Energy & Buildings 329 (2025) 115228

Contents lists available at ScienceDirect

Energy & Buildings

journal homepage: www.elsevier.com/locate/enbuild

Automated model generation and parameter estimation of building energy models using an ontology-based framework

Jakob Bjørnskov [a,*], Muhyiddine Jradi [a], Michael Wetter [b]

[a] SDU Center for Energy Informatics, Maersk Mc-Kinney Moeller Institute, The Faculty of Engineering, University of Southern Denmark, Campusvej 55, Odense, 5230, Denmark
[b] Lawrence Berkeley National Laboratory, 1 Cyclotron Rd, Berkeley, 94720, CA, USA

ARTICLE INFO

Keywords:
Digital twin
Ontology
Data-driven
Building energy model
Building performance simulation

ABSTRACT

This study presents a methodology for automated model generation and parameter estimation of building energy models using semantic modeling and Bayesian estimation. Semantic modeling techniques are used to represent the system components and their interactions, facilitating the automatic generation of a simulation model from dynamic component models. The proposed approach is applied to a case study of a ventilation system where a simulation model is generated, calibrated, and assessed through different performance metrics. These metrics demonstrate the accuracy and reliability of both model point estimates and probabilistic prediction intervals across all model outputs. Overall, the proposed methodology offers a systematic and automated approach to model development and calibration in building energy systems, with potential applications in building performance analysis, monitoring, and optimization.

1. Introduction

The global transition towards future low-carbon energy systems calls for more energy-efficient and flexible buildings [1]. However, increasing efficiency and flexibility brings additional complexity to both the design and operation of buildings. According to Hong et al. [2], the field of building performance simulation (BPS) will play a crucial role in this transition through services such as automated fault-detection, model-aided decision-making, and improved control of buildings. BPS represents the use of computational methods for modeling and simulating energy and indoor climate and has played an important role in the design of energy-efficient and sustainable buildings with decades of simulation program development, e.g. EnergyPlus, DOE-2, and TRNSYS [3–5].

Although the potential benefits of BPS-based services are well-established, broader adoption by the industry faces three fundamental challenges. First, the specification of building system components remains a largely manual process, with engineers spending considerable time interpreting technical documentation and entering specifications into simulation tools. Second, data integration across building systems is hindered by vendor-specific naming conventions and incompatible data formats. Third, maintaining accurate models over time proves challenging as buildings evolve, requiring manual updates and recalibration with each system modification.

Semantic modeling offers solutions to these challenges by providing a modern, web-based, and context-rich framework to standardize component descriptions and improve data integration. This enables a machine-readable framework that automatically interprets specifications, bridges naming conventions, and maintains synchronization between physical and virtual systems. This paper focuses on the operational stage and explores the use of semantic models for the automation of two fundamental tasks related to the BPS modeling workflow: model development and model calibration.

1.1. Model development

Model development is the process of gathering and specifying the set of inputs required by BPS tools, including the specification of geometry, equipment design properties, and system topology. This information is often document-based and spread across diverse sources and formats that lack machine-readability, e.g. piping and instrumentation diagrams (P&ID), representing the system topology of buildings, or design specifications from equipment vendors. The process of gathering and mapping this information is to a large degree carried out manually by a modeling expert, making it time-consuming, costly, and error-prone [6,7].

* Corresponding author.
E-mail address: jabj@mmmi.sdu.dk (J. Bjørnskov).

https://doi.org/10.1016/j.enbuild.2024.115228
Received 25 May 2024; Received in revised form 4 November 2024; Accepted 23 December 2024

Available online 2 January 2025

1.2. Automated building energy modeling

Therefore, automation of model development for BPS has received significant attention in recent research projects such as Annex 60 [8] under the International Energy Agency (IEA), and Project 1 [9] under the International Building Performance Simulation Association (IBPSA). The work conducted in these projects mainly considers Building Information Models (BIM) as the primary information source for the automated generation of simulation models and two general approaches can be highlighted.

The first approach is partly automated and relies on mapping relevant information from the source to an intermediate data model. At this step, the user then manually inserts any additional information required by the simulation tool. As a one-time task, algorithmic mapping rules are then formulated to map from the data model to the simulation tool. The work of Cao et al. [10,11] and Wimmer et al. [12,13] uses the Industry Foundation Classes (IFC) standard as the information source, SimModel [14] as the intermediate data model, and Modelica as the simulation environment.

In the second approach, information is mapped directly from the source to the simulation tool. For instance, Andriamamonjy et al. [15] developed a direct translation workflow from IFC to Modelica through a set of mapping rules. The IFC must follow a set of requirements on the format and structure. Compared with the semi-automated approach, the direct translation approach enables full automation of model development, but it comes with the cost of stricter requirements on the information source. This limits its usefulness in practical use cases where incomplete and inconsistent IFC files frequently occur [16]. In addition, it is unclear whether introducing strict requirements on the information source simply shifts manual labor from the energy modeler to the BIM practitioners with no net gain in time spent.

These BIM-based approaches highlight two fundamental challenges in building energy modeling automation. First, the reliance on complete and consistent IFC files is often impractical in real-world applications, where building documentation may be incomplete or inconsistent. Second, these approaches primarily focus on the initial model creation, with limited support for operational phase requirements such as sensor integration and model updating. These limitations become particularly apparent when models need to be calibrated or updated based on operational data.

1.3. Ontologies and semantic models for improved interoperability

More recent development in automation of model development and information mapping revolves around the use of ontologies, which offer several advantages over traditional BIM approaches. While BIM focuses primarily on geometric and static building information, ontologies represent a more flexible type of data model that describes generalized and structured relationships between concepts for specific domains. This flexibility is particularly valuable for operational applications, where relationships between components, sensors, and control systems need to be explicitly represented. The core motivation behind these structures is to improve interoperability [17], which is crucial for highly complex and heterogeneous systems such as buildings, as also emphasized with the recent launch of the IEA Annex 91, dealing with openBIM and ontologies for building services [18].

Recent work by Roa et al. [19] demonstrated how the Brick ontology combined with the Control Description Language (CDL) can be used to streamline the implementation of advanced building control strategies. As part of a field demonstration, a hot water supply temperature setpoint reset strategy was implemented in a case study building.

Zheng et al. [20], presented a data framework based on the Brick ontology for BPS and building management system (BMS) interoperability. The study considered Brick as a common data model between the BPS and BMS software. Both the BPS model development and mapping to Brick was performed in a semi-automatic fashion, relying on some level of manual work.

Wu et al. [21] presented a direct translation framework from the Brick and BOT ontologies to EnergyPlus. Considering a single floor of a university campus building, the study showed that the energy model development can be fully automated once the semantic model is obtained.

Compared to traditional BIM approaches, semantic models offer several key advantages for operational building services. First, they provide more flexible and extensible ways to represent relationships between building components, sensors, and control systems. Second, they can more easily accommodate incomplete or evolving information, which is common in existing buildings. Third, they enable explicit representation of operational aspects such as sensor locations and control. These capabilities make semantic models particularly suitable for maintaining digital twins that need to stay synchronized with physical systems over time.

While these studies demonstrate the potential of semantic models for automated model generation, they primarily focus on the initial model creation. However, semantic models can provide additional value during the operational phase by maintaining explicit relationships between physical measurements and model variables. This capability is particularly important for model calibration, where understanding the connection between sensors and the components they measure is crucial for automated parameter estimation.

1.4. Model calibration

Even in the ideal case where simulation models can be reliably and effortlessly generated from available design data, additional work is still required to unlock operational services such as performance monitoring or predictive control. Most energy models developed from design conditions suffer from a so-called performance gap, i.e. a gap between predictions and actual measurements [2,22]. Chaudhary et al. [23] report performance gaps varying from 23% to 97% in several studies. Closing this gap requires not only calibration of model parameters but also a clear understanding of which physical measurements correspond to which model variables.

Traditionally, model calibration is performed through an iterative trial-and-error process, where model parameters are manually adjusted until the discrepancy between predictions and measurements is acceptable. This process is complicated by the need to manually establish and maintain mappings between sensors and model variables. As already established, such manual approaches are costly and require domain-specific expertise [24,25]. Consequently, research interest in automated approaches is growing [26], with efforts focusing primarily on optimization-based algorithms and Bayesian parameter estimation techniques [26,27].

Among these studies, Chong et al. [28] proposed a method for automated and continuous calibration of BPS models using the Bayesian formulation proposed by Kennedy and O'Hagan (KOH) [29]. The method also partially automates model development by importing geometry and material information from BIM models following the Green Building eXtensible Markup Language (gbXML). However, specification of system topology and properties as well as the mapping between model outputs and measuring devices is performed manually.

1.5. Aim and contribution

Recent studies have focused on the automation of model development and the implementation of automated model calibration approaches mostly as two isolated processes. However, to calibrate a model, information about which model parameters and measured data to calibrate against is required. This information is not considered in current automation schemes and needs to be established in an automated, yet systematic and comprehensive way. In this work, we propose an ontology-based method that unifies automated model development and calibration. It is aimed at the operational stage of buildings combining design data from semantic models and operational data from measuring devices. For automated model generation, the concept of a signature pattern is introduced, which describes in the language of an ontology how and where a simulation model applies. By solving a graph search problem, these patterns are matched against a semantic model to generate a simulation model. With a standardized format of this simulation model, a general Bayesian parameter estimation problem is formulated, which enables probabilistic parameter estimation and simulation. Both the model generation and parameter estimation methodology are implemented and validated on an end-to-end case study of a ventilation system. Here, the SAREF [30] ontology is used in combination with the SAREF4BLDG [31] and SAREF4SYST [32] extensions to represent the topology and physical properties of the system.

The main contributions of this work can be summarized in five key aspects:

1. **Automated model synchronization:** We present a framework that automatically adapts virtual models to physical system changes by enabling continuous model updating based on both operational data and system modifications, potentially eliminating manual maintenance, and providing direct mapping between physical changes and their virtual representations.
2. **Semantic model integration:** We implement a semantic approach that standardizes building system representation through automated interpretation of sensors and components, flexible integration of diverse data sources, and a standardized framework for describing building systems and their relationships.
3. **Probabilistic parameter estimation:** We use a Bayesian estimation framework that enables uncertainty quantification in model parameters and predictions, providing risk-aware decision support and robust model calibration using operational data.
4. **Component-based architecture:** We create a modular framework that supports reusable component models for different building types and systems, allowing easy extension of capabilities and flexible adaptation to various building configurations.
5. **Demonstration and validation on a real case study:** We validate the framework by considering a case study of a real ventilation system, demonstrating its accuracy and reliability.

These contributions advance different aspects of model integration and automation required for effective digital twin implementation in buildings. The combination of semantic modeling and automated calibration creates a foundation that enables virtual models to stay synchronized with physical systems.

The methodology presented in this paper builds on previous efforts. It has been implemented as part of a larger digital twin (DT) framework [40], available as a reusable open-source Python library [54]. Following the definition by Boje et al. [33] and Grieves [34], a DT refers to a concept with three main constituents; a physical system, a virtual system, and a flow of data linking these two systems. Therefore, the automated model generation and calibration method presented in this paper supports this concept by allowing the virtual system to automatically adapt and re-calibrate based on both structural and operational changes in the building.

The paper is structured as follows: in Section 2, an overview of the workflow and methodology is provided. In Section 3, a method is proposed to generate the simulation model by coupling component models through a comparison of predefined signature patterns with information in the semantic model. Following, the problem formulation for Bayesian parameter estimation of the generated model is presented in Section 4. To validate the methodology, it is implemented on a case study of a ventilation system in Section 5. The results are presented and discussed in Section 6. Finally, a conclusion with the main findings and reflections is presented in Section 7.

2. General methodology and workflow

An overview of the overall methodology and information flow adopted in this study is presented in Fig. 1. One of the core modules of automated model generation is a translator that reads information, applies mapping rules and generates a simulation model. As shown, two information sources are used for this process: 1) a semantic model with information about the physical system such as sensor placement, system topology, and device properties, and 2) a component model library storing component models and signatures that describe how they can be connected to other component models and how they relate to the physical system.

To ensure clarity throughout this paper, we consistently distinguish between the following types of models:

- Semantic models represent building information in a structured, machine-readable format, including system topology, equipment specifications, and sensor placement
- Component models are individual simulation units (for example, a fan or heating coil model) that can be connected to form complete simulation models
- Generated simulation models are complete system models automatically assembled from component models based on semantic model information

Post-translation, three main outputs emerge that are used in the calibration process: a simulation model, the target measuring devices, and the target parameters in the simulation model. Here, the target measuring devices represent the sensors and meters the simulation model is calibrated against, where the target parameters represent the set of model parameters considered in the parameter estimation problem. The specific notation used in the diagram is explained further in Section 3 and Section 4.

3. Ontology-driven model generation

In this section, the translation process is further described including, the semantic model, the component model library, and the translator.

3.1. Semantic model

One of the information sources required by the translation method presented in this section is a semantic model with detailed information about the physical building, such as system topology and equipment specifications. Grounded in an underlying ontology, the format and structure of such a data model are organized using a construct called a *triple*, comprising subject, predicate, and object. In addition, types or classes are assigned to specify its meaning, e.g. `BuildingSpace` → `isSpaceOf` → `Building` (using the SAREF4BLDG ontology). Triples can be combined in a graph-like structure such that the object of one triple might be the subject of another to form arbitrarily large semantic models of real systems in a rich and machine-readable format. Here, the detail and scope by which the semantic model can represent the real system is defined by the ontology. In this paper, the semantics are based on the SAREF ontology [30] with the SAREF4BLDG [31] and SAREF4SYST [32] extensions as introduced in [40]. To properly represent the flow system topology and the placement of equipment, SAREF4SYST is extended with concepts from the Flow Systems Ontology (FSO) according to the alignments described by Kukkonen et al. [35].

