Data-Centric Engineering (2024), 5: e33
doi:10.1017/dce.2024.26

RESEARCH ARTICLE

FSBrick: an information model for representing fault-symptom relationships in heating, ventilation, and air conditioning systems

Min Young Hwang, Burcu Akinci and Mario Bergés

Civil and Environmental Engineering Department, Carnegie Mellon University, Pittsburgh, PA, USA
Corresponding author: Min Young Hwang; Email: minyounh@andrew.cmu.edu

Received: 01 April 2024; **Accepted:** 12 June 2024

Keywords: causal relationships; fault diagnosis; fault-symptom relationships; ontology; semantic information modeling

Abstract

Current fault diagnosis (FD) methods for heating, ventilation, and air conditioning (HVAC) systems do not accommodate for system reconfigurations throughout the systems' lifetime. However, system reconfiguration can change the causal relationship between faults and symptoms, which leads to a drop in FD accuracy. In this paper, we present Fault-Symptom Brick (*FSBrick*), an extension to the *Brick* metadata schema intended to represent information necessary to propagate system configuration changes onto FD algorithms, and ultimately revise FSRs. We motivate the need to represent FSRs by illustrating their changes when the system reconfigures. Then, we survey FD methods' representation needs and compare them against existing information modeling efforts within and outside of the HVAC sector. We introduce the *FSBrick* architecture and discuss which extensions are added to represent FSRs. To evaluate the coverage of *FSBrick*, we implement *FSBrick* on (i) the motivational case study scenario, (ii) Building Automation Systems' representation of FSRs from 3 HVACs, and (iii) FSRs from 12 FD method papers, and find that FSBrick can represent 88.2% of fault behaviors, 92.8% of fault severities, 67.9% of symptoms, and 100% of grouped symptoms, FSRs, and probabilities associated with FSRs. The analyses show that both *Brick* and *FSBrick* should be expanded further to cover HVAC component information and mathematical and logical statements to formulate FSRs in real life. As there is currently no generic and extensible information model to represent FSRs in commercial buildings, *FSBrick* paves the way to future extensions that would aid the automated revision of FSRs upon system reconfiguration.

Impact Statement

As a part of our research vision to create an adaptive fault diagnosis framework robust to system reconfiguration, this article offers a generic and extensible information modeling approach to represent fault-symptom relationships. Through this work, we (i) motivate the need for FSR representation from FD methods in the literature, (ii) compare them against representations possible in current information models in and out of the HVAC sector, (iii) offer a 25 entity and relationship extension to an existing information model, *Brick*, called *FSBrick*, and (iv) evaluate its coverage against three case studies that span multiple real HVAC systems.

This research article was awarded Open Data and Open Materials badges for transparent practices. See the Data Availability Statement for details.

1. Introduction

Faults in heating, ventilation, and air conditioning (HVAC) systems account for 20% of energy consumption in buildings (Deshmukh et al., 2020), which corresponds to approximately 2.4% of the total annual energy use of the United States, equivalent to approximately 2.4 quadrillion BTUs (United States Energy Information Administration [EIA], 2023). However, detecting and diagnosing these faults has proven difficult. A study led by Lawrence Berkeley National Laboratory found that while commercial fault detection tools for HVAC systems reached 83% accuracy, fault diagnosis (FD) only achieved 66% accuracy (Lin et al., 2020).

All FD methods learn the causal relationship between faults and symptoms, either explicitly or implicitly, based on some system assumption (e.g., system configuration). For example, a rule-based method, like air handling unit (AHU) performance assessment rules (APAR) (House et al., 2001), lists known fault and symptom pairs (*explicitly*), and supervised learning methods listed in (Mirnaghi and Haghighat, 2020) learn a function that maps symptoms to known faults (*implicitly*). These fault-symptom relationships (FSR) can change as a result of system reconfiguration (Hwang et al., 2024) (e.g., a fault, such as a fouled heating coil, may no longer be associated with a symptom, Rule 4, in APAR), and having a formal representation of them would facilitate the process of automatically updating the diagnosis methods. However, partly due to not having formal representation, current FD methods do not automatically adapt FSR to system configuration changes, and thus, are more susceptible to becoming inaccurate. For example, in the aforementioned study (Hwang et al., 2024), FD accuracy improved 70% for one of the actuator faults when the method was manually corrected to account for system reconfiguration. Once the assumptions on which the FD method is based on change, the FD method should automatically propagate changes to the relationship between faults and symptoms to maintain diagnosis accuracy.

Currently, subject matter experts (e.g., facility managers) manually modify FD tools in response to system configuration changes (e.g., addition/removal of thermal zones). Within the HVAC domain, both quantitative and qualitative model-based methods require an expert to intervene and change the model to account for system reconfigurations (Zhao et al., 2015, 2017; Yan et al., 2018; Velibeyoglu et al., 2019; Zhu et al., 2019; Qiu et al., 2020; Taal and Itard, 2020; Pradhan et al., 2021). The process history-based methods require labeled data to train classification functions for the new system configuration (Yan et al., 2019; Mirnaghi and Haghighat, 2020).

As shown in Figure 1, we propose a vision where an FD method can continuously adapt to system configuration changes. To fulfill this vision, a semantic model and a corresponding reasoning engine need

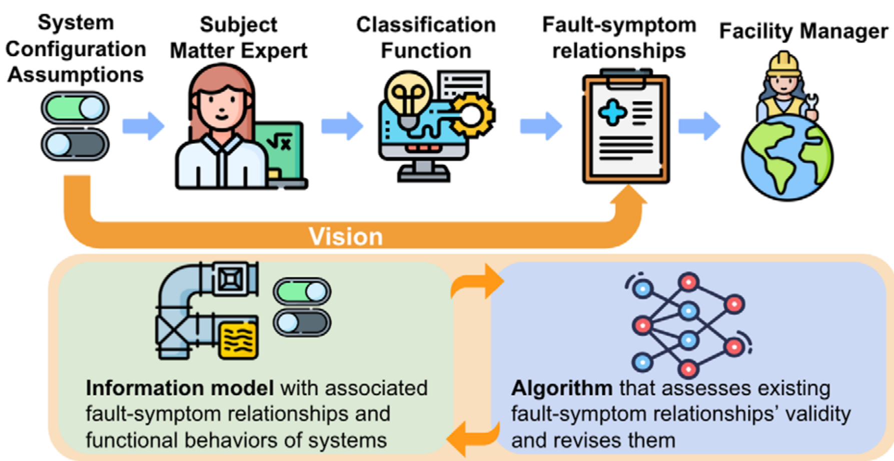

Figure 1. *An illustration of the vision for automatically incorporating system configuration information in the FD method (classification function). We will specifically focus on* FSBrick*, which is a part of the information model.*

https://doi.org/10.1017/dce.2024.26 Published online by Cambridge University Press

to be developed to conjecture how a change in the system configuration will affect the existing FSR (e.g., will the relationship between existing FSR still hold?). Towards this vision, we survey the needs of information representation from existing FD methods, draw inspiration from existing semantic models both in and out of the HVAC sector, and build an extension to *Brick*, which can already represent elements necessary for FSR codification, unlike others. The main contribution of this paper is an extension to *Brick*, called Fault-Symptom Brick (*FSBrick*): a semantic model for commercial HVAC systems to represent part of the information necessary (i.e., FSR) to adapt FD algorithms to system configuration changes.

The rest of the paper is organized as follows. First, in Section 2, we introduce a motivational case study, followed by a literature review on the needs of different FD methods in representing fault-symptom relationships and how existing information models fall short in representing necessary FSR information (Section 3). In addition, *FSBrick* (source code), the proposed extension to one of the existing information models (*Brick*), is motivated by the needs identified in HVAC FD and general information model literature and case study (Section 4). *FSBrick* is then tested for coverage across FSRs from (i) the case study, (ii) 3 AHUs and their Building Automation System (BAS) points, and (iii) 12 FD papers (Section 5). Finally, we have a summary of findings from the analyses and a discussion for future improvements (Section 6).

This paper builds upon previous research (Hwang et al., 2023) by providing (i) a more thorough synthesis of the FSR information requirement through literature review of FD methods (culminating in Table 1), (ii) 5 additional entities and ontological relationships to represent additional needs identified from the synthesis (e.g., grouped symptoms and probabilities for FSR mapping detailed in Section 4), and (iii) an expanded coverage analysis in Section 5 to further demonstrate the applicability and range of *FSBrick.*

2. Motivating case study

To better understand the requirements for the semantic model representing FSRs, we study various system reconfiguration scenarios and the resulting changes to specific faults and their symptoms in these systems. Specifically, we study a simplified thermal resistance-capacitance (RC) network model of a room with one cooling and one heating source calibrated with winter month data from Carnegie Mellon University's (CMU) PhD student room AHU in Porter Hall. Thermal RC network models are commonly used in HVAC system behavior modeling literature (Hazyuk et al., 2012; Kircher and Zhang, 2015; Brastein et al., 2019; Boodi et al., 2020). Additionally, we consider three types of reconfigurations (i.e., addition, deletion, modification) at three different granularity levels (i.e., component level, subsystem level, system level), which were identified to be common system reconfigurations for HVAC systems (ASHRAE, 2018; Hwang et al., 2024).

Many commercial HVAC systems are composed of an AHU, which handles the preparation and distribution of conditioned air for the building, and a Variable Air Volume (VAV) (a terminal unit), which takes the air from the AHU and adjusts the zone temperature to the occupants' liking. However, some FD algorithms (e.g., House et al., 2001; Yan et al., 2018) intended to work on AHUs do not account for the VAV's behavior. Since applying the FD algorithm in the presence of the VAV may violate the assumptions, we are considering this as a system reconfiguration (specifically, a subsystem addition). Upon surveying common system reconfigurations on CMU campus, we concluded that subsystem level addition was the most common, and therefore, we will focus on this case. Another example of a more "physical" subsystem level addition can include multiple terminal units, such as Fan Coil Units and VAVs, working in the same zone due to zone-separating wall demolition, which occurred in Porter Hall three times, and twice in one year (Akcamete, n.d.).

To show how system reconfiguration affects FSR changes, we now parse the subsystem level addition example. To represent the AHU (system configuration before changes), we kept the RC network model of a room with one cooling and one heating source. Six faults were selected based on the cooling and heating manipulations possible in the RC network model. These six faults (i.e., *Cooling Coil Valve Stuck Open*, *Cooling Coil Valve Stuck Closed*, *Cooling Coil Valve Leaking 20%*, *Heating Coil Valve Stuck Open*, *Heating Coil Valve Stuck Closed*, *Heating Coil Valve Leaking 20%*) were inserted to the existing RC

https://doi.org/10.1017/dce.2024.26 Published online by Cambridge University Press

network model to generate the symptoms. We defined a *Supply Air Temperature Alarm* to be our symptom for when the internal temperature of the RC network model fell below 59°F or above 61°F. Threshold-based alarms, such as the one defined here, are commonly used in rule-based methods, such as in House et al. (House et al., 2001). *Cooling Coil Valve Stuck Open* and *Heating Coil Valve Stuck Closed* faults both triggered the *Supply Air Temperature Alarm* in the existing, unreconfigured system. To generate FSRs for the reconfigured system, we inserted one more heating source in the RC network model to represent the addition of a VAV with a reheat subsystem. This time, when the same faults were injected to the reconfigured system, we found that the *Cooling Coil Valve Stuck Open* and *Heating Coil Valve Stuck Closed* faults did not trigger the *Supply Air Temperature Alarm.* Therefore, we found that 2 out of 6 FSRs were altered by reconfiguration.

