Fault Diagnosis Services and Realistic Fault Models for HVAC Systems

DISSERTATION

zur Erlangung des Grades eines Doktors
der Ingenieurwissenschaften (Dr.-Ing.)

Vorgelegt von
M.Sc. Bahareh Kiamanesh

Eingereicht bei der Naturwissenschaftlich-Technischen Fakultät
der Universität Siegen
Siegen, 06. Juni 2023

Betreuer und erster Gutachter
betreut von
Prof. Dr.-Ing. habil. Roman Obermaisser
Universität Siegen

Zweiter Gutachter und Prüfer
Prof. Dr. Raimund Kirner
Universität Hertfordshire

Prüfungskommission
Prof. Dr. Malte Lochau
Prof. Dr. Kristof Van Laerhoven
Universität Siegen

Tag der mündlichen Prüfung
08. Dezember 2023

Fault Diagnosis Service for the Realistic Fault Model in HVC Systems

DISSERTATION
To obtain the degree of doctor
of science engineering

Submitted by
M.Sc. Bahareh Kiamanesh

Submitted to the Faculty of Natural Sciences and Technology
At the University of Siegen
Siegen, 06.06.2023

Acknowledgment

First and foremost, I would like to express my special thanks and sincere gratitude to my scientific advisor Professor Roman Obermaisser for his valuable support and patience in my studies. His constant dedication, guidance, and advice carried me through all the stages of writing my doctoral thesis. Besides, I would like to express my appreciation to Dr. Ali Behravan for providing the basic simulated HVAC system model. Besides, immeasurable appreciation for the help and support are extended to my thesis committee members Professor Raimund Kirner, Professor Kristof Van Laerhoven, and Professor Malte Lochau. I want to acknowledge my respect for their valuable encouragement, insightful comments, and questions. I deeply appreciate and express my respect to my lovely mother, father, and brother for their unconditional support, and love in any circumnutates of my life. Finally, I would like to thank my lovely god for letting me through all the difficulties and express that you were the only one who let me finish my degree.

Bahareh Kiamanesh
Siegen University

Declaration of Authorship

I hereby declare that I am the sole author and composer of this thesis entitled “Fault Diagnosis Services and Realistic Fault Models for HVAC Systems”, and the work contained herein is presented entirely on my own. Where I have consulted the work of others this is always clearly stated. All statements taken literally from other writings or referred to by analogy are marked and their resources are always given. This has been clearly stated where any part of the thesis has previously been submitted for a degree or any other qualification at this University or any other institution. I further declare that I have not submitted this thesis at any other institution to obtain a degree.

List of Publications of this Dissertation

Bahareh Kiamanesh, Ali Behravan, and Roman Obermaisser, “Realistic Simulation of Sensor/Actuator Faults for a Dependability Evaluation of Demand-Controlled Ventilation and Heating Systems”, Energies Journal, Special Issue: Fault Identification and Fault Impact Analysis of Ventilation System in Buildings, vol. 15, no. 8, p. 2878, 2022. DOI: 10.3390/en15082878.

Bahareh Kiamanesh, Ali Behravan, and Roman Obermaisser, “Fault Injection with Multiple Fault Patterns for Experimental Evaluation of Demand-Controlled Ventilation and Heating Systems”, Sensors Journal, Special Issue: Special Issue Sensing Technologies for Fault Diagnostics and Prognosis, vol. 22, no. 21, p. 8180, 2022. DOI: 10.3390/s22218180.

Ali Behravan, Bahareh Kiamanesh, and Roman Obermaisser, "Fault Diagnosis of DCV and Heating Systems based on Causal relations in Fuzzy Bayesian Belief Networks using Relation Direction Probabilities", Energies Journal, Special Issue: Fault Identification and Fault Impact Analysis of Ventilation System in Buildings, vol. 14, no. 20, p. 6607, 2021. DOI: 10.3390/en14206607.

Ali Behravan, Bahareh Kiamanesh, Siddharth Bhandari, and Roman Obermaisser, “Component-Based System Model Design of Multiple-Fault Injection Framework for DCV and Heating Systems”, ESCS'23 - The 21st Int'l Conf on Embedded Systems, Cyber-physical Systems, and Applications, Accepted in May 2023.

Table of Contents

Kurzbeschreibung

Heizungs-, Lüftungs- und Klimasysteme (HLK) sind große, verteilte Systeme mit elektronischen Komponenten, einschließlich Steuerungen, Sensoren und Aktoren, die koordiniert werden müssen, um das beabsichtigte Verhalten zu erreichen. Daher sind HLK-Systeme anfällig für Einzel- und Mehrfachfehler, die sich auf die Elektronik auswirken und einen hohen Energieverbrauch, Unbehagen bei den Bewohnern, eine verschlechterte Raumluftqualität, schlechte thermische Bedingungen und Risiken für kritische Infrastrukturen verursachen können. Darüber hinaus spielen HLK-Systeme in großen kritischen Infrastrukturen in Notfällen eine wesentliche Rolle. Notfallreaktionen erfordern Echtzeitreaktionen, Konsistenz und Fehlertoleranz. Fehlertoleranz ist sowohl bei Betriebs- als auch bei Konstruktionsfehlern unerlässlich. In der Entwicklungsphase fehlertoleranter Systeme ist die Simulation eine gängige Technik, um Einblicke in die Systemfunktionalität, Leistung und Zuverlässigkeit zu erhalten. Sie spart Zeit, senkt die Kosten und vermeidet Risiken, die mit der Durchführung von Tests bei Vorliegen von Fehlern in realen Systemen verbunden sind. Infolgedessen ist die Fehlerinjektion in Simulationsumgebungen eine effektive experimentelle Methode zur Validierung und Bewertung der Zuverlässigkeit von HLK-Systemen.

Die Fehlerinjektion in einer Simulation bietet eine hohe Kontrollierbarkeit und Beobachtbarkeit. Sie ist daher ideal für eine frühzeitige Zuverlässigkeitsanalyse und Fehlertoleranzbewertung. HLK-Systeme in kritischen Infrastrukturen sind sicherheitsrelevante Systeme, die eine angemessene Belüftung und Klimatisierung für die Bewohner gewährleisten sollen. Dementsprechend wird in dieser Arbeit ein simulationsbasiertes Framework zur Fehlerinjektion mit einer Kombination aus zwei Techniken (d.h. Simulatorbefehle und Modifikation des Simulationscodes) mit realistischen Fehlermustern, vorgeschlagen und als generisches und erweiterbares Framework eingeführt. Das Framework zur Fehlerinjektion ist mit Simulationsmodellen anderer elektronischer Komponenten über Ports verbunden. Das Framework zur Fehlerinjektion wurde in einer komponentenbasierten Struktur entwickelt und in MATLAB/Simulink unter Verwendung von Stateflow-Diagrammen mit fehlerfreien und fehlerhaften Systemzuständen simuliert. Zur Bestimmung der Fehlerattribute und des Fehlerortes wird ein automatischer Fehlerinjektionsalgorithmus vorgeschlagen und mit einem Algorithmus zur Generierung von Systemmodellen integriert. Die Systemstruktur ist anpassungsfähig und die Parameter wie die Anzahl der Stockwerke und die Anzahl der Räume auf jedem Stockwerk werden auf der Grundlage der Systemanforderungen definiert. Ein automatisierter Algorithmus zur Injektion von Einzel- und Mehrfachfehlern unterstützt ein umfassendes Spektrum von Fehlern mit den entsprechenden Fehlerattributen, einschließlich Fehlertyp, Zeitpunkt, Ort, Dauer, Intervallzeit und Häufigkeit des Auftretens. Zur Validierung der Fehlerinjektion wird ein szenariobasierter Ansatz verwendet, um die Auswirkungen auf das System und die Qualität der Dienste zu untersuchen. Jedes Szenario besteht aus mehreren Ereignissen und Unterereignissen, die zu mehreren Fehlerinjektionen führen. Der Rahmen für die Fehlerinjektion berücksichtigt ein realistisches Fehlermodell, das weißes Rauschen mit Gaußscher Verteilung als Signalunsicherheiten hinzufügt, und die Reproduzierbarkeit für eine Reihe von spezifischen Fehlerszenarien und für zufällige Fehlerinjektionsszenarien unterstützt. Das Framework umfasst ein mehrdimensionales Fehlermodell und bietet Kompatibilität zu einer Vielzahl anderer Simulationskomponenten. Die experimentellen Ergebnisse der Komponenten mit einfacher und mehrfacher Fehlerinjektion zeigen die Korrektheit, das Systemverhalten, die Genauigkeit und andere Systemparameter wie den Energieverbrauch des Heizgeräts und die Einschaltdauer des Heizgeräts bei Vorliegen verschiedener Fehlerfälle. Die experimentellen Ergebnisse dienen zur quantitativen Bewertung der wichtigsten Leistungsindikatoren wie Energieeffizienz,

Luftqualität und thermischer Komfort. So wirkt sich beispielsweise die Kombination eines Fehlers des CO_2-Sensors mit einem Fehler des Heizungsaktuators um mehr als 70 % auf den Energieverbrauch aus.

Darüber hinaus wird in dieser Arbeit ein neuartiges und allgemeines Fehlerdiagnoseverfahren, das auf der Konstruktion eines Fuzzy Bayesian Belief Network basiert, in ein simuliertes Systemmodell als Überwachungsansatz integriert, um die Ursachen für fehlerhafte Vorgänge auf der Grundlage von Systembeobachtungen und Messungen zu ermitteln. Es wird auch ein datengesteuerter Klassifizierungsalgorithmus vorgeschlagen, der mit wissensgesteuerten Methoden einschließlich der Fuzzy-Theorie und Bayesian Belief Networks kombiniert werden kann und eine genaue Fehlerdiagnose in HLK-Systemen ermöglicht. In dieser Arbeit reduziert der datengesteuerte Ansatz den Zeitaufwand durch Automatisierung und Klassifizierung auf der Grundlage automatischer Ranking-Methoden. Die Fuzzy-Theorie stützt sich auf Überlegungen zu den Unsicherheiten und unterteilt die Systemattribute in mehrere Teilbereiche, um die Wahrscheinlichkeitsberechnungen für kontinuierliche Systemattribute über geeignete Zugehörigkeitsfunktionen auf der Grundlage der Systemspezifikationen zu erleichtern. Die Wahrscheinlichkeiten werden verwendet, um das Bayesian Belief Network auf der Grundlage der Korrelationen der fuzzifizierten Systemattribute unter Verwendung der Theorie der gegenseitigen Information zu konstruieren. Die gegenseitige Information für alle Paare von fuzzifizierten Subdomänen muss berechnet werden, und ein positiver Wert der gegenseitigen Information ist ein Indikator für eine starke Abhängigkeit zwischen zwei Subdomänen. Schließlich unterstützt die Fehlerinjektion die Fehlerdiagnose-Technik, um verschiedene Fehlerfälle zu definieren und die fehlerhaften Ausgangsdaten als Zeitreihe zu erzeugen, die alle gesunden und fehlerhaften Systemmessungen enthält. Der Fuzzy Bayesian Belief Network Algorithmus spezifiziert die strengen Beziehungen, die Richtung und die Wahrscheinlichkeitsmerkmale aller fuzzifizierten Teilbereiche unter Verwendung der erzeugten Zeitreihen durch Injektion der verschiedenen Fehlerfälle.

Die hybride Fehlerdiagnosetechnik verwendet einen datengesteuerten Klassifikator in Kombination mit der Inferenz von Fuzzy-Logik und einem Bayesian Belief Network im Offline- und Online-Modus. Im Offline-Modus wird eine Offline-Bibliothek auf der Grundlage von gerichteten Wahrscheinlichkeitsbeziehungen von Teilbereichen trainiert. Im Online-Modus werden die ähnlichsten Fehler in der Offline-Bibliothek mit den tatsächlichen Fehlerfällen auf der Grundlage der Korrelation von Systemattributen und der Ranking-Methode ermittelt. Die Ergebnisse zeigen eine hohe Genauigkeit bei der Diagnose von permanenten Fehlern in verschiedenen Komponenten von HLK-Systemen.

Abstract

Heating, Ventilation, and Air-Conditioning (HVAC) systems are large-scale distributed systems comprising distributed components, including controllers, sensors, and actuators that must be coordinated to establish the intended behavior. Therefore, HVAC systems are subject to single and multiple faults affecting the electronics, potentially causing high energy consumption, occupant discomfort, degraded indoor air quality, thermal conditions, and risk to critical infrastructures. In addition, in large-scale critical infrastructures, HVAC systems serve an essential role in emergencies. Emergency reactions demand real-time response, consistency, and fault tolerance. Fault tolerance is essential for both operational faults and design faults. In the development phase of fault-tolerant systems, simulation is a common technique to obtain insights into system functionality, performance, and dependability. It saves time, reduces cost and avoids risks of carrying out tests in the presence of faults in real-world systems. As a result, fault injection in simulation environments is an effective experimental method to validate and evaluate the dependability of HVAC systems. Fault injection in a simulation offers high controllability and observability. It is thus ideal for an early dependability analysis and fault-tolerance evaluation. HVAC systems in critical infrastructures are safety-relevant systems that should guarantee adequate ventilation and air conditions for occupants.