An example of a semantic model is shown in Fig. 2. Here, the nodes represent instances of classes, e.g. `<flow temperature sensor 1>` of class `Sensor` or `<supply fan>` of class `Fan`, where the brackets `<>` are used to distinguish instances from classes. In this example, the instances are interrelated through the six types of predicates as shown in the legend, e.g. `observes` which is used to describe the property observed by a sensor. The `suppliesFuildTo` predicate from the FSO

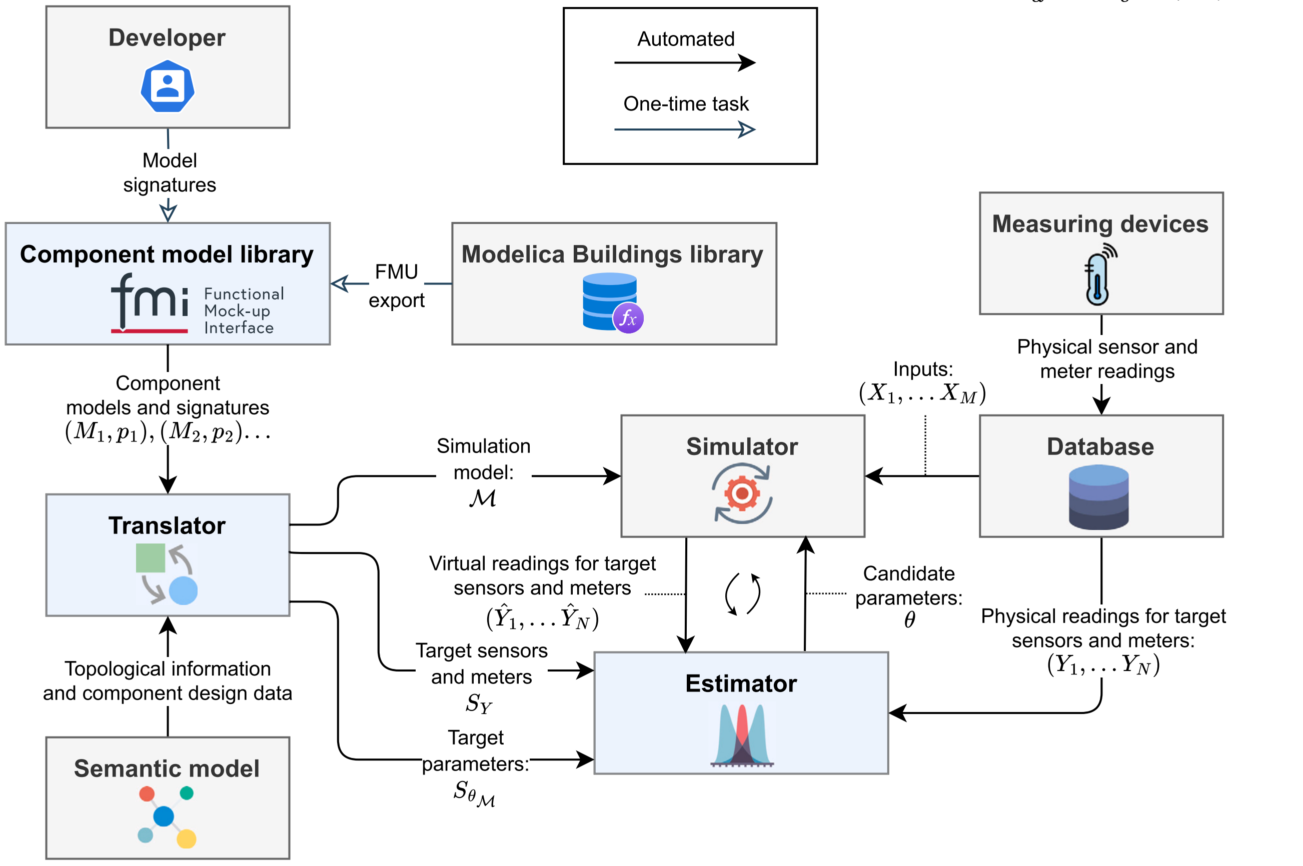
Developer
Model signatures
Automated
One-time task
Component model library
fmi Functional Mock-up Interface
FMU export
Modelica Buildings library
Measuring devices
Physical sensor and meter readings
Component models and signatures
$(M_1, p_1), (M_2, p_2)\ldots$
Inputs: $(X_1, \ldots X_M)$
Simulation model: $\mathcal{M}$
Simulator
Database
Translator
Virtual readings for target sensors and meters $(\hat{Y}_1, \ldots \hat{Y}_N)$
Candidate parameters: θ
Physical readings for target sensors and meters: $(Y_1, \ldots Y_N)$
Target sensors and meters S_Y
Topological information and component design data
Estimator
Target parameters: $S_{\theta_{\mathcal{M}}}$
Semantic model

Fig. 1. Flow diagram showing the information flow between high-level components. The main contributions of this paper are highlighted in light blue boxes. (For interpretation of the colors in the figures, the reader is referred to the web version of this article.)

ontology describes the physical placement of equipment in flow streams. The design properties of equipment, in this case of the nominal airflow of the `<supply fan>`, are described with the `hasPropertyValue`, `isValueOfProperty`, and `hasValue` predicates.

With a basis in this example, it is shown in the following sections how any semantic model can be searched for specific patterns for the automatic generation of a simulation model. Although the description of the developed methodology takes basis in specific ontologies, it should be noted that it generalizes to any ontology.

3.2. Component model library

For this purpose, a modular approach is used where the simulation model is assembled from a set of component models in different configurations to mimic the behavior of the physical system. In this work, we consider the Functional Mockup Interface (FMI) standard [36] as it facilitates the reuse and sharing of dynamic simulation models across a variety of simulation tools and platforms through Functional Mockup Units (FMU). An FMU is a self-contained dynamic simulation model with a set of inputs, outputs, and parameters. In this section, we use five the model classes M_1-M_5 as placeholders for such dynamic models to explain the methodology. For the actual application in Section 5, we use component and system models from the Modelica Buildings library [37].

To ensure the adaptability of the developed models and the automated reuse of the different component models in different applications, a *signature pattern* is defined for each model as a generalized pattern of subject, predicate, and object relations, akin to a blueprint outlining the semantic patterns where the models apply. This concept is illustrated in Fig. 3 where examples of signature patterns p_1-p_5 are shown for M_1-M_5. Each pattern is specified using the triples available from the applied ontology with the nodes representing the ontology classes and the edges

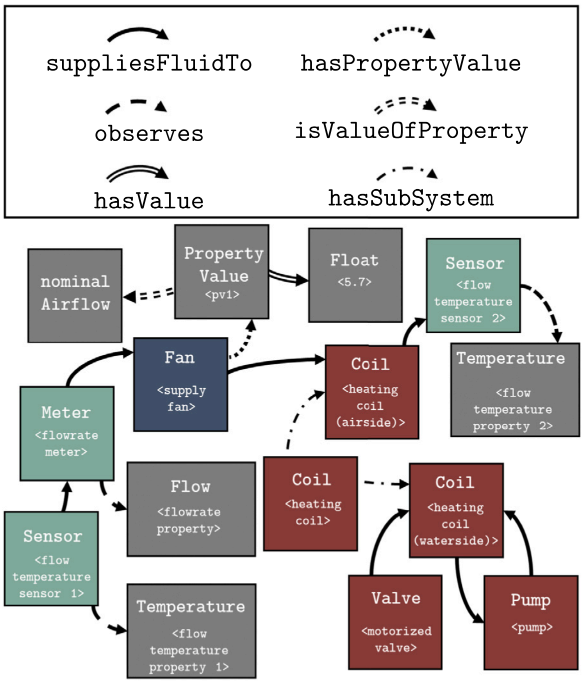

Fig. 2. Example semantic model based on the SAREF, SAREF4BLDG, SAREF4SYST, and FSO ontologies. Inverse predicates have been omitted for clarity.

representing the predicate types. The nodes are marked with a local id, e.g. n_1 in p_1, to distinguish nodes with identical classes. Each node can

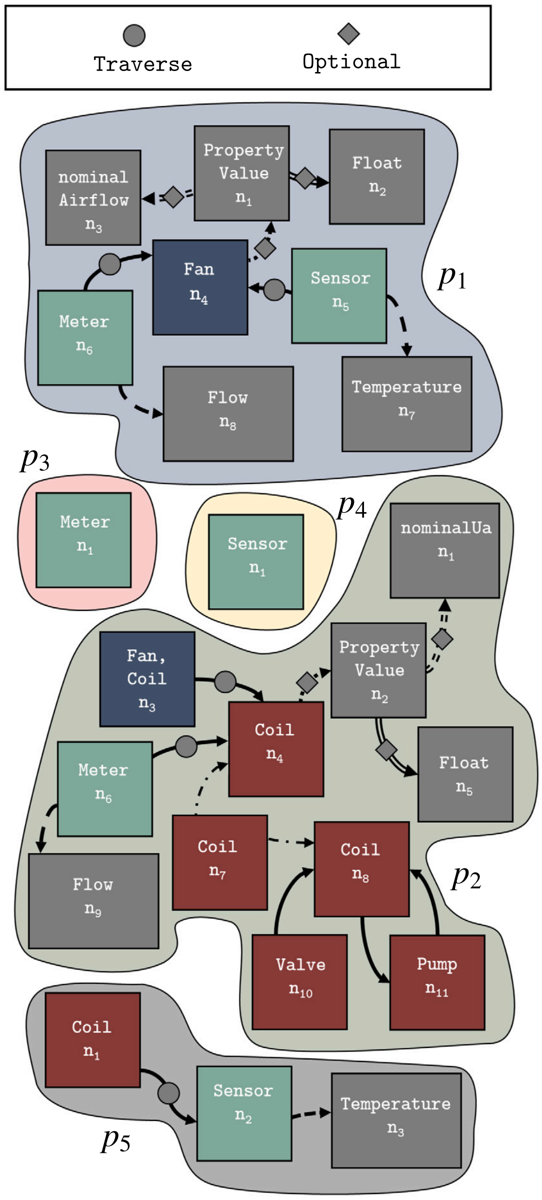

Fig. 3. Semantic triples of the signature patterns p_1-p_5 defined for the five example model classes.

represent multiple classes to support different contexts where the models might apply, e.g. $\mathtt{n}_3$ in p_2, where instances of both class `Fan` and `Coil` are allowed. Taking p_1 as an example, we see that model M_1 applies if the instances of seven triples are present in the semantic model:

1. `Sensor` → `observes` → `Temperature`
2. `Sensor` → `suppliesFluidTo` → `Fan`
3. `Meter` → `observes` → `FlowRate`
4. `Meter` → `suppliesFluidTo` → `Fan`
5. `Fan` → `hasPropertyValue` → `PropertyValue`
6. `PropertyValue` → `isValueOfProperty` → `nominalAirflow`
7. `PropertyValue` → `hasValue` → `Float`

To increase the generality of the signature pattern, rules can be assigned to triples. For instance, as shown in Fig. 3, the second and fourth triple in the list is augmented with the rule `Traverse`, which allows the semantic model to be traversed using the given predicate until an instance of the subject class is found or until the model cannot be traversed any further with the given predicate. This is useful for recognizing certain patterns in flow systems, e.g. to find the nearest temperature sensor or flow meter as shown in the example. In addition, triples can be augmented with the `Optional` rule, which makes the given triple optional, i.e. the pattern can still be matched even though the marked triple is not present in the semantic model. This is used when the mapped information is not strictly required but useful when available, e.g. equipment design parameters that can be estimated through measured data.

Table 1
Example input, output, and parameter mapping of the signature patterns p_1-p_5.

	Nodes	Input		Output	Parameter	
	Modeled	Model ⟷ Source			Model ⟷ Source	
p_1	$\mathtt{n}_4$	$T_{a,in}$ $\dot{m}_a$	$\mathtt{n}_5$: y_{meas} $\mathtt{n}_6$: y_{meas}	$T_{a,out}$ P	$\overline{\dot{m}}_a$	$\mathtt{n}_2$
p_2	$\mathtt{n}_7$ $\mathtt{n}_{10}$ $\mathtt{n}_{11}$	$T_{a,in}$ $\dot{m}_a$	$\mathtt{n}_3$: $(T_{a,out}, T_{a,out})$ $\mathtt{n}_6$: y_{meas}	$T_{a,out}$ $\dot{Q}_{coil}$	$\overline{UA}$	$\mathtt{n}_5$
p_3	$\mathtt{n}_1$			y_{meas}		
p_4	$\mathtt{n}_1$			y_{meas}		
p_5	$\mathtt{n}_2$	y_{meas}	$\mathtt{n}_1$: $T_{a,out}$			

In addition to this specification of triples, input, output, and parameter mapping information is also required for each model as shown in Table 1. The first column specifies the nodes whose behavior is modeled.

Again, taking p_1 as an example, we see that M_1 models the behavior of $\mathtt{n}_4$ (a `Fan`) and requires two inputs. The first input, the inlet air temperature $T_{a,in}$, it receives from the output y_{meas} of the model representing the instance of $\mathtt{n}_5$ while the second input, the air massflow $\dot{m}_a$, is received from the output y_{meas} of the model representing the instance of $\mathtt{n}_6$. The model requires one parameter, the nominal air flow $\overline{\dot{m}}_a$, which is mapped from $\mathtt{n}_2$ (if available). Note that the unit assertions or translations of these parameter values could also be performed by extending the pattern signature with appropriate triples, e.g. `PropertyValue` → `isMeasuredIn` → `UnitOfMeasure`. Finally, the model has two outputs $T_{a,out}$ and P, representing the outlet air temperature and the power consumption of the fan, respectively.

