
Bjørnskov, Jakob; Jradi, Muhyiddine

Published in:
Energy and Buildings

DOI:
10.1016/j.enbuild.2023.113146

Publication date:
2023

Document version:
Final published version

Document license:
CC BY

Citation for published version (APA):

Go to publication entry in University of Southern Denmark's Research Portal
An ontology-based innovative energy modeling framework for scalable and adaptable building digital twins

Jakob Bjørnskov *, Muhyiddine Jradi

SDU Center for Energy Informatics, Marrisk Mc-Kinney Møller Institute, University of Southern Denmark, Campusvej 55, Odense M, 5230, Denmark

ARTICLE INFO

Keywords
Digital twin
Data-driven
Building energy model
Building simulation
Ontology
SAREF

ABSTRACT

Digitalization of buildings and the use of IoT sensing and metering devices are steadily increasing, offering new opportunities for more autonomous, efficient, and flexible buildings. As part of this transformation and inspired by the added value demonstrated in other domains, the concept of a building digital twin that can monitor, simulate, manage, and optimize building operation has received increased interest. To aid such digital twin implementations, accurate and adaptable simulation models are required, which can effectively integrate and utilize the available data. However, traditional building modeling and digital practices, such as Building Information Modeling and white-box modeling tools, are not easily compatible with these requirements. This work presents an innovative and flexible energy modeling framework based on the SAREF ontology. With a basis in the SAREF4BLDG extension for buildings and the defined classes, different models are presented for a selection of typical systems and devices such as spaces, space heaters, dampers, coils, etc. Using the generic semantics and relations of the SAREF4Syst extension, a method for linking and simulating component models is then presented. A proof-of-concept of the modeling framework is provided, showing its application and feasibility to provide a dynamic simulation of the different systems and devices included in a demonstration case. Finally, a future line of work is identified considering the implementation of the modeling framework in an actual building case study, including integration with actual sensing equipment to demonstrate different digital twin services such as performance monitoring, strategy planning, and operational optimization.

1. Introduction

Driven by ambitious environmental goals and the promising potential for improved efficiency and automation by employing the Internet of Things (IoT) and Artificial Intelligence (AI) technologies, the building sector is currently undergoing fast-paced digitalization with increasing adoption of large sensing and metering networks. While the utilization of such data offers many opportunities, it also presents great challenges to status quo data management, building operation, and building energy modeling. As a consequence, the demand is steadily growing for more flexible and scalable data and energy models that integrate seamlessly.

As part of the digitalization of buildings, Building Information Models (BIM) have for the past two decades played a major role, unlocking significant cost savings and improving efficiency by easing communication and information exchange between contractors in the design and construction phases of buildings [1]. However, despite its demonstrated success, the BIM technology is now facing significant challenges as attempts are now made to extend it beyond its intended scope to fulfill the demands of IoT-integrated smart buildings. Boje et al. [2], argued that BIM is not suitable for IoT integration due to the static legacy formats and standards used to represent data. Instead, they highlighted the emerging concept of Digital Twins (DT) in buildings as a promising solution for solving these interoperability issues.

The DT concept was first used by the National Aeronautics and Space Administration (NASA), which defined it as a “comprehensive multi-physical, multi-scale, probabilistic simulation system for vehicles or systems” [3]. Following, the DT concept has been adapted in various other scientific and engineering fields such as product manufacturing, medical sciences, and smart cities [4]. Although DTs as a concept is starting to emerge in the building sector, the concept is still in its infancy with no clear consensus on a common definition and which services it should provide.

In this work, we adopt the definition provided by Boje et al. [2] and Grieves [5]. That is, a DT refers to a concept with three main constituents; a physical system, a virtual system, and a flow of data linking

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

https://doi.org/10.1016/j.enbuild.2023.113146
Received 22 May 2023; Received in revised form 1 May 2023; Accepted 3 May 2023
Available online 9 May 2023
0378-7788/© 2023 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).
these two systems. This is conceptualized in the context of a generic building energy system as shown in Fig. 1. Here, the physical system is the actual asset to be managed, i.e. a building including all systems, devices, spaces, occupants, etc. The virtual system employs relevant simulation models to emulate the behavior of the physical system as closely as possible. The physical and virtual systems are linked through a bidirectional data flow, where the virtual system receives raw data in the form of static design and topology information as well as dynamic sensor and meter data. The virtual system then processes this information to adapt, monitor, simulate, and optimize based on the employed simulation models. This processed information is then communicated back to the physical system for facility management. As a basis for the work presented in this paper and the envisioned DT, three fundamental services, targeting the operational phase of the building life cycle, have been defined as follows:

**Service 1** Efficient collection and management of data generated by sensors, meters, and IoT devices through a user-friendly open-standard context information model

**Service 2** Automated continuous commissioning and performance monitoring in real-time to detect faults and malfunctions, ensuring a smarter and more cost-effective facility management

**Service 3** Operational strategy planning support to enable more informed decision-making by running different control and management scenarios in a zero-risk virtual environment

In recent years, successful implementations of fault detection and performance monitoring, as listed in Service 2, have been reported numerous times with different approaches. Typically, fault detection methods are divided into two categories. The first category is model-based methods, which rely on comparisons between the simulated behavior of a system with actual measurements collected from the physical unit. The second category is model-free methods, which typically utilize only operational data collected from a system to find patterns that distinguish faulty from normal operation. Although model-free methods require no models to implement, their effectiveness is also often very limited in comparison to model-based methods when applied to large-scale distributed systems such as Heating Ventilation Air-Conditioning (HVAC) systems [6]. Therefore, most efforts in fault detection and continuous commissioning of buildings have considered model-based methods, often with the use of white-box simulation tools [7–10]. In addition, an accurate dynamic energy model enables simulation and optimization of future operational scenarios, as required by Service 3, aiding decision-makers with operating the building on a day-to-day basis, taking into account weather forecasts, expected occupancy, and price signals. This potentially increases indoor comfort, reduces operational costs, and improves the overall energy flexibility of the building.

The added value of the outlined services is well-established and has been demonstrated numerous times in case-studies [11–13]. However, using traditional energy modeling tools and being challenged by various interoperability issues, such implementations usually result in highly specialized solutions tailored to each specific building. This building-by-building approach, relies on manual workflows, is costly, hard to scale, and acts as a major barrier to broadly implementing such systems.

Therefore, inspired by the added value provided in different domains, there is a need to promote and develop DT technology for the building sector to harness the benefits provided by advanced sensing and metering devices and provide the outlined services at scale. As a key milestone in such promotion, an automated and adaptable framework for energy modeling of buildings is needed to effectively integrate such models into a building DT.

In this work, an innovative framework is proposed and presented for automated and adaptable energy model development to provide the simulation models required by building DTs. The framework builds upon the Smart Applications REFerence (SAREF) ontology [14] to ensure interoperability by using existing classes, concepts, and relations. A selection of SAREF classes is extended with adaptable data-driven gray-box and black-box models, to describe their dynamic behavior. Furthermore, a generic framework based on the SAREF4SYST extension is presented to describe how these component models interact based on simple topology information. To provide a proof-of-concept, a demonstration case is considered to illustrate the framework implementation considering the simple single-zone system shown in Fig. 1.
The presented framework in this work is capable to serve as a basis for the development and implementation of scalable DTs for building applications, allowing the delivery of major services to enhance the energy efficiency and intelligence quotient of buildings. This includes performance monitoring, data management, performance optimization, and strategy planning.

As established, one of the core elements of DTs is accurate simulation models. Therefore, in the following section, an overview and discussion of different approaches for building energy modeling are provided.

2. Building energy modeling

Over the years, considerable effort has been invested in developing large-scale energy simulation tools, e.g. EnergyPlus, DOE-2, and TRNSYS [15–17]. Models developed in such tools fall under the category of white-box models, which are based entirely on first-principles building physics, typically in the form of mass and energy balance equations, which often lead to large systems of Ordinary Differential Equations (ODE). However, being the case for the majority of large-scale white-box models, they require extensive information on the building and a substantial amount of resources and time to develop. Taking EnergyPlus as an example, a typical workflow consists of three time-consuming phases before an accurate energy model is obtained. First, the building geometry is defined with detailed geometry data from floor plans, cross-sections, 3D models, etc. Second, HVAC systems design data must be specified along with material and thermal properties of the envelope and occupancy, lighting, and equipment schedules of the spaces. Finally, to achieve acceptable prediction accuracy, the model must be calibrated using actual consumption data from the building, a process that often relies on a manual trial-and-error approach that requires domain-specific expertise from the modeler [18]. In addition to the high amount of resources and time that must be dedicated to the manufacturing of these high-resolution white-box models, high computational costs and execution time is also a well-known challenge, which makes them unsuitable for operational optimization purposes [19,20].

