Reasoning about Physical Processes in Buildings through Component Stereotypes

Ganesh Ramanathan
ganesh.ramanathan@siemens.com
Siemens AG
Switzerland

Simon Mayer
simon.mayer@unisg.ch
University of St. Gallen
Switzerland

ABSTRACT

Buildings employ an ensemble of technical systems like those for heating and ventilation and each of them orchestrate complex physical processes. Ontologies such as Brick, IFC, SSN/SOSA, and SAREF have been created to describe the technical systems in a machine-understandable manner. However, such ontologies focus largely on describing system *topology*, whereas several use cases, such as automated fault detection and diagnostics (AFDD), also need knowledge about the physical processes. Physical processes can be described using mathematical simulation models, but this is practically too expensive for building automation systems and their integration with mainstream technical systems ontologies is still under-explored. We propose to address these challenges by introducing the concept of *component stereotypes* that describe the effect of component actuation on the state its underlying physical mechanism. These stereotypes are then linked to actual component instances in the technical system description, thereby accomplishing an integration of structural description with knowledge about physical processes. We contribute an ontology for such stereotypes and evaluate it with respect to the coverage of HVAC components in Brick and its ability to automatically infer relationships between components in a real-world building. We show how the resulting knowledge graph can be queried by AFDD applications to know about expected consequences of an action, or conversely, identify components that may be responsible for an observed state of the process. While we are able to report a coverage of 100% of Brick HVAC components, the automatic inference underreports component dependencies in real-world installations. This points at a group of concepts which we propose should be considered in future versions of the Brick ontology.

ACM Reference Format:
Ganesh Ramanathan and Simon Mayer. 2023. Reasoning about Physical Processes in Buildings through Component Stereotypes. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (BuildSys '23).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
BuildSys '23, November, 2023, Istanbul, Turkey
© 2023 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Technical systems in a building are complex compositions of sub-systems and their components that work in tandem to carry out physical processes like heating, ventilation, and air-conditioning. Traditionally, such systems were described in human-understandable ways like texts and diagrams; however, driven by the need to make such system knowledge more directly accessible to software applications, we are seeing more and more machine-understandable specifications in recent years.

A large part of this development builds on top of active and cross-cutting research in the engineering and Semantic Web communities, where several ontologies like Brick[1] [2], IFC[2] [3], SSN/SOSA[3], SAREF[4], etc. have made it possible to describe buildings and their systems in a manner that permits the creation of interoperable software on top of this knowledge [16].

Today, system design ontologies are used in the building automation domain to describe systems as compositions of sub-systems which are topologically related. For example, using the Brick ontology, we can describe a heating system as a composition of sub-systems for heat generation, distribution, etc., each consisting of interconnected components like pumps, valves, etc. In such models, *components* (sometimes called *equipment*) are the functional units that carry out a well-defined transformation of substance or energy. Finally, the granularity of the decomposition of systems using these abstractions—systems, sub-systems, and components—varies from scenario to scenario and depends on the granularity and abstraction level at which it is desired to describe a system. For example, a fan together with its motor and drive might be considered as a single *component* when viewed from the perspective of a control program since this program is defined in terms of "controlling the speed of the fan" instead of adopting a more sub-system viewpoint (e.g., "actuating the drive which controls the speed of the motor and consequently the speed of the impeller of the fan").

In addition to the structural relationships between sub-systems and components, it is often desired to also encode knowledge about *how the system functions*. We see this need in several existing use cases like automated fault detection and diagnosis (AFDD; see [22]), plausibility checking [12], or control optimization [9]. Such knowledge further enables matching automation program strategies to system deployments, synthesizing control interlocks [15], and may support autonomous software agents to control and coordinate a building automation system [21]. For this purpose, we argue that

[1] https://brickschema.org/ontology/
[2] https://technical.buildingsmart.org/standards/ifc/ifc-formats/ifcowl/
[3] https://www.w3.org/TR/vocab-ssn/
[4] https://saref.etsi.org/core/v3.1.1/

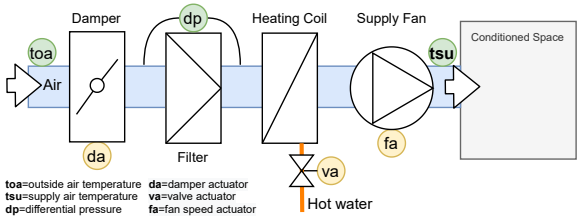

Figure 1: An example of a system: This AHU takes in fresh air and filters and heats it before supplying it to a room. The value of `tsu` in this example is determined by `toa`, `da`, `va`, and `fa`.

machine-understandable system descriptions should not only incorporate structural descriptions, but also a description of the functioning of the system's components and the effects they have on observable physical states of the system's context. For example, although we know that the supply fan of the Air Handling Unit (AHU) shown in Figure 1 is an instance of Brick class `Supply_Fan`, one has to rely on textual descriptions to understand that it functions by transferring kinetic energy from the impeller to the air resulting in increased air pressure and flow. In a system like the AHU, for a software agent to answer the question "Which components and which of their actions influence the supply air temperature?", it would require access to common knowledge (in machine-understandable form) of the physical mechanisms each component of the AHU is involved in.

Traditionally, physical mechanisms are described using semi-formal diagrams, formal mathematical equations, logical rules, state machines, or simulation models. However, creating such models for systems deployed in buildings is prohibitively expensive due to the effort involved in modeling the mechanisms and their mathematical equations. On the other hand, amongst the use cases which can benefit from the knowledge of physical mechanisms and processes, several of them, like AFDD, fault-mode-effect analysis, plausibility checking, and reasoning about control functions, do not require full simulation models, but would already benefit from more abstract knowledge in terms of dependencies between variables that represent the process mechanism. Such information would informally read like "The supply temperature is determined by the airflow rate (influenced by the fan and the damper), the energy input to the heating coil (influenced by water temperature and the flow rate determined by the valve), and the intake air temperature (which is not under system control)" – this would permit a human to for instance understand that too low supply temperature could be remedied by modulating the damper actuator, supply fan speed actuator, or the valve actuator of the heating coil. Such a rule could further make explicit (or integrate) more fundamental knowledge (e.g., that the heating coil is the component that most directly influences supply air temperature). Such knowledge can be further generalized by simplifying the description of the underlying physical mechanism: heating coils use the same physical mechanism as economizers or condensers—on this abstraction level, we may hence refer to all of these types of components as *heat exchangers*.

Towards permitting practically feasible functional description of components, we formulate the following questions:

(1) How can we describe physical mechanisms in a simplified abstraction that is widely valid for components of a kind?
(2) How can we link such (simplified) descriptions of a component's mechanism to its technical interfaces so that we can understand which interfaces are relevant in the functioning?
(3) How can we enable reasoning about the role of components that are interlinked in a process?