3.3. Translator

By defining signature patterns for each model in the component model library, the semantic model can be searched systematically for matching patterns and the corresponding models can be employed in the correct context. For this task, it is useful to represent the semantic model and the signature patterns as a directed graph (V, E, L) with the set of nodes V, the set of edges E, and a labeling function L, mapping nodes or edges to a corresponding label or set of labels. Given this representation, the task of searching for signature patterns in the semantic model is directly related to the *subgraph isomorphism* problem. In the context of this paper, the subgraph isomorphism problem can be stated as follows [38]. Given the pattern signature represented by the graph $p = (V_p, E_p, L_p)$ and the semantic model represented by the graph $G = (V_g, E_g, L_g)$, find the map $f : V_p \to V_g$ such that $\forall u \in V_p$, $L_g\big(f(u)\big) \subseteq L_p(u)$ and $\forall (u, v) \in E_p$, $L_p(u, v) = L_g\big(f(u), f(v)\big)$ and $\big(f(u), f(v)\big) \subseteq E_g$. The condition $L_g\big(f(u)\big) \subseteq L_p(u)$ requires that the node label, i.e. the ontology class, of the semantic model is a subset of the pattern node label. Similarly, $L_p(u, v) = L_g\big(f(u), f(v)\big)$ ensures that the edge label, i.e. the ontology predicate, of the semantic model matches the pattern edge label. Finally, $\big(f(u), f(v)\big) \subseteq E_g$ ensures that the mapped pattern edge $\big(f(u), f(v)\big)$ also exists in the semantic model.

Various algorithms that solve this problem already exist. In this work, we have adapted the VF2 algorithm which applies a depth-first search that simultaneously traverses through the signature pattern and semantic model to add or eliminate nodes as matching candidates based

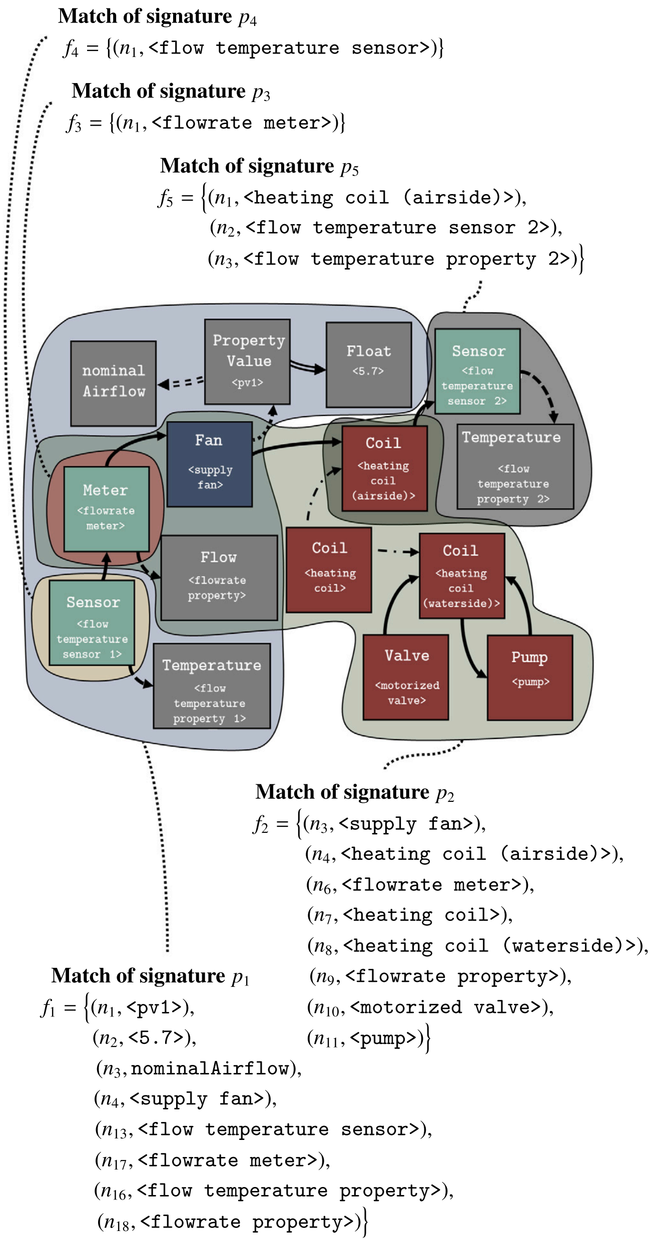
Match of signature p_4
$f_4 = \{(n_1, \texttt{<flow temperature sensor>})\}$
Match of signature p_3
$f_3 = \{(n_1, \texttt{<flowrate meter>})\}$
Match of signature p_5
$f_5 = \{(n_1, \texttt{<heating coil (airside)>}), (n_2, \texttt{<flow temperature sensor 2>}), (n_3, \texttt{<flow temperature property 2>})\}$
nominal Airflow
Property Value <pv1>
Float <5.7>
Sensor <flow temperature sensor 2>
Temperature <flow temperature property 2>
Fan <supply fan>
Coil <heating coil (airside)>
Meter <flowrate meter>
Flow <flowrate property>
Coil <heating coil>
Coil <heating coil (waterside)>
Sensor <flow temperature sensor 1>
Temperature <flow temperature property 1>
Valve <motorized valve>
Pump <pump>
Match of signature p_2
$f_2 = \{(n_3, \texttt{<supply fan>}), (n_4, \texttt{<heating coil (airside)>}), (n_6, \texttt{<flowrate meter>}), (n_7, \texttt{<heating coil>}), (n_8, \texttt{<heating coil (waterside)>}), (n_9, \texttt{<flowrate property>}), (n_{10}, \texttt{<motorized valve>}), (n_{11}, \texttt{<pump>})\}$
Match of signature p_1
$f_1 = \{(n_1, \texttt{<pv1>}), (n_2, \texttt{<5.7>}), (n_3, \texttt{nominalAirflow}), (n_4, \texttt{<supply fan>}), (n_{13}, \texttt{<flow temperature sensor>}), (n_{17}, \texttt{<flowrate meter>}), (n_{16}, \texttt{<flow temperature property>}), (n_{18}, \texttt{<flowrate property>})\}$

Fig. 4. Matching of patterns in the semantic model to generate maps between signature nodes and semantic model instances.

on local information. The original algorithm is described in detail in [39].

Solving the subgraph isomorphism problem for the semantic model in Fig. 2 and each of the patterns p_1-p_5 in Fig. 3 generates the maps f_1-f_5 shown in Fig. 4. As shown, all five signature patterns can be correctly matched once in the semantic model. For each match, the corresponding models are instantiated (referred to as m_1-m_5) and connected using the generated maps and the information shown in Table 1.

For each model instance, the input connections are established by intersecting the identified instances between the 'Input, Source' column and the 'Nodes Modeled' column. For example, to identify the $T_{a,in}$ input of m_1, we use the f_1 map to find that the source of this input is $f_1(\mathrm{n}_5)$ = `<flow temperature sensor>`. Since $f_4(\mathrm{n}_1)$ = `<flow temperature sensor>` with n_1 being a modeled node, the model representing this instance is m_4. Altogether, this means that the output y_{meas} of m_4 maps to the input $T_{a,in}$ of m_1. Completing this process for all inputs and model instances generates the model $\mathcal{M}$ shown in Fig. 5.

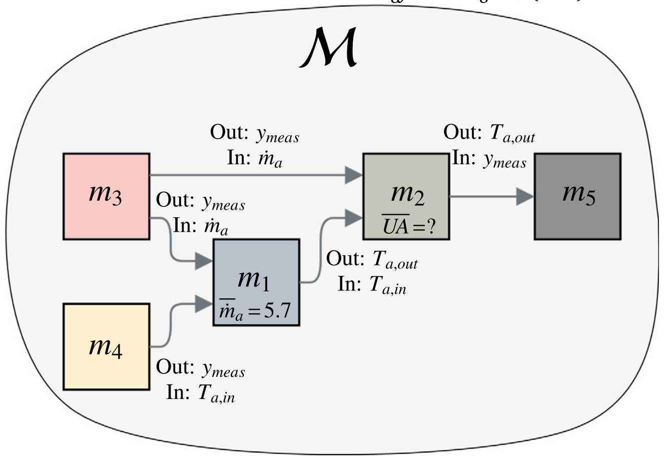

Fig. 5. Resulting simulation model after matching and connecting the component models based on their signature. The boxes represent the component models while the labeled edges represent the inputs and outputs of models.

3.4. Identifying target measuring devices and parameters

After generating a simulation model $\mathcal{M}$, the required information for the parameter estimation stage is gathered. A vital step for both parameter estimation, as well as downstream BPS applications such as performance monitoring, is the comparison of simulated versus measured values. As introduced in earlier work [40], a key concept of the used modeling framework is to incorporate measuring devices into the simulation model on equal terms as other models, and use these as an interface for straightforward comparison between simulated and measured values. Using the notation from [40], the set of measuring devices used for parameter estimation S_Y is given by Equation (1).

$$S_Y = \{ s \in H_y : |s.\text{connectsAt}| = 1 \} \tag{1}$$

In Equation (1), H_y is the set of component models in $\mathcal{M}$ that models nodes with a measuring device class. In the SAREF ontology for instance, these classes would be `saref:Sensor` or `saref:Meter` while in the Brick ontology they would be `brick:Sensor`. The $|s.\text{connectsAt}| = 1$ condition filters for models with 1 input. From the generated maps f_1-f_5, only three contain measuring devices that are also modeled:

- $f_3(\mathrm{n}_1) =$ `<flowrate meter>`
- $f_4(\mathrm{n}_1) =$ `<flow temperature sensor 1>`
- $f_5(\mathrm{n}_2) =$ `<flow temperature sensor 2>`

This gives $H_y = \{m_3, m_4, m_5\}$ and $S_Y = \{m_5\}$ as m_3 and m_4 acts as sources with 0 inputs. The model m_5 thus acts as a virtual sensor and is used as a calibration target for the example model, as its readings can be directly compared to the actual sensor represented by `<flow temperature sensor 2>`.

In addition to identifying the target measuring devices, the target set of parameters $S_{\theta_{\mathcal{M}}}$ must also be identified. This set is defined in Equation (2).

$$S_{\theta_{\mathcal{M}}} = S_\theta \setminus S_{\theta,\text{mapped}} \tag{2}$$

Here, S_θ is the total set of parameters in $\mathcal{M}$, i.e. the entries in the 'Parameter, Model' column in Table 1. $S_{\theta,\text{mapped}}$ is the set of parameters in the simulation model that are successfully mapped from the semantic model with an entry in the generated map. Hence, for the example model $\mathcal{M}$ in Fig. 5, the target parameter set becomes $S_{\theta_{\mathcal{M}}} = \{\overline{UA}\}$ as $S_\theta = \{\overline{\dot{m}}_a, \overline{UA}\}$ and $S_{\theta,\text{mapped}} = \{\overline{\dot{m}}_a\}$ with $\overline{\dot{m}}_a$ mapped to 5.7.

4. Bayesian parameter estimation

Given the simulation model $\mathcal{M}$, the calibration targets $S_{\boldsymbol{Y}}$, and the target parameters of the model $S_{\theta_{\mathcal{M}}}$, parameter estimation can now be performed. A wide variety of methods exist for this purpose, e.g. gradient-based algorithms or heuristic-based methods such as genetic algorithms. These optimization-based algorithms typically provide a point estimate of the best-fitting model parameters given an objective function. In contrast, in a Bayesian framework, model parameters are treated as random variables and the outcome of an estimation process is probability distributions for each parameter in the model. This fundamental difference provides several advantages in favor of Bayesian estimation. 1) It explicitly handles problems where fundamentally different parameter sets reproduce the observed data of the modeled system equally well, addressing the identifyability problem discussed by Reddy [22]. 2) Using the obtained parameter distributions, uncertainty can be propagated through the model to give uncertainty estimates on predictions.

This quantification of uncertainty is especially valuable in the context of automation, where the calibration process is not supervised by an expert modeler and if the model is used as part of a decision-making process [26]. In the following section, the Bayesian estimation problem is introduced along with the sampling technique used to solve this problem.