Accordingly, in this thesis, a simulation-based fault injection framework with a combination of two techniques, simulator command and simulation code modification with realistic fault patterns is proposed and introduced as a generic and extendable framework. The fault-injection framework is integrated and connected to simulation models of other electronic components via the connection of ports. The fault injection framework is developed in a component-based structure, implemented and simulated in MATLAB/Simulink using Stateflow diagrams with healthy and faulty system states. To determine the fault attributes and the fault location, an automated fault injection algorithm is proposed and integrated with a system-model generation algorithm. The system structure is adaptable and its parameters such as the number of floors and the number of rooms on each floor are defined based on the system requirements. An automated single/multiple fault injection algorithm triggers faults and supports a comprehensive range of faults with corresponding fault attributes including the fault type, time, location, persistence, duration, interarrival time and occurrence incidence. To validate the fault injection framework, a scenario-based approach is used to study the system impact and quality of the services. Each scenario consists of multiple events and subevents that result in multiple fault injections. The fault injection framework considers a realistic fault model adding white noise with Gaussian distribution as signal uncertainties and it supports reproducibility for a set of specific fault scenarios and for random fault injection scenarios. The framework incorporates a multi-dimensional fault model and provides compatibility to a wide range of other simulation components. The experimental results of single and multiple fault injection components show the correctness, the system behavior, accuracy, and other system parameters, such as the heater energy consumption and heater duty cycle in the presence of different fault cases. The experimental results serve as a quantitative evaluation of key performance indicators (KPI) such as energy efficiency, air quality, and thermal comfort. For example, combining a CO_2 sensor fault with a heater actuator fault impacts energy consumption significantly by more than 70%.

Furthermore, in this thesis a novel and generic fault diagnostic technique based on the Fuzzy Bayesian Belief Network (FBBN) construction is integrated with a simulated system model as a monitoring approach to determine the causes of faulty operations based on system observations and measurements. A data-driven

classifier algorithm is also proposed to be combined with knowledge-driven methods, including fuzzy theory and Bayesian belief networks, enabling accurate fault diagnosis in HVAC systems. In this thesis, the data-driven approach reduces time consumption through automation and classification based on automated ranking methods. The fuzzy theory relies on reasoning about the uncertainties and divides the system attributes into several subdomains to facilitate the probability calculations for continuous system attributes via proper likelihood membership functions based on the system specifications. The probabilities are used to construct the Bayesian belief network based on the correlations of the fuzzified system attributes using mutual information theory. Mutual information for all pairs of fuzzified subdomains must be calculated and a positive value of the mutual information is an indicator of a strong dependency between two subdomains. Eventually, fault injection supports the fault diagnosis technique to define different fault cases and produce the faulty output data as a time series, including all healthy and faulty system measurements. The FBBN algorithm specifies the stringent relations, direction, and probability features of all fuzzified subdomains using the produced time-series by injecting the different fault cases. The hybrid fault diagnostic technique uses a data-driven classifier in combination with fuzzy logic theory and a Bayesian Belief Network in offline and online modes. Offline mode trains an offline library based on relation-direction-probability relationships of subdomains. Online mode determined the most similar faults in the offline library with actual fault cases based on the correlation of system attributes and the ranking method. The results show high accuracy of diagnosing permanent stuck-at fault in different HVAC system components.

1 Introduction

Buildings are responsible for 40% of global energy usage and contribute 30% of the total Carbon Dioxide (CO_2) emissions [1]. Typically, 20–30% of energy savings in buildings are achievable by recommissioning the Heating, Ventilation, and Air-Conditioning (HVAC) systems to rectify faulty operations [2–4]. HVAC systems are the main reason for global energy dissipation, and CO_2 emissions [5]. For example, the construction and maintenance of building stock are responsible for 36% of the CO_2 emission in the European Union (EU) [6]. In 2018, the building, construction, and processes represented 36% of energy consumption, 39% of energy-related CO_2 emissions, and 50% of global electricity consumption [5, 7]. HVAC systems include the air-handling unit for heating, cooling, and ventilation, aiming for more indoor air quality, comfort, and optimized energy consumption. Demand Controlled Ventilation (DCV) helps HVAC systems as a control strategy to modify and balance the amount of indoor air by controlling and adjusting the damper actuator statuses according to the sensor measurements and nominal values, e.g., CO_2 sensor concentration and temperature sensor measurements. HVAC systems need such specific measurements to keep and ensure proper functionalities. The measurement system requires testing, adjusting, and balancing processes; e.g., the temperature and environmental control systems should be evaluated regularly [8]. This process comprises required tests and monitoring of the temperature, airflow, and other specifications of the HVAC systems. These processes can be applied in both new and existing HVAC systems. It includes factors such as airflow quantities, pressure levels, proper operation and sequencing of the automatic control systems, fan speed, and temperature control system operation [8, 9]. These primary measurement activities are defined as: testing that establishes the quantitative aspects, e.g., the volume of the airflow or heat transfer rate, adjusting that establishes alternations to the system components to achieve proper design requirements, such as changing the temperature settings and balancing that ensures system specifications are equalized among all terminals, and sub-systems, e.g., balancing heat transfer in an indoor environment [8]. In critical infrastructures such as airports and hospitals, HVAC systems play a prominent role in emergency scenarios (e.g., fires and biological hazards). These infrastructures require fault-tolerance strategies for a more reliable and safe HVAC system. In addition, faults in HVAC systems can cause temperature fluctuations, occupancy discomfort, excess ventilation, and overheating. Fault management is a significant component of a Building Management System (BMS) for mitigating faults and their high-level symptoms [10, 11]. For example, Teraoka et al. [10] proposed a fault management framework, BDSher-lock, based on two lists. One list comprises standard checks, and the other contains rules based on anomalies. They use data-driven analysis techniques to investigate the energy impact of the detected faults on the HVAC system.

Simulation is a common technique in early development phases to develop fault-tolerant systems. Simulation is a convenient method to investigate faulty behavior and environments for getting insights into system functionality and fault-tolerant operations, e.g., fault diagnosis [12]. The simulation saves development time and decreases the risks of the system's testing in the real-world under faulty conditions. Simulation models should be validated to increase the system's confidence and accuracy [13]. Therefore, Verification and Validation (V&V) tasks are required elements of each simulation study [14]. Model V&V decreases the cost and associated risks of real-world systems and products [15]. Precision, accuracy, and reproducibility are some example aspects of model validation. Since faults are rare events, faults can be deliberately inserted in simulations. Fault Injection (FI) in a simulation environment offers high controllability and observability. Thus, FI is an ideal experimental method for an early dependability

analysis and fault-tolerance evaluation in complex systems such as HVAC [16]. Dependability analysis is an essential aspect of fault-tolerant systems to improve system efficiency, response time, real-world implementation accuracy, fault handling, and critical conditions avoidance. In addition, the effects of fault scenarios and their impacts can be investigated using FI, e.g., heating cost and CO_2 concentration rates. Considering the fault occurrence probability and probability distributions for signal generation makes the FI more realistic [11, 16]. The complexity of HVAC systems increases by enlarging the system scales and the number of components that causes more susceptibility to faults and a significant increase in energy waste by up to 50%. An example is the steady operation of the heating actuator in the case of a stuck-at-fault in the damper actuator [17, 18]. ASHRAE projects have introduced faults commonly associated with HVAC systems, such as common faults of the Air Handling Unit (AHU) [4, 19, 20]. The number of faults and their diversity increases in distributed buildings. Therefore, it is necessary to provide composable simulation models with FI to investigate fault impacts on a large scale and the ensuing dependability [21]. To ensure the dependability of processes in complex systems, faults should be detected, diagnosed, and removed, which is known as process monitoring [22]. The main aim of process monitoring is achieving the proper functionality in the presence of faults by recognizing anomalies that lead to downtime minimization, plant safety improvement, and cost reduction. Process monitoring comprises four main steps: fault detection, identification, diagnosis, and recovery [22]. Fault detection refers to whether a fault has occurred or not. Fault identification concentrates on plant and subsystem observations to identify the fault effects. Fault diagnosis determines which faults have occurred, their localizations, and the causes of the observed statuses [12, 22].

Furthermore, Fault Detection and Diagnosis (FDD) techniques are essential in safety-critical applications. Advanced designs of complex and critical infrastructures require FDD methods with short latencies to avoid threatening situations [23–25]. For instance, the faults in HVAC systems can be the reason for excess pollutant emissions, e.g., Carbon Monoxide (CO) or CO_2 emissions that cause occupant discomfort and danger to human life [26, 27]. FDD methods for HVAC systems have been introduced since the 1970s [12]. Many authors have provided concepts of FDD techniques. Lan et al. [12] have defined FDD as an investigation of the cause of the conditions or problems, including two stages fault detection and fault diagnosis. They have categorized FDD techniques into two categories, model-based and model-free. A model is a mathematical description of a system, and its details, such as component data, system theory, and processes, require deep expert knowledge [12]. Efficient and accurate FDD techniques in HVAC systems can detect faults before being noticed by occupants, thus significantly decreasing the maintenance cost, repair time, and energy consumption [28]. Automated Fault Detection and Diagnosis (AFDD) methods automatically activate the FDD techniques in online mode upon faulty conditions, increase accuracy, and decrease maintenance costs significantly [18, 25, 29]. There are various types of FDD classifications and methods. Abid et al. [24] gave a general overview of FDD techniques. Most of the FDD methods in the literature are model-based and require explicit mathematical models. The model's accuracy is essential in these methods, which is suitable for small-scale systems with few input and output variables and processes. Unmodeled approaches are known as data-driven approaches. These methods extract features from system measurements (i.e., signals) to diagnose the faults and, finally, the features for fault classification. Signal-based approaches are categorized into two categories: statistics and non-statistics [24, 30, 31]. Abid has mentioned statistical approaches as effective and quick methods [24]. With the growing scale and complexity of the systems, designing the system model with mutual interaction between subsystems and developing the automatic hybrid FDD techniques is a significant challenge. Techniques should be included

in a unified, generic, and qualified framework with suitability for different application domains and target systems [24].

1.1 Thesis Motivation and Objectives

HVAC systems are constructed from heterogeneous and numerous components, including sensors, actuators, communication links, and processes. The occurrence of faults is probable in these complex systems, leading to different consequences. Based on the literature review, we identified major research areas such as developing control strategies using FI for dependability evaluation, experimental evaluation and impact analysis of faults, and hybrid AFDD methods in HVAC systems to deal with fault conditions. These areas significantly affect the repair and maintenance cost and time, indoor thermometric efficiency, poisonous gas emissions, occupancy comfort and safety, system efficiency, system performance, and energy-related attributes such as heating cost and energy consumption. Furthermore, business requirements must be considered, such as inexpensive real-time embedded devices for homes and offices as well as safety-critical infrastructures. To achieve these goals, the thesis addresses the following objectives:

Objective 1: Accurate system modeling with low effort for FDD techniques
The definition of system models and diagnostic models should involve low effort and minimize the need for expert knowledge. For example, using the combination of Bayesian belief networks and fuzzy logic facilitates this goal.

Objective 2: Simulation-based reliability evaluation at early development phases and FI without damage to physical systems
Assured validation results for safety arguments are required**.** This is important because the proposed models and algorithms need to support critical infrastructures such as hospitals and airports. Real-time support, fault tolerance, and consistency must be ensured as well.

Objective 3: Diagnosis with high accuracy
High accuracy of the FDD method is essential in critical infrastructures such as hospitals and airports to ensure the safety, availability, and reliability of the HVAC systems. The system must be able to detect faults precisely with a minimum rate of false positives and false negatives.

Objective 4: Universality and scalability
The goal is to have a generic framework to be integrated with different target systems easily. The proposed system model, algorithms, and FDD techniques should be generic, compatible, transferable, and scalable to other critical and complex infrastructures.

1.2 Thesis Problem Statement

According to the literature review and research gap analysis, the following problems have not been solved in DCV and heating systems.

Problem 1: A realistic simulation-based fault injection framework for DCV and heating systems with a comprehensive fault model that comprehensively covers fault attributes such as type, persistence, injection time, duration, interarrival time, fault occurrence probability, and the number of repetitions in case of intermittent faults for sensors and actuators. A FI framework must activate both single and multiple faults. The FI blocks should be compatible with different target systems and support integration with different applications with low effort.

Problem 2: A composable system model that enables multiple-fault injections in simulated DCV and heating systems to validate fault-tolerance requirements by activating realistic fault scenarios for different system configurations, e.g., multiple floors, rooms, and components.

Problem 3: Experimental evaluation of simulated DCV and heating systems for realistic fault scenarios to investigate the system model's accuracy and correctness. Moreover, realistic experimental data of simulated DCV and heating systems serve for the validation of fault management techniques. The data output (i.e., time series) should be collected from data-collector simulation blocks and include comprehensive information about fault conditions under different fault management approaches, e.g., FDDs such as signal-based methods.

Problem 4: A quick, accurate, and automated hybrid FDD technique for DCV and heating systems that combines the advantages of both knowledge-based and data-driven methods.

1.3 Thesis Contribution

This thesis contains four primary contributions: (1) reliability evaluation using fault injection in simulation, (2) experimental evaluation for realistic scenarios with fault injection, (3) integration of composable models with multiple-fault injection frameworks, and (4) diagnostic services to identify faults in HVAC systems. The contributions and their detailed sub-contributions are described as follows.