Our aim is hence to capture such common knowledge of physical processes in a building automation system and thereby enable the automated answering of queries about a system's overall behavior by permitting software programs (e.g., AFDD) to use formal reasoning on top of this knowledge. To do this, we propose a method to create *stereotypical* behavioral descriptions of *physical mechanisms* like *heat exchange* or *pressurization*. The stereotype would describe the process variables and their inter-relationship in a high-level abstraction instead of a complete mathematical equation. Such a fundamental description may be referred to and, hence, reused by various component kinds – for example, the mechanism of *heat exchange* is used by heating/cooling coils, evaporators, condensers, etc. in HVAC systems. Further, in order to couple a component kind to the description of the physical mechanism(s) it uses, we propose a method to create *stereotypical* descriptions of its *technical interfaces* and couple them to the variables of the physical mechanism – for example, the heating coil in the AHU has inlet and outlet ports for air and hot water. The temperatures and flow rates observed or effected at these ports can be related to the variables involved in the description of the heat exchange mechanism that the heating coil uses. We contribute the concept of such *stereotypes* and position it with respect to related work (see Section 2); we then provide the concepts and relationships required to model stereotypes of physical mechanisms and technical interfaces of system components (see Section 3). The required concepts and relationships are provided in an ontology, which we name *Elementary*[5]. *Elementary* is modeled using OWL, which is part of the Semantic Web technology stack, and widely used to create formal specification of concepts and relationships. In Section 4, we demonstrate the integration of our stereotypes with structurally oriented concepts of system components from the Brick ontology. This shows how structural knowledge can be combined with functional knowledge about the system components, and we illustrate the usage of our ontology by providing several examples of stereotypes along with a complete mapping of Brick HVAC components to proposed stereotypes. Finally, we also provide an evaluation of our approach in a real setting of an office building that consists of about 180 rooms across six floors: We identified kinds of systems in the building that represent different levels of complexities in terms of topology and processes. We then identified kinds of queries found in AFDD use cases and show that our approach enables these to be answered without resorting to the use of full-fledged physics models.

2 RELATED WORK

In most domains, systems engineering involves design of a system through decomposition of requirements to find functional abstractions, which are then organized into a hierarchy of sub-systems (see [18] for an in-depth treatment of this topic). Ontologies such as

[5]Inspired by the word often used by the popular detective character Sherlock Holmes.

Brick, IFC, SSN/SOSA, SAREF, etc. can be used to describe the subsystems, their components, and interdependencies (e.g., in [19]). Such ontologies model one or more of the following key aspects of structural descriptions in systems engineering [6, 25]:

(1) The mereological (compositional) relationships in terms of sub-systems and components. For example, an AHU *consists* of a heating coil, which in turn, has a control valve.
(2) The topological relationships (i.e., the physical process relationship between mereological entities. For example, using the Brick ontology, one can state that a `brick:Radiation_Hot_Water_System` consists of a number of instances of `brick:Radiators` which are of type `brick:Heat_Exchanger` are placed in Rooms, and these are fed by an instance of a `brick:Boiler`.
(3) The domain-specific taxonomy to convey the *functional abstraction* of a component (see [27] for an extensive survey). For example, the class `brick:Fan` intends to describe all things that function like a *stereotypical* fan.

While ontologies such as Brick capture a component's functional abstraction and integrate this with mereological and topological relationships, they do not seek to describe the *physical manifestation of the behavior of a component*, which is what we claim to be required to answer the questions posed in the introduction. We hence seek to further integrate (in a way that is compatible with Brick and other widely used ontologies) knowledge about relevant aspects of this stereotypical behavior. We acknowledge (and support) that this knowledge might be further integrated with full simulation models of a component (e.g., towards realizing verifiably safe cyber-physical systems, see [10]); however, we propose that an abstraction on the level of functional *stereotypes* permits the automated query-answering across a broad range of relevant topics (such as AFDD) while remaining simple enough to be deployed in typical building automation systems.

Physical processes are typically described as sequences of *physical mechanisms*, each of which is managed by a component in the system [4, 6]. Several ontologies (such as OntoCape [17], TSO [19], and PhysSys [7]) have been created for the purpose of describing physical processes and mechanisms. These ontologies, which are grounded in standard physics and engineering knowledge, principally define a *physical process* as a series of *physical mechanisms* involving the transformation of energy, substance, or work. According to Borst [5, 7], these transformees are referred to as physical *stuff*, and the transition of *stuff* between process mechanisms is referred to as *flow*. Examples of such *flow* of *stuff* includes flow of air in a duct, electric current flowing in a wire, heat through a conductor, or force transmitted through a mechanical linkage. Therefore, the term *streams* or *process pathways* is meant for describing and distinguishing the semantics of flow of stuff that occurs through a complex chain of components – for example, *supply*, *extract*, etc. To accomplish the automated inference of answers to questions such as those presented in the introduction, we hence require compatible ways of modeling the individual process mechanisms and of associating them with the respective technical components in a specific deployment.

The association of a technical component with a specific process mechanism is a commonality in these ontological approaches, which also stems from their origin in process and systems engineering theory. Therefore, when components are connected to one another through their *terminals* or *ports*, their mechanisms are deemed to influence one another such that *stuff* flows between them – thereby creating a physical process; however, today, in building automation, neither the technical system description nor the physical process ontologies capture this coupling.

Furthermore, our approach needs to provide a way of describing each process mechanism to convey knowledge about the transformation it performs. This is typically done using mathematical equations where the *state* of the physical mechanism is represented using *process variables* which are measurable quantities. The behavior of individual mechanisms which are described using mathematical models (for example, using the EngMath [11] ontology) can then be used by solvers to determine static and dynamic state changes in the process. The mathematical models are constructed either from first principles of the physics involved or using *system identification*, which is an approach where dynamical systems are observed and a model of their behavior is built using statistical methods. However, creating such models for systems deployed in buildings is typically too expensive in practice due to the effort involved in modeling the mechanisms and their mathematical equations [26]; this is due to models for real-life sub-systems often requiring extensive parameterization together with the computational effort required to simulate them.