The case study example shows us that FSRs change with system reconfiguration and tracking this change automatically is crucial in maintaining FD accuracy. Having FSR representation would facilitate the process of automatically updating the FD diagnosis methods. In the next section, we will review the literature to find information models that may help us represent these FSRs more formally.

3. Literature review

In the case study section, we focused on how rule-based FD methods (which fall under the qualitative model-based FD methods) fell in diagnosis accuracy when system reconfiguration occurred. In this section, we will explore (i) the different types of FD methods in the HVAC sector and the common information requirements for representing faults, symptoms, and fault-symptom relationships; and (ii) how current information modeling sectors represent faults, symptoms, and fault-symptom relationships. The section will wrap up with a discussion surrounding the gaps that still exist in representing faults, symptoms, and fault-symptom relationships in Table 1.

3.1. HVAC fault diagnosis methods

FD methods for the HVAC sector can be classified into the following three categories: (i) qualitative model-based, (ii) quantitative model-based, and (iii) process history-based methods with grey-box or hybrid method extensions for each category (Katipamula and Brambley, 2005a, 2005b; Kim and Katipamula, 2018). The summary of all methods reviewed can be seen in Figure 2.

3.1.1. Qualitative model-based FD methods

Qualitative model-based FD methods involve encoding information about the system's behavior in a knowledge base to refer to when isolating the fault (Chi et al., 2022). Among inductive reasoning methods, which are bottom-up approaches that derive conclusions based on individual observations, are case-based reasoning (Xu et al., 2018) and knowledge-graphs (Chen et al., 2020; Chi et al., 2022) (and their extension into providing causal graphs for grey-box fault diagnosis (Velibeyoglu et al., 2019; Zhu et al., 2019)). Another type of qualitative model-based FD method, the deductive reasoning methods, which are top-down approaches that conjecture about specific cases from general axioms stored in the knowledge base, includes rule-based reasoning (House et al., 2001; Delgoshaei and Austin, 2017) (and their extension into providing causal graphs for bayesian networks (Zhao et al., 2015; Zhao et al., 2017; Taal and Itard, 2020; Pradhan et al., 2021) and ontology-based reasoning (Zhou et al., 2015; Chen et al., 2015) methods. The shared foundation of these methods is their reliance on a knowledge base, which uses a unified taxonomy and ontology for information reuse and reasoning.

Faults in this area of literature were described with respect to their *location* and *behavior.* Location is the equipment in the system (e.g., a descriptive string) that is causing the anomalous behavior, while the behavior describes *how* the equipment is malfunctioning (e.g., a descriptive string). For example, House et al. (House et al., 2001) cite "Leaking heating coil valve" as a fault, where "heating coil valve" refers to the fault location and "leaking" refers to the fault behavior. Similarly, in the built environment, qualitative

https://doi.org/10.1017/dce.2024.26 Published online by Cambridge University Press

Table 1. *Needs identified for HVAC faults, symptoms, and fault-symptom relationships from FD methods literature review in comparison with what current information models can represent. The horizontal line delineates between FD method needs and information model capabilities*

					Information models' ability to represent FSRs for THCS						
			Example data types	Examples	IFC schema	COBie	gbXML	Brick	LBNL fault taxonomy	openCAESAR	Manufacturing ontologies
FD method needs	Fault	Location/system input variable	Descriptive string	E.g., 'heating coil valve'	☑	☑	☑	☑	☑	☐	☐
		Behavior	Descriptive string	E.g., 'stuck'	☐	☐	☐	☐	☑	☐	☐
		Severity/intensity	Float value	E.g., 60%	☐	☐	☐	☐	☑	☐	☐
		Time duration	Reference string	E.g., '2023-11-15 14:42:56'	☐	☐	☐	☑	☐	☐	☐
	Symptom	Measured value/system output variable	Descriptive string	E.g., 'CO2 concentration'	☑	☑	☐	☑	☐	☐	☐
		Behavior	Descriptive string	E.g., 'high'	☐	☐	☐	☑	☐	☐	☐
		Threshold for nominal behavior	Float value	E.g., supply air temperature setpoint (°F) = 60	☐	☐	☐	☑	☐	☐	☐
		Time duration	Reference string	E.g., '2023-11-15 14:42:56'	☐	☐	☐	☑	☐	☐	☐
	Fault-Symptom Relationships	Connection	A dictionary, RDF predicate, or OWL object property	E.g., becauseOf/theEffectIS	☐	☐	☐	☐	☐	☑	☑
		Probability value associated with connection	Float value	E.g., 70%	☐	☐	☐	☐	☐	☐	☐
		Grouped and simultaneous nature	A list of strings or UML association	E.g., simultaneous activation of multiple rules corresponding to a fault {'F1' : ['R1', 'R2', 'R3']}	☐	☐	☐	☐	☐	☐	☐

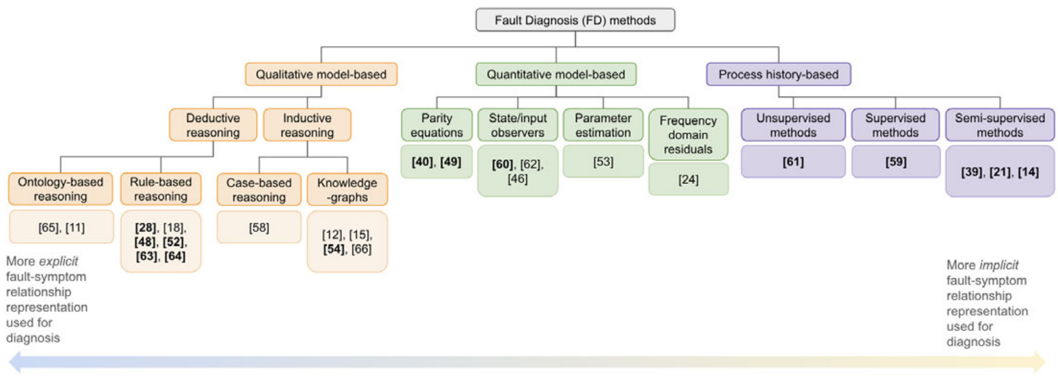

Figure 2. *FD method classification for the HVAC sector and built environment adapted and extended from Katipamula and Brambley (2005a, 2005b), Kim and Katipamula (2018), Venkatasubramanian et al. (2003), and Mirnaghi and Haghighat (2020). The bolded works are HVAC specific.*

model-based FD method works, location (e.g., equipmentComponent—Zhou et al., 2015; faultEquipment—Xu et al., 2018; Fault_name—Chen et al., 2015) and behavior (e.g., failureCause—Zhou et al., 2015; faultCause—Xu et al., 2018) are included in the fault description.

Similarly, symptoms were described with respect to their *sensed value* and *behavior.* The sensed value is the observed variable from sensors deployed in the HVAC system (e.g., a descriptive string) and the behavior is the description of the anomaly (e.g., a descriptive string). For example, in Taal and Itrad (Taal and Itard, 2020), a symptom “high CO_2” is associated with the CO_2 (sensed value) and a higher than nominal measurement (a behavior of the CO_2 concentration with a threshold in mind). In the built environment qualitative model-based FD method works, measured value (e.g., sensed state variables—Velibeyoglu et al., 2019), vibration measurements (Chen et al., 2015) and behavior (e.g., vibration characteristics—Chen et al., 2015) are present in the symptom description.

Fault-symptom relationships were explicitly represented in qualitative model-based FD methods in various ways. House et al. (2001) used a table with faults and symptoms on the axes with check marks to indicate the relationships between them, Zhu et al. (2019) used graphical representation of arrows in between faults and symptoms with probability values associated with them, and Zhou et al. (2015), Chen et al. (2015), and Xu et al. (2018) used ontological relationships, such as “becauseOf/theEffectIs,” “causeIs/toEffect,” and “hasReason/isReasonOf.” Therefore, the fault-symptom relationships can be described by *the connection between faults and symptoms* (e.g., resource description framework [RDF] predicate or web ontology language [OWL] object property), *the one to many nature of the connection* (e.g., unified modeling language [UML] association), and *probability values associated with the connection* (e.g., a float value).

3.1.2. Quantitative model-based FD methods

Quantitative model-based FD methods involve modeling the system and comparing the model’s components, such as outputs, internal states, or unknown inputs, with sensor values from the real system to discern the presence of anomalous behavior, and finally isolate the fault cause (Venkatasubramanian et al., 2003). This comparison between the system model and the system itself can be realized through residual generation. There are different categories of residual generation methods, which is a concept thoroughly explored in the control theory space: (i) parity equations for system output estimation (Qiu et al., 2020), (ii) state observers and also input observers (Zhang et al., 2017; Naderi and Khorasani, 2018; Yan et al., 2018), (iii) frequency domain residuals (Frisk, n.d.), and (iv) parameter estimation (Isermann, 2005; Turner et al., 2017).

To contextualize residual formulation, we define a Linear Time Invariant (LTI) model, Σ', of the system, Σ:

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t) + Fw(t) \\ y(t) &= Cx(t) + n(t)\end{aligned} \tag{3.1}$$

The system model, Σ', has A as the dynamics matrix, B as the control matrix, C as the sensor matrix, and F as the fault relation matrix. The variables $x \in \mathbb{R}^k$ represent the state vector, y the output vector, $w \in \mathbb{R}^j$ the faulty input vector, $u \in \mathbb{R}^i$ the input vector, and n the measurement noise. Constant actuator faults, especially, were described with a fault severity value, f_s, from time t_0 to t_T (Frisk, n.d.; Xu and Zhang, 2004; Qiu et al., 2020):

$$w(t; f_s, t_0, t_T) = f_s \text{ for } t_0 \leq t \leq t_T \tag{3.2}$$

Therefore, faults, for this method class, are defined by their *associated input variable to the system via actuator equipment* (e.g., a descriptive string), *fault severity* (e.g., a float value), and *time duration* (e.g., reference (string) to a multi-dimensional array with timestamps and corresponding values). For example, in Qiu et al. (2020) the stuck actuator fault is described with n, the actuator to the system (associated input variable to the system), which can take a fixed value, R (fault severity) for time duration $t \geq T$.

