Ontology-based approach for fault detection and diagnosis and fault location assessment in air handling units

Sebastian Blechmann[1,*], Hannah Görigk[1], Rita Streblow[1] and Dirk Müller[1]

[1]*Institute for Energy Efficient Buildings and Indoor Climate, Mathieustr. 10, 52074 Aachen, Germany*

Keywords
Linked Data, Brick Schema, Fault Detection and Diagnosis (FDD), Fault Location Assessment, Ontology

1. Extended Abstract

Heating, ventilation and air conditioning systems, e.g. air handling units (AHUs), play a key role in modern building energy systems (BES) as they account for a high share of the energy usage, e.g. 30 % within the U.S. commercial building sector [1]. Unfortunately, many faults occur during the operation of AHUs leading to higher energy usage and the need of expert personnel monitoring the according plants. The latter is a time-consuming task and nowadays assisted by fault detection and diagnosis (FDD) systems. A study by Kramer et al. shows median savings of 6 % building energy usage in using FDD systems in the first year after installation of the FDD system [2]. Fault reports usually do not follow a standardized format, there are no consistent naming conventions, and FDD reports can often only be interpreted by FDD developers [3]. This can lead to high maintenance costs because maintenance personnel needs time to get familiar with the building, its energy system, the data point naming conventions and fault report conventions. To reduce the time maintenance personnel needs to understand fault reports and locate possible faults, this work presents a rule-based approach on semantic knowledge in AHUs to detect and locate faults within them.

For this approach, a graph-based ontology for FDD in AHUs is developed. The main characteristics of the proposed FDD ontology are (1) the connection of AHU components, e.g. sensors and actuators, and their possible faults, (2) the faults are characterized by their fault type, e.g. condition-based, behavior-based, or outcome-based, (3) a class for FDD rules that need to be evaluated in the FDD process is created while semantic web rule language (SWRL) rules connect these FDD rules with components of AHUs, and (4) sensors and their faults are connected to a AHU section class allowing faster fault and sensor location.

The developed FDD ontology describes which faults can occur in components, e.g. temperature sensors or valves, modelled with the BrickSchema ontology (Brick) [4]. This fault mapping as well as the fault types are derived from prior work from Chen et al. [3]. As mentioned, possible fault types are condition-based, behavior-based, and outcome-based. Condition-based faults represent "improper or undesired physical condition in a system or piece of equipment" [3], e.g. a frozen sensor or a stuck valve. Behavior-based faults represent "improper or undesired behavior during the operation of a system or piece of equipment" [3], e.g. a too high or too low supply air temperature. Outcome-based faults represent states "in which a quantifiable outcome or performance metric for a system or piece of equipment deviates from a correct or reference outcome, termed the expected outcome", e.g. too high heating demand.

Figure 1 shows examplary data of a supply air temperature sensor in the supply air section and its possible faults which is monitored by rule 1 of the Air Handling Unit Performance Assessment

Bochum'24: LDAC2024 - Linked Data in Architecture and Construction, June 13–14, 2024, Bochum, Germany
*Corresponding author.
sebastian.blechmann@eonerc.rwth-aachen.de (S. Blechmann); hannah.goerigk@eonerc.rwth-aachen.de (H. Görigk); rstreblow@eonerc.rwth-aachen.de (R. Streblow); dmueller@eonerc.rwth-aachen.de (D. Müller)
https://www.ebc.eonerc.rwth-aachen.de/cms/~dmzz/E-ON-ERC-EBC/ (S. Blechmann)
0000-0002-8135-1843 (S. Blechmann); 0000-0001-7640-0930 (R. Streblow); 0000-0002-6106-6607 (D. Müller)