On the other hand, engineering design primarily relies on considerably *simplified* representations of these equations, which are presented in the form of tables and graphical plots in datasheets and design handbooks, and these are sometimes referred to as *operational model* [28]. For example, the datasheet of a centrifugal fan contains characteristic curves depicting the fan's operation under different conditions. In addition to such information about a specific product's characteristics, an engineer has generalized heuristic knowledge about the physical principles involved – for example, *affinity laws* (see [23]) state the dependency between the speed of a fan impeller and the resulting flow rate. At an even more abstract level of understanding, from such theory of widely applicable physical principles, one can recognize the independent and dependent variables involved in a physical mechanism. Although such simplistic understanding is rarely sufficient when *designing* a system, it does provide valuable knowledge to gain a high-level understanding of a deployed system's functioning. For example, the knowledge that the speed of a fan's impeller determines the outlet flow rate, when coupled with structural knowledge that the fan feeds a heat exchanger and that a heat exchanger (stereotypically) has an inflow flow rate, serves to link the behavior of the fan to that of the heat exchanger. An approach to model such an *elementary* abstraction of the physical principles and their link to the technical components in a building automation system is today missing. While in [13] understanding of simple mechanics is captured using an ontology, this limited to understanding human interactions with mechanical tools and does not address physical processes.

We address this gap in our approach by modeling stereotypical knowledge about the physical mechanisms that a system component is involved in, and by integrating this knowledge in a way that is compatible with widely used ontologies for building automation components. We claim that this enables the system to know

enough about its functioning by allowing formal reasoning on top of this knowledge. This would, for example, enable automatically reasoning about the potential *effects of an automation program* on the physical process, or to determine faults in the system that lead to unexpected functioning of a control program. The integration of mathematical models, which would permit deeper inspection of the system's behavior (at higher cost), remains possible with our approach.

3 APPROACH: COMPONENT STEREOTYPES

In the previous section, we pointed out that physical mechanisms can be described in a generalized manner by solely identifying their independent and dependent variables and their inter-relationships. This already would offer valuable knowledge for several use cases which requires a high-level understanding of the system functions. We also highlighted that ontologies for structural description of systems provide domain-specific taxonomies to classify components according to functional abstraction along with description of their interconnections in the system topology. However, we argue that an approach to interlink structural descriptions and knowledge of physical mechanisms is missing – i.e., an approach is required that goes beyond component classification and topological description to include knowledge of physical mechanisms such that we understand *how* a *system of interconnected components* functions.

To address this, we introduce the idea of *stereotypes* for components of technical systems. To illustrate our idea of a stereotype, let us consider the fan in our AHU example (Figure 1). We can imagine a *stereotypical fan* as having an air inlet and outlet, and a impeller shaft; these form the fan's technical interfaces. We further know that a fan uses the *pressurization* mechanism whereby the speed of the impeller (effected at the shaft) determines the air flow rate (observable at the air outlet and inlet). Now, when presented with an actual instance of a fan, we may integrate this stereotypical knowledge to gain an insight into how this instance can be acted upon and how it would behave in response. We can further use this to investigate the relationships between the fan instance and other technical components in a system such as the AHU.

The idea of a *component stereotype* which we put forward encompasses stereotypical knowledge of *technical interfaces* provided by the component kind and the *physical mechanism* such a component uses. Since a physical mechanism (e.g., heat exchange) may be applicable to more than one component kind, our proposed approach allows description of physical mechanisms in a way that it can be reused by multiple component stereotypes. The resulting component stereotype is also agnostic of the exact process context. For example, the stereotype of the fan would be applicable irrespective of whether it is used for driving airflow for heat exchange or for ventilation. In other words, a stereotype is a reusable artifact for the domain and is not specifically crafted for a particular instance of a component, a system, or the role a component plays in a system. This supports practical applicability of our approach – when supplied with a topological model of component instances, the only remaining task during engineering is to link these instances to suitable stereotypes.

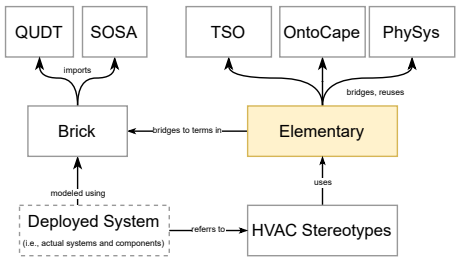

Figure 2: The *Elementary* ontology and its relation to other ontologies. Actual systems are modeled using Brick and each component instance in the system is linked to a stereotype in the domain-specific library of stereotypes for HVAC components.

3.1 Elementary: An Ontology for Stereotyping

We propose the *Elementary* (prefix: `elem:`) ontology[6] to support the creation of stereotypes and the interlinking with topology-oriented system descriptions such as in Brick. *Elementary* addresses three aspects involved in defining and using stereotypes: 1. A way to express physical mechanisms in terms of its relation to process variables, 2. a way to describe the process semantics of interfaces of the stereotypical components and link them to description of mechanisms, 3. support for automatic inference of the effect of interactions between the components on the process which they are part of.

Elementary reuses terms and relationships from several openly available ontologies. An overview of *Elementary* is shown in Figure 2. From TSO, OntoCape, and PhySys, we reuse concepts concerning physical processes, mechanisms, and flow of physical *stuff*. We bridge to Brick (which imports QUDT and SSN/SOSA) to reuse abstract concepts related to system design, and concrete instances depicting substance and quantities. Further, to evaluate our approach, we have created a library of stereotypes for HVAC component kinds (prefix: `hvac`) that are available in the Brick ontology. The actual systems that we used for our evaluation (see Section 4) are modeled using the Brick ontology and are linked to stereotypes in this library. The following snippet in RDF shows how two of the components of our AHU example that are modeled using Brick can be linked to their corresponding stereotypes:

```
urn:fan_01 a brick:Fan;
urn:fan_01 brick:feeds urn:heating_coil_01;
urn:fan_01 elem:hasStereotype hvac:centrifugal-fan.

urn:heating_coil_01 a brick:Hot_Water_Coil;
urn:heating_coil_01 elem:hasStereotype hvac:heat-exchanger.
```

In this snippet, we see two instances which are classified as `brick:Fan` and `brick:How_Water_Coil` and topologically linked using the `brick:feeds` relationship. We will show how the stereotypes `hvac:cent_rif_ugal-fan` and `hvac:heat-ex_chan_ger` are created and how they lead us to infer the semantics of the process.

[6] Available at http://w3id.org/elementary

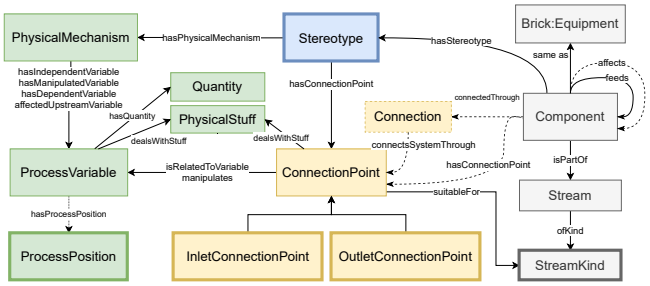

Figure 3: The principal classes of the *Elementary* ontology. Boxes with thick border are classes which we have added while others are reused from existing ontologies. Dashed lines and borders indicate relationships and instances that are inferred.