Therefore, to reduce the amount of manual work of white-box model development, considerable efforts have been put into the automatic translation of the design and geometrical data available in BIM to attain working Energy models for simulation tools (BIM2BEPs) such as EnergyPlus and Modelica. For instance, Ramaji et al. [21] proposed a BIM2BEP transformation tool for conversion from the BIM format International Foundation Classes (IFC) into an OpenStudio model. Andrianamonjy et al. [22] proposed an automated BIM2BEPs workflow between IFC and Modelica. The workflow applies an intermediate representation called Model View Definition (MVD), on which different checks are performed to ensure a given IFC file includes all required information. If the IFC passes this test, it can be directly translated into a working Modelica model. While the workflow is successfully applied to a test case study, the authors also highlighted potential barriers of adoption. Namely, that it relies greatly on information that is typically not available in status quo IFC files.

Despite the significant efforts in BIM2BEPs research, many of the existing interoperability issues such as interdependence on extensive toolchains, inconsistent current practices by BIM practitioners, and loss of information still persist [23,24].

As a result of these shortcomings, data-driven modeling methods have recently gained increasing popularity for building modeling and simulation, due to their lower computational demands and a high potential for automation and integration in smart buildings where sensor networks and metered HVAC components collect large amounts of operational data.

Data-driven modeling methods are typically divided into black-box and gray-box methods. Black-box models are purely data-driven, meaning that no domain-specific model structure is assumed. Given a set of data, specific algorithms are thus applied to find appropriate functional relationships that map system inputs to system outputs. Commonly used black-box models are Artificial Neural Networks (ANN) [25,26], Support Vector Machines (SVM) [27], Decision Trees [28], and state-space models [29]. Due to the absence of domain-specific assumptions, these types of models have been successfully applied across many academic disciplines as well as in industry [30,31].

Typically, black-box models are considered lighter than white-box models in terms of computational complexity and simulation speed. For existing buildings with available historical data, black-box models have been used for aggregated heating and electricity load forecasting [32–34]. These types of models are usually referred to as monolithic models, i.e. whole-building models that do not incorporate any knowledge of the modeled building, besides historical weather and operational data such as heating or electricity consumption. While this offers convenient and fast implementation, there are also certain drawbacks to consider. If the building operation or design is changed, e.g. if system setpoints are changed or if the building is retrofitted, new data have to be collected and the model has to be re-trained. Furthermore, if the data available is scarce or of low quality, the performance of black-box models might be affected. A promising solution to this issue is the use of transfer learning, a technique where knowledge is transferred from one domain to another [35]. Thereby, the knowledge contained in models that are trained on large amounts of data can be reused for training other models, given that the domains have a certain amount of overlap. This concept has been applied for both indoor temperature prediction [36] and energy prediction [37].

One of the often highlighted drawbacks of black-box models is the lack of interpretability, i.e. the model structure and the obtained parameters have no direct physical meaning and the underlying cause of model behavior is thus unknown. This is opposed to gray-box models, the second category of data-driven models. As the name suggests, these types of models are a mix of white-box and black-box models where the model structure is based on physical principles, while the unknown parameters of this model structure are estimated through parameter identification techniques. The physics-derived model structure enables interpretation and validation of the obtained parameters from parameter estimation in a physical context, e.g. as demonstrated by Macarulla et al. [38]. An example of the use of gray-box modeling in buildings is the thermal Resistor Capacity (RC) model, a thermal analog to electrical circuits used to model the thermal dynamics of buildings [39,40].

However, gray-box modeling is a general concept that can also be applied to individual building components such as heat pumps, recovery units, thermal storage tanks, cooling coils, etc., as demonstrated by Afram et al. [41].

In general, the various recent studies on data-driven modeling of building systems and components using black-box and gray-box approaches report clear benefits regarding automation potential and flexibility as well as high prediction accuracy. Considering the discussed challenges of BIM2BEPs automation, these are key properties for applications in a dynamic DT-environment where close integration between data collected from IoT networks and simulation models is fundamental for delivering the outlined DT services.

3. Ontologies and metadata schemes for improved interoperability and building energy modeling

Buildings have a high degree of heterogeneity and complexity, including a wide variety of systems and devices from different contractors and providers. This makes it difficult to develop applications and models that can be scaled and reused across different buildings and use-cases. This has caused the development of ontologies, which attempts to standardize and structure metadata and communication between devices to avoid the status quo many-to-many translation between the multitude of protocols used.

Pritoni et al. [42] provided a review of 40 different ontology and metadata schemas for building design, energy modeling, and building
operation applications. From these 40 different ontologies, five of the most popular ontologies were selected for a deeper comparison based on their applicability in three use cases; Energy auditing, automated fault detection and diagnosis, and optimal control of HVAC systems. The selected ontologies were SAREF, Semantic Sensor Network/Sensor, Observation, Sample, and Actuator (SSN/SOSA), Building Topology Ontology (BOT), Brick, and RealEstateCore (REC).

Generally, it was highlighted that all five ontologies had varying degrees of missing concepts, which impacted their utility in the three use cases, i.e. choosing the right ontology depends on its intended application. For instance, BOT and SSN/SOSA missed central concepts for describing sensed properties, actuator idioms, and units of measurement. BOT, Brick, and REC missed equipment properties to describe e.g. nominal capacities, flow rates, efficiencies, etc. All ontologies except Brick missed schedule concepts for describing control strategies. Generally, Pritoni et al. conclude that none of the five ontologies accounts for all required concepts and that the ontologies must be customized and extended for certain applications.

In this work, SAREF is chosen as a backbone for the energy modeling framework. SAREF was developed in 2015 by Daniele et al. [43] in close collaboration with industry to define a unified reference for recurring concepts and relations in the IoT domain. The core SAREF ontology revolves around the concept of devices, e.g. a lighting switch, sensor, or meter. The ontology has extensions spanning across multiple domains, e.g. SAREF4BLDG [44], SAREF4CITY [45], SAREF4SYST [46], SAREF4ENER [47], etc. Here, SAREF4BLDG is an extension dedicated to the building domain and is based on the ISO-published Industry Foundation Classes (IFC) standard [48]. The extension has 72 classes and 179 object properties that represent typical appliances and devices in buildings. Hence, this ontology allows an almost direct mapping between IFC and SAREF for most of these objects within the building domain. This is a great advantage compared with other relevant ontologies, which could potentially allow for significant reuse of existing well-structured information, contained in the frequently available IFC models for buildings. Another ontology that also has IFC mapping capabilities is IFCow [49]. However, as discussed by Rasmussen et al. [50], IFCow was designed to be backwards compatible with IFC and as a result inherits two main drawbacks; complexity and size. This makes the ontology both hard to understand and hard to use. In comparison, SAREF has a higher degree of modularity and its extension SAREF4BLDG is lighter in complexity and size. In addition, the multiple domains and broader scope of SAREF could allow for further expansion of the DT concept to consider larger systems such as districts or cities at a later stage.

As is the case with the components in IFC, the component classes specified in SAREF4BLDG are data containers with no formal or mathematical description of their dynamic behavior or interaction with other components or systems. For instance, a #4bldg:SpaceHeater object holds the property #4bldg:outputCapacity, but it has no attached model which specifies how this property should be used to simulate actual heat consumption.

Therefore, to build upon the SAREF4BLDG extension and ensure high flexibility, a modular and component-based modeling approach is proposed. Using the component definitions and class hierarchies from SAREF4BLDG, the energy model is thus split into a matching set of component models that can be automatically assembled and adapted, based on the specific use case. Each component should be able to learn from the collected data through various parameter estimation techniques and strive to improve prediction accuracy as the collected data accumulates. This approach will make it possible to continuously monitor and compare the performance of actual components in real-time with their digital counterparts in the digital twin environment, as required by Service 2. Furthermore, a highly modular and flexible model allows for the interchange of component models during the entire building lifecycle, which is necessary to maintain the Digital Twin as a close replica of the actual building, as the installed systems or building use might change over time. Therefore, the component models should be accurate enough to meet the defined services to a satisfactory level, but simple enough to allow for robust and automated model calibration.

With a basis in the components depicted in Fig. 1, a description of the components and their properties from the SAREF4BLDG ontology is given in the Appendix B.3. The superscripts cp and dp stand for property and data property, respectively. An object property points to another object, while a data property holds data in the form of standard types such as integer, string, floating-point, etc. A complete list of available components and a description of the listed properties can be found in the SAREF4BLDG documentation [44].