1.3.1 Reliability Evaluation Using Fault Injection in Simulation

In this thesis, realistic Automated Single-Fault Injection Framework (ASFIF) and Automated Multiple-Fault Injection Framework (AMFIF) are developed to evaluate the reliability of DCV, and heating systems based on different fault scenarios. Associated thesis contributions are explained in 1.3.1.1 and 1.3.1.2 in detail.

1.3.1.1 Single-Fault Injection Framework

A realistic single fault injection framework has been developed with a comprehensive fault model containing the following contributions:

Developing a realistic fault injection framework in DCV and heating system model: A novel simulation-based Fault Injection Framework (FIF) combining simulator command techniques and

simulation code modifications is developed for a realistic and automatic FI in which faults are triggered with an automated fault injection algorithm [11].

Comprehensive fault patterns for single-fault injection in simulated DCV and heating systems: A comprehensive coverage of various fault attributes, such as fault type, time, duration, persistence, interarrival time, and location, has been established in the FI framework [11].

Probability distributions for a more realistic fault injection framework: White noise and uncertainty using Gaussian probability distributions with uniform distributions upon healthy signals, as well as parameter variations upon faulty conditions, have been considered in the model [11].

Automatic scenario generation for reliability evaluation: Reproducibility is supported for a set of specific fault scenarios and random fault injection scenarios [11].

1.3.1.2 Multiple-Fault Injection Framework

The Single fault injection framework has been extended to inject multiple faults as an automated fault injection framework with a multiple fault pattern containing the following contributions:

Modeling patterns of multiple faults in DCV and heating systems based on data from field failure rates and maintenance records: This thesis maps insights from maintenance records to FI patterns with multiple faults. The fault occurrence probability is an important parameter in designing a realistic FI framework because this parameter is affected by environmental conditions, e.g., dust and dirt, seasons and respective temperatures, working conditions, application areas, and the locality of faults in various components of a system. Therefore, the fault model is created using statistical parameters such as fault occurrence probability. Fault occurrence probabilities enable the definition of scenarios and the performance of FI based on different environmental conditions and fault type rates [16].

Injecting multiple faults into a simulated DCV and heating system according to multiple-fault patterns: This thesis introduces an FI framework where faults are activated by a Fault Injection Vector (FIV) that precisely controls the attributes of multiple faults by multi-dimensional matrices, such as timing, locality, type, persistence, and values. The designed FI framework injects multiple faults into multiple zones and multiple components with corresponding fault attributes. An automatic FI algorithm initiates the fault attributes. Fault repetitions and multi-dimensional fault attributes are assigned in a randomized manner. The framework is generic, and the matrices can be customized and extended for different structures and buildings. The thesis shows how fault patterns for multiple faults can be established for a particular structure and environmental conditions based on maintenance records [16].

1.3.2 Experimental Evaluation for Realistic Scenarios with Fault Injection

An experimental evaluation for the simulation-based automated single fault injection and automated multiple fault injection in DCV and heating systems have been carried out in this thesis, considering a comprehensive fault model where faults are activated with realistic fault patterns and combinations for different environmental conditions. Corresponding contributions are explained as follows:

Experimentally evaluating the effects of multiple faults on the behavior of the DCV and heating systems: The thesis provides comprehensive experimental results and insights into the system behavior upon single faults [11] and multiple faults using fault patterns [16]. Due to the use of real-world data and maintenance records in multiple fault injections, the results are realistic. This is a significant result of research on fault management techniques coping with multiple faults, for which no experimental data is available today [16].

1.3.3 Integration of Composable Models with Multiple-Fault Injection Framework

Composable multiple-fault injection framework for DCV and heating systems to ensure scalability and universality: The integration of the multiple-fault injection framework with an automated fault injection algorithm [16] and composable modeling [21] provides more flexibility and scalability for the system modeling, configuration, and faulty scenario generation. The composable model is module-based and builds the system structure and configuration based on the user's demands [21]. The multiple-fault injection framework has been integrated with composable modules, and an automated fault injection algorithm has been integrated with a composable system model generator. This enables multiple-fault injections into multiple floors, rooms, and components and monitoring system behavior and fault impacts in critical infrastructures with numerous components.

1.3.4 Diagnosis Services to Identify Faults in HVAC Systems

In this thesis, the fuzzy theory and Bayesian Belief Network (FBBN) have cooperated to develop an accurate fault diagnosis technique in DCV and heating systems. The main contributions of FBBN have been described as follows. I have contributed to its main idea and implemented the algorithm [18, 25].

Combination of data-driven and knowledge-driven approaches to develop a hybrid FDD technique (FBBN): To develop the FDD technique, a data-driven classifier has been integrated with fuzzy logic and a Bayesian belief network to combine the advantages of both methods. The diagnosis technique has been developed in two modes: offline mode and online mode. The Bayesian network in this approach has been constructed by three main elements: node relations, node directions, and node probabilities. In offline mode, all relations, directions, and probabilities of different nodes have been computed for a specific fault condition and stored in a table called Relation-Direction-Probability (RDP) table. All RDP tables associated with different fault conditions are stored in a library. In online mode, a fault scenario is generated randomly. All signals' relations, directions, and probabilities are calculated, and the RDP table is generated. An automatic classifier compares the RDP table of the actual fault scenario with the RDP tables of the pre-generated library of the offline mode to determine its correlations (similarities) with the library elements. Mutual Information (MI) theory is utilized to find the signals' dependencies or correlations. Fuzzy logic theory and BBN have been combined for node creation and dependency analysis [18].

Extensibility of the fault library for additional fault attributes: The offline fault library has been constructed for stuck-at faults in different time intervals and locations. The fault type, time, and location can be accurately diagnosed for each diagnosis process. This method can be extended to additional faults,

such as multiple faults, more fault types, or other fault attributes. New RDPs for new fault cases should be generated and added to the offline library for this aim. This method uses fuzzy logic, so it provides more understandability. Compared to other knowledge-driven methods, it needs less effort from experts because experts only need to define the fuzzy sets and use the automatic classifier to diagnose the fault cases. The evaluation results show fault diagnosis accuracy when mapping the online fault case to offline fault library elements [18].

Determination of hidden and intrinsic dependencies for signal-based system models: This method has been developed by determining dependencies between different signals or system attributes, such as trends or system statuses in different defined subdomains, by fuzzy logic over time. System attributes are categorized into continuous attributes (e.g., sensor measurements) and discrete attributes (e.g., actuator statuses). Therefore, this method is applicable in system models with different types of attributes when experts cannot quickly determine the hidden dependencies. This introduced FDD technique finds the dependencies automatically among signals' subdomains that change concurrently over time [18].

Scalability and universality of FBBN technique for complex infrastructures: HVAC systems often include numerous types of components, and their complexity increases when increasing their scales. System experts in knowledge-based strategies should analyze many signals, including the sensor's and actuators' measurements. This method automatically finds signals' dependencies and faults in complex structures without experts' inference [18].

Integration of the hybrid FDD technique with DCV and heating system models: The presented diagnostic FDD technique has been integrated with the DCV and heating system models to evaluate the FBBN method with a compatible fault model [18].

Independence of FBBN from prior historical data: Other existing BBN-based methods construct the network and calculate prior conditional probabilities with historical data. In the introduced FBBN method, fuzzy theory categorizes the signal types and their values to different subdomains as fuzzy sets to create the Bayesian network nodes. Node probabilities are computed by calculating the fuzzy sets' conditional probabilities [18].

1.4 Thesis Structure

This thesis is written in eight chapters based on the thesis objectives. The content of each chapter is detailed and summarized as follows.

Chapter 1 is an introductory chapter to clarify the current problems in the field of HVAC systems in buildings that must be solved to achieve good system performance and satisfy quality constraints. Then, the contributions of the thesis and the provided solutions are addressed comprehensively. The proposed techniques are generic and are validated by applying them to a simulated DCV and heating system [32] for accuracy and consistency. Chapter 2 expresses required basic concepts to understand the developed techniques and the system model description. Chapter 3 discusses and specifies the requirements according to the thesis objectives. Then, the applied techniques for each requirement are described. Afterwards, the required state-of-the-art to develop the introduced techniques is investigated and summarized. This chapter

is concluded with a discussion of research gaps based on the literature review to clarify the thesis contributions. Chapter 4 explains the system model of the simulation environment, the physical model of a multi-zone building, a component-based system model equipped with a DCV and heating system, and other embedded subsystems. The component-based system model consists of several repository modules, including different room components, controller components, a fault injection component, and their interrelations. Chapter 4 also explains the development of an automated and generic single/multiple fault injection framework, including a realistic fault model and multiple fault patterns. The interrelations of the automated FI framework and component-based system model components are also illustrated in detail. Then, three algorithms are described using pseudo-code for the automated single fault injection, multiple fault injection, and component-based system model generation. Chapter 5 describes a novel and generic fault diagnostic algorithm, including two main parts: fuzzy Bayesian belief network construction and a classifier diagnostic algorithm based on the constructed FBBN.

A high-level specification is detailed in Chapters 4 and 5. Chapter 6 expresses the implementation details of all proposed techniques, including the component-based system model, the complete fault injection component description, and the extendable multi-dimensional strategy to define fault attributes. An automatic multiple fault injection algorithm is described comprehensively. Indexing is used for mapping the system model components to the multi-dimensional matrixes. An example shows that the matrixes are extendable and mutable with different system structure layouts. Afterwards, the implementation details of the FBBN diagnostic algorithm are explained with an example. Chapter 7 illustrates scenario-based experimental evaluations for the single and multiple fault injection framework by providing different fault cases to investigate the system behavior and their impact on reliability. Then the evaluation results are discussed. The results of all implemented techniques are concluded in Chapter 8. Chapter 9 is the appendix that contains the references, the list of abbreviations, figures, tables, research references, and evaluation results.

2 Basic Concepts

This chapter discusses the main required basic concepts. HVAC systems are designed and developed based on the cyber-physical concept that numerous susceptible components are embedded and cojoined to serve fault tolerance services and dependable functionality. Some basic concepts are described to understand better this thesis's contents, such as cyber-physical systems, embedded systems, real-time embedded systems, dependability analysis, fault injection, fault detection and diagnosis techniques, and probability theory.

2.1 Cyber-Physical Systems and Human-Cyber-Physical Systems

Cyber-physical systems describe the cojoining of physical processes, computations, communications, and integration of internet connectivity among the processes. Cybernetics, which was pioneered in 1948 to develop control systems, is the background of cyberspace and Cyber-Physical Systems (CPS). CPS is an intersection of computation with physical processes, and its behavior is defined by both cyber and physical parts of the system [33].

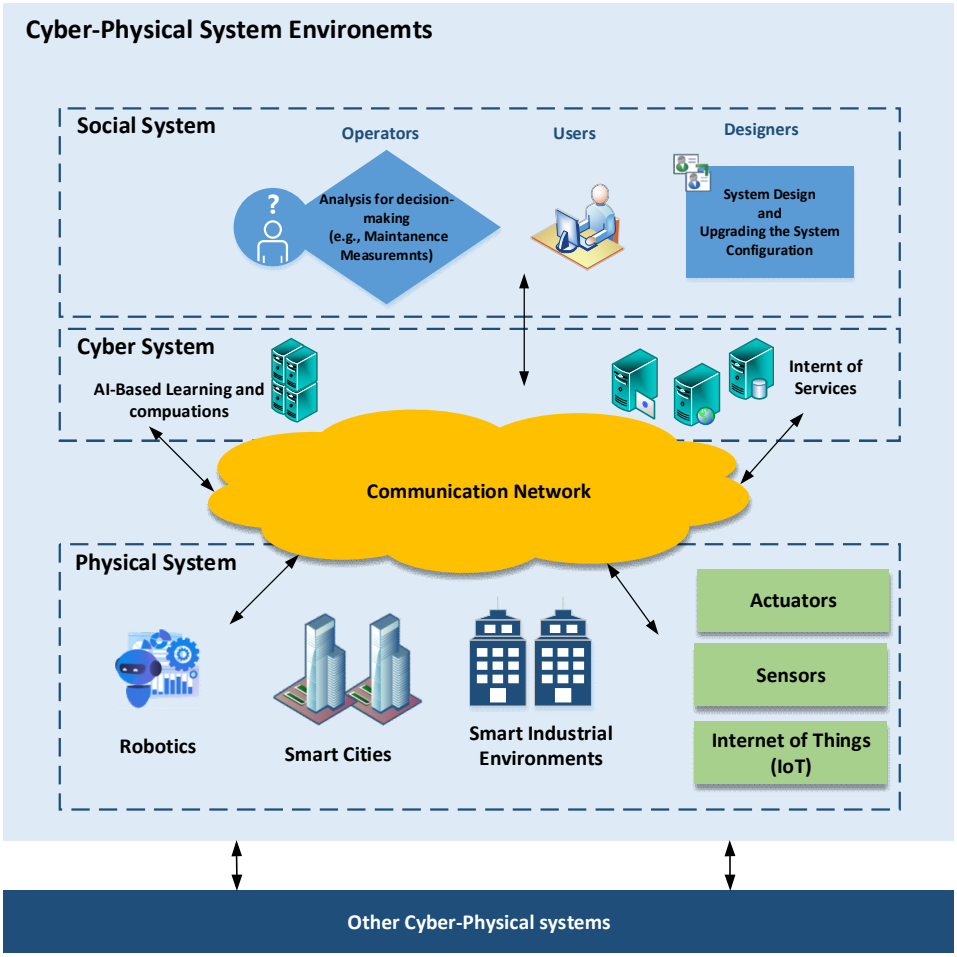

Figure 1. Human-cyber-physical system structure illustration including three primary sub-systems [34, 35].