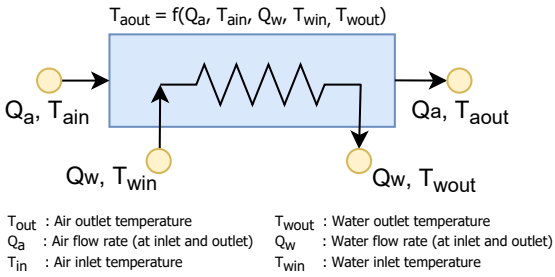

Figure 4: The stereotypical mechanism of heat exchange and its associated variables. The orange circles represent the terminals or ports of a component like heating coil which uses this mechanism.

In the following sections, we describe in detail the three aspects behind our ontology, i.e. means of modeling stereotypes of physical mechanisms, coupling them to stereotypical component interfaces, and finally means of using the stereotypes to infer process relationships of actual components.

3.2 Modeling Physical Mechanisms

Mathematical models of physical mechanisms are typically expressed as equations which refer to *process variables*. More specifically, these are *independent variables* (IV) and *dependent variables* (DV) – in the mathematical equation, IVs appear as parameters of a function whose output are one or several DVs. In other words, by changing values of IVs, the effect of the mechanism can be observed as change of the DVs. The variables are described by: 1. their *position* with respect to the physical mechanism (e.g., at the inlet or outlet), 2. the *physical stuff* whose state a variable represents, and 3. the *quantity* the variable conveys (e.g., temperature, energy content, pressure, or force) [7, 8, 20]. In the context of the AHU example, Figure 4 shows the process variables associated with the process mechanism of *heat exchange*, i.e., the stereotypical mechanism used by the AHU's heating coil. Here, T_{aout} is a DV which is influenced by Q_a, Q_w, T_{ain}, and T_{win} – i.e. the IVs. Though a number of IVs may influence a single DV, only a subset of them is meant to be *manipulated*, i.e, changed by external action – in controls engineering theory, such a variable is called a *manipulated variable* (MV). In the above example, T_win are Q_w are potential MVs.

Although most physical mechanisms that we encounter in building systems exhibit one-to-one relationships between MVs and DVs, there are several cases where a single MV influences more than one DVs (e.g., compression can cause decrease in volume and increase in temperature), or, vice-versa, where several MVs influence a single DV (e.g., in a heating coil where both flow rate and water inlet temperature influence the air temperature). On the other hand, disjointness of MVs and affected DVs in a mechanism indicates the presence of two distinct underlying mechanisms [24] and thus motivates refactoring the model. Our model needs to furthermore consider that a DV of a mechanism might also affect *upstream* mechanisms – for example, flow caused by a centrifugal pressurization mechanism in a fan results in a suction effect at the inlet port which might cause flow in upstream components.

We capture these concepts in the *Elementary* ontology (see Figure 3) beginning with the class `elem:ProcessVariable` to describe a variable using relationships `elem:dealsWithStuff`, `elem:hasProcessPosition`, and `elem:hasQuantity`. Individuals of class `elem:ProcessMechanism` are then related to one or more instances of `elem:ProcessVariable` through the relationships `elem:hasIndependentVariable` (or its sub-type `elem:hasManipulatedVariable`) and `elem:hasDependentVariable`. Finally, to capture upstream effects of physical mechanisms, our model provides a transitive relationship `elem:affectedUpstreamVariable` between a physical mechanism and a DV. Using these classes and relationships, one can define stereotypes of variables and mechanisms that are independent of the specific deployment context and, as we will describe in sub-section 3.4, they can be interlinked with variables in other process mechanism stereotypes. The following snippet of RDF/Turtle shows part of a stereotype for pressurization mechanism:

```
:pressurization rdf:type owl:NamedIndividual,
 elem:PhysicalMechanism ;
 elem:hasIndependentVariable:inlet-air-pressure;
 elem:hasDependentVariable:outlet-air-flow;
 elem:hasManipulatedVariable:shaft-speed;
# An example variable:
:oulet-air-flow rdf:type owl:NamedIndividual ,
 elem:ProcessVariable ;
 elem:dealsWithStuff brick:Air;
 elem:hasProcessPosition elem:outlet;
 elem:hasQuantity brick:Flow.
```

3.3 Linking Components to Mechanisms

The description of a general physical mechanism achieved in the step above allows us to query which IVs of a mechanism can be manipulated and the corresponding DVs that are influenced by these. Next, we require a way that permits us to link ports (interfaces) of a component kind to the stereotype of the physical mechanism it uses. For example, heating coils, such as the one in the AHU, use

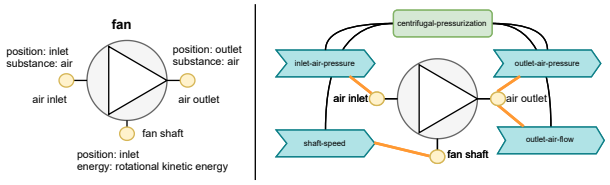

Figure 5: An example of a fan with its stereotypical physical mechanism, associated variables, and relation to the connection points.

the stereotypical heat exchange mechanism that we described as an example in the previous section. Now, we need a way to infer which port (interface) of the heating coil is involved in manipulating the MV representing the water flow rate. A domain expert would, in this case, know that this interface is the water inlet port, similar to the case of a fan, where an expert would know that mechanical force input at the shaft manipulates the fan speed. Based on these observations, to enable the *automatic* inference of such knowledge, we link the *technical interfaces* of a component kind to the (stereotypical) process mechanism the component uses. To model this relationship, we first require a way to describe *stereotypical interfaces* expected in the components and then a way to link these to process variables associated with the corresponding mechanisms.

Several technical systems ontologies, such as OntoCape, PhysSys, and TSO already define the concept of a *terminal*, *connection point*, or *port* to refer to a physical interface offered by the component through which flow of *physical stuff* can be observed or effected [7, 17, 19]. Such a *connection point* is described in terms of the physical *stuff* that flows through it together with the direction with respect to the component in which it flows (e.g., an "air inlet").

To describe technical interfaces, our ontology reuses the class `tso:Con_nect_ion_Point` to represent a terminal along with the relations `elem:dealsWithStuff` which links the terminal to individuals of `elem:PhysicalStuff`. These individuals could then, for instance, be of type `brick:Substance` or `tso:Energy`. Since connection points are semantically qualified as inlet or outlet depending on the direction of the flow, we propose the sub-classes `elem:Inlet-ConnectionPoint` and `elem:OutletConnectionPoint`.