An example of a residual generated for system, Σ, specifically for the parity equations that compare system output, can be described with the following equation:

$$r(t) = y(t) - \widehat{y}(t) \tag{3.3}$$

where $y(t)$ is the system output and $\widehat{y}(t)$ is the predicted system output generated from the system model, Σ'. Similarly for state observer, input observer, frequency domain, and parameter estimation methods, the residual generation process involves a comparison of a vector from the real system $v(t)$ with a vector from the system model $\widehat{v}(t)$, with a predefined threshold, ε for time no earlier than when the fault enters the system ($t_1 > t_0$) to time T (t_T). The comparison vectors, $v(t)$ and $\widehat{v}(t)$, can be compared individually, for N vectors (i.e., $v(t) \in \mathbb{R}^{T \times N}$) with $\varepsilon \in \mathbb{R}^{T \times N}$, or in groups, for M groups with $\varepsilon \in \mathbb{R}^{T \times M}$.

Therefore, symptoms, for this method class, are defined by their *associated system output variable* (e.g., a descriptive string), *possibly grouped and simultaneous nature* (e.g., a list of strings (grouped system output variables)), *predefined threshold for nominal behavior* (e.g., a float value), and *time duration* (e.g., reference (string) to a multi-dimensional array with timestamps and corresponding values). For example, in Yan et al. (2018) the difference between the supply air temperature (associated system output variable) and its set-point (predefined threshold for nominal behavior) is considered for a time window (time duration).

Additionally, some quantitative model-based methods design residuals specifically for each fault (Jain et al., 2019), but others construct "influence structures" or "structured residuals" to classify which faults are present in the system based on the produced set of residuals (Venkatasubramanian et al., 2003; Frisk, n.d.; Svärd, n.d.). These influence structures are tables, much like the one used by House et al. (2001), with faults and symptoms on the axes with 0 or 1 to indicate the presence of relationships between them. Therefore, fault-symptom relationships for quantitative model-based FD methods, when represented, require *the connection between faults and symptoms* (e.g., a dictionary with key and value pairs) and *the one to many nature of the connection* (e.g., list of strings for the value in a dictionary).

3.1.3. Process history-based FD methods

Process history-based FD methods use data from the system to either train a model to recognize fault-symptom relationships (e.g., model-based, supervised methods) or determine a pattern in data (e.g., statistical, unsupervised methods) (Mirnaghi and Haghighat, 2020). Supervised methods, such as ones that use support vector machines (SVMs) or Neural Networks (NN) (Yan et al., 2019), train a diagnostic classifier with labeled fault and symptom pairs, and expect the trained classifier to identify the fault from real system outputs. Semi-supervised methods train a classifier with a limited number of fault training

https://doi.org/10.1017/dce.2024.26 Published online by Cambridge University Press

samples and pull from techniques, such as generative adversarial networks (GANs) (Li et al., 2021), active learning (Fan et al., 2024), and similarity learning (Chen et al., 2023) to also learn from unlabeled data sets. Finally, unsupervised methods require no labeled data, and only generate fault diagnosis results based on patterns in data using methods, such as clustering and associative rule mining (ARM) (Yu et al., 2012).

Among the process history-based methods that use labeled data (supervised and semi-supervised methods), the input to training the classifiers were matrices of system output variables, or features, that were labeled with fault classes. The faults were labeled with *location*, which is the affected system input via the actuator equipment to the system (e.g., descriptive string), *behavior*, which is the description of how the equipment is malfunctioning (e.g., a descriptive string), and *intensity*, which adds a numeric description of the degree of malfunctioning (e.g., a float value). In Yan et al. (2019), one of the labeled faults is "Cooling coil valve stuck (partially open - 15%)". The "cooling coil valve" is the location, "stuck" is the behavior, and "15%" is the intensity. The symptom, or the description of anomalous behavior in the *system output variables* (e.g., a descriptive string), are described with *time series data* (e.g., reference (string) to a multidimensional array with timestamps and corresponding values). For example, Fan et al. (2024) used time-series data of common system output variables, such as air temperatures, water temperatures, flow rates, and differential pressures from fans. The fault-symptom relationships for this method subclass are the *connection between faults and symptoms* (e.g., a dictionary with key-value pairs) and the *one to many nature of the connection* (e.g., list of strings for the value in a dictionary).

For process history-based methods that do not use labeled data, the input for training the classifiers was just the symptoms themselves; the symptoms were described in a similar manner to the supervised and semi-supervised methods, with *time series data* (e.g., reference (string) to a multidimensional array with timestamps and corresponding values) of *system output variables* (e.g., a descriptive string). The output for ARM methods, like Yu et al. (2012), create rules with faults and symptoms with probability values attached. The faults for the output are described with *location* (e.g., a descriptive string), which is the affected system input via actuator equipment, and *behavior* (e.g., a descriptive string), which is the nature of the anomalous behavior. In Yu et al. (2012), one of the faults was "fan frequency is high," which has "fan frequency" as the location, and "high" as the behavior. The fault-symptom relationships are defined by two probability values called "support" and "confidence." Therefore, fault-symptom relationships for this method subclass require the *connection between faults and symptoms* (e.g., a dictionary with key-value pairs) and *probability values associated with the connection* (e.g., a float value).

3.2. Semantic information models

Semantic models are ways for us to represent information in a structured and standardized way that both humans and algorithms can interpret, revisit, and repurpose (Pauwels et al., 2017). Information models (semantic and information models are used interchangeably in this text) are built from (i) taxonomies, which define concepts and concept hierarchies, and (ii) ontologies, which define relationships between concepts (Malin and Throop, 2007). For our framework, we first identified information needed to be represented for FSRs, which we highlighted through a literature review of the FD methods FSR information requirements. We then searched for existing information models within and outside of HVAC and found that while many existing information models can represent partial FSR information, a complete set of descriptions may be missing.

3.2.1. FSR information within HVAC sector

We surveyed information models in the HVAC sector, such as the *IFC schema*, *COBie*, *gbXML*, and *Brick*, and found that while HVAC component information location/system input variable is well represented, fault behavior, fault severity, symptom, and FSR descriptions are incomplete.

The *IFC schema* has HVAC location representation (i.e., with `ifcHvacDomain (HVACie)`) (IFCHVACDOMAIN, n.d.), COBie can store information, such as expected `fanSpeed` and `fanPressureDrop` (COBie Guide, n.d.), and `gbXML` can hold system input information, such as

https://doi.org/10.1017/dce.2024.26 Published online by Cambridge University Press

`AirLoopEquipment` and `equipmentType` (Sun et al., 2020). These information models can store HVAC component location information at different granularities. However, they are not specifically designed for aiding FD and, therefore, do not store information about component fault behavior, fault severities, symptoms, and FSRs.

There have been efforts to extend existing information models for FD. *Brick* has a taxonomy for HVAC components location (i.e., sensor collection point, physical location, equipment family), but also lays the groundwork for representing symptoms and faults (Balaji et al., 2016). Symptoms exist in *Brick* in the form of `brick:Alarm` entities for specific system output variables with characteristic descriptions for certain system output variables (e.g., `brick:High_Supply_Air_Temperature_Alarm`). Additionally, Brick can represent time duration (e.g., `brick:TimeseriesReference` and `brick:hasTimeseriesReference`) and threshold values for some output variables (e.g., `brick:Temperature_Tolerance_Parameter`). The time duration Brick entity, in particular, could also be used to describe the fault's time duration. Fault representation, on the other hand, falls short in *Brick*. `brick:Fault_Status` exists as a *Brick* entity, however, specific types of faults associated with specific HVAC components do not exist as entities in *Brick*. Furthermore, `brick:Fault_Status` does not indicate the behavior of the fault (e.g., is the damper stuck or leaking?) nor the severity of the fault (e.g., how badly (%) is the damper leaking?), which is useful information for facility managers who will interpret FD results and realize it into repairs in the physical system (Balaji et al., 2016) and recognized as necessary from our FD method literature review. *Brick* also offers tags, and more importantly, brick:fault tags, however, tags are created ad-hoc by the user (Fierro et al., 2019). Therefore, tags do not fit what we envision for *FSBrick*, which aims to create consistent information representation for continued revision of FSRs. Additionally, there are no formal ontological relationships in *Brick* to describe FSRs (e.g., `brick:isPartOf`, `brick:isFedBy`, `brick:isPointOf` do not convey diagnosis causal relationships).

Apart from *Brick*, Lawrence Berkeley National Laboratory defined common HVAC faults in a comprehensive taxonomy, where equipment type, component location, and component type, which corresponds to location/system input variable, fault behavior, and fault severity/intensity were outlined (Chen et al., 2021). However, connections with faults and their system-wide symptoms, which were not defined in a taxonomy unlike the faults, were not systematically defined, beyond in diagrams (e.g., fault trees), which again have no formal structure. Additionally, because the work focused on creating a taxonomy for common HVAC faults, symptom taxonomy and ontology were overlooked. Liu et al. (n.d.) have also tried to create an information model for aiding FD processes, which included automatic extraction of functional relationships (e.g., medium flowing in and affected by the component, import, outport, sensor associated with the component) of HVAC components necessary for inputs to FD algorithms from IFC files. However, functional relationships only hint at possible FSRs, and do not explicitly model them. In summary, taxonomy of HVAC systems and ontology for symptoms exist, but they have yet to be combined to represent FSRs.

3.2.2. FSR information outside of HVAC sector

We also surveyed outside of the HVAC domain to see if other fields have addressed supporting the representation of faults, symptoms, and FSRs, since they did not exist in the HVAC literature. In the aerospace sector, NASA has been leading the effort to move away from document-based modeling into model-based systems engineering (MBSE) with SysML as the main language used especially in FSR modeling (Mathur et al., 1998; Day et al., n.d.; Aaseng, 2015; Cornford and Feather, 2016; Izygon et al., 2016; Wang et al., 2016; Infeld et al., 2018; Figueroa et al., 2019). This body of work is especially relevant since MBSE, specifically "State Machine Diagram" and "Requirement Diagram" in SysML, allows derivation of FSRs, in the form of fault trees, failure modes and effects analysis tables, and D-matrices, in a modular fashion with expert's intervention (Hwang et al., 2024). These diagrams, along with an algorithm to traverse them, would result in FSRs. However, NASA users do not have a unified ontology to describe the faults, symptoms, and their relationships, which is also a problem that NASA recognized and began working on through (Malin and Throop, 2007) and openCAESAR (OML Tutorials, n.d.).

https://doi.org/10.1017/dce.2024.26 Published online by Cambridge University Press

The manufacturing sector also has literature on representing faults, symptoms, and FSRs (Chi et al., 2022). Xu et al. (2018) developed an ontology that describes faults through a class `FaultMode` and its two subclasses `FaultCause`, which is akin to our faults, and `FaultEffect`, which is akin to our symptoms, and a relationship `has_reason` and `has_effect` to describe the connections between the faults and symptoms. Similarly, Zhou et al. (2015) connected failure mode (in our case alarms) and failure cause (in our case faults) with a `BecauseOf` relationship. While we can learn how to represent FSRs from the manufacturing sector, the ontology for faults and symptoms is not HVAC specific. Therefore, there is still a gap to address in terms of HVAC FSR representation, which leads us to suggest that we need to create our own formal information model.