4. Component modeling

When modeling buildings, it is common practice to take advantage of the major difference in physical time constants that are associated with the HVAC system and the building envelope and spaces. The HVAC system is often governed by fast-moving dynamics that reach equilibrium within seconds or minutes, while the large thermal inertia of the building envelope and spaces has very slow-moving dynamics. Therefore, it can often be justified to model the HVAC systems as quasi-steady-state systems, which decreases model complexity and lowers the computational burden considerably [51, 52]. For these reasons, building simulation tools such as EnergyPlus is also heavily based on this modeling approach [53]. Therefore, with a basis in the presented components in Table B.3, potential candidate models have been found in the literature for implementation and detailed descriptions are included in Appendix A. The key selection criteria for the models have been simplicity and applicability for data-driven calibration and parameter estimation.

To provide an overview of all the identified models, the inputs, outputs, parameters, and constants have been summarized for each model in Table 1. The parameters for the models in Table 1 can either be given through design specifications or calculated through parameter estimation techniques if time-series data is available for the inputs and outputs. In the case where input-output data can be continuously provided, parameter estimation can be performed on a regular interval to attain a continuously improving performance on the different gray and black-box component models. This method is demonstrated for indoor temperature forecasting by Ruano et al. [54]. Additionally, each physical component can be monitored closely, by comparing its operation with its digital replica to provide fault detection and continuous commissioning capabilities on a component level. If neither design specifications nor operational input-output data is available, default values must be assumed.

Based on the listed inputs and outputs in Table 1, and the model descriptions in the previous sections, certain expected relationships between components arise. For instance, the space heater model in Table 1 expects the massflow m, and the room temperature T, as input, which it can acquire from the valve model and the building space model, respectively. The valve model expects the valve position u, as input, which it can acquire from the controller component etc. In the following section, these types of relations are formalized using simple but broadly applicable concepts from the SAREF4SYST extension.

5. Modeling framework

In the previous section, common building components were presented, along with generic and scalable component energy models. However, to express how the defined components interact, a domain-independent SAREF-compliant methodology for linking the inputs and outputs of component models is defined. The concept of having different self-contained component models defined by an input-output relationship and an overall framework that links these models share many similarities to the co-simulation framework defined in the Functional Mockup Interface (FMI) standard [55]. Here, models are exported from simulation tools as Functional Mockup Units (FMU) and are used as
so-called slaves. The slaves are allowed to communicate at specific time intervals called communication points, managed by a master algorithm. Between communication points, the slaves are simulated independently.

The modeling framework presented in this section shares many of the same principles as co-simulation and is thus well-aligned with the use of FMUs for modeling individual components.

5.1. Linking component models using SAREF4SYST

SAREF has a dedicated extension called SAREF4SYST, which aims at providing a generic framework for representing the topology of systems and how they interact [46]. Through SAREF4SYST, the system topology can be expressed through three generic classes, `s4syst:System`, `s4syst:Connection`, and `s4syst:ConnectionPoint` along with nine applicable properties as shown in Fig. 2.

As shown in Fig. 2, a `s4syst:System` object can be a subsystem of other `s4syst:System` objects specified through the property `s4syst:subSystemOf` or have subsystems itself specified through `s4syst:hasSubSystem`. Various forms of flows between systems, e.g. information or physical quantities can be expressed through the `s4syst:Connection` and `s4syst:ConnectionPoint` classes. In this work, the `s4syst:Connection` class is used to represent system outputs and the `s4syst:ConnectionPoint` class to represent system inputs. Hence, all inputs and outputs in Table 1 can be represented by `s4syst:ConnectionPoint` and `s4syst:Connection` objects respectively, while each component can be represented by `s4syst:System` objects. This concept is shown in Fig. 3, where three SAREF4BLDG components are shown. As recommended in the SAREF4BLDG documentation [56], each component is extended with its identified model as a specialized subclass from section 4. Furthermore, these components all inherit from the `s4syst:System` superclass, enabling them to interact and communicate through the SAREF4SYST ontology. It should be noted that the full SAREF4BLDG class hierarchy is not included for clarity.

In Fig. 3, simple information flows between the three components are shown. The `s4bldg:BuildingSpace` component points to a `s4syst:Connection` object through the `s4syst:connectThrough` property. This `s4syst:Connection` object represents the indoor temperature output `T_s` of the `s4bldg:SpaceHeater` model extension. Through the `s4syst:connectsSystemAt` property, it further points to a `s4syst:ConnectionPoint` object, which represents the input `T_i` of the `s4bldg:SpaceHeater` model extension. This relation is expressed through the `s4syst:ConnectionPoint` property of the `s4syst:ConnectionPoint` object. The same concepts are then applied to relate the `s4bldg:Valve` water massflow output `m_w` to the `s4bldg:SpaceHeater` input `m_i`. Altogether, this example shows how information is translated from model output to model input. Correspondingly, the inverse path from model input to model output is expressed through the `s4syst:connectsAt`, `s4syst:connectsSystemThrough`, and `s4syst:connectsSystemProperties` as shown in Fig. 2. This generic framework and the translation from output to input and vice versa at the `s4syst:Connection` and `s4syst:ConnectionPoint` objects provide the flexibility to exchange information between any type of models independently of model formulation and
input-output terminology. Hence, the presented framework is not specific to the models shown in Table 1, which could be modified or extended to cover other components such as s4bldg:Chiller or s4bldg:Pump, where models from external simulation tools and libraries could be considered, e.g. as FMUs.

5.2. Model execution order

The topology in Fig. 3 defines the required information exchange between components at a given timestep. This produces a scheduling problem, where it must be ensured that all required inputs are available when a model is executed during a simulation time step. For instance, in Fig. 3, both the s4bldg:BuildingSpace and the s4bldg:Valve model must be executed before the s4bldg:SpaceHeater model can be executed. Such task scheduling problems are well-known in computer science and are formally known as topological sorting [57]. Given a directed graph \( G = (V, E) \), with \( V \) being the set of nodes and \( E \) being the set of edges, topological sorting is the ordering of nodes such that for every edge \( (u, v) \in E \) the node \( u \) comes before the node \( v \). This is only possible if the graph has no cycles, i.e. if it is a Directed Acyclic Graph (DAG), in which case it is guaranteed to have at least one solution [57,58]. Hence, if the s4syst:System instances are interpreted as nodes, and the s4syst:Connection, and s4syst:ConnectionPoint pairs are interpreted as directed edges, the correct execution order can be found using topological sorting, given that the graph is acyclic.

5.3. Cycle removal procedure

The no-cycles condition of topological sorting is essential to consider when applying the presented methodology to model actual buildings as feedback control loops are an integral part of all building automation systems. When such control loops are modeled, cycles are introduced. An example of a cycle produced by a controller is shown in Fig. 4. Here, a s4bldg:BuildingSpace model predicts CO₂-concentration \( C_z \), which is used as input \( y \) in a saref:Sensor model. The output \( y_{meas} \) is then used as input in a s4bldg:Controller model. An appropriate input signal \( u \) is then computed, which is sent to a s4bldg:Damper model to calculate the resulting airflow \( m_a \). Finally, this value is sent back to the s4bldg:BuildingSpace model for recalculation of indoor CO₂-concentration \( C_z \) at the next time step, completing the cycle. To systematically deal with such cycles, an algorithm is proposed and is presented in Algorithm 1. Given the set of all s4syst:System objects \( H \) and an initially identical set \( H^* \), the algorithm iterates through the set of all s4bldg:Controller components \( U \in H^* \), identifies reachable components \( V_* \) from \( u \in U \), and removes all edges from \( v \in V_* \) to the controlled component \( u \_controllerProperty \_inPropertyOf \) if such edges exist. For instance, applying Algorithm 1 on the simple system shown in Fig. 4 the Supply Damper would be identified as a reachable component from the CO₂ controller. The Supply damper is connected with the controlled component Space, and this connection is thus removed.

5.4. Topological sorting procedure

After applying Algorithm 1, topological sorting can be applied on the modified set of components \( H^* \). This is shown in Algorithm 2 where an adaptation of Kahn’s topological sorting algorithm is presented, which matches the semantics of SAREF4SYST. The input of this algorithm is the set \( H^* \), while the output is a sequence \( L \) with all the components in \( H \) in a topologically sorted order. The algorithm recursively adds all mapped components \( f(s) \in H \) with \( |s\_connects\_At| > 0 \) (i.e. the component \( s \) has no inputs) to a sequence \( L \) while removing all edges originating from \( s \). In addition, a sequence \( P \) is created which holds integers starting from 0 that represent the execution priority of the components in \( L \). Components in \( L \) with the same priority can be executed in parallel. For instance, applying Algorithm 2 on the simple system shown in Fig. 3 produces the sequence \( L = \{ \text{Valve, Space, Space heater} \} \) and the priority sequence \( P = (0, 0, 1) \). This implies that the Valve and Space models can be executed in parallel and that they must be executed before the Space heater model.