4.1. Problem formulation

Consider a dynamic simulation model $\mathcal{M}$ parameterized by the vector of parameters $\boldsymbol{\theta}_{\mathcal{M}} \in \mathbb{R}^P$ with the time series inputs $\boldsymbol{X} = (\boldsymbol{X}_1, \ldots \boldsymbol{X}_M)$ with $\boldsymbol{X}_i \in \mathbb{R}^T \forall_{i\in 1..M}$, outputs $\hat{\boldsymbol{Y}} = (\hat{\boldsymbol{Y}}_1, \ldots \hat{\boldsymbol{Y}}_N)$ with $\hat{\boldsymbol{Y}}_i \in \mathbb{R}^T \forall_{i\in 1..N}$, and timestamps $\boldsymbol{t} \in \mathbb{R}^T$. Here, $P = |S_{\theta}|$, M, $N = |S_{\boldsymbol{Y}}|$, and T represents number of parameters, inputs, outputs, and timesteps, respectively. The relationship between inputs and outputs is then given by Equation (3).

$$\hat{\boldsymbol{Y}} = \mathcal{M}\left(\boldsymbol{X}, \boldsymbol{t}, \boldsymbol{\theta}_{\mathcal{M}}\right) \tag{3}$$

Using KOH's Bayesian model formulation [29], the relationship between observations $\boldsymbol{Y}_y$, model response $\hat{\boldsymbol{Y}}_y$, and discrepancy $r_y(\cdot)$ is given by Equation (4).

$$\boldsymbol{Y}_y = \hat{\boldsymbol{Y}}_y + r_y(\cdot) \ \forall_{y\in 1..N} \tag{4}$$

Here, the purpose of the discrepancy term $r_y(\cdot)$ is to account for structural inadequacies in the model and is modeled as a Gaussian process with a mean function of 0 and covariance function $k_y(\boldsymbol{x}_i, \boldsymbol{x}_j)$ as given in Equation (5).

$$r_y(\boldsymbol{x}_i, \boldsymbol{x}_j) \sim \mathcal{GP}\left(\boldsymbol{0}, k_y(\boldsymbol{x}_i, \boldsymbol{x}_j)\right) \ \forall_{y\in 1..N} \tag{5}$$

In this work, we model the covariance function using the Matern 3/2 kernel as shown in Equation (6).

$$k_y(\boldsymbol{x}_i, \boldsymbol{x}_j) = a_y \left(1 + \sqrt{3} d_y(\boldsymbol{x}_i, \boldsymbol{x}_j)\right) \exp\left(-\sqrt{3} d_y(\boldsymbol{x}_i, \boldsymbol{x}_j)\right) \ \forall_{y\in 1..N} \tag{6}$$

Where a_y is a parameter controlling the variance of the kernel function. Using this kernel, the covariance decays with the distance $d_y(\boldsymbol{x}_i, \boldsymbol{x}_j)$, which is computed between two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ using Equation (7).

$$d_y(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{(\boldsymbol{x}_i - \boldsymbol{x}_j)^\intercal \boldsymbol{C}_y^{-1} (\boldsymbol{x}_i - \boldsymbol{x}_j)} \ \forall_{i,j\in 1..T} \tag{7}$$

The metric $\boldsymbol{C}_y$ is populated with the hyper parameters $\boldsymbol{l}_y = [l_{y,1}, \ldots l_{y,|\boldsymbol{x}_i|}]^\intercal$ on the diagonal, determining the scale of the inputs as shown in Equation (8).

$$\boldsymbol{C}_y = \operatorname{diag}(\boldsymbol{l}_y) \tag{8}$$

All the gaussian process hyperparameters are then given by the vector $\boldsymbol{\theta}_{\mathcal{GP}}$, as shown in Equation (9).

$$\boldsymbol{\theta}_{\mathcal{GP}} = [a_1, \boldsymbol{l}_1^\intercal, \ldots a_N, \boldsymbol{l}_N^\intercal]^\intercal \tag{9}$$

Following, the full vector of parameters $\boldsymbol{\theta}$ is then given by Equation (10).

$$\boldsymbol{\theta} = [\boldsymbol{\theta}_{\mathcal{M}}^\intercal, \boldsymbol{\theta}_{\mathcal{GP}}^\intercal]^\intercal \tag{10}$$

Having defined the independent gaussian processes for each output y, the likelihood of making all observations $(\boldsymbol{Y}_1, \ldots \boldsymbol{Y}_N)$, given the parameters $\boldsymbol{\theta}$, is given by Equation (11).

$$\ln \mathcal{L}(\boldsymbol{Y}_1, \ldots \boldsymbol{Y}_N | \boldsymbol{\theta}) = \sum_{y=1}^{N} -\frac{1}{2}\left(\boldsymbol{r}_y^\intercal \boldsymbol{K}_y^{-1} \boldsymbol{r}_y + \ln\det \boldsymbol{K}_y + T \ln 2\pi\right) \tag{11}$$

Where $\boldsymbol{r}_y$ is the residual between observations and model response, as given in Equation (12).

$$\boldsymbol{r}_y = \boldsymbol{Y}_y - \hat{\boldsymbol{Y}}_y \ \forall_{y\in 1..N} \tag{12}$$

Each element of row i and column j of the covariance matrix $\boldsymbol{K}_y$ is computed using the kernel function and is given by Equation (13).

$$\boldsymbol{K}_{y,ij} = k_y(\boldsymbol{x}_{y,i}, \boldsymbol{x}_{y,j}) + \sigma_y^2 \delta_{ij} \ \forall_{i,j\in 1..T} \ \forall_{y\in 1..N} \tag{13}$$

Where the points $\boldsymbol{x}_{y,i}$ and $\boldsymbol{x}_{y,j}$ are defined as the i^{th} and j^{th} sample from a subset B_y of the model inputs, i.e. $\boldsymbol{x}_{y,i} = [\boldsymbol{X}_{k,i}, \ldots]^\intercal_{\forall k\in B_y}$ with $B_y \subseteq S_{\boldsymbol{X}}$. The set B_y is used as a filter to make sure that only relevant model inputs are used in the Gaussian process for model output y. Measurement error is modeled as i.i.d. white noise by adding variance σ_y^2 at the diagonal through the Kronecker delta function δ_{ij} which is 1 for $i = j$ and otherwise 0.

As earlier stated, the goal of Bayesian estimation is to estimate the distributions of a set of model parameters, given some observed data of the modeled system. Following Bayes' rule, this distribution is given by Equation (14).

$$\pi(\boldsymbol{\theta}|\boldsymbol{Y}) = \frac{\mathcal{L}(\boldsymbol{Y}|\boldsymbol{\theta})\pi_0(\boldsymbol{\theta})}{\int_{\mathbb{R}^P} \mathcal{L}(\boldsymbol{Y}|\boldsymbol{\theta})\pi_0(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta}} \tag{14}$$

Where $\pi(\boldsymbol{\theta}|\boldsymbol{Y}_1, \ldots \boldsymbol{Y}_N)$ is the posterior density of observing the parameters $\boldsymbol{\theta}$ given the observations $\boldsymbol{Y}_1, \ldots \boldsymbol{Y}_N$. The integral in the denominator, which evaluates to a constant, ensures that the posterior integrates to 1. $\pi_0(\boldsymbol{\theta})$ is the prior density of observing $\boldsymbol{\theta}$. This distribution encodes information about the parameters before observing $\boldsymbol{Y}_1, \ldots \boldsymbol{Y}_N$, typically in the form of domain-specific knowledge. For the application shown in this work, we assume uniform distributions defined through upper and lower bounds on the estimated parameters. Uniform priors are also known as uninformative priors because they do not encode any information about the parameters other than the defined bounds. Another commonly used prior is the normal distribution. However, defining a meaningful prior normal distribution for each parameter would require case-specific expert knowledge of the modeled system, which defeats the purpose of the applied automation methodology.

4.2. Markov chain Monte Carlo sampling

Although both the likelihood $\mathcal{L}(\boldsymbol{Y}|\boldsymbol{\theta})$ and the prior $\pi_0(\boldsymbol{\theta})$ in Equation (14) are known, in most cases, obtaining a closed-form solution or performing any form of expectation calculations using traditional numerical integration methods is infeasible [41]. Instead, the numerical sampling method Markov Chain Monte Carlo (MCMC) is frequently used for drawing samples from the posterior using different variants, e.g. Metropolis-Hastings [42,43] and the Affine Invariant MCMC Ensemble sampler [44].

4.3. Parallel tempering ensemble MCMC

In an automated model generation and calibration framework, where important characteristics of the generated model and estimation problem vary (e.g. number of target measuring devices and target parameters), a robust MCMC sampler that can deal with a wide variety of likelihood functions is important. Here, traditional MCMC schemes are often ineffective as they can get stuck at local high-probability regions of the parameter space, which prevents the algorithm from converging to the posterior in a reasonable amount of time [45]. In this work, a specific version of MCMC called Parallel tempering MCMC (PTMCMC) is therefore used. PTMCMC makes use of multiple chains that are run in parallel to sample from tempered versions of the posterior $\pi_T(\boldsymbol{\theta}|\boldsymbol{Y})$, which is given in Equation (15).

$$\pi_T(\boldsymbol{\theta}|\boldsymbol{Y}_1,...\boldsymbol{Y}_N) \propto \mathcal{L}(\boldsymbol{Y}_1,...\boldsymbol{Y}_N|\boldsymbol{\theta})^{1/T}\pi_0(\boldsymbol{\theta}) \tag{15}$$

The exponent in Equation (15) flattens the likelihood function to make the parameter space easier to explore. When $T \to \infty$, the posterior becomes the prior, which in this work is a uniform distribution that can easily be explored. During runtime, samples are swapped periodically between chains of different temperatures to allow solutions to propagate from hot chains to colder chains. Further details on the PTMCMC algorithm are provided by [44,45].

4.4. Posterior predictive distribution and inference

As a key property of all MCMC variants, the distribution of the drawn samples converges to the desired posterior distribution in the limit as the chain progresses and the number of samples increases. The first part of the chain where convergence has not been reached is known as the burn-in phase and is discarded. The sampler convergence, i.e. the number of burn-in steps, can be determined by computing the integrated autocorrelation time (IAT) and comparing it with the total number of steps [41], where IAT is a measure of the number of steps required on average for an independent sample to be drawn.

The remaining k samples drawn from the posterior distribution, i.e. $\Theta_k = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 ... \boldsymbol{\theta}_k\}$ where $\boldsymbol{\theta}_p \sim \pi(\boldsymbol{\theta}|\boldsymbol{Y}_1,...\boldsymbol{Y}_N)\forall_{p\in 1..k}$ can then be used for various applications. In the context of BPS, the straightforward application is to produce the *posterior predictive distribution* which represent probabilistic predictions for future or unseen conditions of the modeled system. Given the parameter sample $\boldsymbol{\theta} \in \Theta_k$, the training data $(\boldsymbol{X}, \boldsymbol{Y})$ and test data $(\boldsymbol{X}^*, \boldsymbol{Y}^*)$, the predictive distribution of model discrepancy $\boldsymbol{r}_y^*$ is first obtained by conditioning on the observations as given in Equation (16) [46].

$$\boldsymbol{r}_y^*|\boldsymbol{\theta}, \boldsymbol{X}, \boldsymbol{r}_y, \boldsymbol{X}^* \sim \mathcal{N}\left(\bar{\boldsymbol{r}}_y^*, \mathrm{cov}(\boldsymbol{r}_y^*)\right) \tag{16}$$

Where $\bar{\boldsymbol{r}}_y^*$ is the mean vector given by Equation (17) and $\mathrm{cov}(\boldsymbol{r}_y^*)$ is the covariance matrix given by Equation (18).

$$\bar{\boldsymbol{r}}_y^* = \boldsymbol{K}_y^*\left(\boldsymbol{K}_y\right)^{-1}\boldsymbol{r}_y \tag{17}$$

$$\mathrm{cov}(\boldsymbol{r}_y^*) = \boldsymbol{K}_y^{**} - \boldsymbol{K}_y^*\left(\boldsymbol{K}_y\right)^{-1}(\boldsymbol{K}_y^*)^{\intercal} \tag{18}$$

The elements of matrix $\boldsymbol{K}_{y,ij}^*$ are calculated as the covariance between the test and training inputs as shown in Equation (19) while the elements of matrix $\boldsymbol{K}_{y,ij}^{**}$ are calculated as the covariance between the test inputs as shown in Equation (20).

$$\boldsymbol{K}_{y,ij}^* = k_y(\boldsymbol{x}_{y,i}^*, \boldsymbol{x}_{y,j})\ \forall_{i\in 1..T^*}\ \forall_{j\in 1..T}\ \forall_{y\in 1..N} \tag{19}$$

$$\boldsymbol{K}_{y,ij}^{**} = k_y(\boldsymbol{x}_{y,i}^*, \boldsymbol{x}_{y,j}^*)\ \forall_{i,j\in 1..T^*}\ \forall_{y\in 1..N} \tag{20}$$

Finally, the posterior predictive distribution $\boldsymbol{Y}_y^p$ for one parameter sample is given by the sum of the model response $\hat{\boldsymbol{Y}}_y^*$ as shown in Equation (21) and the predictive distribution for the discrepancy $\boldsymbol{r}_y^*|\boldsymbol{\theta}, \boldsymbol{X}, \boldsymbol{r}_y, \boldsymbol{X}^*$ as shown in Equation (22).

$$\hat{\boldsymbol{Y}}_y^* = \mathcal{M}\left(\boldsymbol{X}^*, \boldsymbol{t}^*, \boldsymbol{\theta}_{\mathcal{M}}\right) \tag{21}$$

$$\boldsymbol{Y}_y^p = \hat{\boldsymbol{Y}}_y^* + \boldsymbol{r}_y^*|\boldsymbol{\theta}, \boldsymbol{X}, \boldsymbol{r}_y, \boldsymbol{X}^*\ \forall_{y\in 1..N} \tag{22}$$

5. Case study and implementation

To validate the proposed methodology, we apply it to a case study. The presented methodology has been implemented as part of a reusable Python library [54]. The considered case study is a section of a real ventilation system in a university building that supplies around 20 zones with air and has a capacity of 34.000 m^3/h. A diagram is shown in Fig. 6 with the installed measuring devices and equipment.

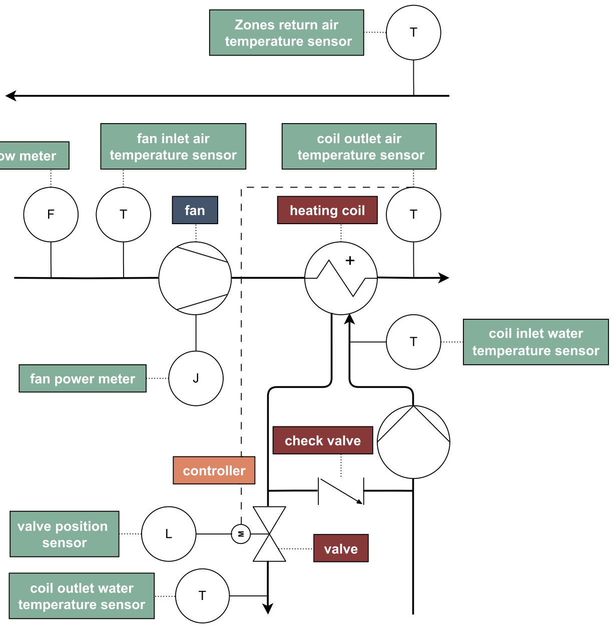
Zones return air temperature sensor
T
fan flow meter
fan inlet air temperature sensor
coil outlet air temperature sensor
F
T
fan
heating coil
T
+
T
coil inlet water temperature sensor
fan power meter
J
check valve
controller
valve position sensor
L
valve
coil outlet water temperature sensor
T

Fig. 6. Diagram of the ventilation case study system.