Figure 1 illustrates a CPS structure comprising its elements and internal and external interactions. CPS has gained attention for its great potential to design intelligent systems that integrate cyber technologies into the physical world and contain collaborative communication objects to control and monitor real-world physical processes. A CPS interacts with physical systems (sensors, actuators, human-machine interfaces, working spaces such as IoT-based manufactures, devices, and smart cities), social systems (humans including designers, operators, and users) and other CPS environments via cyber systems (communication networks and internet connectivity) . Such systems are also known as Human-Cyber-Physical Systems (HCPS) [34, 35]. An HCPS is an intelligent system with significant applications in digital-networked manufacturing [34]. Social systems design, analyze the output, make decisions, learn based on human-based cognitive characteristics, and interact with other systems, such as physical and cyber systems. Cypher systems control, analyze, compute, and make decisions based on expert knowledge. Expert knowledge in the cyber system can be fused and ameliorated by Machine Learning (ML) and Artificial Intelligence (AI) through interaction with the physical world [34]. The physical system executes physical device tasks and generates data for the other systems, e.g., sensing and actuating. A CPS can interact with cloud platforms to reduce the amount of required local resources. Embedded systems are typically a part of CPS where control is based on continuous dynamic feedback. Therefore, error detection and time constraints must be satisfied for stability of control and recovery tasks. At the same time, CPSs solve the consistency in large-scale, complex, and multi-dimensional environments [36]. CPSs have a wide range of applications, such as Autonomous Automobile Systems (AAS), smart greenhouses, water distribution, healthcare systems, and smart buildings.

2.2 Embedded Systems

An embedded system is a computer system that includes an embedded device programmed and optimized to perform a specific application. This embedded device is usually hidden inside a device and interacts with the environment, e.g., sensors rather than users, to provide the services [33].

2.3 Real-Time Systems

A real-time computer system is a computer system facilitated by real-time computing. Real-time computing is the ability of the system to react at constrained points in time, known as deadlines. Embedded devices implement real-time computing. Real-time systems control the actuators and get information from sensors simultaneously. The system should react to the system events and inputs and provide outputs considering the timing requirements. Timing constraints include the finish time or both the start and finish time [37]. The control system structure of real-time systems and their communication can be designed in different ways using periodic, aperiodic, or hybrid timing models [37].

2.4 Real-Time Embedded Systems

A combination of embedded systems and real-time computing results in real-time embedded systems. Not all embedded systems include real-time features. The embedded systems included in the real-time operations are real-time embedded systems. Embedded real-time systems are extensively used in many

applications, including automotive systems, industrial automation, aerospace, and home security systems [37–39].

2.5 Finite-State Machines

State Machines (SMs) are an appropriate and common solution to model the behavior of embedded real-time systems. When the system states are finite, we denote the SM as a Finite State Machine (FSM). Compositions of FSMs include Concurrent State Machines (CSMs) and Hierarchical State Machines (HSMs). Figure 2 illustrates the notation of the state machine for the concurrent and hierarchical composition, in which patterns, inputs, outputs, and variables are combined to build a complex system [40]. The states of the system and its transitions can be represented in a diagram. In a system model with discrete dynamics, each reaction maps valuations of the inputs to valuations of the outputs. The number of states in a finite sate machine is finite [40]. Below is the Equation 1 shows the set of states for the FSM diagram with *i* states.

$$States = \{State_0 \ , \ State_1 \ , \ ..., \ State_i\} \qquad \text{Equation 1}$$

$State_1$ is the initial state at the beginning of the sequence of states, and the FSM can change from one state to another in response to the inputs. The change from one state to another one is called a transition [40].

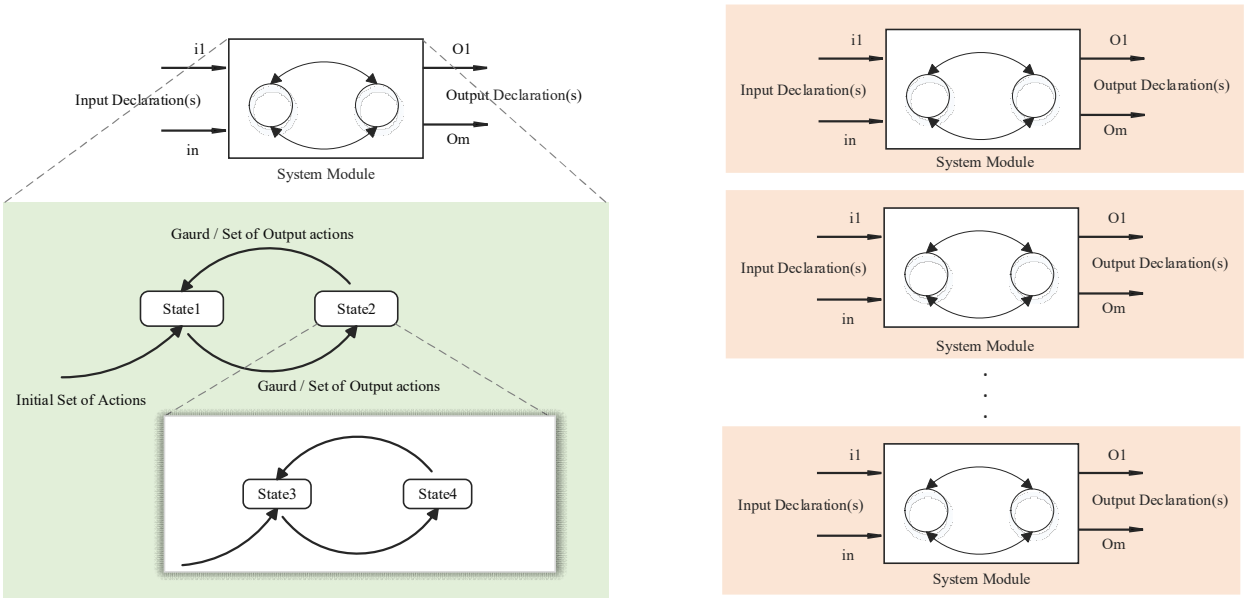

Figure 2. Concurrent and hierarchical composition notations of state machines [40]

2.6 Dependability Analysis

Real-time computing systems are characterized by functionality, performance, cost, and dependability [41]. Dependability analysis can be applied to software and hardware and ensures that the system operates appropriately [42]. In literature, there are various definitions for dependability [42–44]. Parhami [44] provided different definitions for dependability in 1988. For example, dependability is the probability of the system's correct services or justifiable confidence that a computer system performs specified actions or delivers expected results in a trustworthy and timely manner [44]. Trivendi et al. [43] have defined the dependability of a system as justifiable confidence for a specified action that the results are correct and prompt and used for the fault tolerance measurements. Sometimes the term dependability is used interchangeably with the term reliability [42, 44, 45]. Dependability integrates attributes such as availability, reliability, safety, confidentiality, integrity, maintainability, security [41], correctness, and robustness [43].

2.7 HVAC systems

HVAC systems are large-scale distributed embedded systems with different components such as sensors, actuators, and controllers interconnected with different wire-bound and wireless communication networks. In addition, HVAC refers to systems that perform designed processes to regulate the interior air conditions to maintain desirable and acceptable temperature, humidity, ventilation, and the safety of occupants for diverse application domains such as commercial buildings, industrial environments, and office buildings [46–49]. Furthermore, HVAC systems in buildings aim to maintain the thermal conditions in a comfort zone and qualify the air conditions. Various control strategies should be considered based on geographical locations and conditions to ensure interior comfort levels [50]. Indoor Air Quality (IAQ) and proper ventilation are challenging concerns due to energy conservation issues [51]. Therefore, DCV, besides heating systems, is a type of HVAC system with a control strategy that modifies the amount of fresh air from the environment delivered to a room by automatically adjusting damper actuators and ensures thermal comfort for the occupants. Furthermore, in critical infrastructures such as airports and hospitals, HVAC systems serve an essential role in emergencies. For example, in case of a fire, HVAC systems need to remove toxic gases while slowing down the expansion of the fire. Such an emergency reaction is associated with stringent requirements for real-time response, consistency, and fault tolerance:

Real-time: Emergency situations and state changes of the emergency must be detected within bounded delays. After that, suitable control strategies must be implemented that depend on real-time requirements for control stability.

Fault-tolerance: Numerous faults can occur in emergency scenarios, such as communication faults, faulty sensors, and actuators (e.g., heat-induced damage, and obstructions). Besides, design faults that are not observable may be triggered under normal fault-free conditions. Therefore, fault tolerance is required for operational and design hardware faults.

Consistency: The distributed control of the actuators must be coordinated to establish the intended behavior (e.g., channels for the airflow, and fire mitigation). Therefore, system-wide consistency of state information and control decisions must be ensured despite adverse conditions (e.g., faults, stuck statuses of actuators).

2.8 Fault, Failure, and Failure Propagation

Research on dependability resulted in fault-tolerant computing in the 1960s [44]. In fault-tolerant systems, faults from unknown origins and events should be tolerated to achieve more system reliability if a defective hardware or software component results in a "*fault*". The provided "*service*" may be contaminated by a defective component, leading to an "*error*". An erroneous system state causes sub-system "*malfunction*" and system performance degradation. Then, it causes a system-level "*failure*" or a failed computer system [44]. Faults propagate from the component level into the system level or the other systems. Faulty components are commonly denoted as system fault containment regions (FCRs). Figure 3 shows that a root fault in a component (i.e., component-level) leads to the failure of that component, causing a fault for the other components, and then the fault of the entire system has the potential for a system-level failure [11]. A fault-tolerant system may contain "*symptoms*". A Symptom is an external manifestation of system failures, e.g., the management system's alarm, log file, or notification [52].

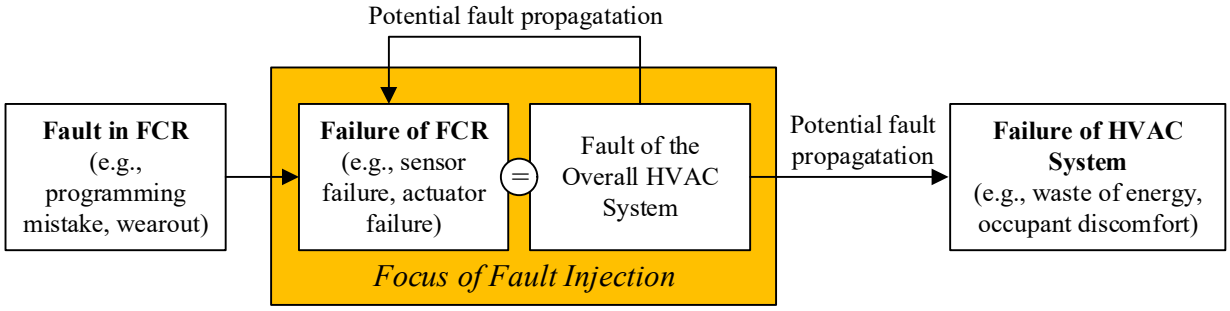

Figure 3. Fault and failure propagation [11].

2.9 Fault Injection, Fault Detection, and Diagnosis

Fault injection in fault tolerant systems is defined as the dependability validation technique where the system's behaviors and observations in the presence of faults are induced by the introduction of faults [53, 54]. Different objectives of the FI can be mentioned, such as understanding the fault impacts and workloads, assessing the efficiency of fault-tolerant systems, forecasting faulty behavior, estimating failure coverage, identifying faulty links, and studying the system behavior under different fault conditions [53].

FDD can be integrated with simulated system models as a monitoring process to detect and predict the presence of defects based on system observations to determine the causes of faulty operations. FDD techniques are able to diagnose the fault types, locations, and other specifications. Furthermore, FDD techniques provide corrective instructions for each diagnosed fault case [28, 55].

2.10 Correlation and Mutual Information in Probability Theory

In probability theory [56, 57], correlation is a statistical relation or association between two random variables. Correlation or dependency is applicable to find the similarities of signals [58]. Mutual information (MI) is a concept rooted in information and probability theory introduced by Shannon in 1948 [59–61]. Typically, MI of two random variables is a statistical measure of the mutual dependence (correlation) of two random variables [62]. MI measures information about one random variable by observing the other random variables [18, 25, 63], although in reality the number of sources that transmit the information is not specified. In this situation, the system should consider a multivariate or multi-dimensional model to calculate a contingency table of more than two variables [64]. Multivariate information is suitable for error analysis [64]. For example, Srinivasa [64] and Batina et al. [65] described the MI for the multivariate sources and multi-dimensional variables. In a general form, Multivariate Mutual Information (MMI) can be defined as [64]:

$$I_N(X_1;X_2;...;X_N) = I_N(X_1;X_2;...;X_{N-1}) - I_N(X_1;X_2;...;X_{N-1} \mid X_N) \quad \text{Equation 2}$$

$$I_N(X^N) = I_{N-1}(X^{N-1}) - I_{N-1}(X^{N-1} \mid X^N), \text{ where } X^K = (X_1, X_2, ..., X_k). \quad \text{Equation 3}$$

$$I_N(X_1;X_2;...;X_n) = ((H(X_1) + H(X_2) - ... + H(X_n)) + ...(-1)^{N-1} H(X_1;X_2;...;X_n) \quad \text{Equation 4}$$

H(x) is defined as:

$H(X) = -\sum_{x \in X} p_X(x) \log(P_X(x))$, where p(x) is the conditional probability distribution. Equation 5

For instance, for three random inputs of X, Y, and V, the MI can be calculated as [64] :

$$I(V, X; Y) = H(X, V) + H(Y) - H(X, V, Y) \quad \text{Equation 6}$$

Where $I(X;Y) = H(X) + H(Y) - H(X,Y)$. Equation 7

MI has several applications such as FDD, classification, test permutation [66], feature selection in engineering and machine learning [67–71], graph and Bayesian network (BN) optimization [72], test selection [73], medical applications [62] and image processing [74]. MI can be calculated upon a pair of random variables, signals, or processes. For Example, Intan et al. [62] have used MI upon fuzzy sets to track medical records by combining fuzzy logic and BN methods.