5.5. Simulation procedure

When the execution sequence \( L \) is found using Algorithm 2, the system can be simulated by simply traversing through this sequence...
Algorithm 1 Removes cycles created by feedback control loops. DFS($V$, $s$) is a recursive depth-first search to find the set of reachable components $V$ from the component $s$.

1: Let $H$ be the set of all instances with superclass s4sys: System
2: Let $H'$ contain a copy of all the elements in $H$ such that the elements in $H'$ can be modified to contain no cycles
3: Let $U$ be the subset of all instances with the class s4bldg: Controller of $H'$
4: function DFS($V$, $s$)
5: $V' = V \cup \{s\}$
6: for all $v \in s$.connectedThrough do
7: $cp = c$.connectsSystemAt
8: $s' = cp$.connectionPointOf
9: if $s' \notin V$ then
10: $V = DFS(V', s')$
11: end if
12: end for
13: return $V$
14: end function
15: for all $u \in U$ do
16: uw = u.controlsProperty.isPropertyOf
17: $V_u = DFS(0, u)$
18: for all $s \in V_u$ do
19: for all $v \in s$.connectedThrough do
20: $cp = c$.connectsSystemAt
21: $s' = cp$.connectionPointOf
22: if $uw = s'$ then
23: $uw$.connectsAt = $uw$.connectsAt \ {cp}
24: $s$.connectedThrough = $s$.connectedThrough \ {c}
25: end if
26: end for
27: end for
28: end for

Algorithm 2 A SAREF4SYST-specific adaptation of Kahn’s topological sorting algorithm [59].

1: Let $f$.H = $H$ be a function that maps from the modified components $h' \in H'$ to the original components $h \in H$
2: Let $S$ be the subset of $H'$ with no inputs, i.e.
3: $S = \{s \in H' : \{l.connectsAt\} = \emptyset\}$
4: Let $P = ()$ be an empty sequence which will be populated with the components of $H$ in a sorted order
5: Let $p = 0$ be the priority counter
6: The notation $X^{-}(s)$ is used to append the element $s$ to the sequence $X$
7: while $|S| > 0$ do
8: $S' = \emptyset$
9: for all $s \in S$ do
10: $L = L \cup f(s)$
11: $P = P^{-}(p)$
12: for all $v \in s$.connectedThrough do
13: for all $cp \in c$.connectsSystemAt do
14: $s' = cp$.connectionPointOf
15: $s'.connectsAt = s'.connectsAt \ {cp}$
16: if $v$.connectsAt = 0 then
17: $S' = S' \cup \{v\}$
18: end if
19: end if
20: end for
21: end for
22: $P = p + 1$
23: $S = S'$
24: end while

of components, while gathering the needed inputs and executing the attached model. Executing this routine once completes one system timestep of length $\Delta t$ and can be repeated indefinitely given the required inputs, such as weather data and schedule values, are available. This is formulated in Algorithm 3, where each component model is assumed to have an input and output dictionary which are indexed by the properties inputName and outputName of the associated s4sys: ConnectionPoint and s4sys: Connection objects, respectively. These property names are simply the mathematical notation.
for the inputs and outputs shown in Table 1 translated into a representative name, e.g. "indoorTemperature" to represent \( T_\text{in} \). In addition, each component model is assumed to have a \( \text{do}\_\text{step()} \) method, implementing the governing equations defined in section 4 or executing external models or libraries such as exported FMU models.

Algorithm 3 An algorithm for simulating building devices and systems given the execution sequence \( L\).

1: Let \( \Delta t_{\text{start}} \) and \( \Delta t_{\text{end}} \) be the start and end simulation time
2: Let \( \Delta t \) be the simulation time step
3: \( t \leftarrow \Delta t_{\text{start}} \)
4: while \( t < \Delta t_{\text{end}} \) do
5: for \( i = 1 \ldots |L| \) do
6: \( s \leftarrow L_i \)
7: for all \( op \in s'.\text{connectsAt} \) do
8: \( s' \leftarrow s'.\text{connectsSystem} \)
9: \( s' \leftarrow s'.\text{connectsSystem} \)
10: \( s'.\text{input}[c'.\text{inputName}] \leftarrow s'.\text{output}[c'.\text{outputName}] \)
11: end for
12: \( s'.\text{output} \leftarrow s'.\text{do\_step}(s'.\text{input}) \)
13: end for
14: \( t \leftarrow t + \Delta t \)
15: end while

6. Demonstration case and implementation

To validate the proposed modeling framework, the single-zone system shown in Fig. 1 is considered as a demonstration case. The modeling framework is implemented in Python as a reusable library [60], following the semantic structure and guidelines provided by the SAREF ontology. However, to accurately model the system, basic topological information is required. To ensure that the model can easily be constructed and adapted without manually specifying all the logical connections between models as presented in section 5, a simple model-independent representation is instead used as a starting point. Based on this representation, the logical connections between models can then be inferred based on simple rules. For this purpose, the single-zone system has been formulated using the SAREF ontology semantics and concepts as shown in Fig. 5 only considering the actual topology of the physical system.

6.1. Available topology information

The system consists of one \( \text{s4bldg:BuildingSpace} \) instance, containing one \( \text{s4bldg:SpaceHeater} \), one \( \text{s4bldg:ShadingDevice} \), one \( \text{s4bldg:Valve} \), two \( \text{s4bldg:Damper} \), two \( \text{s4bldg:Controller} \), and two \( \text{saref:Sensor} \) instances as represented by the \( \text{s4bldg:contains} \) property. The \( \text{s4bldg:BuildingSpace} \) instance is linked with one \( \text{saref:Temperature} \) instance representing indoor temperature and one \( \text{saref:CO2} \) instance representing CO2-concentration through \( \text{saref:hasProperty} \), which is inherited through the \( \text{saref:FeatureOfInterest} \) class. The \( \text{saref:CO2} \) subclass of \( \text{saref:Property} \) has been added as an extension to the core SAREF ontology similar to the work carried out by Weerd et al. [61], as it is not included by default. The measured properties of the two \( \text{saref:Sensor} \) instances are then determined through the \( \text{saref:measuresProperty} \), pointing either to the \( \text{saref:Temperature} \) instance or the \( \text{saref:CO2} \) instance. Similarly, the controlled properties of the \( \text{s4bldg:Controller} \) instances are determined through the \( \text{saref:controlsProperty} \). To determine which devices are actuated by the controllers, each \( \text{s4bldg:Controller} \) instance is associated with a \( \text{saref:LevelControlFunction} \) instance (subclass of \( \text{saref:Function} \)) through the \( \text{saref:hasFunction} \) property, which further points to a \( \text{saref:SetLevelCommand} \) instance (subclass of \( \text{saref:Command} \)) through the \( \text{saref:hasCommand} \) property. Finally, the \( \text{saref:SetLevelCommand} \) instances each point to a \( \text{saref:MultiLevelState} \) instance (subclass of \( \text{saref:State} \)) through the \( \text{actsUpon} \) property. The specific subclasses used here indicate that the actuated \( \text{s4bldg:Damper} \) and \( \text{s4bldg:Valve} \) instances can be regulated with a level-setting, e.g. from 0-100%. Alternatively, for representing binary control of other objects such as windows or doors, the \( \text{saref:OpenCloseFunction} \) is used. For the shown system, the supply and exhaust dampers share the same state, resulting in balanced supply and exhaust airflows. However, for controlling supply and exhaust airflows separately, each damper instance should have its own \( \text{saref:MultiLevelState} \) instance each associated with two different \( \text{s4bldg:Controller} \) instances.

To describe the topology of the ventilation system components and their order of placement in the flow path, the \( \text{s4syst:connectTo} \) property is used, which is inherited from the \( \text{s4syst:System} \) class. This property is symmetric, meaning that two connected components both point to each other with the \( \text{connectsTo} \) property (although only one path is shown in Fig. 5). Hence, the \( \text{s4bldg:AirToAirHeatRecovery} \) instance is connected to both the supply and exhaust \( \text{s4bldg:Fan} \) instances while the exhaust fan is further connected to an exhaust node instance. Here, the Node class represents nodes in a flow network and keeps track of flow and temperature before one stream branches out or after multiple streams join together. In the demonstration case, there is only one space, resulting in the same flow properties before and after the Node instances. However, this addition is necessary in the case of multiple spaces and a more complex flow path topology. The exhaust Node instance is further connected to an exhaust \( \text{s4bldg:Damper} \) instance. The supply \( \text{s4bldg:Fan} \) is connected to the heating \( \text{s4bldg:Coil} \) instance, which is connected to a supply Node instance. Finally, this supply Node instance is connected to a supply \( \text{s4bldg:Damper} \) instance.