Four components from the supply side are modeled: a fan, a heating coil, a controller, and a valve. The system employs demand-controlled ventilation (DCV) and the fan is operated at variable speed to maintain a constant supply air pressure. The heating coil is responsible for heating the supply air to a desired setpoint temperature, which is calculated using a piece-wise linear function of the measured return air temperature. This is achieved by actuating the position of a motorized valve to regulate the waterflow. The required control signal for the valve position is determined by a PI controller. Weather data, in the form of outdoor temperature, was obtained from the nearest weather station through the API of the Danish Meteorological Institute [47].

The semantic model used to represent the case study system is shown in Fig. 7. Similar to the example presented in Section 3, the SAREF, SAREF4BLDG, SAREF4SYST, and FSO ontologies are used. As recommended by the FSO ontology [35], the `<heating coil>` is split into an airside and waterside to avoid ambiguous configurations where it is impossible to determine whether components placed before and after the coil are on the airside or waterside. As shown in Fig. 7, the `suppliesFluidTo` predicate is used to encode that the `<fan>` is placed before the `<heating coil (airside)>` in the flow path. Many of the shown predicates have an inverse, e.g. `hasFluidSuppliedBy` pointing from the `<heating coil (airside)>` to the `<fan>`.

The use of these predicates also enables the representation of measuring devices as part of the flow stream, e.g. the `<fan flow meter>`, which is placed between the `<fan inlet air temperature sensor>` and the `<fan>`. Measuring devices not physically part of the flow stream can also be related to a specific device through the `saref:ob-`

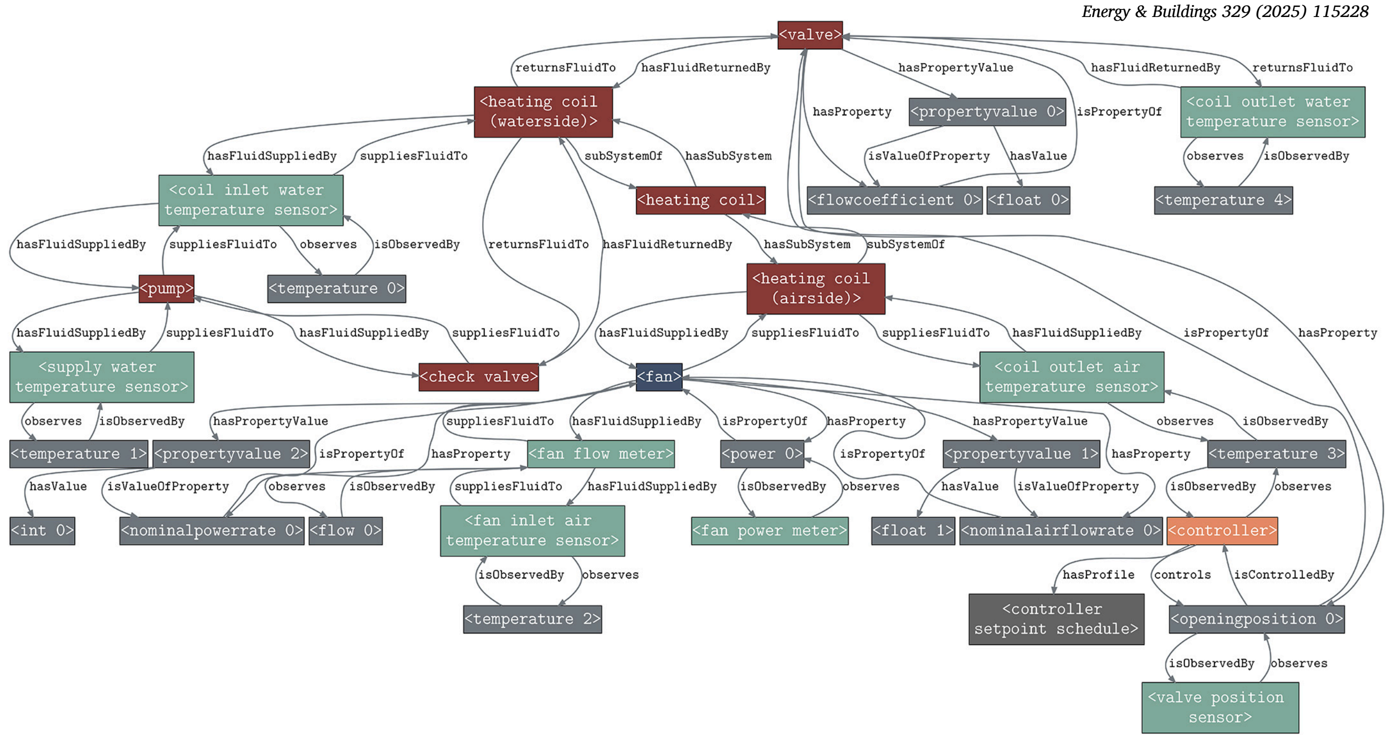

Fig. 7. Semantic model of the case study system.

serves and saref:isPropertyOf as is the case for the <fan power meter>.

5.1. Hot water loop model

The hot water loop containing the heating coil, motorized valve, check valve, and pump is modeled jointly in one FMU as shown in Fig. 8. Although it is generally preferable to export each component model separately to maximize reusability, a more specialized component is preferred in this case to reduce complexity and increase efficiency of the generated model. In addition, the hot water loop configuration for the case study system is fairly common which further justifies the development of a specialized model.

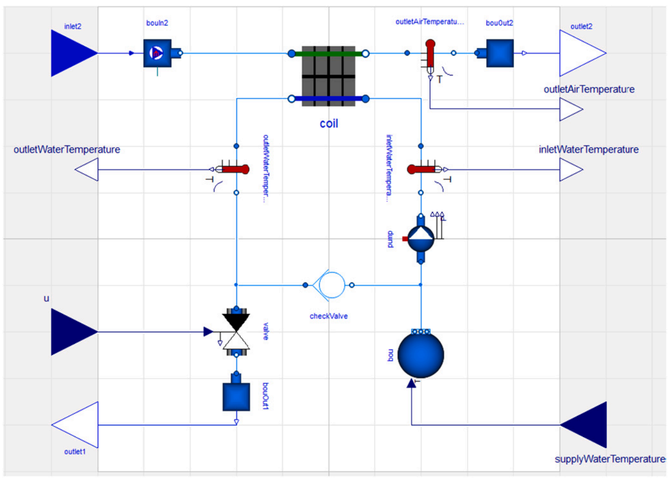

Fig. 8. Modelica diagram of the exported hot water loop model.

For the heating coil, the Buildings.Fluid.HeatExchangers.-DryCoilDiscretized model is used, which models the heating coil as a dynamic heat exchanger. The considered parameters for this model

Table 2
Mapping of inputs, outputs, and parameters of the hot water loop signature pattern.

Nodes	Input		Output	Parameter	
Modeled	Model ⟷ Source			Model ⟷ Source	
n$_2$	$T_{c,a,in}$	n$_{16}$: $(T_{f,out}, T_{c,a,out})$	$T_{c,a,out}$	$\overline{UA}$	n$_{18}$
n$_6$	$T_{w,sup}$	n$_{15}$: y_{meas}	$T_{c,w,in}$	τ_w	–
n$_{11}$	$\dot{m}_{c,a}$	n$_1$: $\dot{m}_a$	$T_{c,w,out}$	τ_m	–
n$_{12}$	u_v	n$_{14}$: u		τ_a	–
				$\overline{\dot{m}}_{c,w}$	–
				$\overline{\dot{m}}_{c,a}$	–
				ΔP_c	–
				K_v	n$_{24}$
				$\overline{\dot{m}}_v$	–
				K_{cv}	n$_{21}$
				$\overline{\dot{m}}_{cv}$	–
				ΔP_p	–
				ΔP_s	–
				ΔP_{res}	–

are the nominal heat transfer coefficient $\overline{UA}$, the time constants of the waterside, airside, and metal τ_w, τ_a, τ_m, the water and air nominal mass flows $\overline{\dot{m}}_{c,w}$, $\overline{\dot{m}}_{c,a}$ as well as the pressure drop across the heating coil ΔP_c.

Assuming a linear valve characteristic, the motorized valve is modeled using Buildings.Fluid.Actuators.Valves.TwoWayLinear, which is parameterized by the flow coefficient K_v, the nominal water flowrate $\overline{\dot{m}}_v$, and a fixed resistance ΔP_{res}, that represents all system pressure losses after the motorized valve. The check valve is modeled with Buildings.Fluid.FixedResistances.CheckValve and is also parameterized by a flow coefficient K_{cv} and the nominal water flowrate $\overline{\dot{m}}_{cv}$. The pump is assumed to provide a constant pressure rise ΔP_p and the pressure drop between the source and sink is assumed to be constant and is given by the parameter ΔP_s. The signature pattern of the model is given by Table 2 and Fig. 9.

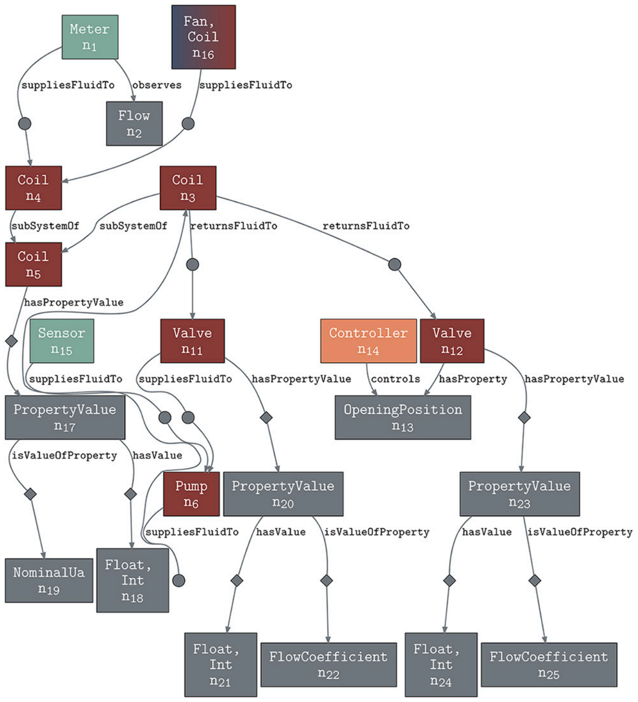

Fig. 9. Semantic triples of the hot water loop signature pattern.

5.2. Controller

The controller is modeled using `Buildings.Controls.-Continuous.LimPID`, configured as a reverse-acting PI controller parameterized by the proportional gain K_P and the integral time constant T_I. The Modelica diagram is shown in Fig. 10. Compared to the hot water loop model, the controller model applies for PI controllers in general, as also indicated by the signature pattern as given in Table 3 and Fig. 11.

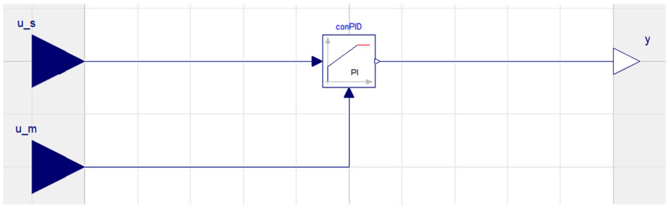

Fig. 10. Modelica diagram of the exported controller model.

Table 3
Mapping of inputs, outputs, and parameters of the controller signature pattern.

Nodes	Input		Output	Parameter	
Modeled	Model $\longleftrightarrow$ Source			Model $\longleftrightarrow$ Source	
n$_1$	y_{set}	n$_4$: y_{sched}	u	K_P	–
	y_{meas}	n$_2$: y_{meas}		K_I	–

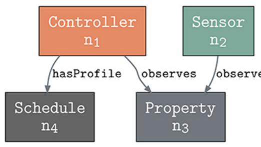

Fig. 11. Semantic triples of the controller signature pattern.

5.3. Fan

The fan component is modeled using the polynomial power curve used by EnergyPlus [48] and the Modelica diagram is shown in Fig. 12. This model is parameterized by the nominal power consumption $\overline{P_f}$ and the nominal air massflow $\overline{\dot{m}}_{f,a}$ along with the power coefficients c_1-c_4.

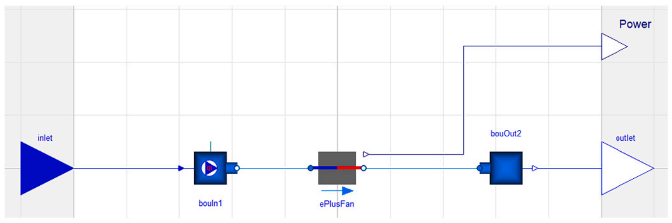

Fig. 12. Modelica diagram of the exported fan model.