2.11 Bayesian Belief Network

As a probabilistic model, the Bayesian Network (BN) models nonlinear dynamics in a discrete manner and measures the probability of faults [18, 75]. A Bayesian network expresses a system model graphically for reasoning about uncertainties and constructing nodes, arcs, and their conditional dependencies

originating from Bayes theory [76, 77]. Each node represents a random variable of the system. Each arc represents a directed connection between two nodes and the graph should be acyclic. The BN model expresses probabilistic belief about variables and it can be updated automatically when new information (evidence) is available [77]. The BN model introduces system states (called belief states). Each belief state is the best possible belief given all available evidence [75]. It means that the strength of the relationship between nodes is quantified by conditional probability distributions [78]. The network structure is based on the qualitative relationships between variables. Nodes are connected once one node affects the other ones. The arc between nodes indicates the direction of this effect.

There are some structural terminologies for BN models. For example, a node can be a child, parent, ancestor, or descendant. A node is a parent if there is an arc from the former node to the latter node. A node is an ancestor when it is in a directed chain of nodes with the earlier node, and the later node is a descendant. The Markov blanket of a node is constructed of related nodes, including parents, children, and children's parents. The first nodes in a chain are root nodes that represent original causes. The last nodes are leaves that represent the final effects, and the nodes in between are intermediate nodes. When there is a new observation in the system from system variables, a new conditional probability should be calculated. The conditioning process is known as *probability propagation*, *inference*, or *reasoning*. There are different types of reasoning: (1) diagnostic reasoning is the reasoning from symptoms to cause, (2) predictive reasoning is the reasoning from new information about causes to new beliefs about effects, (3) intercausal reasoning is the reasoning about the mutual causes of a common effect, and (4) combined reasoning is a combination of the other types [76].

3 Related Works

The related work chapter investigates the state of the art and identifies the primary research gaps, concentrating on the thesis requirements and utilized techniques. First, all thesis requirements and their applied techniques have been listed and introduced in Section 3.1. Then, applied techniques have been detailed and summarized as the state of the art in top-down order. A simulation-based system model and fault injection techniques to change the system behavior are necessary to evaluate the reliability at an early development phase of the system. For this purpose, their state-of-the-art is studied. A complete fault model should be defined to develop the fault injection framework. Thus, fault modeling techniques and fault classifications in HVAC systems are studied. The state-of-the-art for single and multiple-fault injection techniques, experimental evaluation techniques, and FDD techniques in HVAC systems are also studied. Finally, all research gaps are concluded in a separate subsection to clarify thesis contributions.

3.1 List of Requirements and Applied Techniques

The requirements have been listed based on the thesis objectives. Techniques have been developed to achieve thesis objectives summarized in Table 1 and discussed in this section in detail**.**

Table 1. An overview of the thesis requirements and developed techniques.

Techniques	**Requirements**			
	Accurate system modeling with low effort	**Reliability Evaluation**	**Diagnosis with high accuracy**	**Universality and scalability**
Single Fault Injection Framework (SFIF)	×	×	×	×
Multiple Fault Injection Framework (MFIF)	×	×	No	×
Component-Based System Model enabled for Multiple Fault Injection	×	×	No	×
Hybrid Fault Detection and Diagnosis Using Fuzzy logic and Bayesian Belief Network (FBBN)	×	×	×	×

Requirement 1: Accurate system modeling with low effort for FDD techniques

Techniques: automation is one crucial factor for modeling the control strategies in HVAC systems in modern buildings to reduce efforts for experts and minimize required knowledge. In addition, simulation is a convenient way to evaluate various system models [12]. As a result, a generic and simulation-based fault injection framework in this thesis triggers different types of faults (single and multiple) automatically in a Matlab/Simulink environment [11, 16]. This thesis also provides automatic scenario generation for experimental evaluation and data generation for FDD techniques in DCV and heating systems. Different random or customized fault scenarios are reproduced and constructed automatically for automated fault injection framework [11, 16]. This thesis also has integrated the automated multiple fault injection framework [16] and composable system models [21] to facilitate the system model configuration for designers and users based on their requirements. Fault injection based on the system configuration is

automated. Finally, the introduced FDD technique, FBBN, diagnoses the faults using an automatic classifier algorithm with fewer experts' efforts by combining the Bayesian belief network and Fuzzy logic [18].

Requirement 2: Simulation-based reliability evaluation at early development phases and fault injection without damage to physical systems

Techniques: dependability analysis at an early development phase significantly improves cost, time, and performance and prevents damage to the system-under-test and real-world systems. A composable model enables the injection of multiple faults using different system configurations , and the reliability of the system model in the FIF can be evaluated for different fault scenarios [11, 16]. A comprehensive fault model (fault profile) with a description of fault attributes is required to cover all faults specifications and scenarios [11]. In the case of multiple fault injection, the fault pattern should match the system characteristics with the fault occurrence probabilities. Therefore, history data of fault occurrence probabilities and calculations for different fault events in DCV and heating systems have been studied [16].

Requirement 3: Diagnosis with high accuracy

Techniques: a hybrid FDD technique has been introduced, named the FBBN technique. To develop the FBBN, two different techniques, including fuzzy logic and Bayesian belief network, have been combined to benefit the advantages of both FDDs [18].Fuzzy logic is a knowledge-driven method that facilitates the model description using expert knowledge. BBN is a statistical method that constructs the network with arcs and nodes based on probability theory and its associated methods, such as mutual information, conditional probability distributions, and joint probability distributions. The state-of-art for different diagnosis techniques, their general classifications, and other related topics have been investigated comprehensively. The FBBN diagnosis method has been tested and experimented with in DCV and heating systems. FBBN accurately diagnoses the time and location of single stuck-at faults [18].

Requirement 4: Universality and scalability

Techniques: the developed methods in this thesis, such as FBBN [18] and AMFIF [11, 16], are generic and scalable. They are capable of being integrated into other simulation-based target systems with low effort. For example, the AMFIF has been integrated with a component-based HVAC system model by adding multiple fault injector components to the system model blocks. The system model reconstructs the wanted configuration, and the AMFIF injects the faults based on the indexing procedure. Furthermore, FBBN is integrated with models of an example DCV and heating system to evaluate the method's accuracy and scalability [18]. In FBBN, the correlation of the system attributes (continuous and discrete signals) is determined automatically. When increasing the number of attributes, FBBN is adapted easily by defining the corresponding fuzzy sets [18].

3.2 State-of-the-art in Simulation Modeling Techniques for HVAC Systems

The building sector consumes around 40% of total energy and produces 36% of Greenhouse Gas (GHG) emissions [1, 79]. Therefore, smart buildings are equipped with building automation systems and management control systems that enhance the optimization of the HVAC systems [80]. Smart automation

systems allow management of indoor environmental efficiency, e.g., heating, ventilation, air conditioning, lighting, heating cost, and energy consumption. Model Predictive Control (MPC) and model-based optimization are practical solutions where mathematical models are utilized for control processes [81]. For dependability evaluation and system performance analysis during the design phase, a model is required to simulate different control strategies, e.g., monitoring the system energy consumption and indoor air quality. There are two types of control strategies for HVAC systems: local control functions and supervisory control functions [1, 80]. Local control functions include basic controls and automation. They can be categorized into sequencing control and process control strategies. Examples of control variables are damper position or valve positions, their statuses changing in a sequence. Supervisory functions are also known as optimal control. They provide so-called total system monitoring and overall control of the local sub-systems, including setpoints and schedules [1, 80]. Supervisory control optimizes the operation of HVAC systems by providing a system approach. It considers system-level or subsystem-level characteristics and interactions in the overall system. Supervisory control approach requires system models, component models, and optimization techniques. Figure 4 illustrates the control strategies in HVAC systems in detail.

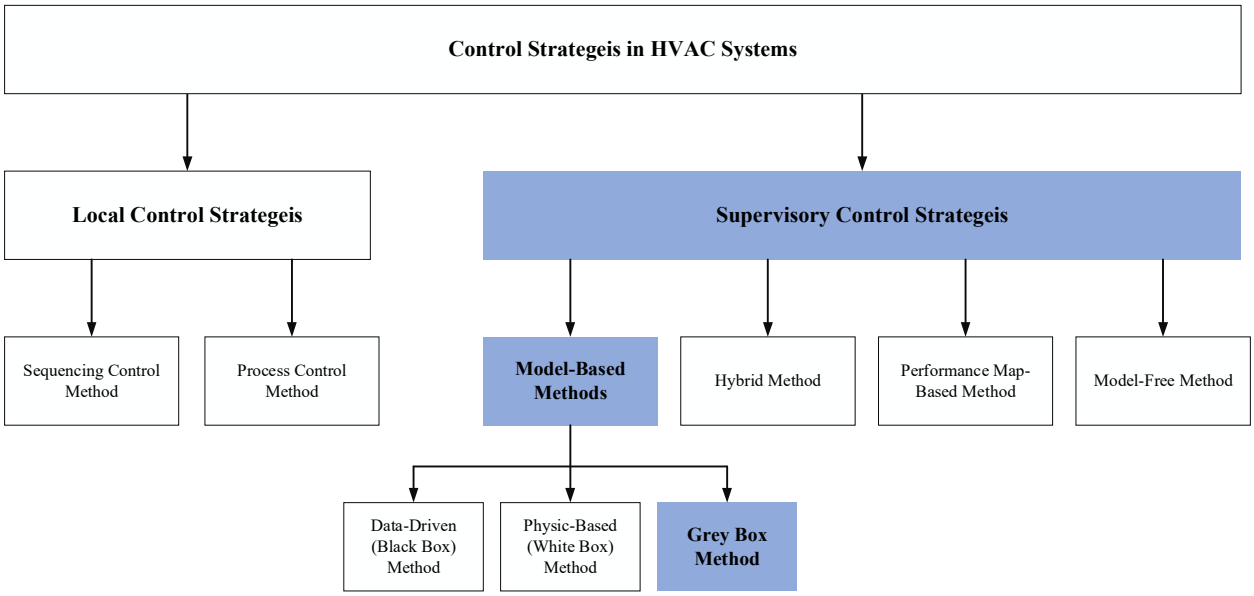

Figure 4. Control strategies in HVAC systems [80, 82, 83].

HVAC systems typically have a complex internal structure and are developed in different application domains, such as residential buildings, commercial buildings, office buildings, and storage of goods [84]. For an accurate analysis of HVAC systems, all individual components should be modeled based on the underlying physical phenomena. HVAC models have mainly been classified as data-driven (black-box, inverse, or empirical), physics-based (white-box, forward, or mathematical), and grey-box models (hybrid) [82, 83].

Data-driven approaches are inductive models in which the input and output data relationships are defined by experimental techniques and Artificial Neural Networks (ANN) using data measured and gathered under specified tests. There are various techniques in data-driven modeling, e.g., data mining algorithms, fuzzy logic models, statistical and stochastic models, state-space models, and case-based reasoning [82]. The data-driven approach results in linear/nonlinear, static/dynamic, explicit/implicit, discrete, and stochastic models [82].

The physics-based approach uses a deductive model known as the analytical first principal model. Their modeling is based on the physical principles and details of the processes. In these models, usually, time-domain differential equations are converted to frequency functions [85]. The main physics-based applications in HVAC systems are the zone models, cooling and heating models, duct, damper, valve, fan, pump, and storage tank models. The physics-based approach results in linear/nonlinear, static/dynamic, explicit/implicit, and continuous models [82].

Grey-box is a hybrid model-based approach and benefits both data-driven and physic-based approaches qualities. The basis of this approach is the physics-based approach for the system structure, and the parameters of this model are calculated by estimation algorithms using the details of the processes [82].

Gershenfeld et al. [86] have classified the mathematical models, including explicit/implicit, linear/non-linear, static/dynamic, deterministic/probabilistic, and discrete/continuous categories. In linear models, system observations are linear. In non-linear models, system observations are represented by non-linear equations and are the standard output of the white-box and grey-box methods. Static models are time-independent, and dynamic models are defined by differential equations and are time-dependent. Inputs of explicit models are known, and a finite set of computations must compute the outputs. In implicit models, inputs and outputs are known, but iterative methods determine the input and output relations. Discrete models have a discrete sample space, and continuous models have a continuous sample space (e.g., values of sensor measurements). In deterministic models, system states are determined by previous states (i.e., previous values of system variables). However, in probabilistic models, system states are defined by probability distributions.