Having created the means to specify process semantics for the connection points of a component, we now describe our approach to linking it to the stereotypes of the physical mechanism and variables we achieved in the previous step.

Considering the example of the fan in the left part of Figure 5, a typical domain expert would be aware of these facts:

- The fan uses the process mechanism stereotype of `centrifugal pressurization`;
- the air pressure and volumetric flow can be observed at the inlet and outlet connection points; and
- the shaft of the fan's impeller (i.e., another connection point) can be supplied with torsional force to increase its speed.

This association is illustrated on the right side of Figure 5.

The linking of the process variables to the corresponding connection points enables software agents to automatically infer that changing the mechanical power input to the impeller shaft will result in change in volumetric flow rate at the fan's air outlet. Therefore, a component kind stereotype (like `hvac:centrifugal-fan`) should link the description of its typical technical interfaces to the description of a general physical mechanism (like `hvac:centrifugal-pressurization`) that is used by it.

To accomplish this, we provide the relationship `elem:isRelatedToVariable` with `elem:ConnectionPoint` as domain and `elem:ProcessVariable` as range. In many cases, the designer of the component stereotype furthermore knows if an `elem:ConnectionPoint` is meant to be used for manipulation of a process variable. In such cases, the relationship should be specified using the property sub-type `elem:manipulates`.

The following snippet shows an example of these relationships applied in the context of the definition of a stereotypical fan (in RDF/Turtle).

```
:centrifugal-fan rdf:type owl:NamedIndividual,
 elem:Stereotype ;
 elem:hasConnectionPoint :cp_fan_air_inlet;
 :cp_fan_air_outlet, :cp_fan_shaft;
 elem:hasProcessMechanism hvac:pressurization.

:cp_fan_air_outlet rdf:type owl:NamedIndividual ,
 elem:OutletConnectionPoint ;
 elem:dealsWithStuff brick:Air;
 elem:isRelatedToVariable hvac:inlet-air-pressure.

:cp_fan_shaft rdf:type owl:NamedIndividual ,
 elem:InletConnectionPoint ;
 elem:dealsWithStuff elem:mechanical_force;
 elem:manipulates hvac:shaft-speed.
```

Based on this example, the following SPARQL query shows how the relationship between the component's interfaces and the associated physical mechanism can be used to determine what inputs influence the airflow rate.

```
SELECT * WHERE {
 urn:fan_01 elem:hasStereotype ?s.
 ?s elem:hasProcessMechanism ?m.
 ?m elem:hasIndependentVariable ?iv.
 ?m elem:hasDependentVariable ?dv.
 ?dv elem:dealsWithStuff brick:Air.
 ?dv elem:hasQuantity brick:Flow.
 ?s elem:hasConnectionPoint ?cp.
 ?cp elem:manipulates ?iv;
}
```

3.4 Inferring Process Relationships

In this section, we explain how the stereotypes of the individual components which are interconnected in a system collectively help infer the *process relationships*. For example, in the AHU (Figure 1), increasing the air flow rate (by increasing the fan speed) affects the air outlet temperature of the heating coil (assuming constant energy input to the heating coil). To infer this relationship, we require an approach that automatically links the DVs of one component to the IVs of other affected components.

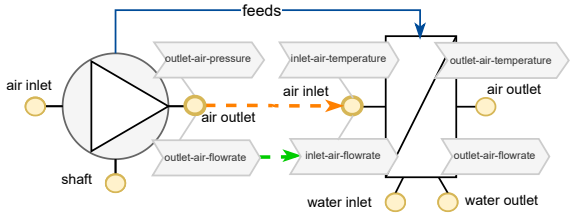

Figure 6: Modeling a connection between two components (fan and heating coil) using a generic feeds relationship is insufficient to infer which specific connection points are connected (e.g., it is not specified whether the fan air outlet is connected to the heating coil's water inlet or its air inlet). However, by considering the characteristics of the connection points, inferences can be drawn about their relationship (orange dashed line), and further, about relationship between the variables (green dashed line).

Often topological descriptions of technical systems, like those based on Brick or IFC, do not model connection points of components (like for example in TSO) explicitly, but instead link the components with one another using a generic relationship such as `brick:feeds`. On the one hand, using a generic relationship between components makes it easier to create system design descriptions, but on the other hand, it is not expressive enough when we want to understand the process relationships – Figure 6 illustrates this situation.

In a system scenario, components are parts of *streams* (see Section 2): a concept which is essential in describing semantics of a connection point and process relationships between components. For example, an AHU economizer has two air inlet connection points, one of which is meant for a *fresh-air* stream, while the other is meant for an *extract-air* stream. In the HVAC domain, *supply-air*, *extract-air*, etc. are examples of well-known stream kinds.

Our ontology supports the integration of streams by providing a class `elem:Stream` and to permit assigning these to domain-specific `elem:StreamKind` (through the property `elem:hasStreamKind`). Component instances may be assigned to one or more `elem:Streams` through the `elem:isPartOf` relationship. Similarly, a `elem:ConnenctnionnPonint` may be specified to be suitable for a particular stream kind using the relationship `elem:suitableFor`.

In order to infer connection between connection points described by the respective stereotypes of two components (in a `feeds` relationship), we match the following characteristics of the connection points:

(1) *Matching Stream*: Both components should be part of the same `elem:Stream`
(2) *Compatible Directionality*: Only an `elem:OutletConnectionPoint` of the source component can connect to an `elem:InletConnectionPoint` of the target component.
(3) *Compatible Stuff*: Both connection points should deal with the same (or compatible) physical *stuff*.
(4) *Matching StreamKind*: In case, multiple connection points of a component stereotype have the same attributes in terms of directionality and physical stuff, the `elem:StreamKind` of the connection points should match.

For two `elem:ConnectinonPoints` which are inferred to be connected using this set of rules, the DV of the `elem:OutletnConnectionnPort` is consequently inferred to elem:manipunlates the IV related to the `elem:InletConnectionPort` to which it connects. If the downstream component has a DV which is an `elem:affectedUpstreamVariable` of the mechanism, then it is inferred to elem:manipulate the corresponding IVs on the upstream component.

We hence propose to infer process dependencies between actual components with the help of their respective stereotypes. For example, using this approach we come to know that the outlet air temperature of an instance of a `brick:Heating_Coil` is influenced by the connected instances of `brick:Fan` (influences air flow) and `brick:Heating_Valve` (influences inlet water flow). This enables both, automatically inferring potential influences of manipulation of a component interface on the system functioning (for e.g., in fault detection) and tracing potential causes of an observed effect to components and their interfaces (e.g, in fault diagnosis).