For Fig. 5, inverse properties also exist for most of the shown properties, although these are omitted for clarity. For example, \( \text{s4bldg:isContainedIn} \) and \( \text{saref:isContained} \) are inverse properties of \( \text{s4bldg:contains} \) and \( \text{saref:measures} \) properties.

6.2. Obtaining a working energy model

With the SAREF representation in Fig. 5 of the considered single-zone system, the topological context of all devices and components is defined. However, this representation cannot be directly used with the presented algorithms in section 5 to simulate the system. Therefore, inputs and outputs of each model extension, as described in section 5, are first matched using the topological relationships shown in Fig. 5 and the \( \text{s4syst:Connection} \) and \( \text{s4syst:ConnectionPoint} \) classes as demonstrated in section 5.

For example, a connection between the \( \text{s4bldg:BuildingSpace} \) instance and the \( \text{CO2} \) \( \text{saref:Sensor} \) instance is inferred through the properties \( \text{saref:measures} \) \( \text{Property} \) and \( \text{saref:isPropertyOf} \) of the \( \text{saref:Sensor} \) and \( \text{saref:CO2} \) instance, respectively. A connection between the \( \text{saref:Sensor} \) and \( \text{s4bldg:Controller} \) is established through the \( \text{saref:controls} \) \( \text{Property} \) and \( \text{saref:is-} \) \( \text{MeasuredByDevice} \) \( \text{Property} \) properties.

The resulting connections are shown in Fig. 4. Completing this process for all \( \text{s4syst:System} \) instances produces the graph shown at the top of Fig. 6. Here, each pair of \( \text{s4syst:Connection} \) and \( \text{s4syst:ConnectionPoint} \) instances linking two models are represented as one edge labeled with the shorthand "C" and "CP" representing the \( \text{outputName} \) and \( \text{inputName} \) properties, respectively. As mentioned previously, these property names correspond to the mathematical notation of input and outputs in Table 1. In addition to the already discussed Node class, two other helper classes \( \text{OutdoorEnvironment} \) and \( \text{Schedule} \) have also been introduced. \( \text{OutdoorEnvironment} \) represents relevant outdoor conditions such as outdoor temperature and solar irradiation, which affects different systems, in this
case, the s4bldg:BuildingSpace and s4bldg:AirToAirHeatRecovery instances. Schedule is a generic class that outputs predetermined schedules for a specific period, e.g. daily occupancy or setpoint profiles.

With the inputs and outputs defined for all components in the system, the three algorithms from section 5 can be applied to simulate the system. An overview of this workflow is shown in Fig. 6. As a first step, Algorithm 1 is applied to the set of components $H$, to identify and remove cycles caused by the controllers. This results in the updated set $H^*$, where four edges caused by the two controllers are removed. As shown in Fig. 6, all four removed edges point to the s4bldg:BuildingSpace instance, which holds the controlled properties. The obtained set of components $H^*$ is then used as input for Algorithm 2 from which a sequence $L$ is obtained, containing the components of $H$ in a topologically sorted order. This sequence is then used in Algorithm 3 for simulating the original system $H$. To run this simulation, parameters have been specified for each model as listed in Table B.2. The parameters are based on a classroom in an actual university building. This is the case for the temperature prediction model for the s4bldg:BuildingSpace instance, which was trained on actual data collected from the classroom as described in our previous work [62,63]. For other components such as the s4bldg:Fan and s4bldg:AirToAirHeatRecovery instances, standard values, suitable for the demonstration case, are used. It should be stressed that the simulation is not intended to validate the accuracy of the presented component models, but rather to provide a proof-of-concept and validation of the overall modeling framework, information carryover, algorithms, and workflow.

6.3. Simulation results

For simulation, a 4-day winter period using actual historical weather data from Denmark is considered. The results for some of the components are shown in Fig. 7 where the output of each model is shown with the black dashed line while the colored lines show model inputs. The weather data used for the simulation are shown in Fig. 7i as a separate plot, characterized by low solar irradiance and temperatures.

Results for the temperature and CO2 models attached to the s4bldg:BuildingSpace instance are shown in the two top plots, Fig. 7a and Fig. 7b. Fig. 7c shows the inputs and outputs of the temperature s4bldg:Controller instance. As shown, a constant 23°C setpoint $T_{\text{set}}^\text{in}$ is provided as input to the s4bldg:Controller model during the whole period, resulting in a fluctuating valve position, which closes when the indoor temperature $T_\text{in}$ exceeds $T_{\text{set}}^\text{in}$ and opens when the temperature drops below $T_{\text{set}}^\text{in}$. From Fig. 7e, the resulting heat consumption $Q_\text{H}$ is shown for the s4bldg:SpaceHeater instance following the same fluctuating pattern, based on the water massflow $m_\text{w}$.

Due to the low solar irradiation in the winter period, the shading position $\alpha$ has a limited effect on the indoor temperature and is therefore set to 0 during the entire period. However, as expected, running simulations for summer periods, it is observed that shade position has a high impact on indoor temperature, due to the higher irradiation levels.
Fig. 6. Overview of Algorithm 1, Algorithm 2, and Algorithm 3 applied to the demonstration case.

Fig. 7b shows the simulation results for the CO₂ model attached to the s4bldg:BuildingSpace instance, where an artificially generated occupancy profile N_occ is used as input. Here, m_u represents both the supply and exhaust airflow as they are equal. Fig. 7d shows the CO₂ setpoint C(set) is set constant at 600 ppm, which the controller tracks by inversely actuating the supply and exhaust damper positions u_d. As shown in Fig. 7b, the predicted CO₂-concentration C(z) is sensitive to rapid changes in occupancy and airflow, causing the tracking error to occasionally approach 100 ppm during occupancy periods.

The predicted power consumption \(W_f\) for the supply s4bldg:Fan instance is shown in Fig. 7f. As expected, this consumption closely follows the supply airflow m_u. The exhaust fan power consumption is identical due to identical models and balanced supply and exhaust airflows and is therefore not shown.

Fig. 7g shows the predicted outlet temperature at the supply side T(sup,out) of the s4bldg:AirToAirHeatExchanger instance. The inputs here are inlet temperature T_in, i.e. the outdoor temperature, the exhaust inlet temperature T(exh,in), i.e. the indoor air temperature predicted by the s4bldg:BuildingSpace model, and the airflows at the supply and exhaust side m_u.

The simulated heat consumption for the heating s4bldg:Coil instance \(Q_{heating}\) is shown in Fig. 7h. The consumption closely follows the trend of the supply airflow m_u due to a constant setpoint T(set) and a relatively invariant inlet air temperature T_in between 18-20 °C.
Fig. 7. Results of a 4-day winter period simulation for the demonstration case, employing the presented component models and modeling framework. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)
6.4. Discussion

As demonstrated from the generated results, the modeling framework enables direct dynamic simulation of the components and systems represented through SAREF semantics. Although the demonstration takes basis in a simple single-zone system, the framework is not case-specific and can readily be applied to larger systems.

The majority of the presented component models are validated through other studies and implemented in simulation tools. However, their performance and effectiveness, as part of the presented modeling framework and a dynamic DT environment, are important to establish. Implementation of the framework in a real building case study is therefore planned as part of the next steps. Here, model parameters should be specified through a combination of available design data, and parameter estimation using measured operational data. The semantic alignment between SAREF4BLDG and IFC potentially provides an advantage in the task of obtaining the required design information and properties of the modeled devices and systems.

In addition, given the fact that sensor and meter equipment is virtually represented through the `saref:Sensor` and `saref:Meter` classes, the task of retrieving and comparing actual readings from the physical system and simulated readings from the virtual system for the same device is conceptually straightforward, which facilitates the implementation of continuous commissioning and parameter estimation. Relating to the demonstration case, the virtual temperature `saref:Sensor` instance reads the simulated indoor temperature shown in Fig. 7a while the CO₂ `saref:Sensor` instance reads the simulated CO₂ concentration shown in Fig. 7b. These readings are logged as `saref:Measurement` instances and retrieved through the `saref:hasFunction` and `saref:hasSensingRange` properties. Therefore, if actual readings from the two physical sensors are represented using the same concepts, the primary communication point from the physical system to the virtual system would thus pass through the physical and virtual `saref:Sensor` and `saref:Meter` instances. Here, robust and automated live communication with the physical IoT devices and preprocessing of the data will play a crucial role in the overall DT solution for proper use in continuous commissioning and parameter estimation.