Each model has its specific purpose. For example, dynamic models are commonly used for modeling the slow moving dynamics of temperature and humidity, and static models for fast-moving dynamics (e.g., CO_2 concentration) [82]. In zone modeling of the physics-based approach, the zone temperature maintains steady by balancing the room's heat and energy. Heat transfer usually occurs in the system through the supply of air conditioning, walls and windows, and internal or external gains, e.g., humans or solar energy. Heat transfer models commonly use the heat conduction equation model, heat balance method, and weighting factors [25, 82, 87–90]. In damper modeling of the physics-based approach, the airflow rate of the damper depends on the control signals that control the damper status [25, 82, 91, 92].

Many model-based techniques in HVAC systems have been discussed in the literature. Ciprian et al. [81] have simulated a virtual prototype for energy management in HVAC systems using Matlab/Simulink. A system model can be modeled and evaluated with simulation tools [11, 16, 25]. Simulation is used to imitate, describe, and analyze a system's real-world behaviors and operations over time. It helps to design a real-world system and its control and automation applications [93, 94]. Its environmental parameters and conditions are set, and simulation results can be compared with real-world scenarios. There are many tools and environments for simulation-based techniques. MATLAB/Simulink as a user-friendly tool, and SimScape as a powerful tool to model the physical models, components, and connections in the Simulink environment are practical tools for modeling HVAC systems [95].

Multi-zone buildings have been modeled based on heat, heat flow, and moisture. Their mathematical equations are described using white-box modeling in [63, 65–67]. Mathematical models are very popular for HVAC systems in representing processing signals [83]. Signals are constructed according to the physical principles in sub-systems, components, and links between inputs and outputs. Behravan et al. [32] modeled a grey-box and scalable multi-zone office building simulated in Matlab/Simulink. The model contains thermal dependencies among rooms, the outside environment, and indoor spaces. Heat transfer in the building has been modeled using equations. The number of occupants and their changing patterns are

modeled by a counting sensor. Their model also includes the demand-controlled ventilation sub-system in which the CO_2 concentration signal was based on the calculation of CO_2 concentration according to equations [25, 32]. Karmacharya et al. [96] have modeled a simplified building-HVAC system using MATLAB/Simulink. Their model predicts temperature variation, energy consumption, and comfort levels. Different physical properties should be used for the estimations, such as environmental conditions, and heating system. There are different approaches for modeling the heat flow, such as lumped capacitance models, distributed parameters, finite element or finite difference method, and impulse response factor method [25, 32, 83, 96]. Gouda et al. [97] have modeled the HVAC system's robustness and control feedback. Their room model is based on lumped capacitance modeling. Kassas et al. [98] have implemented the HVAC system for residential buildings using Matlab/Simulink to predict energy consumption. Their model predicts the temperature variation and energy required for occupants' comfort. Asad et al. [99] have developed an adaptive model of HVAC systems for reliability analysis. For energy efficiency and decision-making, they have provided a Model-Based Real-Time Optimization (MRTO) in which the set points for local-loop operation should be optimized regularly. Usually, real-time optimization techniques aim for optimal operation settings to improve the system's Quality of Service (QoS) [99].

The Department of Defense (DoD) has addressed the application-oriented simulation for easier construction and analysis of the systems using composable models [100]. In composability, the system modules are reusable and can be composed at different levels of perspectives and scales. Modules can be selected, combined, or recombined based on user requirements [101]. In composability, a library of system blocks (i.e., modules) is required that can be extended for desired levels. Modules should be interconnected and achieve accurate interoperability. Advantages of composable models are higher quality, comprehensiveness, consistency, validity, time-saving, lower complexity, and lower cost [21, 100, 101]. Figure 5 represents the composability concept in which a repository of different modules with different specifications is required to build the simulation structures and their interconnections.

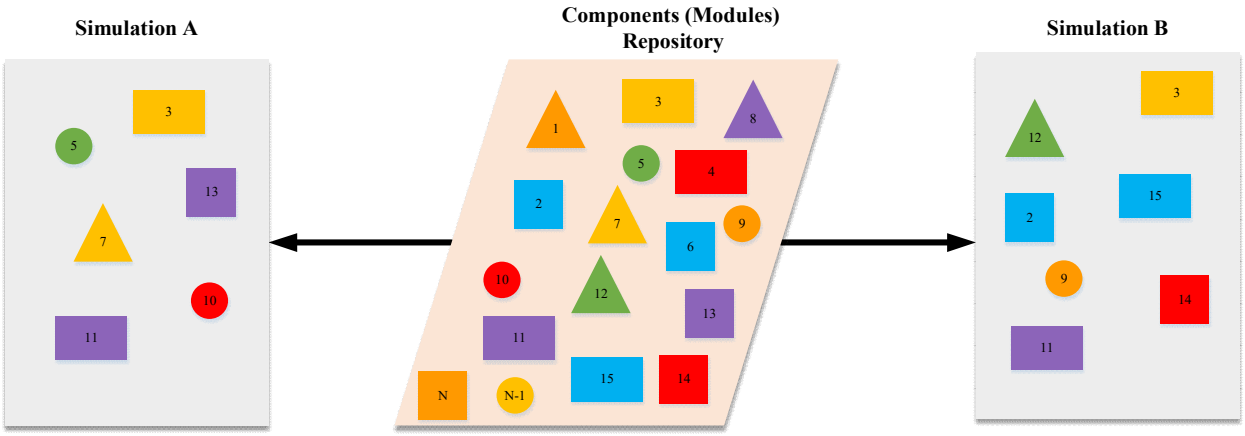

Figure 5. Example of a composable model and simulations including a repository with N modules and two different simulations of A and B with different component combinations [21, 101].

Behravan et al. [21] have presented a composable model for DCV and heating systems to decrease the complexity of the system construction in the case of reusable components and interrelations based on user demands. In their model, different system configurations at different levels can be constructed. The system model is constructed based on the number of rooms, their indexes, and the number of floors. Siegele et al.

[102] have presented an object-oriented system model using Matlab programming. This model has a library of Simulink blocks and a basic HVAC library. The structure of the model and the variables should be defined beforehand. Then the thermal zones and building models can be created according to the defined variables. However, this model supports only single faults with a limited fault model.

In this thesis, a component-based system model has been developed to support multiple fault injection in multiple floors, rooms, and components. This model has been implemented in Matlab /Simulink and validated for different fault scenarios and system model configurations. The system model is constructed based on the system components integration e.g., room, corridor and fault injection components according to user demands for different floors and rooms. In this model, each fault scenario comprises other sub-scenarios. The impact of faults for each sub-scenario has been analyzed and illustrated precisely for the entire scenario, such as CO_2 concentration changes, temperature changes, heating cost, damper, and heater actuators statuses.

3.3 State-of-the-Art of Fault Modeling in HVAC Systems

A fault model is an engineering model that represents all possible ways that a system or device can be faulty [103]. Using a fault model, the consequences of a specific fault can be predicted [104]. Fault model attributes and manifestations should be extracted based on the application requirements and the system's environment [105]. Faults are classified based on the six main criteria of the phases of creation (development, and operational), system boundaries (internal, and external), domain (hardware, and software), phenomenological causes (natural, and human-made), intents (accidental, and deliberate), and persistence (permanent, transient, and intermittent) [41]. A fault model can consist of one or all these criteria based on the system requirements. Fault model accuracy and quality increase the accuracy of control strategies [106]. Complex infrastructures such as distributed HVAC systems are integrated with numerous components. Due to their complexity, many kinds of faults and errors emerge. Therefore, a comprehensive fault model is a required assumption to investigate component faults and their consequences on the system's behavior. The fault model should fit to the system model and introduce each fault's attributes, e.g., fault location and persistence [11, 16]. Faults degrade system performance, therefore modeling of fault sources and fault propagation among components allows scalable compositional safety analysis in hazard identification, and fault impact and probabilistic fault model analysis [107, 108].

Many authors have studied fault modeling in HVAC systems and other related applications. For example, in the area of digital circuits Polian et al. [109] have presented several logical fault models with Missing Gate Faults (MGF), e.g., single, multiple, partial, and repeated MGFs. They have considered different types of faults. Multiple MGFs occur in one or more consecutive gates, and repeated MGFs occur in several cycles for the stuck-at-fault type. To investigate their fault models, they have used an Automatic Test Pattern Generation (ATPG) method in which several test vectors have been used for fault detection. Joshi et al. [110] have discussed behavioral fault modeling in aircraft wheel brake systems. Their fault model explanation includes internal and external fault activation. Internal faults are limited to the component boundaries. Internal faults are dormant and independent from other component faults. External faults originate from out-of-component boundaries due to the propagation of other component faults. Fault propagation increases the system's complexity. They have also considered fault persistence (transient and permanent) and the duration for their defined fault model and rules for fault propagation. However, they have only modeled the stuck-at, burst, and leak faults in the valve and pipe. Gosh et al. [106] have also

presented behavioral fault modeling for testing in digital designs. They established several test vectors, applied them to the digital structures, and compared output responses to the nominal values. They have defined a fault model as a required assumption for test generation to model system failures. However, they have only modeled stuck-at faults for pin faults. Da silva et al. [111] used knowledge-based fault modeling for sensors in aerospace. They have used a combination of object-oriented modeling and rules for sensor faults, including bias, drift, and loss of signal faults. However, they have not considered the duration, interarrival time, and persistence in their presented fault model. Najeh et al. [112] defined a new fault model for symptom generation in the building. They have defined a symptom as a measurable change in normal system behaviors. They have considered a rule-based behavioral test for symptom generation. There are several factors for applying the test, such as door, damper, and weather conditions. They have used the tests for more reliable fault diagnosis, e.g., without occupancy verification, the test may be erroneous or unreliable.

Fault modeling and fault coverage in other FI techniques and FI in DCV and heating systems are significantly studied. For example, Maleki et al. [113] have simulated FI for an Advanced Driver Assistance System (ADAS). They have used a fault model for their FI method, considering different types of faults, such as stuck-at-value and single/double bit-flip, and fault persistence, such as transient and semi-permanent faults. Song et al. [114] have developed a simulation-based interface for fault injection, including the list of components, potential failures, and different faults such as open-pole, open/short circuits, and drift in the verification procedure of circuits of a radar. However, they have not considered persistence, duration, and other fault types. Gil-Tomás et al. [115] have modeled intermittent faults for dependability evaluation for a microcomputer system using the Markov model. Faults are also injected into one or multiple locations. However, fault types are limited to stuck-at, burst, and delay faults. Behravan et al. [18, 21, 32, 116–118] have introduced a fault model for FI in DCV and heating systems comprising different fault types such as gain fault, off-set fault, stuck-at value, stuck-at open/close, stuck-at off/on and single-location and injection time. However, their fault model is limited to permanent faults.

In this thesis, Kiamanesh [11, 16] has introduced a fault model for DCV and heating systems with different fault attributes such as fault type, including gain fault, offset fault, stuck-at value, stuck-at open/close, stuck-at off/on, out-of-bound fault, and data-loss fault. In addition, fault persistence including transient, intermittent, and permanent faults, single/multiple locations, fault injection time, fault duration, fault interarrival time, and fault occurrence probability are supported.

3.3.1 State-of-the-art of Fault Classifications in HVAC Systems

Inaccurate measurements due to hardware faults are inevitable in HVAC systems and lead to more energy consumption and low air quality. Bondavalli et al. [119] classified physical faults into two categories (1) permanent and (2) temporary faults. Permanent faults lead to abnormal behavior and wrong signals which continue constantly. The respective component should be removed or repaired to handle a permanent fault. Temporary physical faults are classified into internal (usually intermittent) and external (transient). An intermittent fault occurs regularly and continuously at the exact location, while a transient fault arises at random locations [119]. Many reasons exist for intermittent faults in different systems. Wakil et al. [120] discussed various intermittent fault causes in embedded electronic modules. They explained that most of them are caused by interconnections and marginal design, e.g., corroded wires, cracked solder joints, corroded or loose connectors, and broken wires [120, 121]. Layali et al. [122] mentioned that the primary cause of intermittent faults is device wear out or the tendency of solid-state to degrade with time, stress,

and time-dependent dielectric breakdown (TDDB), supposing the stress conditions persist in the long term. Such faults may eventually lead to permanent defects. Different transient and intermittent faults include short transients, long transients, and short intermittent faults. Intermittent faults may disappear or become permanent [123].

In this thesis, HVAC hardware faults are classified by their duration into permanent, transient, short, and long intermittent faults [11, 16]. Permanent faults are caused by a defect in a component that requires the repair or replacement of the component. Examples of permanent faults in HVAC systems are a damper stuck-at a closed position or a depleted battery in a sensor. Transient faults occur far more often than permanent faults, and they are harder to detect [53]. They are usually caused by environmental conditions such as powerline fluctuations, high-energy particles, and electromagnetic interference. Intermittent faults are temporary malfunctions of a device that are repetitive and occur mostly at irregular time intervals [124]. Intermittent faults have different root causes, such as unstable hardware, varying hardware states, design faults, and wear-out. Intermittent faults can be repaired by replacement or redesign. Most systems incorporate many embedded electronic modules and components to increase the performance of the monitored system. For such complex systems, especially in the vehicle industry-trains, ships and aircraft-intermittent faults become challenging because they increase due to thermal stress, vibration, moisture, and other stresses. In these systems, there are many reasons for intermittent faults, such as loose or corroded wires, cracked solder joints, corroded connectors, loose crimp connections, hairline cracks in a printed circuit, broken wires, and unsoldered joints. For example, Wakil et al. discussed intermittent faults and electrical continuity in electrical interconnections [120]. They mentioned some common causes of intermittent faults that can be classified into manufacturing imperfection, connection degradation, interface/coupling, poor design, and intermittent connectivity [120, 125]. Examples of intermittent faults in HVAC systems are sensors that are not well-calibrated, software faults, and loose power or communication line contacts. In our proposed FI framework, one intermittent fault with two or three repetitions can be modeled in the case of short intermittent faults. The number of repetitions for long intermittent faults can be defined flexibly according to the system requirements.