4 EVALUATION

We report on two types of evaluations we have performed on our approach: 1. To examine the expressivity and generalizability (see Section 4.1) and 2. to study its effectiveness when reasoning about the component's role in a process (see Section 4.2).

4.1 Applying Stereotypes to Brick Components

The use of the Brick ontology to model technical systems in real buildings has shown that its component classes are sufficient to cover common HVAC applications in buildings [2], and we use this result to demonstrate that our proposed approach can be used to describe the component stereotypes for these classes. From the subclasses of `brick:HVAC_Equipment`, we identified all classes that represent components – i.e., those which use a distinct *physical mechanism*.

After the identification of all such components[7] in the Brick ontology, we created stereotype definitions consisting of description of physical mechanisms and technical interfaces. This creation of stereotypes was done together with a HVAC domain expert, and we then related these definitions to the classes of components in Brick. Table 1 shows that our proposed stereotypes can be applied to all Brick components and can hence be used to describe what action can be taken on a component together with its physical effects.

For the stereotypes which we associated to the Brick classes, we then verified if it is possible to find out the relationships between the input interface, the process variable it manipulates, the process variable that will be affected, and the output interface of the component where the effect can be observed.

To evaluate this, we created a SPARQL query to run on the HVAC stereotypes ontology, and presented the results (see Table 2) to three HVAC domain experts. The domain experts confirmed that our system's results were correct and sufficient to gain a high-level understanding of the IVs, MVs, and DVs in the components and to identify the connection points where the MVs can be manipulated.

[7] Some Brick classes like `brick:AHU` implement much more complex behavior than individual components; we consequently treat them as systems of components and they are hence not included in our stereotype matching table.

	Brick Class	Mechanism Stereotype
1	Boiler	heat-generation
2	Heat_Exchanger	heat-exchange
3	Cooling_Tower	heat-exchange
4	Economizer	heat-exchange
5	Cold_Deck	heat-exchange
6	Radiator	heat-exchange
7	Condenser	heat-exchange
8	Space_Heater	radiative-heating
9	Air_Diffuser	throttling
10	VAV	throttling
11	HVAC_Valve	throttling
12	Damper	throttling
13	Isolation_Valve	throttling
14	Bypass_Valve	throttling
15	Fan	pressurization
16	Pump	pressurization
17	Compressor	pressurization
18	Filter	particulate-separation
19	Humidifier	vapor-humidification
20	Motor	electromagnetic-actuation

Table 1: Classes of components available in Brick and the proposed stereotype of the mechanism it employs.

4.2 Understanding Processes using Stereotypes

To evaluate whether our approach is indeed sufficient to reason about a component's role in a real process, we examined the HVAC sub-systems in an office building consisting of about 180 rooms located across six floors which are served by central plants like AHUs, boilers, and chillers. From the system's engineering data available to us in this context, we chose the following system kinds (SK) in the order of increasing complexity (see Figure 7):

(1) SK1: Fresh-air AHU (FAHU). Topologically and process-wise, this represents a simple *sequence of components and their physical mechanisms* in a single stream.
(2) SK2: Recirculation AHU (RAHU). In addition to the features of a FAHU, a RAHU also has the merging and splitting of air streams. This hence *extends the simple sequence* of components to a situation where components in different streams influence each-other.
(3) SK3: Energy-recovery AHU (ERAHU). In this case, an energy recovery component (heat exchanger) influences two process streams by transferring energy from one to another. This represents systems where an individual component manages flows of physical stuff (and, hence, dependencies) across different streams: Other than in the RAHU where the recirculated air stream enters the supply-side components directly, the ERAHU only transfers heat (and not air mass) between the two streams.
(4) SK4: Parallel-Fan AHU (PFAHU): This case introduces the parallel operation of two components (fans) which separate from a stream and later rejoin. This represents systems that feature parallel mechanisms that are stereotypically equivalent and feed jointly into a downstream component.

Though these SKs are based on different configurations of AHUs, the process topology kinds they represent are also seen in other systems such as in chiller and boiler plants.

We defined two test cases to evaluate if topological descriptions where components are associated to stereotypes of their process mechanisms can be used to infer 1. the process dependencies between the components and 2. the effect of a component on a process variable observed at a given position in the stream. We now describe the two test cases and the results in further detail.

Test Case 1: Given two components whose connection is described using a generic *feeds* relation (such as `brick:feeds`), can our approach identify the *physical process relationship* between them?

Results: Table 3 summarizes the results for each pair connected components in each of the SKs. For each of the connections, our approach helps to identify which process variables are affected on the upstream and downstream sides. A HVAC domain expert examined the results and confirmed that the inferences were correct and no connection or associated effects were missing.

Test Case 2: Find component(s) of a system which play a role in modulating a process variable observed at specified test points (TPs) in the process (e.g., "Which components influence the supply temperature observed at the outlet of the heating coil?").

This test case is motivated from the use case of system diagnostics where a maintenance engineer attempts to identify components whose operation is likely to influence the process variable under question (see [1] for further details). To aid the evaluation, our approach permits the definition of TPs at various locations in each of the SKs chosen for evaluation (see TP markers in Figure 7). The TPs were chosen based on key process positions which are relevant for an AFDD use case [14]. We asked a HVAC domain expert to list all components both upstream and downstream of the TPs which would influence temperature and air flow rate at each TP. We then formulated SPARQL queries to automatically find instances of `elem:Components` that influence the temperature and air flow at each TP, and applied these queries to each of the four AHU configurations SK1-SK4.

Results: Our proposed automatic approach achieved 100% matching accuracy of influencing components in SK1 and SK3. At all TPs in these system kinds, our approach correctly identified the chain of dependencies between the components. For example, in TP3 in SK1, the dependencies for air flow are `sufan` and `oad`, and the dependencies for air temperature are `oa`, `oad`, `hcl`, `vlv`, and `sufan`. Our approach performs equally well for SK3 because the energy transfer between two streams (supply and extract) is captured in the mechanism of the energy recovery component (`erc`), and therefore the flow of air in the streams are independent.

However, in SK2, our approach fails to correctly identify the influencing components at TP2, i.e., at one of the seven TPs; and in SK4, it fails with respect to two TPs – for all these errors, our approach underreported influencing components, i.e., while its precision remained at 100%, its recall dropped to 20% for TP2 in SK2 and to 66%/66% for TP3/TP4 in SK4.