4. Architecture of *FSBrick*

After surveying the available information models and FD methods' information representation needs, we moved forward with extending *Brick* to accommodate fault and FSRs because it already had a Resource Description Framework (RDF) format of HVAC component taxonomy to serve as a basis for fault representation and a more complete representation capability compared to other surveyed information models. RDF format is particularly beneficial for searching through all the faults, symptoms, and FSRs in the HVAC system using SPARQL queries as potential candidates for revisions required in an adaptive system, as envisioned. The *Brick* entities were preserved and repurposed as much as possible. Existing information models using the *Brick* ontology should not have issues using *FSBrick* since elements were added, and no existing entities or relationships were manipulated or subtracted. The tables in this section will provide piece-by-piece examples of how *Brick* and *FSBrick* entities can be combined to formulate FSR and Figure 3 will give a complete FSR example. The *FSBrick* Github repository contains (i) the extended Brick.ttl file and (ii) the data used for the coverage analysis in Section 5.

4.1. Representing faults

As seen in Table 1, fault behavior is not currently represented in *Brick*, and must be extended through the addition of *FSBrick.* Additionally, *FSBrick* must also account for new ontological relationships that will allow users to append fault severity information to the rest of the fault information.

4.1.1. Representing and connecting fault behavior

For *FSBrick*, we adapted the fault taxonomy developed by Chen et al. (2021, in particular, fault nature in Table 4) into the *Brick* ontology to account for the missing fault representation, specifically fault behavior. This work describes what behavior of faults (e.g., "Stuck", "Leakage") are possible for which specific HVAC component or location, which we used to create new fault entities in *FSBrick.* The summary of *FSBrick* fault representation is presented in Table 2 for faults not related to sensors or controls. The *Brick* entities introduced in the leftmost column in Table 2 (e.g., `brick:Reheat_Valve`) can be connected to these new *FSBrick* fault entities with a new ontological relationship `fsbrick:isFault/hasFault`, akin to how Xu et al. (2018) organized faults. Figure 3 shows an example implementation, and we can see that `bldg:Chilled_Water_Valve` is connected to `fsbrick:Valve_Leakage` via *FSBrick* relationship `fsbrick:hasFault`.

4.1.2. Connecting fault severities

Additionally, we used existing *Brick* entities to describe the fault severity with a new relationship, `fsbrick:isSeverity/hasSeverity`. Fault severity is described by two *Brick* entities: one quantity indicator to define what quantity is affected by the fault and one float value to explain to what degree the quantity is affected. For example, valves already have a *Brick* entity called `brick:Position`, which specifies in percentages what position the valve is in. The `Position` entity can also be connected to an XSD double via the `brick:value` relationship as seen in Figure 3. Rightmost column of Table 2 also gives insight into which *FSBrick* fault entities can be matched with existing *Brick*

https://doi.org/10.1017/dce.2024.26 Published online by Cambridge University Press

Table 2. A snippet of fault nature taxonomy from Chen et al. (2021) for faults and how they can be combined with existing Brick *entities to create new* FSBrick *entities for nonsensor or control-related faults*

Brick entities for HVAC components	Fault nature taxonomy (from (Chen et al., 2021))	New *FSBrick* entities examples	*Brick* entities for fault severity
Valves (e.g., `brick:Reheat_Valve`, `brick:Return_Heating_Valve`, `brick:Steam_Valve`)	Stuck	`fsbrick:Valve_Stuck`	`brick:Position` with `brick:Value Literal:XSD double`
	Leakage	`fsbrick:Valve_Leakage`	`brick:Flow` with `brick:Value Literal: XSD double`
Coils (e.g., `brick:Coil`, `brick:Heating_Coil`, `brick:Cooling_Coil`)	Fouling	`fsbrick:Coil_Fouling`	`brick:Flow` with `brick:Value Literal: XSD double`
Dampers (e.g., `brick:Outside_Damper`, `brick:Exhaust_Damper`, `brick:Return_Damper`)	Stuck	`fsbrick:Damper_Stuck`	`brick:Position` with `brick:Value Literal:XSD double`
	Leakage	`fsbrick:Damper_Leakage`	`brick:Flow` with `brick:Value Literal: XSD double`
	Malfunctioning	`fsbrick:Damper_Malfunctioning`	`brick:Flow` with `brick:Value Literal: XSD double`
Fans (e.g., `brick:Fan`, `brick: Supply_Fan`, `brick:Return_Fan`)	Stuck	`fsbrick:Fan_Stuck`	`brick:Rotational_Speed` with `brick: Value Literal:XSD double`
	Malfunctioning	`fsbrick:Fan_Malfunctioning`	`brick:Rotational_Speed` with `brick: Value Literal:XSD double`
Filters (e.g., `brick:Filter`)	Block	`fsbrick:Filter_Block`	`brick:Flow` with `brick:Value Literal: XSD double`
Air Plenums (e.g., `brick:Air_Plenum`, `brick:Supply_Air_Plenum`)	Block	`fsbrick:Air_Plenum_Block`	`brick:Flow` with `brick:Value Literal: XSD double`
	Leakage	`fsbrick:Air_Plenum_Leakage`	`brick:Flow` with `brick:Value Literal: XSD double`
Pumps (e.g., `brick:Pump`)	Stuck	`fsbrick:Pump_Stuck`	`brick:Position` with `brick:Value Literal:XSD double`
	Leakage	`fsbrick:Pump_Leakage`	`brick:Flow` with `brick:Value Literal: XSD double`

entities to describe fault severity. For example, brick:Rotational_Speed has an applicable unit "RAD-PER-MIN" which can tell us how the rotational speed has been affected due to the fsbrick:Fan_Malfunctioning.

4.2. Representing symptoms

As mentioned in Section 3, *Brick* already has measured value/system output variable information, behavior, time duration, and threshold representation for the HVAC system. *Brick* already has alarms that allow for alerting operators to off-nominal conditions that correspond with common sensors found in HVAC systems. While the motivational case study showcased just one threshold-based alarm, we want to note that the *Brick* alarm class goes beyond solely representing symptoms for rule-based methods. The alarm class can be generalized to specify any anomalous behavior in the system. *Brick* has entities that allow us to define what off-nominal conditions are with parameters and setpoints, which can serve as thresholds. For example, brick:Temperature_Setpoint and brick:Temperature_Tolerance_Parameter (threshold for nominal behavior) can be connected to a brick:Air_Temperature_Alarm to imply that the alarm will sound when the monitored temperature (measured value/system output variable) reaches beyond an acceptable threshold. We selected alarms that specified the medium (air) and quantity measured (temperature) for our classification, such as brick:Air_Temperature_Alarm. For example, we can have brick:Water_Temperature_Alarm connected to brick:Chilled_Water with a brick:isPoint relationship to imply that the chilled water temperature is behaving anomalously. There is no Chilled_Water_Temperature_Alarm in *Brick*; Nor do we feel the need to add it to *FSBrick*'s entity list because the combination of entities and relationship already imbues the meaning we want.

The above representation is a specific example implementation for a qualitative model-based method symptom, however, the alarm class, as mentioned, can be generalized. The same alarm, brick:Air_Temperature_Alarm (measured value/system output variable) can be connected to a brick:TimeseriesReference with a brick:hasTimeseriesReference relationship (time duration). This alarm could also be associated with a general brick:Limit or brick:Tolerance_Parameter entity (threshold for nominal behavior), which can also have size $\mathbb{R}^{T \times N}$ through its connection with a time series (brick:TimeseriesReference with a brick:hasTimeseriesReference). Symptom behavior can also be implied through the comparison of the measured value/system output variable and the threshold for nominal behavior. Figure 3 illustrates these relationships with the brick:Supply_Air_Temperature_Alarm and brick:Mixed_Air_Temperature_Alarm. Therefore, we will focus more on developing the FSRs that are missing from *Brick*, such as the grouped and simultaneous nature of symptoms.

4.2.1. Representing grouped and simultaneous nature

In *FSBrick*, we created a grouped entity called fsbrick:Grouped_Symptom, which uses fsbrick:hasSymptom relationships (which will be explored in more detail in the following section) to connect individual *Brick* alarms to indicate the grouped nature of the symptom. The simultaneity of the individual symptom in the group would be implied through the shared time series reference start times and end times connected to each alarm entity. This extension was based off of the n-ary relations that openCAESAR (OML Tutorials, n.d.) adopted to have a "relation entity" that can hold information other than the relationship between between two entities. In Figure 3, symptoms brick:Mixed_Air_Temperature_Alarm and brick:Supply_Air_Temperature_Alarm are grouped together through the fsbrick:Grouped_Symptom entity.

4.3. Representing FSRs

Lastly, although *Brick* does not have a system in place currently to represent fault-symptom relationships, we can borrow from the aerospace and manufacturing industry to extend FSR connection, probability, and one-to-many relation concepts to *FSBrick*.

4.3.1. Representing FSR connections

As explained earlier, Zhou et al. (2015) connect faults and symptoms with a `BecauseOf` relationship, and for *FSBrick*, we propose a new relationship, `fsbrick:isSymptomOf/hasSymptom`, to convey the same message in a more "*Brick*" manner (i.e., is/has ontological relationships). An example of this architecture is provided in Figure 3. A chain of entities and relationships connect the fault, `fsbrick:Valve_Leakage`, through `fsbrick:hasSeverity`, and `fsbrick:hasSymptom` (indirectly) to the `brick:Supply_Air_Temperature_Alarm`, which represents our FSR. In cases where there are no grouped symptoms, the `fsbrick:hasSymptom` relationship would be directly attached to the fault and the respective symptom (see Figure 4).

4.3.2. Representing and connecting FSR one-to-many relations

Similarly, we propose a new relationship to connect faults with grouped symptoms, `fsbrick:isGroupedSymptom/hasGroupedSymptom`. In Figure 3, `bldg:Percent_Limit`, which is the last element of our fault description (i.e., fault severity), is connected to the `bldg:Grouped_Symptom` entity with the proposed `fsbrick:hasGroupedSymptom` relationship.