Faults in HVAC systems can also be distinguished based on the design, developmental and operational phases. The phase of a fault denotes when a fault occurs, e.g., during the design, development, or operational time of a system's life cycle. A developmental fault occurs before the equipment installation. Developmental faults can be physical faults in production (e.g., inaccurate mask alignment) or design faults (e.g., incorrect positioning of sensors, improper scheduling of operations). An operational fault occurs after the equipment installation phase. An example is wear-out of electronic components. Torabi et al. [126] reviewed common human-made errors in different stages of creation in HVAC systems with multiple zones: preconstruction, construction, and operation phases. Frank et al. [127] discussed common faults and their relevance in the design and operation stages for HVAC systems, rooftop units (RTU), lighting, and refrigeration faults. Faults may propagate through components, phases, and other systems. For example, developmental faults in a component (i.e., FCR) can result in multiple failures in that component and may cause faults in other components. Faults may also propagate through different systems. It means that a root fault in a component leads to the failure of that component which is a fault for other components or the entire system, which can lead to a system failure [11, 16]. Table 2 describes how a component fault leads to a system failure by illustrating examples of fault propagations in HVAC systems. Faults in FCRs (system components), such as inappropriate programming and improper setpoints, lead to component faults and potentially propagate to system-level failures. Each row of Table 2 represents the fault propagation example in HVAC systems from the component-level faults to system-level failures and the failures' impacts, such

as energy waste, poor indoor thermal conditions, and occupant discomfort [16]. Fault detection and diagnosis in air-handling systems are complex because of fault propagation across components. Yan et al. [128] captured fault propagation impacts in an efficient manner using dynamic hidden Markov models to identify failure modes since they contain state transition matrices depending on other components and do not generate joint states.

Table 2. Fault propagation examples in HVAC systems [11, 16, 47, 129–131]

Nr.	Component Faults	Phases	Component Failure (System Fault)	System Failure	Impacts
1	Wrong scheduling of the processing unit, e.g., an incorrect sequence of operations	Developmental fault: design fault	Stuck-at fault Gain fault Offset fault Out-of-bounds fault	Delay High/low/wrong sensor measurements	Equipment life Energy consumption Thermal comfort Indoor air quality
2	Programming mistakes	Developmental fault: design fault	Stuck-at fault Gain fault Offset fault Out-of-bounds fault	Delay High/low/wrong sensor measurements	Equipment life Energy consumption Thermal comfort Indoor air quality
3	Wrong setpoints too high/low	Developmental fault: design fault	Stuck-at fault Gain fault Offset fault Out-of-bounds fault	High/low/wrong temperature. High/low/wrong CO_2 concentration	Equipment life. Occupant thermal comfort. Energy consumption
4	Oversized equipment at the design phase, e.g., incorrect perimeter heating system sizing	Developmental fault: design fault	Stuck-at fault Gain fault Offset fault Out-of-bounds fault	High/low/wrong temperature. High/low/wrong CO_2 concentration	Equipment life Occupant thermal comfort. Energy consumption
5	Improper design	Developmental fault: design fault	Stuck-at fault Gain fault Offset fault Out of bounds Data loss	Delay High/low/wrong sensor measurements	Equipment life Occupant thermal comfort. Energy consumption
6	Inaccurate location of sensors and valves, e.g., wrong thermostat location, Occupancy-sensor misplacement	Developmental fault: design fault	Stuck-at fault Gain fault Offset fault Out of bounds Data loss	Delay High/low/wrong sensor measurements	Equipment life. Occupant thermal comfort. Energy consumption
7	Missing insulation for ductwork or pipes	Developmental fault	Stuck-at fault Gain fault Offset fault	Delay High/low/wrong sensor measurements	Occupant thermal comfort. Indoor air quality
8	Poor coordination of the processing unit	Developmental fault	Stuck-at fault Gain fault Offset fault Out of bounds Data loss	Delay Missing information	Occupant thermal comfort. Indoor air quality Delay
9	Air-duct leakages	Operational faults	Stuck-at fault Gain fault Offset fault	Wrong actuator signals	Equipment life Thermal discomfort Indoor air quality Energy consumption

10	Inappropriate voltage	Operational faults	Stuck-at fault Gain fault Offset fault Out of bounds Data loss	Wrong actuator signals High/low/wrong measurements Missing information	sensor	Equipment life Thermal discomfort Energy Consumption Life risk Fire risk
11	Poor preventive maintenance	Operational faults	Stuck-at fault Gain fault Offset fault Out of bounds Data loss	Delay Missing information		Equipment life Energy consumption Life risk Fire risk

3.4 State-of-the-art of Fault Injection and Experimental Evaluation in HVAC Systems

Modern smart buildings play an important role in the economy, ecology, and human well-being. They are equipped with various electronic components, including different actuators, sensors, and automatic control systems called Building Management Systems (BMS) [132, 133]. The user's comfort is important and affected by the operation of heating, ventilation, and air conditioning system, which is a significant source of energy consumption. The efficient operation of an HVAC system affects the efficiency of the overall system, which is the BMS [133]. In addition, many sensors and actuators are integrated with an HVAC system, and the interactions of these components are fault prone. Without fault-tolerance techniques, the system may face unpredictable conditions. Therefore, a dependability analysis of critical infrastructure is essential. A system is deemed critical when the normal functionality of the provided services by the system is vital for the end users or the environment [134]. For the assessment of quality constraints such as resource usage, resource availability [135], thermal conditions, occupant comfort, and dependability of a system under faults, different approaches, including analytical modeling [136] and experimental methods such as FI [116, 117, 137] are discussed in the literature. FI brings high controllability and observability in a simulation environment. Arlat et al. [138] have introduced an FI methodology for two main goals: validation and design aid. They have also described different modeling abstraction levels, including axiomatic, empirical, and physical models. Axiomatic models emphasize analytic models such as Markov graphs and fault trees. Empirical models relate to more complex and detailed behavior and structural descriptions, such as simulation and physical models implemented as hardware and software features. Fault injection was recognized as a powerful and effective experimental method and extensively used for the validation and dependability evaluation of a target system under faults [139].

3.4.1 Fault Injection Techniques in HVAC Systems

FI was introduced in the early 1970s to study fault impacts and verify fault-tolerant capabilities by deliberately injecting faults into a modeled system [134]. Several surveys studied FI methodologies [53, 134, 140–142]. Briefly, FI techniques can be categorized into four methodologies: (1) physical fault injections, including hardware-based fault injection (HaFI) and soft-ware-based fault injection (SoFI) methods; (2) simulation-based fault injection (SiFI) methods; (3) emulation-based fault injection (EmFI) methods; and (4) hybrid fault injection (HyFI) methods [53, 137, 141, 143]. The advantages and

disadvantages of each method were systematically discussed in [140]. An overview of FI techniques and their positive and negative points are comprehensively described in [53, 140–142].

SiFI is most popular for early experimental evaluations among all FI techniques. A SiFI analyzes a target system by simulating fault effects. It is well-known for its wide range of advantages, such as flexibility, adaptability, visibility, and controllability [139]. However, one disadvantage of the simulation techniques is computation time [1]. SiFI supports the adaptation of tests to various traffic scenarios and avoids costly or dangerous physical FI in the real world [113]. SiFI has a low cost, high controllability, high safety, and high fault coverage [142]. SiFI is categorized into three different subcategories in the literature: the simulation command technique, simulation code modification technique, and simulation modification technique with different levels of abstraction [139]. In the simulation command technique, the simulation model does not change and uses commands to inject faults into the target system model. Built-in simulator commands are used to modify the values of signals and variables [134]. Simulation code modification modifies the system description by adding FI components called saboteurs or mutants to existing component descriptions [53]. Simulation-based fault injectors, such as saboteurs and mutants, are responsible for the deliberate insertion of faults. Fault injectors provide this opportunity to change one or more signal values or timing characteristics. The simulator modification technique changes the simulation kernel and not the target simulation model. Each technique has corresponding advantages and disadvantages.

Many researchers have focused on SiFI, which can be discussed from the point of view of different applications, some specifically for HVAC systems. Maleki et al. [113] proposed a simulation-based injector called SUFI to activate faults in Advanced Driver Assistance Systems (ADAS). The fault model in this framework covers transient and permanent faults such as stuck-at values. Chao et al. [123] proposed a SiFI framework called FSiFI to study the propagation of faults and symptoms. They analyzed the transient faults affecting different SPARC processor components, such as ALU, decoders, and register files. Song et al. [114] proposed a method for verifying radar systems using PSPICE for the simulation environment. The simulation represents the circuit model of the radar in the simulation software. The software provides the behavioral model, and the user can extend the model or use models built by the software. Gil-Tomás et al. [115] designed an SFI to inject intermittent faults to evaluate the dependability of submicron complementary metal-oxide-semiconductor (CMOS) technologies. A wide set of intermittent faults was injected, and coverages and latencies were measured from the simulation traces. In addition, a Markov model was generated for a reliability evaluation. Evangeline et al. [144] designed an SFI for digital circuits using the software from Xilinx. They modeled transient and permanent faults for stuck-at values, stuck-at bits, and faulty input data words. Salih et al. [145] proposed a fault injection model for highly automated vehicles. They developed a model of fault injection for the steering system to study the impact of steering system sensor failures. Their model was implemented in the MATLAB/Simulink environment. However, there are few scientific studies specifically on SiFI in HVAC systems. Hyvarinen et al. [146] categorized faults as design, installation, abrupt, and degradation. Examples in HVAC systems are sensor faults, such as invalid and incorrect sensor readings or noise, and actuator faults, such as stuck-at faults that account for 20% of energy waste, along with thermal discomfort and CO_2 emissions in HVAC systems [26, 27, 32, 116, 146]. Simulation-based fault injection models are beneficial for learning about system behavior by evaluating concrete fault scenarios. Some researchers developed simulation-based fault injection system models. Behravan et al. [25, 116, 117] implemented simulation-based fault injection models for DCV, and heating systems in multi-zone office buildings. In [116], Behravan et al. extended the simulated HVAC system models, providing them with FI capabilities of permanent stuck-at faults for the sensors, stuck-at

opened/closed damper actuators, and stuck-at heater actuators. Simulated temperature sensors and CO_2 sensors were also equipped with FI blocks. The supported fault types include gain faults, offset faults, and stuck-at values (e.g., stuck-at open/close in damper actuator, stuck-at off/on in heater actuator) [117]. Behravan et al. [147] also introduced a command-based fault injection framework with a compositional model in Matlab, where the Matlab code is mapped to the simulation blocks in Simulink. Further, Behravan et al. [18] proposed an automated FI tool to systematically inject different faults with different fault injection times.

3.4.2 Multiple-Fault Injection in HVAC Systems and Other Domains

FI provides insights into the system's behavior by deliberately introducing faults in different scenarios and conditions. Single-fault injection, single-fault detection and diagnosis have been investigated vastly. However, many HVAC systems (e.g., hospitals, airports, multi-story office buildings) are large-scale distributed systems with thousands of components, including sensors, actuators, computational nodes, and communication links which are vulnerable and prone to multiple faults. Today's FI frameworks for HVAC systems are based on this single-fault hypothesis. However, systems face multiple faults in reality [135]. Therefore, FI must investigate the effects of multiple faults simultaneously. This is in significant contrast to smaller scale systems (e.g., automotive electronics, medical equipment) where a single fault hypothesis is predominant [148] and considering a single fault at a given point in time is sufficient. Gil-Tomás et al [115] also express the importance of multiple faults due to technology scaling. FI experiments consist of simulation executions of the target system where any number of faults can be injected in one or multiple components at one or several points in time and with random fault time durations. In a simulation framework, faults can be injected using a set of input patterns via an automated FI code or FI dashboard in hardware or software.

Multiple faults have been investigated in domains other than HVAC systems. Yalcin et al. [149] have injected different hardware faults, such as transient, intermittent, permanent, and multi-bit faults, in simulations of processors. Multi-bit faults occur when a fault affects multiple bits simultaneously, such as spatial multi-bit upsets. Stroud et al. [150] have described single and multiple stuck-at-fault simulations for gate-level faults. Multiple faults are injected randomly or clustered for testing multiple fault detection capabilities. A list of fault groups has been considered for injecting multiple faults. Each fault group contains a number of gate-level stuck-at faults with a number of potential fault sites and possible combinations of single and multiple stuck signals at the gate level. Faults are injected randomly or in a cluster-based manner. The selection can be changed from a random sample to a deterministic function in the clustered FI. It modifies for clustered defects that tend to form a list of faults that are tightly coupled based on the degree of the cluster.