7. Conclusion and future work

As part of the global energy transition, increasing the smartness of buildings is a key milestone, improving automation, efficiency, and flexibility. As part of this transformation, increased availability, sharing, management, and use of IoT devices and sensor networks both entail new opportunities as well as challenges. Inspired by the demonstrated success and applications of the DT concept in other fields and industries undergoing a similar digital transformation as the building sector, the building DT is now emerging as a promising solution to handle and utilize these data streams for different important services such as fault detection, operational optimization, and strategy planning.

One of the fundamental constituents of a building DT which enables this is a dynamic simulation model. However, to avoid manual and time-consuming modeling workflows, and to ensure the adaptability and scalability of building DTs, close integration between the simulation models and IoT-devices at the building site is essential.

This work presents an innovative energy modeling framework that builds directly upon the semantics of the SAREF ontology with an emphasis on modular data-driven models. With a basis in the SAREF4BLDG extension, different candidate models were considered as extensions for a selection of classes. Using the existing ontology patterns from the SAREF4SYST extension, a generic method for systematically linking model inputs and outputs between components was presented. To enable direct dynamic simulation of the different components and systems, three essential algorithms were then presented. As a proof-of-concept, the framework was applied to model and simulate a generic single-zone system with common building components and systems. The presented framework is considered the foundation for future work that will consider an actual building as a case study with the integration of actual devices, real-time sensor equipment, parameter estimation, and delivery of different services. This will demonstrate the feasibility of using the DT services to enhance the efficiency and smartness of buildings.

The use of SAREF ontology as a semantic backbone for the modeling framework improves interoperability and integration with the actual physical system and devices and avoids translation to simulation software. This potentially enables broader adoption and scalability of building DTs. In addition, using the multiple domains covered by the SAREF extensions such as industry, energy, water, smart cities, etc., the DT concept could potentially be expanded to consider streets, districts, and cities.

CRediT authorship contribution statement

Jakob Bjørnskov: Conceptualization, Methodology, Software, Formal analysis, Investigation, Writing - original draft, Writing - review & editing, Visualization, Project administration.

Muhyiddine Jradi: Conceptualization, Methodology, 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:

Jakob Bjørnskov reports financial support was provided by Danish Energy Agency.

Data availability

Data will be made available on request.

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.

Appendix A

A.1. Valve

The valve model is based on the model used by Zaicz et al. [64], and is given by the equations (A.1)-(A.2).

\[ u_v = \frac{u_v}{\sqrt{\omega_v^2 (1 - N_v) + N_v}} \]

\[ u_v = \tilde{u} \cdot \dot{m}_{\text{max}} \]

In equation (A.1), the effectively applied step position \( \tilde{u} \) is calculated based on the control input \( u \in [0,1] \), and the valve authority \( N_v \in [0,1] \). Following, the massflow can then be calculated in equation (A.2) using the maximum massflow \( \dot{m}_{\text{max}} \).

A.2. Space heater

The space heater model is based on the energy balance equation (A.3).

\[ C_s \frac{dT_s}{dt} = \dot{m}_{w,\text{in}}(T_{w,\text{in}} - T_{w,\text{out}}) - U(A(T_s - T_e)) \]

equation (A.3) represents the energy balance of the space heater with the stored heat on the left-hand side and the ingoing and outgoing heat flows on the right-hand side. Here, \( T_s \) is the effective temperature of the space heater, \( T_{w,\text{in}} \) is the water inlet temperature, and \( T_{w,\text{out}} \) is the
water outlet temperature. \( C_r \) is the heat capacity of the space heater, \( m_r \) is the water massflow, \( c_{pa} \) is the specific heat capacity of water, \( UA \) is the heat transfer coefficient, and \( T_r \) is the room temperature.

Xu et al. [65] models the effective temperature \( T_{ei} \) of a space heater as one lumped element governed by equation (A.3) with a varying nonlinear \( UA \) based on flow conditions. Taherimasa et al. [66] consider the space heater as multiple finite elements that each are governed by equation (A.3) with a constant \( UA \). For simplicity, this work considers the space heater as one lumped element with a constant \( UA \) and \( T_r = T_{w,ext} \) is assumed, i.e. all energy transferred to the room is driven by the outlet water temperature. Through Backward Euler discretization [67] of equation (A.3), an explicit expression for the effective space heater temperature \( T_{ei} \) at time \( t \) can thus be obtained as shown in equation (A.4).

\[
T_{ei + 1} = \frac{\left( T_{w,ext}m_w c_{pw} + UA T_{ei}\right) \Delta t + T_{ei}\Delta t}{UA + m_w c_{pw}} + C_r \tag{A.4}
\]

Here, \( \Delta t \) is the timestep and \( T_{ei-1} \) is the effective space heater temperature from the previous timestep. Based on the obtained space heater temperature \( T_{ei} \), the heat transferred to the space heater from the heating system can be calculated in equation (A.5).

\[
Q_{he} = m_w c_{pw} (T_{air,in} - T_{w,ext}) \tag{A.5}
\]

A.3. Heating coil

The heating coil model is based on a simple heat balance equation by considering the sensible heat transfer from the coil to the air and is given by equation (A.6).

\[
Q_{he} = \begin{cases} m_w c_{pa}(T_{coil,sat} - T_{air,in}) & \text{if } T_{air,in} < T_{coil,sat} \\ 0 & \text{otherwise} \end{cases} \tag{A.6}
\]

In equation (A.6), \( m_w \) is the massflow of air, \( c_{pa} \) is the specific heat capacity of air, \( T_{coil,sat} \) is the temperature setpoint of the supply airflow, and \( T_{air,in} \) is the inlet air temperature of the airflow with respect to the heating coil. The model assumes that the heating coil is ideal and can always meet the specified supply air setpoint.

A.4. Cooling coil

Similarly to the heating coil, the cooling coil model also is based on a simple heat balance equation by considering the sensible heat transfer from the air to the coil and is given by equation (A.7).

\[
Q_{he} = \begin{cases} m_w c_{pa}(T_{air,in} - T_{coil,sat}) & \text{if } T_{air,in} > T_{coil,sat} \\ 0 & \text{otherwise} \end{cases} \tag{A.7}
\]

In equation (A.7), \( m_w \) is the massflow of air, \( c_{pa} \) is the specific heat capacity of air, \( T_{coil,sat} \) is the temperature setpoint of the supply airflow, and \( T_{air,in} \) is the inlet air temperature of the airflow with respect to the cooling coil.

A.5. Air to air recovery

The heat recovery model is based on the EnergyPlus Air System Air-To-Air Sensible and Latent Effectiveness Heat Exchanger model [53] and is given by equations (A.8)-(A.15).

\[
f_{flow} = \frac{\left( m_{a,in} + m_{a,exh} \right)}{m_{a,max}} \tag{A.8}
\]

\[
e_{75\%} = \begin{cases} \varepsilon_{75\%}^1 & \text{if } T_{a,in} < T_{a,exh,in} \\ \varepsilon_{75\%}^2 & \text{otherwise} \end{cases} \tag{A.9}
\]

\[
e_{100\%} = \begin{cases} \varepsilon_{100\%}^1 & \text{if } T_{a,in} < T_{a,exh,in} \\ \varepsilon_{100\%}^2 & \text{otherwise} \end{cases} \tag{A.10}
\]

\[
e_s = \varepsilon_{75\%} + \left( \varepsilon_{100\%} - \varepsilon_{75\%} \right) \frac{f_{flow} - 0.75}{1 - 0.75} \tag{A.11}
\]

\[
f_{flow} = \frac{m_a}{m_{a,max}} \tag{A.17}
\]

\[
f_p = c_1 + c_2 f_{flow} + c_3 f_{flow}^2 + c_4 f_{flow}^3 \tag{A.18}
\]

\[
W_{fan} = f_p W_{fan,max} \tag{A.19}
\]

In equation (A.17), the flow fraction \( f_{flow} \) is first calculated based on the current massflow \( m_a \) and the maximum massflow \( m_{a,max} \). Following, in equation (A.18), the part load ratio \( f_p \) is calculated, based on the flow fraction and the power coefficients \( c_1 \) to \( c_4 \), where \( c_1 + c_2 + c_3 + c_4 = 1 \). In equation (A.19), the part load ratio is then used to calculate the fan power consumption \( W_{fan} \) based on the power consumption at maximum airflow \( W_{fan,max} \).