In addition, to model the temperature rise across the fan, `Buildings.Fluid.Interfaces.TwoPortHeatMassExchanger` is used, where heat added to the air stream $\dot{Q}_{toAir}$ is modeled as a fraction f_{tot} of P_f as shown in Equation (23).

$$\dot{Q}_{toAir} = f_{tot} P_f \tag{23}$$

The signature pattern of the model is given by Table 4 and Fig. 13. As shown in Table 4, 2 of the 7 model parameters can be mapped from the semantic model.

Table 4
Mapping of inputs, outputs, and parameters of the fan signature pattern.

Nodes	Input		Output	Parameter	
Modeled	Model $\longleftrightarrow$ Source			Model $\longleftrightarrow$ Source	
n$_3$	$\dot{m}_{f,a}$	n$_1$: $\dot{m}_a$	P_f	$\overline{P_f}$	n$_5$
	$T_{f,in}$	n$_2$: y_{meas}	$T_{f,out}$	$\overline{\dot{m}}_{f,a}$	n$_8$
				c_1	–
				c_2	–
				c_3	–
				c_4	–
				f_{tot}	–

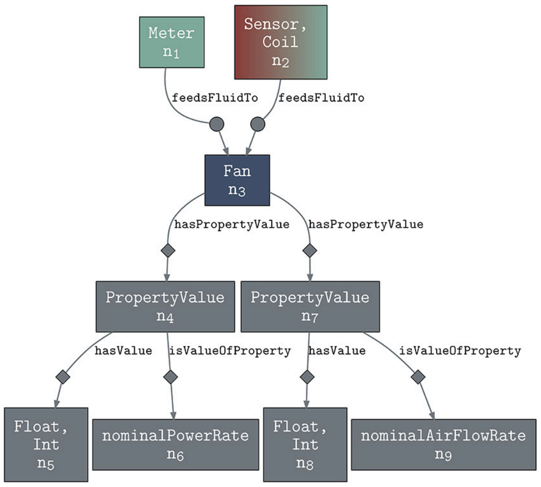

Fig. 13. Semantic triples of the fan signature pattern.

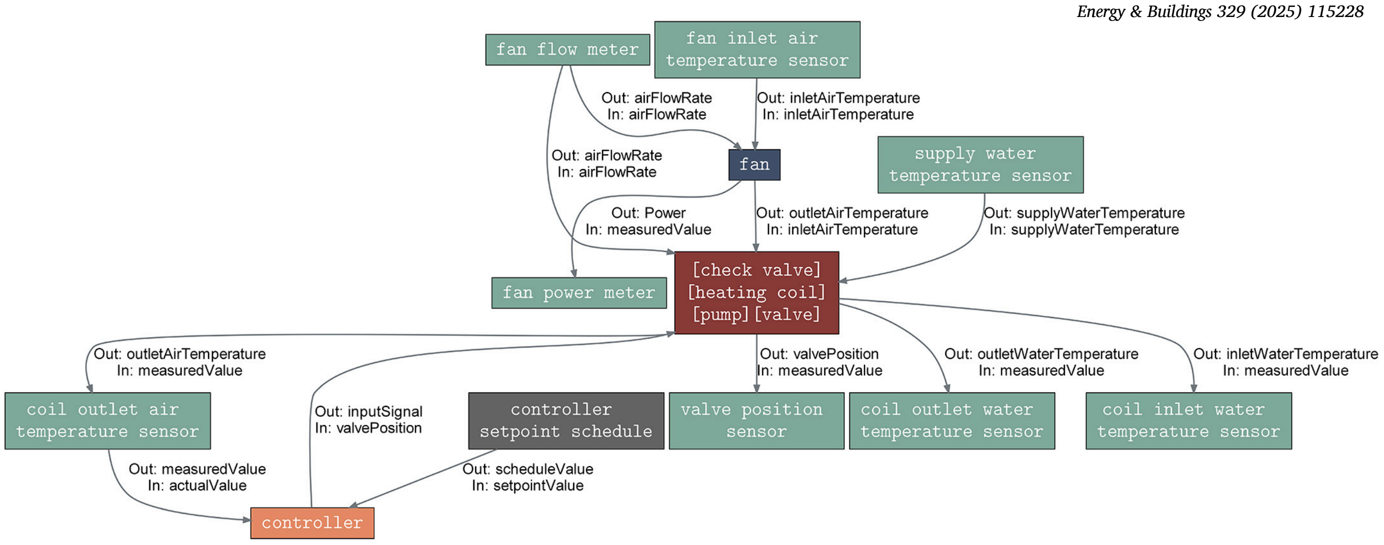

Fig. 14. Simulation model of the case study system generated from a semantic model and signatures using the methodology described in Section 3. The boxes represent the component models while the labeled edges represent the inputs and outputs of models.

6. Results and discussion

In this section, we present and discuss the case study implementation results, including the output of the model generation stage, and the parameter estimation process.

6.1. Model generation

Using the proposed model generation method with the semantic model of the case study system and the signature patterns defined throughout Section 5, a simulation model is generated as shown in Fig. 14. The three signature patterns defined for the fan, controller, and heating coil water loop models have all been identified in the semantic model with the models and connections instantiated and established accordingly. The instantiated heating coil water loop model represents four instances from the semantic model: `<heating coil>`, `<check valve>`, `<valve>`, and `<pump>` as dictated by Table 2. The remaining models represent the instances from the semantic model in a one-to-one fashion. Note that Fig. 14 contains a feedback loop caused by controller model, meaning that both estimation and inference is performed in a closed-loop configuration.

The `<fan flow meter>`, `<controller setpoint schedule>`, `<fan inlet air temperature sensor>`, and `<supply water temperature sensor>` constitute the system boundary and provide actual measured data as input, while the remaining 5 measuring devices are identified as the target set S_Y considered for estimation. As part of the translation process, 3 parameter values are automatically mapped from the semantic model to the simulation model, i.e. $S_{\theta,\text{mapped}} = \{\overline{\dot{m}}_{f,a}, \overline{\dot{P}}_f, K_v\}$. Hence, with a total set S_θ of 23 parameters, the target set of parameters $S_{\theta_\mathcal{M}}$ contains 20 of the listed parameters from Tables 2–4.

6.2. MCMC parameter sampling

For parameter estimation, sensor and meter data for a one-day winter period is used to sample from the posterior distribution using the algorithm described in Section 4.3. The resulting traceplots are shown in Fig. 15, where one sample represents the vector of parameters $\boldsymbol{\theta}$. Only the model parameters $\boldsymbol{\theta}_\mathcal{M}$ are shown for readability, although similar trace plots can be shown for the Gaussian process hyper-parameters $\boldsymbol{\theta}_{\mathcal{GP}}$. The first axis is the sampling step, while the second axis is the sampled values of the parameters. As explained in Section 4.4, the IAT-based criterion proposed by Foreman-Mackey et al. [44] is used to determine convergence. In this work, this criterion is considered met when the number of steps is 20 times IAT. In Fig. 15, this is indicated by the intersection of the IAT in red and the black dashed line, occurring around step 2000. The samples after this intersection represent the posterior distribution of interest while samples before are discarded as burn-in. In qualitative terms, convergence is identified by the transition to a scattered distribution with no systematic trends. After convergence, the chain is run for an additional 4000 steps to produce a population of samples from the posterior distribution.

6.3. Posterior distribution

As a general tool to visualize and summarize the sampled posterior, a corner plot is often used. Fig. 16 shows the corner plot of the resulting samples from the posterior distribution. The pairwise correlations are shown between the parameters as a 20x20 matrix with the marginalized distributions of the parameters along the diagonal represented by a histogram. Again, only the model parameters $\boldsymbol{\theta}_\mathcal{M}$ are shown for readability, although a full 49x49 matrix plot can be generated, which includes the Gaussian process hyperparameters $\boldsymbol{\theta}_{\mathcal{GP}}$. The quantile statistics are shown at the top of each histogram in the format $\theta = Q_{50\%}{}^{+(Q_{84\%}-Q_{50\%})}_{-(Q_{50\%}-Q_{16\%})}$, corresponding to $\pm 1\sigma$ for a normal distribution. For instance, the nominal water massflow of the heating coil $\overline{\dot{m}}_{c,w}$ has a median value of 0.72 kg/s across all samples with an estimated upper bound of 0.76 kg/s and lower bound 0.69 kg/s. The corner plot can also be used as a diagnostic tool to identify unidentifiable or correlated parameters. Parameter identifiability refers to whether unique parameter values can be determined from the available measurements. Parameters are considered unidentifiable when different combinations can produce the same model output. For instance, the power coefficient pairs c_1, c_2 and c_2, c_3 and c_3, c_4, and c_1, c_4 are inversely correlated. This implies that lowering e.g. c_1 can be compensated by increasing c_2. The opposite is true for the pairs c_1, c_3 and c_2, c_4 which are positively correlated.

6.4. Inference and posterior predictive distribution

To measure the performance of the calibrated model and the validity of the sampled posterior distribution, the posterior predictive distribution is calculated across a week of test data. As discussed earlier, one of the advantages of Bayesian parameter estimation is the inherent capability to provide uncertainty estimates or prediction intervals (PI) on predictions. However, to ensure that these estimates are reliable, the average coverage error (ACE) is used. The ACE metric measures the re-

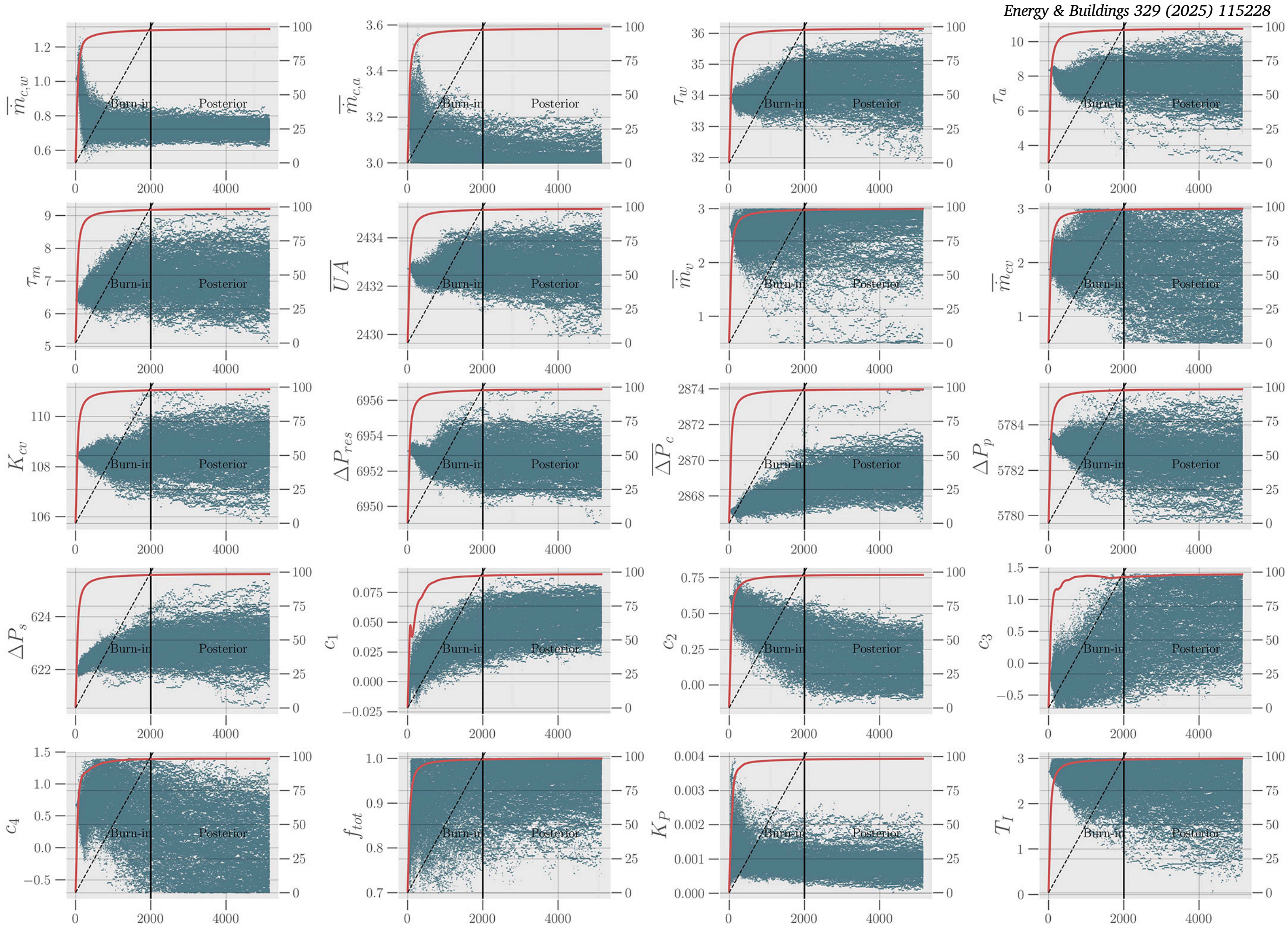

Fig. 15. Trace plots of the sampled parameters $\theta_{\mathcal{M}}$. The red line represents the IAT while the black dashed line represent the convergence criteria.

Table 5
Performance metrics of the calibrated model including assessment of the PIs 50%-99% for a week of test data.