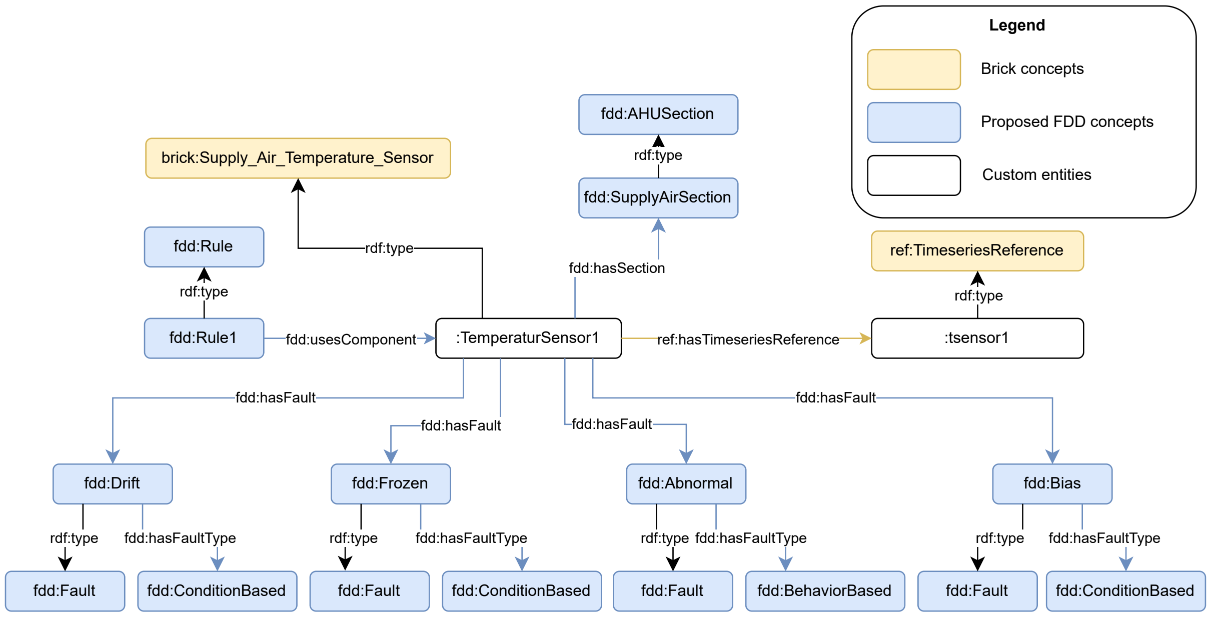

Figure 1: Example data of the proposed FDD ontology with a temperature sensor modelled in Brick. It is connected to its possible faults, rule 1 of the APAR monitoring the sensor data, all modelled with the proposed FDD ontology, and the time series reference modelled in Brick.

Rule 1:
```
FDD:hasSection(?c, FDD:MixedAirSection) ∧
 brick:Mixed_Air_Temperature_Sensor(?c) ∧
 brick:Supply_Air_Temperature_Sensor(?d) ∧
FDD:hasSection(?d, FDD:SupplyAirSection)
⇒ FDD:usesComponent(FDD:rule1, ?d) ∧ FDD:usesComponent(FDD:rule1, ?c)
```

Figure 2: SWRL representation of the APAR rule 1 according to [5].

Rules (APAR) by Schein et al. [5]. Rule 1 verifies whether the supply air temperature T_{sa} is lower than the mixed air temperature T_{ma} plus the temperature change over the supply air fan ΔT_{sf} minus a plant specific factor ϵ. It is expressed in equation 1 as

$$T_{sa} < T_{ma} + \Delta T_{sf} - \epsilon \tag{1}$$

The examplary SWRL rule for rule 1 is depicted in figure 2.

For the assessment of rule 1, temperature data for a mixed air temperature sensor modelled as *brick:Mixed_Air_Temperature_Sensor* and a supply air temperature sensor modelled as *brick:Supply_Air_Temperature_Sensor* need to be queried. If these components are present in the knowledge graph, a relationship between rule 1 and the sensors is added to the knowledge graph. The temperature change over the fan ΔT_{sf} and the plant-specific parameter ϵ are parameters of the Python code.

During the development of the proposed FDD ontology, a similar, independently developed, approach was published by Hwang et al. [6]. While Hwang et al. follow the same approach in modelling faults to certain brick components, they do not use standardized naming conventions or fault types and do not model AHU sections to locate the fault. Furthermore, faults like malfunctioning, stuck, leakage, fouling, and block are modelled for components like valves, coils, pumps, dampers and fans in [6], yet,common sensor faults are not modelled. The proposed FDD ontology in this paper combines the idea of formal fault mapping of common sensor faults following structured naming conventions assisting maintenance personnel to locate faults easier.