A.7. Damper

The damper model describes the airflow through the damper \( a \) as a function of the damper position \( a \) and the parameters \( b \) and \( c \). It is based on the model presented by Huang [69] and is given by equation (A.20).

\[
a = a \phi^b \omega + c \tag{A.20}
\]

Constraints can be imposed on the parameters \( a, b, c \) in equation equation (A.20) such that \( m_a = 0 \) when \( a = 0 \) and \( m_a = m_{a,max} \) when \( a = 1 \). Hence, if \( a \) is given, \( b \) and \( c \) are given by equations (A.21)-(A.22) to satisfy these two constraints.
\[c = -a\]  
\[b = \ln \left( \frac{m_{\text{max}} - c}{a} \right)\]  

(A.21)  
(A.22)

A.8. Controller

Properly designed and implemented Building Management Systems (BMS) are vital for the building to adapt and react effectively to internal and external dynamic changes such as occupancy and climatic conditions [70]. The BMS is typically connected with a set of controllers that control the operation of various building systems. Controllers can come in different forms, e.g. rule-based control, conventional feedback control, predictive control, etc. However, the general aim of controllers is to modulate system inputs such that a predefined goal is achieved. For conventional controllers, the goal is typically to drive a specific system property toward the desired setpoint value. One of the most fundamental and widely used controllers is the Proportional Integral Derivative (PID) controller [71]. In discrete time, the controller behavior is described by equations (A.23)-(A.24) [72].

\[e_t = y_{\text{set}} - y_{\text{meas}}\]  

(A.23)  
\[u_t = u_{t-1} + K_p(e_t - e_{t-1}) + K_i \frac{\Delta t}{\Delta t} (e_t - 2e_{t-1} + e_{t-2})\]  

(A.24)

Where \(e_t\) is the residual between the setpoint \(y_{\text{set}}\) and the measured value \(y_{\text{meas}}\) at time \(t\). In equation (A.24), the output signal of the controller \(u_t\) is calculated based on the previous signal \(u_{t-1}\), the previous residuals \(e_{t-1}\), \(e_{t-2}\), the parameters \(K_p\), \(K_i\), \(K_d\), and the timestep \(\Delta t\). Changing the type of controller is simply a matter of adjusting the parameters \(K_p\), \(K_i\), \(K_d\), i.e., a PI controller can be modeled with \(K_d = 0\) and \(K_i = 0\), while a PI controller can be modeled with \(K_p = 0\) and \(K_d = 0\). Additionally, a reverse-acting PID controller can be modeled with \(K_p > 0\), \(K_i > 0\), \(K_d > 0\).

A.9. Building space

Indoor comfort is the driving force of essentially all energy use in buildings. Most BMS have automated control of indoor temperature to follow specified setpoint schedules. Additionally, Demand Controlled Ventilation (DCV) is a common strategy to ensure that mechanical ventilation is only operating during occupancy presence in the building with the aim of lowering power consumption and ensuring proper Indoor Air Quality (IAQ). For this purpose, CO\(_2\) concentration is predominately measured as an indicator of IAQ and is controlled to follow certain setpoint schedules, by modulating the ventilation airflow [73]. Therefore, the task of accurately modeling and predicting temperature and CO\(_2\) concentration is essential to properly represent the actual building space in the building DT. Therefore, the space component must be capable of predicting both the dynamic temperature and CO\(_2\) concentration response, depending on the operation of the HVAC system, weather conditions, and occupancy use. The dynamic CO\(_2\) concentration in rooms is typically modeled with the fundamental mass balance given by equation (A.25) [38,70].

\[\frac{dC_t}{dt} = C_{\text{sup}} m_{\text{sup}} - C_t m_{e,\text{exh}} + K_{\text{occ}} N_{\text{occ}}\]  

(A.25)

Where, \(m_t\) is the mass of the air contained in the room, \(C_t\) is the room CO\(_2\) concentration, and \(C_{\text{sup}}\) is the CO\(_2\) concentration of the supply airflow, which can be assumed equal to the outdoor air CO\(_2\) concentration level at 400 ppm [70,74]. \(K_{\text{occ}}\) is the rate of CO\(_2\) mass generated per occupant and \(N_{\text{occ}}\) is the number of occupants in the room. Pantazaras et al. [70], estimated \(K_{\text{occ}}\), while synthetic Energy-Plus data was used as input for air flows and occupancy. Macarulla et al. [38], extended equation (A.25) to form a stochastic differential equation. Different configurations were tested for the estimated parameters. However, for the best-performing model, the actual occupancy count was used as input, while \(K_{\text{occ}}\) and the air flows were estimated as constants.

However, similarly to the transformation from equation (A.3) to equation (A.4), an explicit expression can be obtained for the room CO\(_2\)-concentration with Backward Euler discretization of equation (A.25). The obtained expression is given by equation (A.26), where the timestep \(\Delta t\) has been introduced.

\[C_{t+1} = m_t C_{t+1} + C_{\text{sup}} m_{\text{sup}} \Delta t + K_{\text{occ}} N_{\text{occ}} \Delta t\]  

(A.26)

Indoor temperature forecasting has been studied extensively using different modeling methods covering both white-box, gray-box, and black-box approaches. However, recently, black-box approaches in the form of ANNs have become increasingly popular. Specifically, to deal with the transient and dynamic nature of indoor temperature, a specific branch of ANNs called Recurrent Neural Networks (RNN) is typically used. Under this category, the Long Short-Term Memory (LSTM) architecture is often used, due to its gated structure, which rectifies certain undesirable traits of its predecessor, the vanilla RNN [75]. The adaptability and prediction accuracy of LSTM networks have been demonstrated across many disciplines [76], including for the task of indoor temperature forecasting [77,78].

We have previously developed and demonstrated the use of the LSTM architecture [62,63] for indoor temperature modeling of a large set of spaces in a case study building. Therefore, this black-box approach is also applied in this work, which also demonstrates the flexibility of the proposed framework and the ability to combine different modeling approaches in a unified manner. The overall model architecture is summarized by equations (A.27)-(A.31), employing two sequential LSTM networks A and B.

\[N_{\text{A},t-1} = \{T_{z,t-1}, T_{\text{out},t-1}, \Phi_{z,t-1}, u_{\text{sd},t-1}, u_{\text{jd},t-1}, u_{\text{jd},t-1}\}\]  

(A.27)  
\[(c_{z,t}, h_{z,t}) = \text{LSTM}_A (N_{\text{A},t-1}, c_{z,t-1}, h_{z,t-1})\]  

(A.28)  
\[N_{\text{B},t-1} = h_{z,t}\]  

(A.29)  
\[(c_{z,t}, h_{z,t}) = \text{LSTM}_B (N_{\text{B},t-1}, c_{z,t-1}, h_{z,t-1})\]  

(A.30)  
\[\Delta T_{z,t} = h_{z,t}\]  

(A.31)

Equation (A.27) defines the input \(X_t\) for LSTM_A. In this case, the inputs include indoor temperature \(T_z\), outdoor temperature \(T_{\text{out}}\), shortwave irradiation \(\Phi\), space heater valve position \(u_{\text{sd}}\), damper opening position \(u_{\text{jd}}\), and shades opening position \(u_{\text{jd}}\). These inputs were chosen considering the thermal heat balance of the modeled spaces, and commonly available data in buildings. However, the inputs should be modified based on available sensors and equipment installed for the modeled space.

In equation (A.28), the input \(X_{\text{B},t-1}\) as well as the state vectors \(c_{z,t-1}\) and \(h_{z,t-1}\) are provided as input for LSTM_B, which computes the updated state vectors \(c_{z,t}\) and \(h_{z,t}\), equation (A.29) then defines the input \(X_{\text{B},t-1}\) for LSTM_B as the updated hidden state \(h_{z,t}\). Using this input \(X_{\text{B},t-1}\) as well as the state vectors \(c_{z,t-1}\) and \(h_{z,t-1}\), equation (A.30) then computes the updated state vectors \(c_{z,t}\) and \(h_{z,t}\), where the hidden state \(h_{z,t}\) is treated as prediction target \(\Delta T_{z,t}\) as shown in equation (A.31). Here, \(\Delta T_{z,t}\) is the indoor air temperature difference between the previous and current timestep. After training, the model is used in inference for indoor temperature forecasting through a closed-loop configuration, given by equation (A.32). Here, the indoor temperature of the next timestep is obtained by simply adding the predicted temperature change \(\Delta T_{z,t}\) with the known indoor temperature of the previous timestep \(T_{z,t-1}\). The newly obtained temperature \(T_{z,t}\) can then be fed back as input in equation (A.27), closing the loop.

\[T_{z,t} = T_{z,t-1} + \Delta T_{z,t}\]  