Metric	u_v	$T_{c,w,in}$	$T_{c,w,out}$	$T_{c,a,out}$	$\dot{P}_f$
MAE($Q_{50\%}$)	0.06	0.54 °C	0.37 °C	0.19 °C	46 W
RMSE($Q_{50\%}$)	0.07	0.72 °C	0.49 °C	0.29 °C	62 W
ACE$_{50\%}$	-4%	6%	-3%	-5%	6%
ACE$_{55\%}$	-3%	7%	-3%	-6%	5%
ACE$_{60\%}$	-3%	7%	-3%	-8%	4%
ACE$_{65\%}$	-2%	7%	-3%	-9%	3%
ACE$_{70\%}$	-2%	8%	-2%	-10%	2%
ACE$_{75\%}$	-2%	8%	-2%	-11%	1%
ACE$_{80\%}$	-1%	7%	-1%	-11%	0%
ACE$_{85\%}$	0%	6%	1%	-11%	-1%
ACE$_{90\%}$	2%	4%	3%	-10%	-2%
ACE$_{95\%}$	2%	2%	2%	-8%	-3%
ACE$_{99\%}$	0%	0%	0%	-2%	-3%

liability of probabilistic models by calculating the difference between the expected and actual percentage of observations within a given PI and should be as close to 0 as possible [49]. In addition, the root mean square error (RMSE) and mean absolute error (MAE) of the median prediction $Q_{50\%}\left(\boldsymbol{Y}_y^p\right)$ is also calculated, which enables comparison with deterministic models. Here, the median is chosen as the point estimator for robustness compared to other options such as the mode or mean, which can be sensitive to outliers.

The results of applying these metrics on the posterior predictive distribution are shown in Table 5. As shown, the automated model generation and calibration method provides fairly accurate point predictions for all five measuring devices with an RMSE of 0.72 °C, 0.49 °C, 0.29 °C for the `<inlet water temperature sensor>`, `<outlet water temperature sensor>`, and `<outlet air temperature sensor>`, respectively. The `<valve position sensor>` and `<fan power meter>` has an RMSE of 0.07 and 62 W, respectively.

The ACE is calculated for a broad range of PIs from 50% to 99%. As shown, the 99% PI is the most reliable with absolute ACE values less than or equal to 3% across all 5 measuring devices. For the `<valve position sensor>`, `<inlet water temperature sensor>`, and `<water outlet temperature sensor>`, ACE$_{99\%}$ is 0%, meaning that 99% of all observations are contained in the PI as expected. The PIs between 75% and 85% are not as reliable with -11% ACE for the `<outlet air temperature sensor>`. This means that the PI is too narrow and contains 11% fewer observations than expected. If the model predictions are used as part of a decision-making process, commonly chosen PIs are 90%, 95% or 99%, but they should be chosen based on the specific application and the tradeoff between risk and utility of the generated interval [50].

The model predictions are shown for a 1-day period of the test data in Fig. 17. As also indicated by the before-mentioned RMSE metrics, the median of the predictive distributions (black dashed) agrees well with the observed data (red) and is able to capture the dynamics of the system. For instance, the flow temperatures of the system is highly sensitive to the valve position (u_v), which is set by the controller. At around 10:00 the setpoint of the controller is changed from 21 °C to 22 °C, causing the valve position to open slightly, resulting in the desired outlet air temperature change. The median point estimates are not able

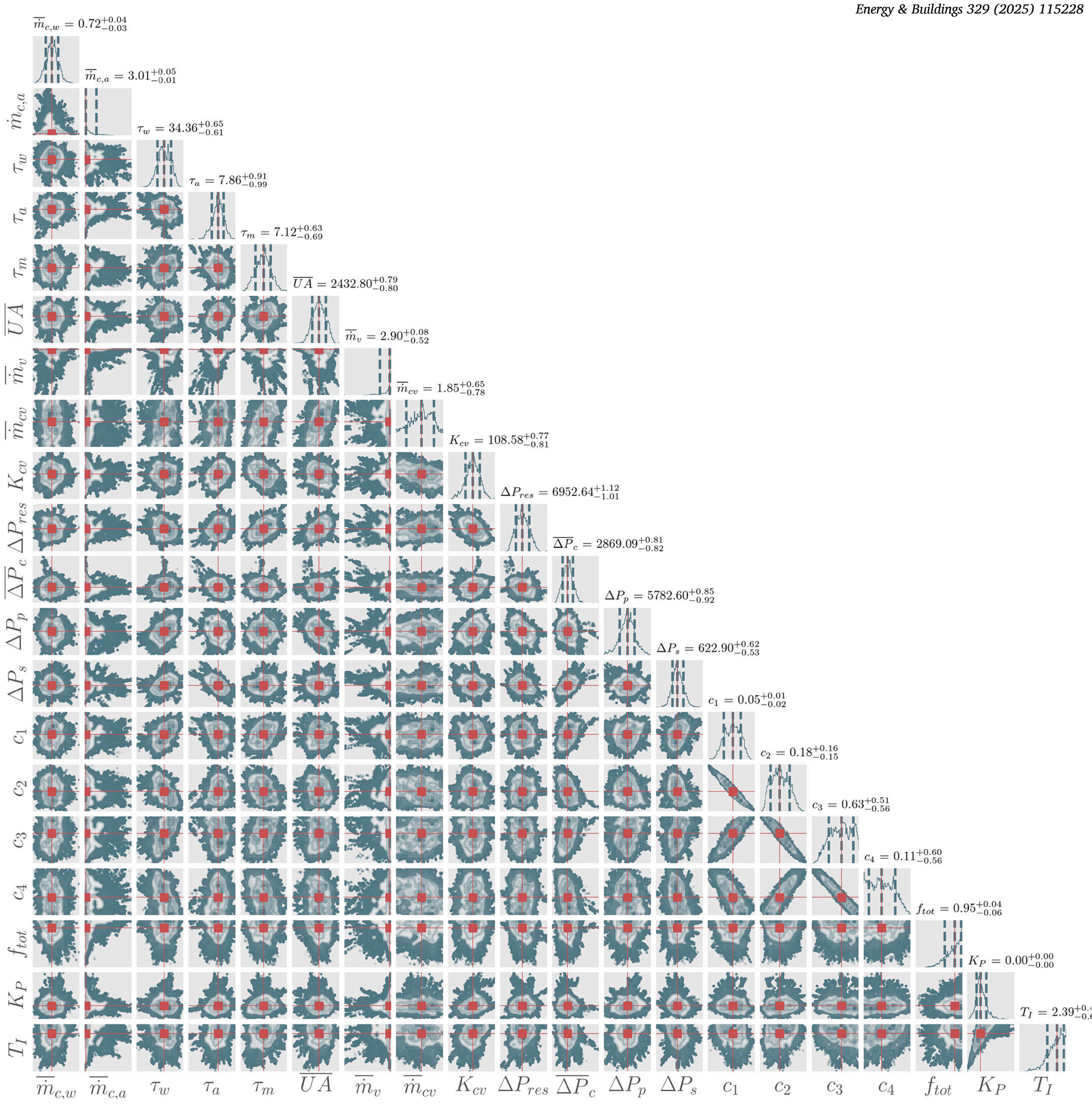

Fig. 16. Corner plot summarizing the posterior distribution of the sampled parameters. The pairwise correlation between the parameters are shown as a matrix plot while the marginalized distribution for each parameter is shown at the diagonal as a histogram. The statistics at the top of the histograms show the quantiles in the format $\theta = Q_{50\%}{}^{+(Q_{84\%}-Q_{50\%})}_{-(Q_{50\%}-Q_{16\%})}$. These values are also shown as vertical dashed lines in the histograms.

to capture all fast dynamics, especially towards the end of the period where the flow temperatures and valve position start to fluctuate in the real system. However, this uncertainty is reflected in the 99% PI, which covers these fluctuations.

6.5. Limitations and future work

While this case study uses one day of winter data to demonstrate the core methodology, many systems and practical implementations require longer calibration periods to ensure the whole operational range is covered. For real-world applications, calibration data should typically include:

1. Multiple seasons to capture different weather conditions
2. Different occupancy patterns, e.g. during weekdays, weekends, and holidays
3. Various operational modes and control sequences

The framework can handle longer time series, with computational requirements scaling approximately linearly with the length of the calibration period. However, to further investigate how the proposed method scales with both system size and length of the calibration period, larger systems and longer calibration periods should be explored in future work.

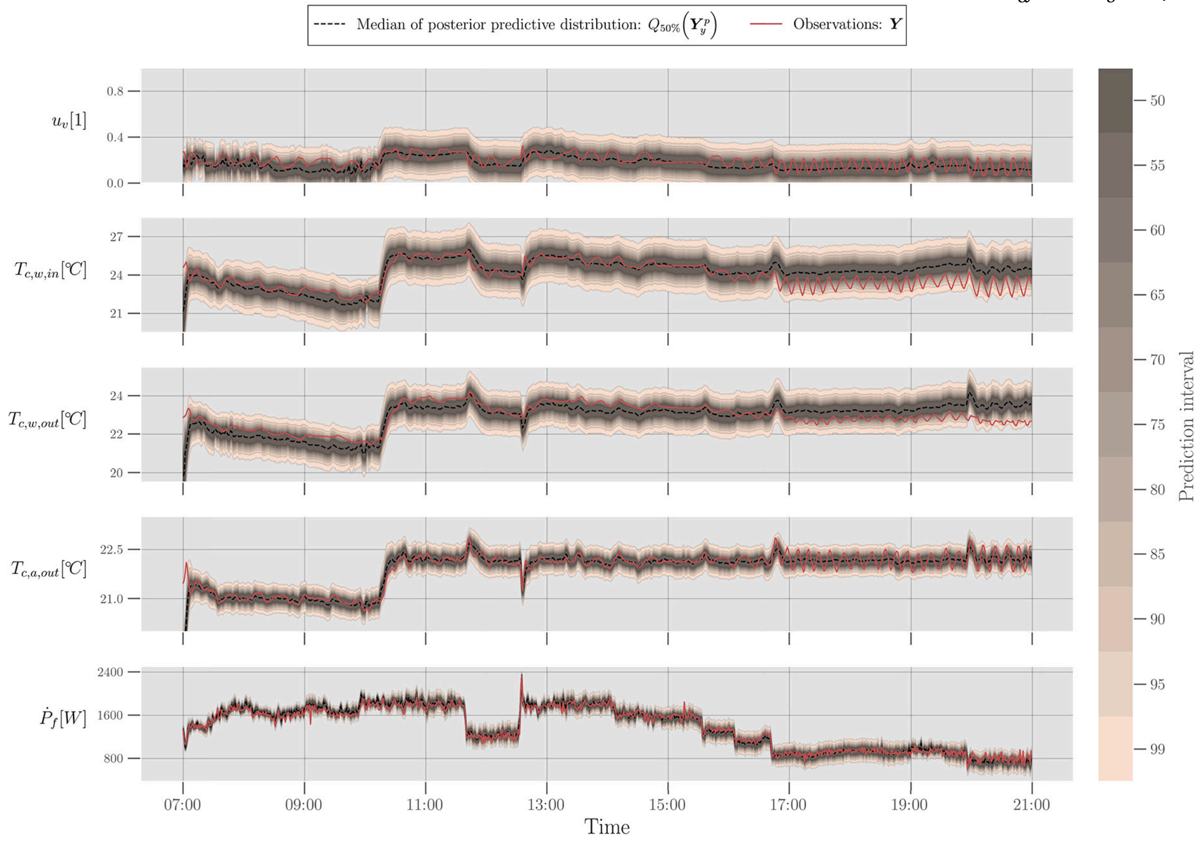

Fig. 17. The median and PIs of the posterior predictive distributions compared with observations for a 1-day period of the test data.

In general, the flexibility of the proposed method is highly dependent on the number and diversity of components present in the component model library. While this paper demonstrates the methodology using HVAC components, the framework is designed to handle any building system that can be represented in the semantic model. Future work should apply the proposed methodology to a diverse set of case studies and expand the component model library to include models for building envelope, occupancy patterns, lighting systems, and other building systems. It is expected that the required effort should decrease and the level of automation increase for each new use case as new components are added to the library. In this regard, as the component library grows, a robust method for prioritizing between multiple fitting models in the translation stage must be developed. A simple option is to allow the user to determine the desired detail of the model by setting a high-level complexity option similar to the translation method proposed by Reynders et al. [51]. Their method enables automated translation between models of different complexity levels through predefined reduction rules. Inspired by this concept, component models in the library could be tagged with complexity levels 1, 2, 3, etc. For instance, multiple heating coil models could exist in the library:

1. A simplified ideal model
2. A detailed model with temperature-dependent performance that takes thermal mass into consideration (such as the one used in the case study)

During model translation, when multiple component models match the semantic signature, the framework selects the best-fitting model based on the user-specified complexity level. This approach maintains the automated workflow while giving users control over the computational complexity and level of detail in the generated simulation model.

The proposed translation and parameter estimation methodology is independent of the ontology used for the semantic model. However, for any given ontology and semantic model used as input, a corresponding library with ontology-specific signature patterns is required. The signature patterns developed for the case study in this work are thus only viable for semantic models following the SAREF ontology and its extensions. Therefore, to accommodate other ontologies and improve interoperability, earlier work on domain ontology mapping between e.g. SAREF and Brick [52,53] could be leveraged for automated translation of either the signature patterns or the semantic model used as information source.

7. Conclusion

In conclusion, this paper addresses the pressing need for automation and interoperability in the development and calibration of BPS models. By examining the challenges faced in current practices, particularly in model development and calibration stages, the paper proposes a comprehensive approach that integrates automated model generation and calibration processes using an ontology-based method.

The proposed approach leverages semantic models to automate model generation, introducing the concept of signature patterns to describe component model applicability. Through a graph search process, these patterns are matched against semantic models to generate simulation models, thus streamlining the model development process. Based on the generated simulation model, the paper presents a general Bayesian parameter estimation problem to enable probabilistic parameter estimation and simulation based on operational data from measuring devices.