The information about the actual plants, their configuration, sensors, actuators, and controls are stored in a knowledge graph in a graph database. The FDD program is written in Python language running in a loop. During each run, it loops through all FDD rules and verifies whether all components needed to evaluate a rule are present in the knowledge graph. If a component needed for a certain rule is missing the evaluation for that rule is skipped. At the current stage of implementation, the implemented rules are based on the APAR. In the evaluation process of a rule, the FDD program follows the *brick:hasTimeseriesReference* property for every component towards the URI where the time series data is stored, fetches and evaluates it. In that process, the FDD program verifies whether the rule's statement is valid and if the fetched time series data shows any anomalies in accordance to the components' possible faults, e.g. whether the time series data for a sensor is frozen.

In this work, data from a real AHU is sent to a cloud platform developed in previous work every second via the message queuing telemetry transport (MQTT) protocol [7]. To validate the framework, we manipulated the real time series data of the AHU and the FDD program checks the data every 30 minutes. Depending on whether a fault is present or not, fault messages are gathered in a list and sorted according to their incidence. The incidence is calculated with the cumulative sum control chart (CUSUM) in order to allow a certain prioritization for maintenance personnel. The fault messages include the fault name that is composed by the section of the equipment type, in this case AHU, the AHUSection, the component type and the fault nature, e.g. frozen, all according to Chen et al. [3]. All this information is retrieved through queries to the graph database. The notation allows maintenance personnel to locate the faults fast.

The developed approach is validated using historic data of a real AHU on our premises. We injected faults into the historic data, e.g. drift, stuck, bias, fouling, abnormal, and frozen.

Before injecting the faults, we carried out a baseline experiment in which pre-existing faults in the time series data of the AHU were identified. These faults were excluded from the analysis in the subsequent tests where faults have been injected. The isolation of the injected faults allowed a more in-depth analysis of the fault detection rate without interference of pre-existing faults. Furthermore, the manipulation of the time series data allowed verifying the FDD algorithm for all implemented faults in different configurations and combinations for the same time period, thus providing better fault isolation results. The injected faults are pre-defined based on fault-specific characteristics, e.g. a frozen sensor is mimicked by repeating the last value at a certain point in time in the time series data untill the end of the analyzed time period. A frozen sensor itself is identified as such if the time series data are constant for more than 80 % of a certain time period, e.g. 30 minutes. Different combinations of faulty data were injected and analyzed, subsequently, the faults were isolated and the fault report, as shown in table 1, is created or supplemented. The found isolated faults are compared to the injected faults and 2x2 confusion matrix are used to determine the quality of the FDD algorithm.

Table 1
Examplary fault report
(BB = *behavior-based*, CB = *condition-based*)

N	time	sensor	fault	type	section	cusum	faulty values
1	13:00	Sensor1	Abnormal	BB	SupplyAirSection	0.7	'value': 18.4, 'setpoint': 21
2	13:00	Sensor2	Drift	CB	CondenserSection	0.7	'value': 10.4, 'setpoint': 13
1	13:30	Sensor1	Abnormal	BB	SupplyAirSection	1.4	'value': 18.5, 'setpoint': 21
2	13:30	Sensor2	Drift	CB	CondenserSection	1.4	'value': 10.3, 'setpoint': 15

The results indicate a high rate of fault detection, e.g. average accuracy of 99.3 %, precision of 97.85 %, sensitivity of 95.6 %, and Matthews Correlation Coefficient of 98.4 %. Although the FDD algorithm checks for self-injected faults, the FDD algorithm does not deliver perfect results. This is due to the fact that some parameters, e.g. the ϵ of rule 1, are not chosen perfectly. Yet, the approach is considered valid for FDD in the presented framework with self-injected faults as the majority of the injected faults is found and the fault reports are created correctly to allow fast fault location assessment and fault prioritization through the use of CUSUM.