4.3.3. Representing and connecting FSR probabilities

In the n-ary relation documentation for openCAESAR, the World Wide Web Consortium had a working page (W3C, 2006) on attaching meaning to relationships by creating a "relation entity," which

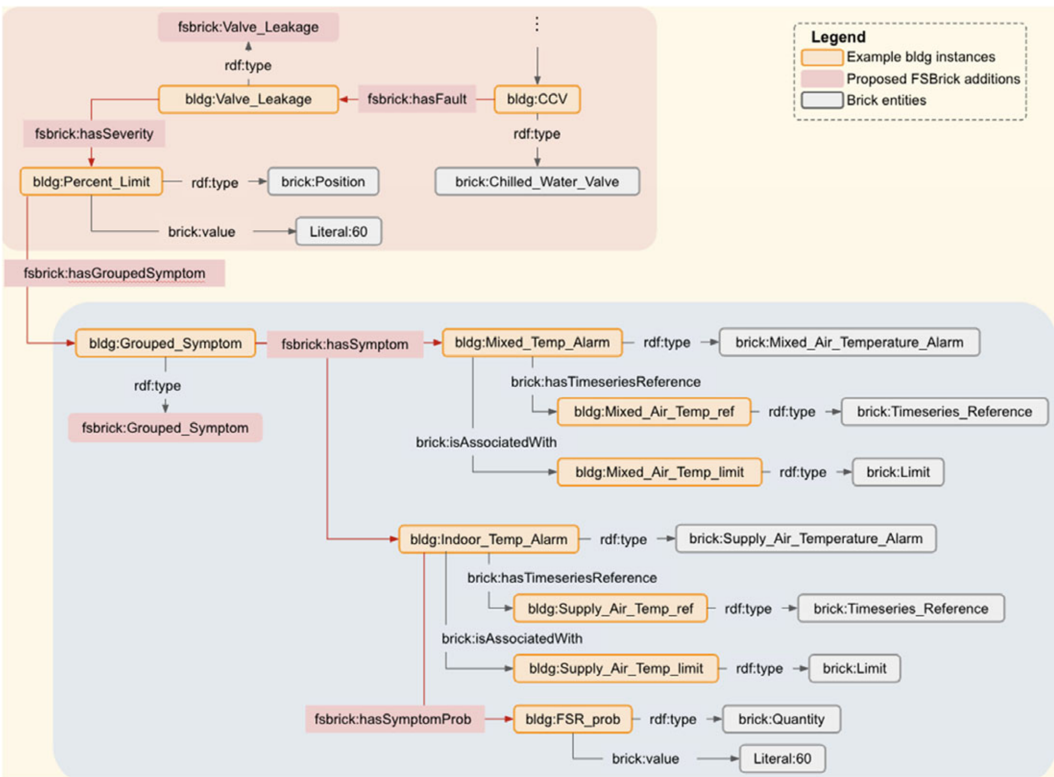

Figure 3. *Additions to the chilled water valve to represent fault, symptoms, and fault-symptom relationships. New entities added include* `fsbrick:Valve_Leakage` *and* `fsbrick:Grouped_Symptom`*. New ontological relationships include:* `fsbrick:hasFault`*,* `fsbrick:hasSeverity`*,* `fsbrick:hasGroupedSymptom`*,* `fsbrick:hasSymptom`*, and* `fsbrick:hasSymptomProb`*. The red box highlights fault representation and the blue box highlights symptom representation.*

https://doi.org/10.1017/dce.2024.26 Published online by Cambridge University Press

Table 3. *List of* Brick *symptoms that are connected to* Brick *entities to build FSRs*

Brick example entities for HVAC mediums	*Brick* example symptoms
`brick:Supply_Air`	`brick:Air_Temperature_Alarm`
`brick:Outside_Air`	`brick:Humidity_Alarm`
`brick:Return_Air`	`brick:Air_Flow_Alarm`
`brick:Zone`	`brick:CO2_Alarm`
`brick:Filter`	`brick:Pressure_Alarm`
`brick:Chilled_Water`	`brick:Water_Temperature_Alarm`
`brick:Cooling_Valve`	`brick:Valve_Position`

openCAESAR (OML Tutorials, n.d.) also uses. One of the suggested additional information to attach to the relation entity was a probability that describes the strength of the diagnosis certainty. Similarly, we can express the strength of the causal relationship between faults and symptoms with a `brick:Quantity` and `Literal` that ranges between 0 and 1. We can also attach this causal strength to the `brick:Alarm` entity to represent FSR probabilities. For example, in Figure 3, we see that `brick:Supply_Air_-Temperature_Alarm` is attached to a `brick:Quantity` and a `Literal` through the proposed relationship `fsbrick:hasSymptomProb`.

5. Applied study

The *FSBrick* architecture's coverage (defined as % of entities mapped) was tested through surveying its ability to represent FSRs of (i) an example from the motivating case study, (ii) 3 AHUs and their BAS points, and (iii) 12 manuscripts in the FD literature. Challenges and shortcomings with the current iteration of *FSBrick* are also explored in this section.

5.1. FSBrick mapping in the motivating case study

We map the subsystem addition example presented in the motivating case study as an initial check for *FSBrick*'s coverage. In the case study, we see that the *Cooling Coil Valve Stuck Open* and *Heating Coil Valve Stuck Closed* faults both triggered the *Supply Air Temperature Alarm* for the existing system, consisting solely of the AHU. This FSR is recorded in Figure 4, where the *Cooling Coil Valve Stuck Open* fault, expressed by the chain of `brick:Cooling_Valve`, `fsbrick:hasFault`, `fsbrick:Valve_Stuck`, `fsbrick:hasSeverity`, `brick:Position`, `brick:value`, `Literal:100`, is connected to the symptom, `brick:Air_Temperature_Alarm`, with a `fsbrick:hasSymptom` relationship. Similar representation is also displayed for the `Heating Coil Valve Stuck Closed` fault and `brick:Air_Temperature_Alarm` symptom. For the reconfigured case where both *Cooling Coil Valve Stuck Open* and *Heating Coil Valve Stuck Closed* faults no longer display the `brick:Air_Temperature_Alarm` symptom, we can simply disconnect the two faults and symptom by erasing the `fsbrick:hasSymptom` connection as seen in Figure 5.

5.2. FSBrick mapping to the FD case study in Carnegie Mellon University's Porter Hall

Additionally, we applied *FSBrick* and *Brick* to a real-life case study using Building Automation System (BAS) points. This application was done to showcase *FSBrick*'s ability to represent FSRs in *real-life building* HVAC systems, such as the AHUs in CMU's Porter Hall, as opposed to the simulated case study examples in the last subsection. We queried CMU's HVAC FD platform to survey their fault database and collect nonsensor or command faults that occurred between 5/15/23 and 6/15/23. Table 4 shows the faults that were flagged by the platform's diagnosis systems (notice that only one has a severity associated with it).

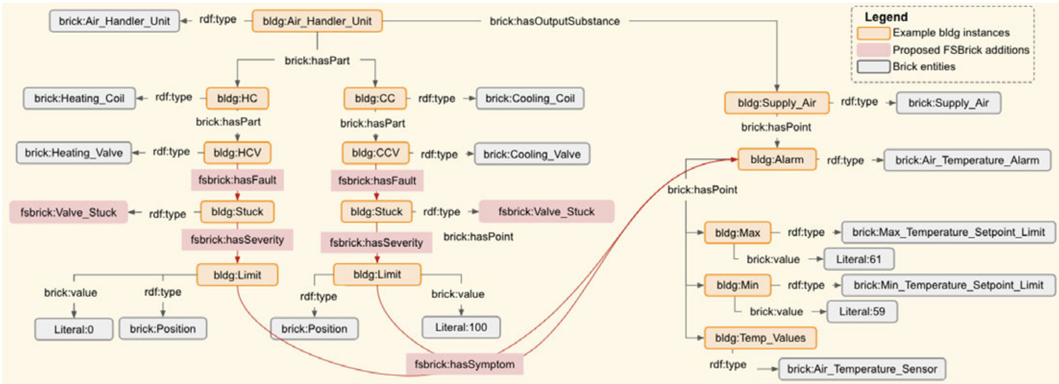

Figure 4. *Representing the FSRs for* Cooling Coil Valve Stuck Open *and* Heating Coil Valve Stuck Closed *faults from the motivating case study before the reconfiguration.*

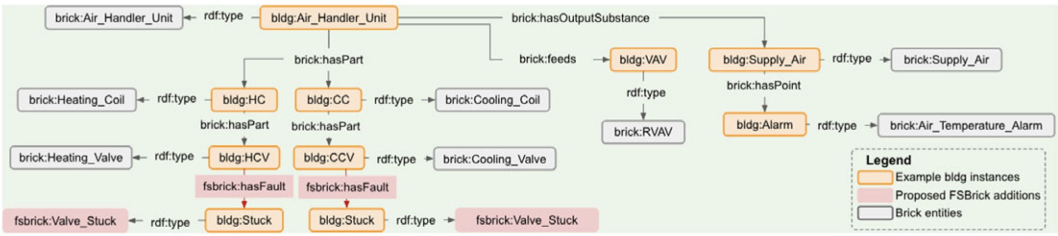

Figure 5. *Representing the FSRs for* Cooling Coil Valve Stuck Open *and* Heating Coil Valve Stuck Closed *faults from the motivating case study after the reconfiguration. The severity and alarm detail entities were taken out to avoid repetitive information.*

Table 4. *Mapping from the FD platform fault name to FSBrick and Brick entities. The dates below the AHU names correspond to the 24-hour period in which the fault was present.*

AHU #	FD platform fault name	Fault mapping	Fault severity mapping
AHU3 (5/25/23)	Heating Valve Stuck Closed	`fsbrick:Valve_Stuck` for `brick:Hot_Water_Valve`	`brick:Position` with `Literal:0`
AHU9 (6/7/23)	Cooling Valve Leaking	`fsbrick:Valve_Leakage` for `brick:Cooling_Valve`	No severity in name
AHU2 (6/11/23)	Exhaust Air Damper is Open But No Airflow	`fsbrick:Damper_Malfunctioning` for `brick:Exhaust_Damper`	No severity in name

The dates below the AHU names correspond to the 24-hour period in which the fault was present.

In parallel, we also pulled a subset of the available BAS points in the same 24-hour period that the faults were detected in and converted them into alarms, if an entity had setpoints and sensor values. The threshold parameters were selected to our best judgment, since the importance of the analysis is placed on representing FSRs and not the accuracy of the relationships. The alarm would ring if the following inequalities were not met for more than an hour:

- *|Outside Air Airflow—Outside Air Airflow Minimum Setpoint| < 200 cfm.*
- *Outside Air damper position—Outside Air Damper minimum % open > x* (varied with the 3 AHUs: AHU2 100%, AHU3 55%, AHU9 30%).
- *Exhaust Air damper position—Exhaust Air Damper minimum % open > y* (varied with the 3 AHUs: AHU2 100%, AHU3 55%, AHU9 60%).
- *|PreHeat Water Supply Temperature—PreHeat Water Supply Temperature Setpoint| < 2.5 F.*
- *Return Air CO_2 Maximum Setpoint—Return Air CO_2 > 0 ppm.*
- *|Supply Air Static Pressure Actual—Supply Air Static Pressure Setpoint| < 0.5 in H_2O.*
- *|Supply Air Airflow—Supply Air Airflow Setpoint| < 200 cfm.*
- *|Supply Air Temperature Actual—Supply Air Temperature Setpoint| < 2.5°F.*

Out of the 3 faults, 1 fault severity, 8 symptoms, and 16 BAS points observed, we were not able to assign *Brick* entity for one of the symptoms (`brick*:Damper_Position_Alarm` does not exist, and the asterisk specifies this) and one of the BAS points (`brick*:CO2_setpoint_limit` does not exist). This was to bring attention to the fact that *Brick* itself may need to be expanded to accommodate for FSRs.