The proposed methodology is implemented and validated through a case study on a ventilation system, comprised of 8 measuring devices, a controller, a fan, and a heating coil water loop. This system is represented through a semantic model using the SAREF ontology with the SAREF4BLDG and SAREF4SYST extensions. Signature patterns are developed and dynamic component and system models from the Modelica Buildings library are exported as functional mockup units

(FMU). Using the proposed methodology, a simulation model is generated, calibrated, and assessed through different performance metrics. These metrics demonstrates the accuracy and reliability of both model point estimates and probabilistic prediction intervals across all model outputs.

In summary, this paper contributes an automated approach to the development and calibration of BPS models, addressing important challenges in the field. By integrating semantic models with data-driven simulation models, this work could drive forward the development of building digital twins and facilitate the implementation of various building services that can enhance building operation. In this regard, the probabilistic properties of the generated and calibrated model could provide a strong basis for operational decision-making.

CRediT authorship contribution statement

Jakob Bjørnskov: Writing – review & editing, Writing – original draft, Visualization, Software, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Muhyiddine Jradi:** Writing – review & editing, Supervision, Funding acquisition. **Michael Wetter:** Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Muhyiddine reports financial support was provided by Danish Energy Agency. Michael Wetter reports financial support was provided by US Department of Energy. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work is carried out under the 'Twin4Build: A holistic Digital Twin platform for decision-making support over the whole building life cycle' project, funded by the Danish Energy Agency under the Energy Technology Development and Demonstration Program (EUDP), ID number: 64021-1009.

This research was supported by the Assistant Secretary for Energy Efficiency and Renewable Energy, Office of Building Technologies of the U.S. Department of Energy under contract number DE-AC02-05CH11231.

Data availability

Data will be made available on request.

References

[1] B. Bleys, S. Jensen, A. Marszal, Annex 67 – energy flexible buildings, REHVA J. 6 (2018) 32–35.
[2] T. Hong, J. Langevin, K. Sun, Building simulation: ten challenges, Build. Simul. 11 (2018), https://doi.org/10.1007/s12273-018-0444-x.
[3] D. Crawley, L. Lawrie, F. Winkelmann, W. Buhl, Y. Huang, C. Pedersen, R. Strand, R. Liesen, D. Fisher, M. Witte, J. Glazer, Energyplus: creating a new-generation building energy simulation program, Energy Build. 33 (2021) 319–331.
[4] F. Winkelmann, B. Birdsall, W. Buhl, K. Ellington, A. Erdem, J. Hirsch, S. Gates, Doe-2 supplement: Version 2.1e, 1993.
[5] S. Klein, W. Beckman, Trnsys 16: a transient system simulation program: mathematical reference, TRNSYS 5 (2007) 389–396.
[6] V. Bazjanac, T. Maile, J.O. Donnell, C. Rose, N. Mrazović, Data Environments and Processing in Semi-Automated Simulation with Energyplus, 2011.
[7] H. Kim, Z. Shen, I. Kim, K. Kim, A. Stumpf, J. Yu, Bim ifc information mapping to building energy analysis (bea) model with manually extended material information, Autom. Constr. 68 (2016) 183–193, https://doi.org/10.1016/j.autcon.2016.04.002, https://www.sciencedirect.com/science/article/pii/S0926580516300656.
[8] M. Wetter, C. van Treeck, IEA EBC annex 60: new generation computing tools for building and community energy systems, https://www.iea-annex60.org/pubs.html, 2017.
[9] M. Wetter, C. van Treeck, L. Helsen, A. Maccarini, D. Saelens, D. Robinson, G. Schweiger, IBPSA project 1: BIM/GIS and modelica framework for building and community energy system design and operation – ongoing developments, lessons learned and challenges, IOP Conf. Ser. Earth Environ. Sci. 323 (2019) 012114, https://doi.org/10.1088/1755-1315/323/1/012114.
[10] J. Cao, T. Maile, J.O. Donnell, R. Wimmer, C. Treeck, Model Transformation from Simmodel to Modelica for Building Energy Performance Simulation, 2014.
[11] J. Cao, R. Wimmer, M. Thorade, T. Maile, J.O. Donnell, J. Rädler, J. Frisch, C. Treeck, A flexible model transformation to link bim with different modelica libraries for building energy performance simulation, https://doi.org/10.26868/25222708.2015.2446, 2015.
[12] R. Wimmer, T. Maile, J.O. Donnell, J. Cao, C. Treeck, Data-Requirements Specification to Support Bim-Based Hvac Definitions in Modelica, 2014.
[13] R. Wimmer, J. Cao, P. Remmen, T. Maile, J.O. Donnell, J. Frisch, R. Streblow, D. Mueller, C. Treeck, Implementation of Advanced Bim-Based Mapping Rules for Automated Conversions to Modelica, 2015.
[14] J.O. Donnell, R. See, T. Maile, V. Bazjanac, P. Haves, Simmodel: A Domain Data Model for Whole Building Energy Simulation, 2011.
[15] A. Andriamamonjy, D. Saelens, R. Klein, An automated ifc-based workflow for building energy performance simulation with modelica, Autom. Constr. 91 (2018) 166–181, https://doi.org/10.1016/j.autcon.2018.03.019, https://www.sciencedirect.com/science/article/pii/S0926580517308282.
[16] K. Katsigarakis, G. Lilis, G. Giannakis, D. Rovas, An ifc data preparation workflow for building energy performance simulation, https://doi.org/10.35490/EC3.2019.188, 2019.
[17] IEEE Standard Glossary of Software Engineering Terminology, IEEE Std 610.12-1990, 1990, pp. 1–84.
[18] IEA Energy in Buildings and Communities, EBC Annex 91: Open BIM for Energy Efficient Buildings, Tech. Rep., IEA Energy in Buildings and Communities, 2023, https://annex91.iea-ebc.org/.
[19] C.D. Roa, p. Raftery, A.K. Prakash, T. Peffer, Field demonstration of the brick ontology to scale up the deployment of ashrae guideline 36 control sequences (6 2023), https://doi.org/10.20357/B7J88Q, https://www.osti.gov/biblio/1996791.
[20] Z. Zheng, E. Yahia, E. Farazdaghi, R. EL Meouche, F. Ababsa, P. Béguery, A data framework enabling bem and bms interoperability based on semantic web technologies and brick ontology, https://doi.org/10.26868/25222708.2023.1366, 2023.
[21] Z. Wu, J.C. Cheng, Z. Wang, H.H. Kwok, An ontology-based framework for automatic building energy modeling with thermal zoning, Energy Build. 296 (2023) 113267, https://doi.org/10.1016/j.enbuild.2023.113267, https://www.sciencedirect.com/science/article/pii/S0378778823004978.
[22] A. Reddy, Literature review on calibration of building energy simulation programs: uses, problems, procedure, uncertainty, and tools, ASHRAE Trans. 112 (2006) 226–240.
[23] G. Chaudhary, J. New, J. Sanyal, P. Im, Z. O'Neill, V. Garg, Evaluation of "autotune" calibration against manual calibration of building energy models, Appl. Energy 182 (2016) 115–134, https://doi.org/10.1016/j.apenergy.2016.08.073, https://www.sciencedirect.com/science/article/pii/S030626191631159X.
[24] J. Sanyal, J. New, Simulation and big data challenges in tuning building energy models, in: 2013 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES), IEEE, 2013, pp. 1–6.
[25] A. Garrett, J. New, T. Chandler, Evolutionary tuning of building models to monthly electrical consumption, ASHRAE Trans. 119 (2013) 89–100.
[26] A. Chong, Y. Gu, H. Jia, Calibrating building energy simulation models: a review of the basics to guide future work, Energy Build. 253 (2021) 111533, https://doi.org/10.1016/j.enbuild.2021.111533, https://www.sciencedirect.com/science/article/pii/S0378778821008173.
[27] O. Vera-Piazzini, M. Scarpa, Building energy model calibration: a review of the state of the art in approaches, methods, and tools, J. Build. Eng. (2023) 108287, https://doi.org/10.1016/j.jobe.2023.108287, https://www.sciencedirect.com/science/article/pii/S2352710223024701.
[28] A. Chong, W. Xu, S. Chao, N.-T. Ngo, Continuous-time Bayesian calibration of energy models using bim and energy data, Energy Build. 194 (2019) 177–190, https://doi.org/10.1016/j.enbuild.2019.04.017, https://www.sciencedirect.com/science/article/pii/S0378778818339094.
[29] M.C. Kennedy, A. O'Hagan, Bayesian calibration of computer models, J. R. Stat. Soc., Ser. B, Stat. Methodol. 63 (2001), https://api.semanticscholar.org/CorpusID:119562136.
[30] M. Laura Daniele, R. Garcia-Castro, M. Lefrançois, M. Poveda-Villalon, Saref: the smart applications reference ontology, [Online], Available: https://saref.etsi.org/core/.
[31] M. Poveda-Villalón, R. Garcia-Castro, Saref extension for building, [Online], Available: https://saref.etsi.org/saref4bldg/.
[32] M. Lefrançois, Saref4syst: an extension of saref for typology of systems and their inter-connections, [Online], Available: https://saref.etsi.org/saref4syst/.
[33] C. Boje, A. Guerriero, S. Kubicki, Y. Rezgui, Towards a semantic construction digital twin: directions for future research, Autom. Constr. 114 (2020) 103179, https://doi.org/10.1016/j.autcon.2020.103179, https://www.sciencedirect.com/science/article/pii/S0926580519314785.

[34] M. Grieves, Digital Twin: Manufacturing Excellence Through Virtual Factory Replication, 2015.

[35] V. Kukkonen, A. Kücükavci, M. Seidenschnur, M.H. Rasmussen, K.M. Smith, C.A. Hviid, An ontology to support flow system descriptions from design to operation of buildings, Autom. Constr. 134 (2022) 104067, https://doi.org/10.1016/j.autcon.2021.104067, https://www.sciencedirect.com/science/article/pii/S0926580521005185.

[36] Modelica Association, Functional Mock-up Interface for model exchange and co-simulation version 2.0, Tech. Rep., Modelica Association, Linköping, Jul. 2014, https://fmi-standard.org.

[37] M. Wetter, K.S. Benne, A. Gautier, T.S. Nouidui, A. Ramle, A. Roth, H. Tummescheit, S.G. Mentzer, C. Winther, Lifting the garage door on spawn, an open-source bem-controls engine, https://api.semanticscholar.org/CorpusID:221857040, 2020.

[38] J. Lee, W.-S. Han, R. Kasperovics, J.-H. Lee, An in-depth comparison of subgraph isomorphism algorithms in graph databases, Proc. VLDB Endow. 6 (2012) 133–144, https://api.semanticscholar.org/CorpusID:9694765.

[39] L. Cordella, P. Foggia, C. Sansone, M. Vento, A (sub)graph isomorphism algorithm for matching large graphs, IEEE Trans. Pattern Anal. Mach. Intell. 26 (2004) 1367–1372, https://doi.org/10.1109/TPAMI.2004.75.

[40] J. Bjørnskov, M. Jradi, An ontology-based innovative energy modeling framework for scalable and adaptable building digital twins, Energy Build. 292 (Aug. 2023), https://doi.org/10.1016/j.enbuild.2023.113146.

[41] J.R. Goodman, J. Weare, Ensemble samplers with affine invariance, https://api.semanticscholar.org/CorpusID:54842399, 2010.

[42] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller, Equation of state calculations by fast computing machines (3 1953), https://doi.org/10.2172/4390578, https://www.osti.gov/biblio/4390578.

[43] W.K. Hastings, Monte Carlo sampling methods using Markov chains and their applications, Biometrika 57 (1) (1970) 97–109, https://doi.org/10.1093/biomet/57.1.97, https://academic.oup.com/biomet/article-pdf/57/1/97/23940249/57-1-97.pdf.

[44] D. Foreman-Mackey, D.W. Hogg, D. Lang, J. Goodman, `emcee`: the MCMC hammer, Publ. Astron. Soc. Pac. 125 (925) (2013) 306–312, https://doi.org/10.1086/670067.

[45] W.D. Vousden, W.M. Farr, I. Mandel, Dynamic temperature selection for parallel tempering in Markov chain Monte Carlo simulations, Mon. Not. R. Astron. Soc. 455 (2) (2015) 1919–1937, https://doi.org/10.1093/mnras/stv2422.

[46] C.E. Rasmussen, C.K.I. Williams, Gaussian Processes for Machine Learning., Adaptive Computation and Machine Learning, MIT Press, 2006.

[47] DMI Open Data, Meteorological Observation Data.

[48] U.S. Department of Energy, Energyplus version 9.5.0 documentation, energyplus engineering reference, Tech. Rep., U.S. Department of Energy, 2020, https://energyplus.net/sites/all/modules/custom/nrel_custom/pdfs/pdfs_v9.5.0/EngineeringReference.pdf.

[49] P. Pinson, G. Kariniotakis, H. Nielsen, T. Nielsen, H. Madsen, Properties of Quantile and Interval Forecasts of Wind Generation and Their Evaluation, vol. 2, 2006.

[50] J. Landon, N.D. Singpurwalla, Choosing a coverage probability for prediction intervals, Am. Stat. 62 (2) (2008) 120–124, https://doi.org/10.1198/000313008X304062.

[51] G. Reynders, A. Andriamamonjy, R. Klein, D. Saelens, Towards an ifc-modelica tool facilitating model complexity selection for building energy simulation, https://doi.org/10.26868/25222708.2017.621, 2017.

[52] G.F. Schneider, Towards aligning domain ontologies with the building topology ontology conference, https://api.semanticscholar.org/CorpusID:29323176, 2017.

[53] G. Schneider, Automated Ontology Matching in the Architecture, Engineering and Construction Domain - a Case Study, 2019.

[54] J. Bjørnskov, twin4build: A python package for Data-driven and Ontology-based modeling and simulation of buildings, [Online], Available: https://github.com/JBjoernskov/Twin4Build.