Regarding SK2, our approach correctly identified that at all TPs flow is influenced by `oad`, `sufan`, `exfan`, and `ehd`. However, at TP2, our approach identifies only the outdoor air (`oa`) temperature as the variable that influences temperature at TP2 while an expert

Mechanism Stereotype	Input Interface	Manipulated Variable	Affected Variable	Output Interface
heat-generation	electrical-input	electrical-power	water-outlet-temperature	water-outlet
heat-exchange	primary-fluid-inlet	primary-fluid-flowrate primary-fluid-inlet-temperature	secondary-fluid-outlet-temperature primary-fluid-outlet-temperature	primary-fluid-outlet secondary-fluid-outlet
radiative-heating	electrical-input	electrical-power	air-temperature	conditioned-space
throttling	actuation-input	orifice-opening-ration	fluid-flow-rate fluid-outlet-pressure	fluid-outlet
fluid-pressurization	shaft	mechanical-work	fluid-outlet-pressure fluid-inlet-pressure fluid-flow-rate	fluid-outlet fluid-inlet
particulate-separation	none	none	fluid-outlet-particle-count	fluid-outlet
vapor-humidification	steam-inlet	steam-flowrate	air-outlet-humidity	air-outlet
electro-magnetic-actuation	electrical-input	electrical-power	mechanical-work	shaft

Table 2: Querying the stereotype mechanism description suggests which *Input Interface* of the component (that uses the mechanism) can be acted upon towards affecting which *Manipulated Variable*. It further links this manipulation to the *Affected Variable* and gives the *Output Interface* at which this effect can be observed.

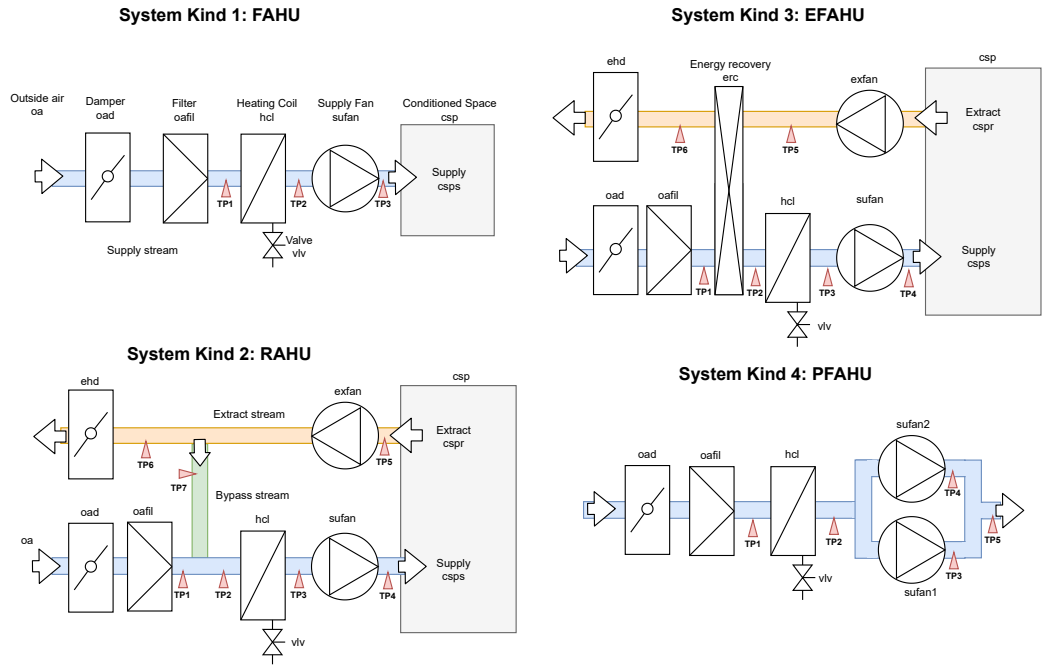

Figure 7: The four system kinds identified for our evaluation. The labels on different components (like *oad* or *sufan*) are referred to in our results. The red triangles represent the test points identified in our evaluation.

SK1			SK2			SK3			SK4		
Connection	**DSE**	**USE**	**Connection**	**DSE**	**USE**	**Connection**	**DSE**	**USE**	**Connection**	**DSE**	**USE**
`oa` → `oadmp`	none	flow									
`oadmp` → `oafil`	flow	none				exfan → erc	flow	none	hcl → sufan1	none	flow
`oafil` → `hcl`	none	none	exfan → hcl	flow	none	erc → ehdmp	none	none	hcl → sufan2	none	flow
`hcl` → `sufan`	none	flow	exfan → ehdmp	flow	flow	oafil → erc	none	none	sufan1 → cspc	flow	none
`vlv` → `hcl`	air T	none				erc → hcl	air T	none	sufan2 → cspc	flow	none
`sufan` → `cspc`	flow	none									

Table 3: Interconnection of components identified by HVAC domain expert and the downstream effects (DSE) and upstream effects (USE) they have on each other in terms of process variables they influence.

identifies five additional factors: `cspr`, `exfan`, `sufan`, `ehd`, and `oad`; these factors determine how much air from the bypass will mix with outside air, and thereby influence the temperature at TP2. The reason for this underreporting of dependencies in our approach is that it is missing a mechanism to model mixing physical stuff from two streams. This problem could be remedied in our HVAC stereotypes ontology by modeling *conditioned space* as a stereotypical component with thermal (e.g., for heat gain) and hydrodynamic (e.g., for air flow) physical mechanisms; however, for our evaluation we chose to use a system that is modeled *exclusively* based on an already available ontology – in this case, Brick. An important learning in this regard is that entities that may not *directly* appear as components in a system topology (such as splits or joins in ducts, etc.) may play an important role in physical processes – these should, hence, be added to ontologies such as Brick.

With respect to TP3 and TP4 in SK4, our approach of chaining variable dependencies in upstream and downstream directions does not handle a case where operation of one component affects the IV of another component in a *parallel* stream. Concretely, the expert identified that operation of `sufan1` will affect the inlet pressure of `sufan2` (and vice-versa), and hence the flow rate at both, TP3 and TP4, are dependent on *both* fans. This issue could be remedied in the same way as for SK2 – that is, by modeling the splitting and joining of streams as explicit components.

5 CONCLUSIONS AND FUTURE WORK

Our approach introduces a novel method of ontology-guided modeling of stereotypical behavior of components that are commonly used in building systems. We have shown that with this approach we can create stereotypes corresponding to HVAC component classes in Brick, which can then be associated to actual instances of components in the system. Further, we validated using real-life engineering information that the stereotypes capture a high-level abstraction of the underlying physical mechanisms which is sufficient to correlate possible actions on the component to the observable effects. This knowledge about the local behavior of each component in a system then helps us infer their cumulative effect on the physical process conducted by the system. We evaluated our approach on systems deployed in real life which displayed largely positive results; however, in process topologies which contain mixing mass flow streams, we identified the need to extend our system with a way to model physical laws that apply to the system in its entirety. Our work seeks to bring together research on semantic modeling of system design and physical processes so that it opens up the possibility for software programs such as for AFDD and plausibility checking to use machine-understandable knowledge to reason about behavior of a system. For higher-fidelity system simulation, our approach is compatible with the attaching of complete simulation models to building automation components.