In Figure 6, we can see an example of how *FSBrick*, *Brick*, and CMU's BAS points can be used together to represent FSRs for AHU3. To describe the *Heating Valve Stuck* fault, we related `brick:Hot_Water_Valve` to the `fsbrick:Valve_Stuck` entity with a `fsbrick:hasFault` relationship. To convey that it is a *Valve Stuck Closed* fault, we related a `brick:Position` of `Literal:0` with `fsbrick:hasSeverity`. To describe the *|Outside Air Airflow—Outside Air Airflow Minimum Setpoint| < 200 cfm* symptom, we related `brick:Air_Flow_Alarm` to `brick:Outside_Air`. In addition, we attached `brick:Tolerance_Parameter`, `brick:Min_Air_Flow_Setpoint_Limit`, and `brick:Air_Flow_Sensor` to `Literal:200`, *AHU3:OA Airflow Min Setpoint*, and *AHU3: OA Airflow* respectively, to convey the alarm's parameters. Lastly, we connected the fault with the symptom using the `fsbrick:hasSymptom` relationship. Along with the visualization results for AHU3, we also provided FSR representation using *FSBrick*, *Brick*, and CMU's BAS points for all AHUs in Table 5.

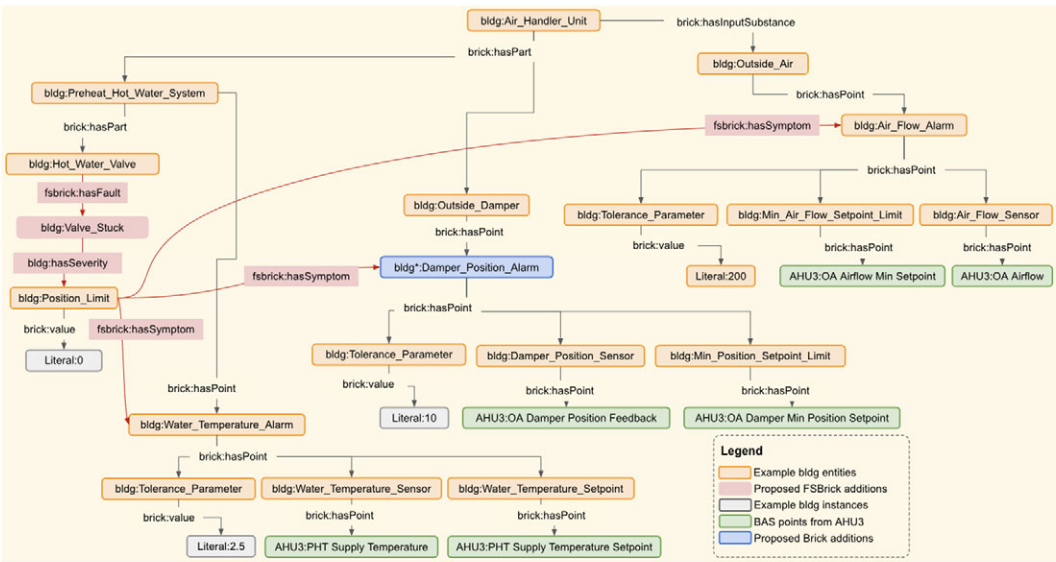

Figure 6. *FSR with* FSBrick *for AHU3's BAS points. The connection between* Brick *and* FSBrick *entities were deleted to avoid repetition. However, all example bldg entities were named verbatim after* Brick *entities.* `bldg:Valve_Stuck` *and* `bldg*:Damper_Position_Alarm` *were colored by their* FSBrick *or* Brick *entity colors.*

https://doi.org/10.1017/dce.2024.26 Published online by Cambridge University Press

https://doi.org/10.1017/dce.2024.26 Published online by Cambridge University Press

Table 5. *FSR mapping for* FSBrick, Brick, *and BAS points from CMU. Note that the "–" holds repeating information*

AHU #	Symptom mapping	BAS symptom element name	Brick symptom element mapping
AHU3 (5/25/23)	`brick:Air_Flow_Alarm` for `brick:Outside_Air`	AHU3:OA (Outdoor Air) Airflow AHU3:OA Airflow Min Setpoint	`brick:Air_Flow_Sensor` `brick:Min_Air_Flow_Setpoint_Limit`
	`brick*:Damper_Position_Alarm` for `brick:Outside_Damper`	AHU3:OA Damper Position Feedback AHU3:OA Damper Min Position Setpoint	`brick:Damper_Position_Sensor` `brick:Min_Position_Setpoint_Limit`
	`brick:Water_Temperature_Alarm` for `brick:Preheat_Hot_Water_System`	AHU3:PHT (Preheat Hot Water) Supply Temperature AHU3:PHT Supply Temperature Setpoint	`brick:Water_Temperature_Sensor` `brick:Water_Temperature_Setpoint`
AHU9 (6/7/23)	`brick:Air_Flow_Alarm` for `brick:Outside_Air`	–	–
	`brick*:Damper_Position_Alarm` for `brick:Outside_Damper`	–	–
	`brick:Water_Temperature_Alarm` for `brick:Preheat_Hot_Water_System`	–	–
	`brick:Pressure_Alarm` for `brick:Supply_Air`	AHU9:SA Static Pressure Actual AHU9:SA Static Pressure Setpoint	`brick:Static_Pressure_Sensor` `brick:Static_Pressure_Setpoint`
	`brick:Air_Temperature_Alarm` for `brick:Supply_Air`	AHU9:SA Temperature Actual AHU9:SA Temperature Setpoint	`brick:Air_Temperature_Sensor` `brick:Air_Temperature_Setpoint`
AHU2 (6/11/23)	`brick:Air_Flow_Alarm` for `brick:Outside_Air`	–	–
	`brick:Water_Temperature_Alarm` for `brick:Preheat_Hot_Water_System`	–	–

5.3. FSBrick mapping to FSRs in literature

Finally, we chose to build a database of fault behaviors, fault severities, symptoms, grouped symptoms, FSRs, and probability values for FSRs from a subset of the HVAC FD literature we surveyed previously (House et al., 2001; Liang and Du, 2007; Zhao et al., 2015; Zhao et al., 2017; Yan et al., 2018; Yan et al., 2019; Qiu et al., 2020; Taal and Itard, 2020; Li et al., 2021; Pradhan et al., 2021; Chen et al., 2023; Fan et al., 2024) and perform a coverage analysis similar to the one done for Balaji et al. (2016). In that study, *Brick*'s applicability and effectiveness were tested by the ability to map five campus HVAC data points (e.g., from BMS, other metadata formats, and building infrastructure) to *Brick.* The match percentage was calculated by field experts assessing if point names could be manually converted to a *Brick* entity. From the literature, we collected unique descriptors for 86 fault behaviors, 125 fault severities, 159 symptoms, 25 grouped symptoms, 98 FSRs, and 29 probability values for FSRs for nonsensor-related faults, available on a Github repository with *FSBrick.ttl* file. We want to mention that Fan et al.'s (2024) FSRs came from Granderson and Lin (n.d.), which provides us another opportunity to check *FSBrick*'s coverage for another real-life HVAC test bed. Additionally, some works in Figure 2 did not explicitly list some fault, symptom, and FSR elements; hence, they were left out of this evaluation. The results of the coverage analysis can be seen in Table 6. We will refer to this table in the following paragraphs to discuss our results.

5.3.1. Fault behavior mapping

Of the 86 fault behaviors, 88.23% of them were converted into 13 unique *FSBrick* fault entities. Most fault descriptors from literature were sorted into `fsbrick:Valve_Stuck`, `fsbrick:Damper_Stuck`, `fsbrick:Valve_Leakage`, and `fsbrick:Coil_Fouling`. The fall in % entities mapped came from (i) *Brick* missing an entity to describe ducts, and therefore, we could not account for faults like *AHU duct leaking before/after supply fan* and (ii) some fault descriptors were more like symptoms rather than faults. For example, *heating coil reduced capacity* can be due to heating coil fouling, but the authors did not specify further. Therefore, we could not conjecture what *FSBrick* entity would fit best.

5.3.2. Fault severity mapping

Of the 125 descriptors for fault severities, 92.8% were connected to *FSBrick* fault entities with 3 unique combinations of existing *Brick* and *FSBrick* ontological relationships. The unique combination consisted of a link to the *FSBrick* fault entity with `fsbrick:hasSeverity` ontological relationship to (i) `brick:Flow`, `brick:Position`, or `brick:Rotational_Speed` and (ii) variable quantitative descriptors (e.g., 60%) with `brick:value` and `Literal:XSD double`. Qualitative descriptions, such as *exhaust air damper stuck fully open* were converted to `Literal:100`, to the best of our knowledge. The fall in coverage came from (i) *Brick* missing descriptors for elements like surface area and (ii) failure to convert some qualitative descriptors (e.g., *complete failure*) into either *FSBrick* or *Brick* entities.

5.3.3. Symptom mapping

Of the 159 symptom descriptors, 67.9% were mapped to various alarm entities as mentioned in Table 3. The fall in mapping score came from missing entities in *Brick*, such as the lack of a flow alarm for water when there is one for air (i.e., `brick:Air_Flow_Alarm` under `brick:Air_Alarm` but no `brick:Water_Flow_Alarm` under `brick:Water_Alarm`). The other limitation of *FSBrick* and *Brick* was incorporating mathematical operations. For example, some of the symptoms we could not represent were *Difference between return air and mixed air temperatures* and *Supply fan power consumption is a polynomial function of supply air flow rate.* The symptoms that we had envisioned usually consisted of entities within the medium and quantity being measured. For example, the `brick:Air_Temperature_Alarm` associated with `brick:Supply_Air` would imply that the supply air is out of the range of its setpoint +/- the threshold. Therefore, it was difficult to represent House et al.'s (2001) rules that required comparison across multiple mediums. If we subtract the rules from our dataset, *FSBrick* reaches up to 79.8% coverage for symptoms that only concern themselves with one medium and quantity.