In the future, it is planned to implement more complex faults, such as control faults. Furthermore, the semantic enriched plant configuration, fault types and fault locations shall be used to discover fault-symptom relationships. Additionally, the implemented APAR rules in the FDD algorithm are currently just looking for the exact implementation of component characteristics as depicted in the original APAR rules. E.g. rule 1 is only checking for supply air temperature sensors modelled as *brick:Supply_Air_Temperature_Sensor* and mixed air temperature sensors as *brick:Mixed_Air_Temperature_Sensor*. This means that the knowledge graph needs to be modelled exactly in the same way as the FDD rules are implemented. If modelled differently, the FDD algorithm will skip certain rule validations. The approach of computational quantities and decision trees to find component characteristics modelled differently, e.g. a supply air temperature sensor modelled as *brick:Air_Temperature_Sensor* in the supply air modelled as *fdd:SupplyAirSection*, by Mavrokapnidis et al. [8] can help to expand the use of the proposed framework.

Formalizing the relationship between components of AHUs and faults, fault types, and locations in machine and human readable format bear huge advantages in FDD and resolving faults in AHUs by assisting maintenance personnel with standardized semantic-enriched fault reports.

Yet, further investigation on the proposed approach is needed to quantify the benefits, e.g. maintenance time and cost.

References

[1] W. Goetzler, R. Shandross, J. Young, O. Petritchenko, D. Ringo, S. McClive, Building Technologies Office Corporate, Energy Savings Potential and RD&D Opportunities for Commercial Building HVAC Systems, Technical Report DOE/EE–1703, 1419622, 2017. URL: http://www.osti.gov/servlets/purl/1419622/. doi:10.2172/1419622.

[2] H. Kramer, G. Lin, C. Curtin, E. Crowe, J. Granderson, Proving the Business Case for Building Analytics (2020). URL: https://escholarship.org/uc/item/5m66941j. doi:10.20357/B7G022.

[3] Y. Chen, G. Lin, E. Crowe, J. Granderson, Development of a Unified Taxonomy for HVAC System Faults, Energies 14 (2021) 5581. URL: https://www.mdpi.com/1996-1073/14/17/5581. doi:10.3390/en14175581, number: 17 Publisher: Multidisciplinary Digital Publishing Institute.

[4] B. Balaji, A. Bhattacharya, G. Fierro, J. Gao, J. Gluck, D. Hong, A. Johansen, J. Koh, J. Ploennigs, Y. Agarwal, M. Berges, D. Culler, R. Gupta, M. B. Kjærgaard, M. Srivastava, K. Whitehouse, Brick: Towards a Unified Metadata Schema For Buildings, in: Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments, ACM, Palo Alto CA USA, 2016, pp. 41–50. URL: https://dl.acm.org/doi/10.1145/2993422.2993577. doi:10.1145/2993422.2993577.

[5] J. Schein, S. T. Bushby, N. S. Castro, J. M. House, A rule-based fault detection method for air handling units, Energy and Buildings 38 (2006) 1485–1492. URL: https://linkinghub.elsevier.com/retrieve/pii/S0378778806001034. doi:10.1016/j.enbuild.2006.04.014.

[6] M. Y. Hwang, B. Akinci, M. Berges, 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, BuildSys '23, Association for Computing Machinery, Istanbul, Turkey, 2023, pp. 69–78. URL: https://dl.acm.org/doi/10.1145/3600100.3623729. doi:10.1145/3600100.3623729.

[7] S. Blechmann, I. Sowa, M. H. Schraven, R. Streblow, D. Müller, A. Monti, Open source platform application for smart building and smart grid controls, Automation in Construction 145 (2023) 104622. URL: https://www.sciencedirect.com/science/article/pii/S0926580522004927. doi:https://doi.org/10.1016/j.autcon.2022.104622.

[8] D. Mavrokapnidis, G. Fierro, I. Korolija, D. Rovas, A Programming Model for Portable Fault Detection and Diagnosis, in: Proceedings of the 14th ACM International Conference on Future Energy Systems, ACM, Orlando FL USA, 2023, pp. 127–131. URL: https://dl.acm.org/doi/10.1145/3575813.3595190. doi:10.1145/3575813.3595190.