REFERENCES

[1] Esteban Arroyo, Alexander Fay, Moncef Chioua, and Mario Hoernicke. 2014. Integrating plant and process information as a basis for automated plant diagnosis tasks. In *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*. IEEE, 1–8.

[2] Bharathan Balaji, Arka Bhattacharya, Gabriel Fierro, Jingkun Gao, Joshua Gluck, Dezhi Hong, Aslak Johansen, Jason Koh, Joern Ploennigs, Yuvraj Agarwal, et al. 2016. Brick: Towards a unified metadata schema for buildings. In *Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments*. 41–50.

[3] Jakob Beetz, Jos Van Leeuwen, and Bauke De Vries. 2009. IfcOWL: A case of transforming EXPRESS schemas into ontologies. *Ai Edam* 23, 1 (2009), 89–101.

[4] Jan Benjamin, Pim Borst, Hans Akkermans, and Bob Wielinga. 1996. Ontology Construction for Technical Domains. In *Advances in Knowledge Acquisition: 9th European Knowledge Acquisition Workshop, EKAW'96, Nottingham, UK, May 14-17, 1996. Proceedings*, Vol. 1076. Springer Science & Business Media, 98.

[5] Pim Borst, Hans Akkermans, and Jan Top. 1997. Engineering ontologies. *International journal of human-computer studies* 46, 2-3 (1997), 365–406.

[6] Pim Borst, J Akkermans, Anita Pos, and Jan Top. 1995. The PhysSys ontology for physical systems. In *Working Papers of the Ninth International Workshop on Qualitative Reasoning QR*, Vol. 95. 11–21.

[7] Pim Borst, J Akkermans, Anita Pos, and Jan Top. 1995. The PhysSys ontology for physical systems. In *Working Papers of the Ninth International Workshop on Qualitative Reasoning QR*, Vol. 95. 11–21.

[8] Michel Cessenat et al. 2018. *Mathematical modelling of physical systems*. Springer.

[9] Parastoo Delgoshaei, Mark A Austin, Amanda J Pertzborn, Mohammad Heidarinejad, and Vikas Chandan. 2017. Towards a Semantically-enabled Control Strategy for Building Simulations: Integration of Semantic Technologies and Model Predictive Control. In *Proceedings of the 15th IBPSA Conference San Francisco, CA, USA, Aug. 7-9.*

[10] Luis Garcia, Stefan Mitsch, and André Platzer. 2019. HyPLC: Hybrid programmable logic controller program translation for verification. In *Proceedings of the 10th acm/ieee international conference on cyber-physical systems*. 47–56.

[11] Thomas R Gruber and Gregory R Olsen. 1994. An ontology for engineering mathematics. In *Principles of Knowledge Representation and Reasoning*. 258–269.

[12] Markus Gwerder, Reto Marek, Andreas Melillo, and Maria Husmann. 2022. Data Integrity Checks for Building Automation and Control Systems. In *CLIMA 2022 conference*.

[13] Jack Hodges. 1995. Functional and physical object characteristics and object recognition in improvisation. *Computer Vision and Image Understanding* 62, 2 (1995), 147–163.

[14] John M House, Hossein Vaezi-Nejad, and J Michael Whitcomb. 2001. An expert rule set for fault detection in air-handling units/discussion. *Ashrae Transactions* 107 (2001), 858.

[15] Heiko Koziolek, Julius Rückert, and Andreas Berlet. 2020. Industrial plant topology models to facilitate automation engineering. In *Systems Modelling and Management: First International Conference, ICSMM 2020, Bergen, Norway, June 25–26, 2020, Proceedings 1*. Springer, 91–108.

[16] Simon Mayer, Jack Hodges, Dan Yu, Mareike Kritzler, and Florian Michahelles. 2017. An open semantic framework for the industrial Internet of Things. *IEEE Intelligent Systems* 32, 1 (2017), 96–101.

[17] Jan Morbach, Andreas Wiesner, and Wolfgang Marquardt. 2009. OntoCAPE—A (re) usable ontology for computer-aided process engineering. *Computers & Chemical Engineering* 33, 10 (2009), 1546–1556.

[18] Claus Ballegaard Nielsen, Peter Gorm Larsen, John Fitzgerald, Jim Woodcock, and Jan Peleska. 2015. Systems of systems engineering: basic concepts, model-based techniques, and research directions. *Comput. Surveys* 48, 2 (2015), 1–41.

[19] Nicolas Pauen, Dominik Schlütter, Jérôme Frisch, and Christoph van Treeck. 2021. TUBES System Ontology: Digitalization of building service systems. In *Proceedings of the 9th Linked Data in Architecture and Construction Workshop*.

[20] Heinz A Preisig. 2021. Ontology-based process modelling-with examples of physical topologies. *Processes* 9, 4 (2021), 592.

[21] Ganesh Ramanathan. 2021. Autonomous buildings. In *Proceedings of the 8th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*. 246–247.

[22] Ganesh Ramanathan, Maria Husmann, Christoph Niedermeier, Norbert Vicari, Kimberly Garcia, and Simon Mayer. 2021. Assisting automated fault detection and diagnostics in building automation through semantic description of functions and process data. In *Proceedings of the 8th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*. 228–229.

[23] Maurice Stewart. 2018. *Surface production operations: volume IV: pumps and compressors*. Gulf Professional Publishing.

[24] Jan L Top and Hans Akkermans. 1991. Computational and Physical Causality.. In *IJCAI*. 1171–1176.

[25] LC Van Ruijven. 2013. Ontology for systems engineering. *Procedia Computer Science* 16 (2013), 383–392.

[26] Michael Wetter. 2019. A view on future building system modeling and simulation. In *Building performance simulation for design and operation*. Routledge, 631–656.

[27] Lan Yang, Kathryn Cormican, and Ming Yu. 2019. Ontology-based systems engineering: A state-of-the-art review. *Computers in Industry* 111 (2019), 148–171.

[28] Masaharu Yoshioka, Yasushi Umeda, Hideaki Takeda, Yoshiki Shimomura, Yutaka Nomaguchi, and Tetsuo Tomiyama. 2004. Physical concept ontology for the knowledge intensive engineering framework. *Advanced engineering informatics* 18, 2 (2004), 95–113.