Table 6. *Percentage mapped results and sample examples for faults, fault severities, and symptoms collected from HVAC literature and fitted to* FSBrick

	Fault behavior	Fault severity	Symptom	Grouped Symptom	FSRs	Probability values for FSRs
% Mapped	88.2%	92.8%	67.9%	100%	100%	100%
E.g.	`fsbrick:Cooling_Valve_Stuck` *can represent:* Cooling coil valve stuck closed/open, AHU cooling coil valve stuck higher/lower than normal, Stuck cooling coil, Cooling coil valve stuck	`fsbrick:hasSeverity` + `brick:Position` + `value, literal` *can represent:* Percentages (e.g., 0%, 5%, 15%, 100%), Qualitative descriptors (e.g., stuck at max, stuck at min)	`fsbrick:hasSymptom` + `brick:Valve_Position_Alarm` *can represent:* Cooling coil control signal open/close valve, predicted control signal of cooling coil valve vs actual value (positive max, positive, negative, negative min)	`fsbrick:Grouped_Symptom` + `fsbrick:hasSymptom` *can represent:* High CO_2 AND air flow rate = 0 at the same time	`fsbrick:hasSymptom` *can represent:* Recirculation damper stuck causing mixed air temperature alarm AND outlet water temperature alarm	`fsbrick:hasSymptomProb` + `value, literal` *can represent:* The probability of cooling coil valve fully stuck open fault causing symptom E12 and E42 be at 28%

5.3.4. Grouped symptom mapping

We collected 25 grouped symptoms from literature, which mostly consisted of statements, such as the one in Qui et al. (2020), where an FSR says the *air valve stuck* fault will exhibit *low room air temperature*, *increased fan energy consumption*, and *increased water pump energy consumption* at the same time. If the symptom could be represented with *FSBrick*, the coverage result was 100% for this category. However, we want to bring attention to the fact that while the current *FSBrick* implementation can represent logic statements, like "AND," it has difficulty representing others, like "OR", and "NOT." This is a problem that we foresee in future usages of *FSBrick*, although not one we encountered during our literature search.

5.3.5. FSR mapping

98 FSRs were collected, and it was possible to map all faults and symptoms with the `fsbrick:hasSymptom` relationship, if the subject and object of the RDF graph could be represented with *Brick* and *FSBrick*.

5.3.6. FSR probability mapping

We collected 29 FSR probabilities for the coverage analysis. Some FSRs were posterior probabilities linking faults and all symptoms together (Pradhan et al., 2021), which would mean that the `fsbrick:Grouped_Symptom` entity would be connected to a `brick:Value Literal:XSD double` with `fsbrick:hasSymptomProb`. Other FSRs had faults attached to individual symptoms and had probability values for these individual relationships. *FSBrick* represented these relationships with connections from faults to individual symptoms (e.g., alarms) via `fsbrick:hasSymptom`. The individual symptoms were also attached to `brick:Value Literal:XSD double` with the `fsbrick:hasSymptomProb` relationship. These two representations covered 100% of the FSRs we found in the literature.

6. Discussion and conclusion

Current FD methods do not automatically account for system reconfiguration, where existing FSRs will need to be checked and revised. To do so, we must create formal representation for existing FSRs that contain semantic information. We presented *FSbrick*, which was a first attempt at representing FSRs on top of an existing information model, namely *Brick*. We chose *Brick* because its development towards representing FSR was further along than other semantic models. *Brick* already had (i) HVAC equipment necessary for fault representation and (ii) symptoms in the form of alarms, thresholds, and setpoints. While we chose to build upon *Brick* for the current iteration of representing FSRs, there is merit in exploring the incorporation of this work in more commonly used schemas, like HVACie, and even graphical modeling languages, like SysML.

FSBrick adds (i) entities to describe fault behaviors (16 *FSBrick* entities), fault severities (2 *FSBrick* entities), and grouped symptoms (3 *FSBrick* entities) and (ii) ontological relationships to connect fault entities to symptom entities (2 *FSBrick* entities) and symptom entities to probability associated with the FSR (2 *FSBrick* entities). We conducted three studies to show *FSBrick*'s applicability and coverage: showcasing *FSBrick*'s usage on the motivational case study, applying *FSBrick* to represent FSRs in 3 different AHUs and their BAS points at CMU, and analyzing the % entities mapped on FSRs found in 12 FD papers across all method types. Through our analyses, we discovered that *Brick* itself can be extended to better accommodate for FSR representation, as it lacked infrastructure to describe some HVAC components and properties. *FSBrick* can also be improved further, to include mathematical and logical expression representation in symptoms and FSRs. In this iteration of this work, simultaneous alarm activation (e.g., "AND" relationship) could be represented by *FSBrick* with the addition of the grouped symptom entity. However, other logical expressions, such as "NOT" and "OR" could not be represented. These elements will be explored in future works to aid the automated revision of FD algorithms upon system reconfiguration.

https://doi.org/10.1017/dce.2024.26 Published online by Cambridge University Press

Overall, this work is in line with the building energy academic community's efforts to streamline the adaptation of smart analytics and control applications by standardizing descriptions for HVAC operations. `FSBrick`, in particular, offers an information representation approach for automating fault diagnosis. `FSBrick` is also the first step in creating an adaptive fault diagnosis framework robust to system reconfiguration. This framework has the potential to reduce inaccuracies in automated fault diagnosis methods deployed in commercial building HVAC systems, which will decrease energy waste and increase occupant comfort.

Data availability statement. FSBrick ontology files and the database used to conduct the three applied studies can be found on the github page: https://github.com/INFERLab/FSBrick. The .ttl file stored in this repository can be used in conjunction with the `Brick` ontology to represent HVAC fault-symptom relationships. The three Excel files contain raw data utilized in Section 5: Applied Study. Each file is appropriately labeled as #1, #2, and #3, corresponding to their respective case studies. More details on how to use FSBrick and the data are documented in the `README` associated with this repository.

Acknowledgments. Special thanks to *Brick* subject matter experts, Dr. Gabe Fierro, Eric Paulson, and Connor Cantrell, for providing relevant background reading materials for this research.

Author contribution. Investigation: Min Young Hwang. Writing—Original Draft: Min Young Hwang. Funding Acquisition: Burcu Akinci, Mario Bergés. Writing—Review and Editing: Min Young Hwang, Burcu Akinci, Mario Bergés.

Funding statement. This effort is supported by NASA under grant number 80NSSC19K1052 as part of the Space Technology Research Institute (STRI) Habitats Optimized for Missions of Exploration (HOME) project.

Competing interest. Mario Bergés holds concurrent appointments as a Professor of Civil and Environmental Engineering at Carnegie Mellon University and as an Amazon Scholar. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the individuals thanked, NASA, Amazon, nor Carnegie Mellon University.

Ethical standard. The research meets all ethical guidelines, including adherence to the legal requirements of the study country.

References

Aaseng GB (2015) Scaling up model-based diagnostic and fault effects reasoning for spacecraft. In *AIAA SPACE 2015 Conference and Exposition*. American Institute of Aeronautics and Astronautics. https://doi.org/10.2514/6.2015-4465.

Akcamete A (n.d.) *A Formal Approach for Managing Facility Change Information and Capturing Change History as Part of Building Information Models (BIMs). Ph.*D Dissertation. PA: Carnegie Mellon University.

ASHRAE (*2018*) *Guideline 36-2018. High-Performance Sequences of Operation for HVAC Systems*. American Society of Heating, Refrigerating, and Air-Conditioning Engineers, Inc.

Balaji B, **Bhattacharya A**, **Fierro G**, **Gao J**, **Gluck J**, **Hong D**, **Johansen A**, **Koh J**, **Ploennigs J**, **Agarwal Y**, **Berges M**, **Culler D**, **Gupta R**, **Kjærgaard MB**, **Srivastava M and Whitehouse K** (2016). Brick: Towards a unified metadata schema for buildings. In *Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments*. Palo Alto, CA: ACM, 41–50. https://doi.org/10.1145/2993422.2993577.

Balaji B, **Weibel N and Agarwal Y** (2016) Managing Commercial HVAC Systems: What Do Building Operators Really Need? https://doi.org/10.48550/arXiv.1612.06025.

Boodi A, **Beddiar K**, **Amirat Y and Benbouzid M** (2020) Simplified building thermal model development and parameters evaluation using a stochastic approach. *Energies 13*(11), 2899. https://doi.org/10.3390/en13112899.

Brastein OM, **Lie B**, **Sharma R and Skeie NO** (2019) Parameter estimation for externally simulated thermal network models. *Energy and Buildings 191*, 200–210. https://doi.org/10.1016/j.enbuild.2019.03.018.

Chen R, **Zhou Z**, **Liu Q**, **Pham DT**, **Zhao Y**, **Yan J and Wei Q** (2015) Knowledge modeling of fault diagnosis for rotating machinery based on ontology. In *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*. IEEE, 1050–1055.

Chen X, **Jia S and Xiang Y** (2020) A review: Knowledge reasoning over knowledge graph. *Expert Systems with Applications 141* (2020), 112948.

Chen Y, **Lin G**, **Crowe E and Granderson J** (2021) Development of a unified taxonomy for HVAC System Faults. *Energies 14* (17), 5581. https://doi.org/10.3390/en14175581.

Chen Z, **Xiao F and Guo F** (2023) Similarity learning-based fault detection and diagnosis in building hvac systems with limited labeled data. *Renewable and Sustainable Energy Reviews 185*, 113612. https://doi.org/10.1016/j.rser.2023.113612.

Chi Y, **Dong Y**, **Wang ZJ**, **Yu FR and Leung VCM** (2022) Knowledge-based fault diagnosis in industrial internet of things: a survey. *IEEE Internet of Things Journal 9*(15), 12886–12900. https://doi.org/10.1109/JIOT.2022.3163606.

COBie Guide (n.d.) COBie Guide - Public Release 3 PDF. Available at https://www.bimpedia.eu/static/nodes/1010/COBie_Guide_-_Public_Release_3.pdf.

Cornford SL and Feather MS (2016) *Model Based Mission Assurance in a Model Based Systems Engineering (MBSE) Framework: State-of-the-Art Assessment.* Technical Report JPL-Publ-17-26.

Day J, **Donahue K**, **Ingham M**, **Kadesch A**, **Kennedy A and Post E** (n.d.) Modeling Off-Nominal Behavior in SysML. In *Infotech@Aerospace 2012*. American Institute of Aeronautics and Astronautics. https://doi.org/10.2514/6.2012-2576.

Delgoshaei P and Austin MA (2017) Framework for knowledge-based fault detection and diagnostics in multi-domain systems: application to heating ventilation and air conditioning systems.

Deshmukh S, **Glicksman L and Norford L** (2020) Case study results: Fault detection in air-handling units in buildings. *Advances in Building Energy Research 14*(3), 305–321. https://doi.org/10.1080/17512549.2018.1545143.

Fan C, **Wu Q**, **Zhao Y and Mo L** (2024) Integrating Active Learning and Semi-Supervised Learning for Improved Data-Driven HVAC Fault Diagnosis Performance. *Applied Energy 356*, 122356. https://doi.org/10.1016/j.apenergy.2023.122356.

Fierro G, **Koh J**, **Agarwal Y**, **Gupta RK and Culler DE** (2019) Beyond a house of sticks: Formalizing metadata tags with brick. In *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation (BuildSys '19)*. New York, NY: Association for Computing Machinery, pp. 125–134. https://doi.org/10.1145/3360322.3360862.