(A.32)
### Appendix B

#### Table B.2
Overview of parameter and constant values used for the demonstration case.

<table>
<thead>
<tr>
<th>Component</th>
<th>Parameters</th>
<th>Constants</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valve</td>
<td>$N_r = 0.8$</td>
<td>$m_{a,\text{max}} = 0.21$ kg/s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$c_{\text{p,\text{air}}} = 4180$ J/kg/K</td>
</tr>
<tr>
<td></td>
<td>$\dot{m}_{\text{w,\text{max}}} = 0.21$ kg/s</td>
<td></td>
</tr>
<tr>
<td>Space heater</td>
<td>$U_A = 162$ W/K</td>
<td>$c_{\text{p,\text{air}}} = 4180$ J/kg/K</td>
</tr>
<tr>
<td></td>
<td>$C_r = 125000$ J/K</td>
<td>$\Delta t = 600$ s</td>
</tr>
<tr>
<td></td>
<td>$c_{\text{p,\text{air}}} = 4180$ J/kg/K</td>
<td></td>
</tr>
<tr>
<td>Heating coil</td>
<td>$c_{\text{p,\text{air}}} = 4180$ J/kg/K</td>
<td></td>
</tr>
<tr>
<td>Air to air recovery [1]</td>
<td>$c_{\text{p,\text{air}}} = 4180$ J/kg/K</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$m_{a,\text{max}} = 1.7$ kg/s</td>
<td>$\epsilon_{75%} = 0.79$, $\eta = 0.79$</td>
</tr>
<tr>
<td>Supply Fan[2]</td>
<td>$c_1 = 0.027828$</td>
<td>$\rho_a = 1.225$ kg/m$^3$</td>
</tr>
<tr>
<td></td>
<td>$c_2 = 0.026583$</td>
<td>$\epsilon_{100%} = 0.79$, $\delta = 0.79$</td>
</tr>
<tr>
<td></td>
<td>$c_3 = -0.087069$</td>
<td>$\epsilon_{100%} = 0.79$, $\delta = 0.79$</td>
</tr>
<tr>
<td></td>
<td>$m_{a,\text{max}} = 1.7$ kg/s</td>
<td>$\eta = 0.79$</td>
</tr>
<tr>
<td></td>
<td>$C_{\text{f,ex,\text{in}}} = 1500$ W</td>
<td></td>
</tr>
<tr>
<td>Supply Damper[3]</td>
<td>$a = 5$</td>
<td>$m_{a,\text{max}} = 1.63$ kg/s</td>
</tr>
<tr>
<td>Exhaust Damper[3]</td>
<td>$a = 5$</td>
<td>$m_{a,\text{max}} = 1.63$ kg/s</td>
</tr>
<tr>
<td>Temperature controller</td>
<td>$K_p = 0.05$</td>
<td>$m_l = 571.5$ kg</td>
</tr>
<tr>
<td></td>
<td>$K_i = 0.8$</td>
<td>$\Delta t = 600$ s</td>
</tr>
<tr>
<td></td>
<td>$K_d = 0$</td>
<td>(continued on next page)</td>
</tr>
<tr>
<td>CO2 controller</td>
<td>$K_p = 0.001$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$K_i = -0.001$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$K_d = 0$</td>
<td></td>
</tr>
<tr>
<td>Space[4]</td>
<td>$K_{\text{occ}} = 8.316 \cdot 10^{-4}$ kg/person</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$m_l = 571.5$ kg</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\Delta t = 600$ s</td>
<td></td>
</tr>
</tbody>
</table>

[1] Efficiency values $\epsilon_{75\% \text{, } \eta}$, $\epsilon_{100\% \text{, } \delta}$ was selected based on the AHRI directory of certified product performance for a specific product [79].

[2] Power coefficients $c_1$-$c_4$ was selected based on the single-zone default values provided by the ANSI/ASHRAE/IES standard 90.1 [80].

[3] Equation (A.21) and equation (A.22) are used to calculate $c$ and $\delta$.

[4] CO$_2$-generation per occupant $K_{\text{occ}}$, obtained for a classroom from [81]. The LSTM model contains too many parameters to show.

#### Table B.3
A subset of class descriptions and related properties from the SAREF4BLDG ontology extension for components commonly considered in building energy modeling [44].

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valve</td>
<td>$A$ valve is used in a building services piping distribution system to control or modulate the flow of the fluid.</td>
<td>closeOffRating®, flowCoefficient®, size®, testPressure®, valveMechanism®, valveOperation®, valvePattern®, workingPressure®</td>
</tr>
<tr>
<td>Space Heater</td>
<td>Space heaters utilize a combination of radiation and/or natural convection using a heating source such as electricity, steam or hot water to heat a limited space or area. Examples of space heaters include radiators, convectors, baseboard and finned-tube heaters.</td>
<td>bodyMass®, energySource®, heatTransferDimension®, heatTransferMedium®, numberOfPanels®, numberOfSections®, outputCapacity®, placementType®, temperatureClassification®, thermalEfficiency®, thermalMassHeatCapacity® (continued on next page)</td>
</tr>
</tbody>
</table>
Table B.3 (continued)

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Coil</strong></td>
<td>A coil is a device used to provide heat transfer between non-mixing media. A common example is a cooling coil, which utilizes a finned coil in which circulating chilled water, antifreeze, or refrigerant that is used to remove heat from air moving across the surface of the coil. A coil may be used either for heating or cooling purposes by placing a series of tubes (the coil) carrying a heating or cooling fluid into an airstream. The coil may be constructed from tubes bundled in a serpentine form or from finned tubes that give a extended heat transfer surface.</td>
<td>airflowRateMax°C&lt;br&gt;airflowRateMin°C&lt;br&gt;nominalLatentCapacity°C&lt;br&gt;nominalSensibleCapacity°C&lt;br&gt;nominalUAs&lt;br&gt;operationTemperatureMax°C&lt;br&gt;operationTemperatureMin°C&lt;br&gt;placementType°C</td>
</tr>
<tr>
<td><strong>Air To Air Heat Recovery</strong></td>
<td>An air-to-air heat recovery device employs a counter-flow heat exchanger between inbound and outbound air flow. It is typically used to transfer heat from warmer air in one chamber to cooler air in the second chamber (i.e., typically used to recover heat from the conditioned air being exhausted and the outside air being supplied to a building), resulting in energy savings from reduced heating (or cooling) requirements.</td>
<td>hasDefrost°C&lt;br&gt;heatTransferTypeEnum°C&lt;br&gt;operationTemperatureMax°C&lt;br&gt;primaryAirflowRateMax°C&lt;br&gt;primaryAirflowRateMin°C&lt;br&gt;secondaryAirflowRateMax°C&lt;br&gt;secondaryAirflowRateMin°C</td>
</tr>
<tr>
<td><strong>Fan</strong></td>
<td>A fan is a device which imparts mechanical work on a gas. A typical usage of a fan is to induce airflow in a building services air distribution system.</td>
<td>capacityControlType°C&lt;br&gt;motorDriveType°C&lt;br&gt;nominalAirflowRate°C&lt;br&gt;nominalPowerRate°C&lt;br&gt;nominalRotationSpeed°C&lt;br&gt;nominalStaticPressure°C&lt;br&gt;nominalTotalPressure°C&lt;br&gt;operationTemperatureMax°C&lt;br&gt;operationTemperatureMin°C&lt;br&gt;operationalRiterial°C</td>
</tr>
<tr>
<td><strong>Damper</strong></td>
<td>A damper typically participates in an HVAC duct distribution system and is used to control or modulate the flow of air.</td>
<td>airflowRateMax°C&lt;br&gt;bladeAction°C&lt;br&gt;bladeEdge°C&lt;br&gt;bladeShape°C&lt;br&gt;bladeThickness°C&lt;br&gt;closeOffRating°C&lt;br&gt;faceArea°C&lt;br&gt;frameDepth°C&lt;br&gt;frameThickness°C&lt;br)frameType°C&lt;br&gt;leakageFullyClosed°C&lt;br&gt;nominalAirflowRate°C&lt;br&gt;numberOfBlades°C&lt;br&gt;openPressureDrop°C&lt;br&gt;operation°C&lt;br&gt;operationTemperatureMax°C&lt;br&gt;operationTemperatureMin°C&lt;br&gt;orientation°C&lt;br&gt;temperatureRating°C&lt;br&gt;workingPressureMax°C</td>
</tr>
<tr>
<td><strong>Controller</strong></td>
<td>A controller is a device that monitors inputs and controls outputs within a building automation system. A controller may be physical (having placement within a spatial structure) or logical (a software interface or aggregated within a programmable physical controller).</td>
<td>contains°C&lt;br&gt;hasSpace°C&lt;br&gt;isSpaceOf°C</td>
</tr>
<tr>
<td><strong>Building Space</strong></td>
<td>An entity used to define the physical spaces of the building. A building space contains devices or building objects.</td>
<td>contains°C&lt;br&gt;hasSpace°C&lt;br&gt;isSpaceOf°C</td>
</tr>
</tbody>
</table>

References


Energy & Buildings 292 (2023) 113146


Chartered Institution of Building Services Engineers, Cibse guide a: Environmental design, chapter 4.2, Table 4.2, 2007.