Figueroa F, **Walker M and Underwood LW** (2019) NASA Platform for Autonomous Systems (NPAS). In *AIAA Scitech 2019 Forum*. San Diego, CA: American Institute of Aeronautics and Astronautics. https://doi.org/10.2514/6.2019-1963.

Frisk E (n.d.) *Residual Generation for Fault Diagnosis.*

Granderson J and Lin G (n.d.) *Inventory of Data Sets for AFDD Evaluation.*

Hazyuk I, **Ghiaus C and Penhouet D** (2012) Optimal temperature control of intermittently heated buildings using model predictive control: Part I – Building modeling. *Building and Environment 51*, 379–387. https://doi.org/10.1016/j.buildenv.2011.11.009.

House JM, **Vaezi-Nejad H and Whitcomb JM** (2001) An expert rule set for fault detection in air-handling units / discussion. *ASHRAE Transactions 107*, 858.

Hwang MY, **Akinci B and Berges M** (2023) FSBrick: An information model for representing fault-symptom relationships in HVAC systems. In *Proceedings of the 10th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, pp. 69–78.

Hwang MY, **Akinci B and Berges M** (2024) Updating subsystem-level fault-symptom relationships for Temperature and Humidity Control Systems with redundant functions. *Journal of Space Safety Engineering 11*(1), 2–12.

IFCHVACDOMAIN (n.d.) https://standards.buildingsmart.org/IFC/RELEASE/IFC2x3/TC1/HTML/ifchvacdomain/ifchvacdomain.htm.

Infeld SI, **Goggin D**, **Vipavetz K and Grondin T** (2018) A SysML Model Template for NASA Concurrent Engineering Studies. In *AIAA SPACE and Astronautics Forum and Exposition*. Orlando, FL: American Institute of Aeronautics and Astronautics. https://doi.org/10.2514/6.2018-5392.

Isermann R (2005) Model-based fault-detection and diagnosis–status and applications. *Annual Reviews in Control 29*(1), 71–85.

Izygon M, **Wagner H**, **Okon S**, **Wang L**, **Sargusingh M and Evans J** (2016) Facilitating R&M in spaceflight systems with MBSE. In *Annual Reliability and Maintainability Symposium (RAMS)*, pp. 1–6. https://doi.org/10.1109/RAMS.2016.7448031.

Jain P, **Poon J**, **Singh JP**, **Spanos C**, **Sanders SR and Panda SK** (2019) A digital twin approach for fault diagnosis in distributed photovoltaic systems. *IEEE Transactions on Power Electronics 35*(1), 940–956.

Katipamula S and Brambley MR (2005a) Methods for fault detection, diagnostics, and prognostics for building systems—A review, part I. *Hvac&R Research 11*(1), 3–25.

Katipamula S and Brambley MR (2005b) Methods for fault detection, diagnostics, and prognostics for building systems—A review, part II. *Hvac&R Research 11*, 2 (2005), 169–187.

Kim W and Katipamula S (2018) A review of fault detection and diagnostics methods for building systems. *Science and Technology for the Built Environment 24*, 1 (2018), 3–21.

Kircher KJ and Zhang KM (2015) On the Lumped Capacitance Approximation Accuracy in RC Network Building Models. *Energy and Buildings 108*, 454–462. https://doi.org/10.1016/j.enbuild.2015.09.053.

Li B, **Cheng F**, **Cai H**, **Zhang X and Cai W** (2021) A semi-supervised approach to fault detection and diagnosis for building HVAC systems based on the modified generative adversarial network. *Energy and Buildings 246*, 111044. https://doi.org/10.1016/j.enbuild.2021.111044.

Liang J and Du R (2007) Model-based fault detection and diagnosis of hvac systems using support vector machine method. *International Journal of Refrigeration 30*(6), 1104–1114. https://doi.org/10.1016/j.ijrefrig.2006.12.012.

Lin G, **Kramer H and Granderson J** (2020) Building fault detection and diagnostics: Achieved savings, and methods to evaluate algorithm performance. *Building and Environment 168*, 106505. https://doi.org/10.1016/j.buildenv.2019.106505.

Liu X, **Akinci B**, **Garrett JH and Bergés M**. [n.d.]. Requirements and development of a computerized approach for analyzing functional relationships among HVAC components using building information models.

Malin JT and Throop DR (2007) Basic concepts and distinctions for an aerospace ontology of functions, entities and problems. In *2007 IEEE Aerospace Conference*, pp. 1–18. https://doi.org/10.1109/AERO.2007.352806.

Mathur A, **Deb S and Pattipati KR** (1998) Modeling and real-time diagnostics in TEAMS-RT. In *Proceedings of the 1998 American Control Conference. ACC (IEEE Cat. No.98CH36207)*, Vol. *3*, pp. 1610–1614. https://doi.org/10.1109/ACC.1998.707278.

https://doi.org/10.1017/dce.2024.26 Published online by Cambridge University Press

Mirnaghi MS and Haghighat F (2020) Fault detection and diagnosis of large-scale HVAC systems in buildings using data-driven methods: A comprehensive review. *Energy and Buildings 229*, 110492. https://doi.org/10.1016/j.enbuild.2020.110492.

Naderi E and Khorasani K (2018) Inversion-based output tracking and unknown input reconstruction of square discrete-time linear systems. *Automatica 95*, 44–53.

OML Tutorials (n.d.) OML Tutorials. Available at https://www.opencaesar.io/oml-tutorials/.

Pauwels P, **Zhang S and Lee Y-C** (2017) Semantic Web Technologies in AEC Industry: A Literature Overview. *Automation in Construction 73*, 145–165. https://doi.org/10.1016/j.autcon.2016.10.003.

Pradhan O, **Wen J**, **Chen Y**, **Lu X**, **Chu M**, **Fu Y**, **O'Neill Z**, **Wu T and Candan KS** (2021) Dynamic bayesian network-based fault diagnosis for ASHRAE guideline 36: High performance sequence of operation for HVAC systems. In *Proceedings of the 8th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation (BuildSys '21)*. New York, NY: Association for Computing Machinery, pp. 365–368. https://doi.org/10.1145/3486611.3491124.

Qiu A, **Yan Z**, **Deng Q**, **Liu J**, **Shang L and Wu J** (2020) Modeling of HVAC systems for fault diagnosis. *IEEE Access 8*(2020), 146248–146262. https://doi.org/10.1109/ACCESS.2020.3015526.

Sun R, **Hu Z**, **Gowri K and Xu W** (2020) Improving the interoperability of gbXML data model through redefining data mapping rules of HVAC systems.

Svärd C (n.d.) Residual generation methods for fault diagnosis with automotive applications.

Taal A and Itard L (2020) Fault detection and diagnosis for indoor air quality in DCV Systems: Application of 4S3F method and effects of DBN probabilities. *Building and Environment 174*, 106632. https://doi.org/10.1016/j.buildenv.2019.106632.

Turner WJN, **Staino A and Basu B** (2017) Residential HVAC fault detection using a system identification approach. *Energy and Buildings 151*, 1–17. https://doi.org/10.1016/j.enbuild.2017.06.008

United States Energy Information Administration (2023) U.S. Energy Consumption by Source and Sector, 2022. Available at https://www.eia.gov/totalenergy/data/monthly/pdf/flow/total_energy_2022.pdf

Velibeyoglu I, **Noh HY and Pozzi M** (2019) A graphical approach to assess the detectability of multiple simultaneous faults in air handling units. *Energy and Buildings 184*, 275–288. https://doi.org/10.1016/j.enbuild.2018.12.008.

Venkatasubramanian V, **Rengaswamy R**, **Yin K and Kavuri SN** (2003) A review of process fault detection and diagnosis: Part I: Quantitative model-based methods. *Computers & Chemical Engineering 27*, 3 (2003), 293–311.

W3C (2006) Defining N-ary Relations on the Semantic Web. Available at https://www.w3.org/TR/swbp-n-aryRelations/.

Wang L, **Izygon M**, **Okron S**, **Garner L and Wagner H** (2016) Effort to accelerate MBSE adoption and usage at JSC. In *Space 2016 AIAA*, Long Beach, CA.

Xu A and Zhang Q (2004) Residual generation for fault diagnosis in linear time-varying systems. *IEEE Transactions on Automatic Control 49*(5), 767–772.

Xu F, **Liu X**, **Chen W**, **Zhou C and Cao B** (2018) Ontology-based method for fault diagnosis of loaders. *Sensors 18*(3), 729. https://doi.org/10.3390/s18030729.

Yan K, **Ji Z**, **Lu H**, **Huang J**, **Shen W and Xue Y** (2019) Fast and accurate classification of time series data using extended elm: application in fault diagnosis of air handling units. *IEEE Transactions on Systems, Man, and Cybernetics: Systems 49*(7), 1349–1356. https://doi.org/10.1109/TSMC.2017.2691774.

Yan Y, **Luh PB and Pattipati KR** (2018) Fault diagnosis of components and sensors in hvac air handling systems with new types of faults. *IEEE Access 6*, 21682–21696. https://doi.org/10.1109/ACCESS.2018.2806373.

Yu Z, **Haghighat F**, **Fung BCM and Zhou L** (2012) A novel methodology for knowledge discovery through mining associations between building operational data. *Energy and Buildings 47*, 430–440. https://doi.org/10.1016/j.enbuild.2011.12.018.

Zhang M, **Li Z-t**, **Cabassud M and Dahhou B** (2017) Unknown input reconstruction: A comparison of system inversion and sliding mode observer based techniques. In *36th Chinese Control Conference (CCC)*, pp. 7172–7177. https://doi.org/10.23919/ChiCC.2017.8028488.

Zhao Y, **Wen J and Wang S** (2015) Diagnostic bayesian networks for diagnosing air handling units faults – Part ii: Faults in coils and sensors. *Applied Thermal Engineering 90*, 145–157. https://doi.org/10.1016/j.applthermaleng.2015.07.001.

Zhao Y, **Wen J**, **Xiao F**, **Yang X and Wang S** (2017) Diagnostic bayesian networks for diagnosing air handling units faults – part i: faults in dampers, fans, filters and sensors. *Applied Thermal Engineering 111*, 1272–1286. https://doi.org/10.1016/j.applthermaleng.2015.09.121.

Zhou A, **Yu D and Zhang W** (2015) A research on intelligent fault diagnosis of wind turbines based on ontology and FMECA. *Advanced Engineering Informatics 29*(1), 115–125. https://doi.org/10.1016/j.aei.2014.10.001.

Zhu Q-X, **Luo Y and He Y-L** (2019) Novel multiblock transfer entropy based bayesian network and its application to root cause analysis. *Industrial & Engineering Chemistry Research 58*(12), 4936–4945. https://doi.org/10.1021/acs.iecr.8b06392.

Cite this article: Hwang MY, Akinci B and Bergés M (2024). *FSBrick*: an information model for representing fault-symptom relationships in heating, ventilation, and air conditioning systems. *Data-Centric Engineering*, 5, e33. doi:10.1017/dce.2024.26