Tarrilo et al. [151] introduced a multiple-bit-flip FI platform. They triggered multiple faults in SRAM-based FPGAs, which are sensitive to soft errors, unexpected bit-flips, and critical errors. They injected single-event upsets (SEUs) and multiple-bit upsets (MBUs) for functional errors. The location of each malfunction is chosen from a list of locations. Kundu et al. [152] injected multiple faults to diagnose chips at the logic level. Arlat et al. [153] compared physical and software-based FI for the MARS fault-tolerant distributed real-time system. They addressed the respective impacts of FI techniques using a testbed and test scenarios. Zhong et al. [154] investigated operational single and multiple-fault impacts for HVAC systems under different climates. The effect of faults in HVAC systems may depend on climate changes. They also evaluated the system's impacts on thermal comfort, performance, and energy usage. They ranked

single and multiple faults for each climate condition. However, they did not carry out simulation-based multiple FI. Sangchoolie et al. [155] evaluated the impacts of single and multiple bit-flip errors. They used the open-source fault injector tool LLFI, which injects faults into the low-level virtual machine (LLVM). To realize the injection of multiple faults, they extended LLFI to facilitate the injection of multiple bit-flips. LLFI defines single bit-flip errors as time location pairs. To model multiple bit-flip errors, they developed the time location parameters that enable clustering the error space into different classes of errors. Tadeusiewicz et al. [156] introduced a method for simulating multiple faults in AC circuits. They used a systematic approach to perform the combination of multiple faults. The FI procedure uses admittance and impedance matrices for the faulty circuit nodes and fault combinations.

Lisboa et al. [157] described soft errors that may appear at the same time. Robust operators are introduced, and the operator's behavior is analyzed by simulating single and multiple faults. Papadimitriou et al. [158] introduced a multiple-fault injection methodology for digital circuits. Fault modeling at the register transfer level (RTL) can occur early in the design phase and facilitates the analysis of the gate-level models. They injected multiple faults by partitioning the RTL description of the circuits. Then, faults are injected in two groups. Firstly, faults are injected into one or more flip-flops, and the second group includes faults occurring in the combinational part of the circuits. Wang et al. [159] discussed hierarchical model-based diagnosis (MBD) for multiphase faults and hitting calculation sets (MHS), which serve for stability and reliability in power distribution networks. They calculated the system performance when the distributed network has multiple multiphase faults. The hierarchical MBD comprises different parts, including an offline model library, fault observations, and online identification of faulty elements. Takahashi et al. [160] introduced and simulated the diagnosis of single and multiple faults in combinational circuits. Kim et al. [161] introduced the modeling and simulation of multiple faults. The multiple-fault model consists of a set of lines. For example, the stuck-at fault consists of two lines, stuck-at-1, and stuck-at-0. Any fault combination can be modeled by activating these lines.

3.4.3 Experimental Evaluation in HVAC Systems

The experimental evaluation of HVAC systems in the design phase is an important subject [162] to enhance the system's efficiency, resource usage [163], economic effectiveness [164], thermal comfort [20], and reduce CO_2 emissions [7, 164–167]. Extensive research has presented experimental evaluations of energy consumption for HVAC systems. Antonopoulos et al. [168] proposed an experimental assessment of the energy savings of Air Conditioning (AC). Al-Deen et al. [169] evaluated the energy consumption of HVAC systems under different climate conditions. Vishwanath et al. [5] investigated the HVAC cooling energy consumption and cost associated with experiments conducted in large buildings. Andrés et al. [165] performed a real-scale experimental evaluation for regulating thermal control in lightweight constructions. Krajcik et al. [170] performed an experimental evaluation of residential rooms for sustainable heating/cooling and efficient energy consumption. Arteconi et al. [171] introduced an experimental assessment of a ground-coupled heat pump (GCH), an alternative to traditional systems for heating and cooling.

In this thesis, a realistic, Automated, and Simulation-based Fault Injection Framework (ASFIF) is introduced by combination of two simulation-based FI techniques, simulator command and simulation code modification for reliability evaluation in DCV and heating system. The FI framework incorporates saboteurs as fault injector blocks. In addition, an automated fault injector algorithm automatically activates fault cases with certain fault attributes according to the fault model. The proposed fault injection framework

supports a comprehensive range of faults and various fault attributes, including fault persistence, fault type, fault location, fault duration, and fault interarrival time. This framework considers noise in a DCV and heating system as a type of HVAC system since it has been demonstrated that any fault injection scenario is accompanied by impacts on energy consumption, occupancy comfort, and a fire risk. It also supports the reproducibility for a set of specific fault scenarios and random fault injection scenarios. The system model was implemented and simulated in Matlab/Simulink, and fault injector blocks were developed as Stateflow diagrams. An experimental evaluation serves as the assessment of the presented fault injection framework with a defined example of fault scenarios [11].

In this thesis, the FI framework is extended to support the injection of multiple faults with exact control of the timing, locality, and values in fault injection vectors [16]. Furthermore, the multiple fault injection framework requires an adaptable fault model for multiple-fault introduction. Therefore, modeling patterns of numerous faults in HVAC systems based on data from field failure rates and maintenance records are defined. A multi-dimensional fault model is defined, including the probability of the occurrence of different sensor and actuator faults. The automated multiple fault injection framework has been evaluated experimentally to analyze the system behavior under different faulty conditions [16]. The experimental results serve as a quantitative evaluation of key performance indicators (KPI) such as energy efficiency, air quality, and thermal comfort. Comprehensive experimental results provide insights into the system's behavior for concrete example scenarios using patterns of multiple faults. An overview of the SiFI techniques is provided and summarized in Table 3.

Table 3. Overview of simulation-based on fault injection techniques.

Ref.	Fault Profile and Attributes										
	Fault Type	**Fault Persistence**	**Fault Duration**	**Fault Interarrival Time**	**Fault Occurrence rate**	**Probability Distributions**	**Single Fault Injection**	**Multiple Fault Injection**	**Multiple Location**	**Simulation Technique**	**Injection Method**
[113]	Stuck-at value. Single bit-flip Double bit flip	Transient Semi-permanent	No	No	No	No	Yes	No	No	SUMO	Not specified
[123]	No	Transient	No	No	No	No	Yes	No	No	SAM	Not specified
[114]	Circuit faults	No	No	No	No	No	Yes	No	No	PSPICE and ADS	Not specified
[115]	Circuit faults Single or multiple	Intermittent	No	No	No	No	Yes	No	No	VHDL-based fault injection tool (VFIT)	Not specified
[144]	Stuck-at bit Stuck-at value Input data word	Transient Permanent 6-bit LFSR	No	Yes	No	No	Yes	No	No	Xilinx software and 4-bit adder and C17 benchmark circuit	Not specified
[18]	Stuck-at open/close Stuck-at off/on	Permanent	Yes	No	No	No	Yes	No	No	MATLAB/Simulink and MATLAB Programming	Automatic Injection by Script
[117]	Stuck-at value Stuck-at open/close Stuck-at off/on	Permanent	No	No	No	No	Yes	No	No	MATLAB/Simulink	Manual Injection by a visual and graphical dashboard

[116]	Gain fault Off-set fault Stuck-at value Stuck-at open/close Stuck-at off/on	Permanent	No	No	No	No	Yes	No	No	MATLAB/Simulink	Manual Injection by a visual and graphical dashboard
[11]	Gain fault Offset fault stuck-at value Stuck-at open/close Stuck-at off/on Out-of-bound fault Data-loss fault	Permanent Transient Static Intermittent with 2 or 3 Repetitions	Yes	Yes	No	Gaussian and Normal Distributions	Yes	No	No	MATLAB/Simulink and and MATLAB Programming Static Stateflow diagram implementation with 3 faulty states and 1 healthy sate	Automatic Injection by Script
[16]	Gain fault Offset fault stuck-at value Stuck-at open/close Stuck-at off/on Out-of-bound fault Data-loss fault	Permanent Dynamic Intermittent with n Repetitions	Yes	Yes	Yes	Gaussian and Normal Distributions	Yes	Yes	Yes	MATLAB/Simulink and and MATLAB/Programming Dynamic Stateflow diagram implementation with 1 faulty state and 1 healthy sate	Automatic Injection by Script Using multi-dimensional attributes

3.5 State-of-the-art of Fault Detection and Diagnosis Techniques in HVAC Systems

In HVAC systems, a fault occurrence may decrease energy efficiency, system performance, and occupant discomfort and lead to dangerous conditions in complex and critical infrastructures (e.g., inadequate ventilation upon a fire in a hospital or an airport). Therefore, optimization and control strategies such as FDD techniques and testing play a key role in these systems to improve maintenance and cost of energy. Ahamed et al. [1] reviewed the application of Computational Intelligence (CI) for prediction, optimization, control, and diagnosis in HVAC systems. They have classified the CI techniques into (1) prediction, including artificial neural network and support vector machines, (2) optimization, including stochastic approaches and Intelligent Agents (IG), and (3) control and diagnosis, including expert systems, fuzzy logic, and pattern recognition-based methods. Pattern recognition-based methods consist of different categories such as principal component analysis, Bayesian networks, clustering, and pattern matching. CI is an advanced research field using computation technologies and was initiated by the Institute of Electrical and Electronics Engineers (IEEE) Neural Networks Council in 1990 [1]. One of the most common CI techniques in HVAC systems are fuzzy logic-based controllers and detectors to overcome system uncertainties [1].

Achieving optimized operations in HVAC system is challenging due to the non-linearities of system indicators such as energy consumption. Indoor air quality, CO_2 concentration, energy management, and thermal comfort are major optimization objectives in buildings [1]. Many authors have studied these optimization objectives in HVAC systems. For example, Yu et al. [172] reformulated the energy cost minimization problem as a Markov game. An HVAC control algorithm has been proposed to solve the Markov game based on multi-agent deep Reinforcement Learning (RL). In this article, the authors formulated a long-term HVAC energy cost minimization problem related to multi-zone commercial

buildings. The solution does not require model building with thermal dynamics and has been formulated as a Markov game. It used an HVAC control algorithm to solve the Markov game. Yu et al. [173] have proposed another algorithm based on reinforcement learning for smart home energy management. They formulated the problem as a Markov Decision Process (MDP). They presented an energy management algorithm based on RL policy gradients. Temporally coupled operational constraints are associated with energy storage systems and HVAC systems. The authors have proposed an energy management algorithm based on deep deterministic policy gradients (DDPGs) to address the challenge that actions affect future decisions. Based on the current observations, the algorithm makes decisions about Energy Storage System (ESS) charging/discharging power and HVAC input power. Huang et al. [174] proposed a machine learning approach called Non-Intrusive Load Monitoring (NILM) to disaggregate heating usage. High-frequency measurements are mapped to knowledge that can improve energy efficiency in the residential sector. The main goal of the work is to use smart measurement data to identify heating and cooling usage levels for a smart home. This method uses a Markov model to capture the dependence of heating usage on the outdoor temperature. This proposed method provides details on heating usage patterns and is more flexible in incorporating other system-specific information. Wu et al. [175] have formulated a multi-room HVAC control problem as an event-based optimization, where decisions are made when certain events occur. They developed an approximate solution to simplify the calculation process, focusing on local event-based policies. The size of the state and space increases exponentially with the number of rooms. It could become extremely large for practical problems. It is challenging to solve the problem directly with MDPs, and event-based optimization provides an alternative. Shanin et al. [176] have developed software solutions for a housing and utility condition monitoring system. Their system processes sensor readings using statistical and probabilistic models such as linear regression and the Hidden Markov model to classify equipment's regular and faulty operating modes. An AHU for HVAC systems conditions and circulates air in rooms. The cooling coil and the supply air fan are essential components of an AHU.

Fault detection and diagnosis is a process that localizes faults and determines the fault type [177]. FDD methods are developed with several objectives including cost-effective maintenance policy, improving productivity standards and ensuring safety-critical aspects [178]. Many techniques for fault diagnosis have been pioneered since the 1960s [177], and methods have been reviewed and classified widely in many studies. Steinder et al. [52] have concentrated on fault localization techniques in complex communication systems to find the exact source of a failure from. They have classified the fault localization techniques into three categories containing Artificial Intelligence (AI) techniques, model traversing techniques, and fault propagation models including Bayesian Networks (BNs). Park et al. [179] reviewed FDD methods in industrial processes. They have provided a general implementation procedure for the FDD methods consisting of four steps: industrial processes or systems, data collection and analysis, feature extraction and selection, and model training and validation. They have classified FDD methods based on the system characteristics into data-driven methods further categorized into dynamic, nonlinear, non-gaussian, time-varying/multimode, and non-stationary systems, model-based methods further categorized into quantitative model-based methods, qualitative model-based methods, and process history-based methods, knowledge-based methods and hybrid methods. Isermann [180] has also classified the analytical fault-detection methods into detection with single signals, and detection with multiple signals and process model-based and multi-variant data analysis. Isermann has also classified the fault diagnosis methods into classification methods, which include pattern recognition, statistical classification, approximation methods, density-based methods, AI methods, and inference methods which are binary reasoning and approximate reasoning. The statistical classification method includes the Bayes classifier and decision tree. Miljković

[181] classified the fault detection methods into data methods and signal models, process model-based methods, and knowledge-based methods. FDD techniques in the building energy system field can also be categorized into knowledge-driven and data-driven methods [182]. Knowledge-driven methods resemble the diagnostic thinking of domain experts with a high capacity for reasoning uncertainties; they can work with different fault severities. In contrast, methods in the data-driven category mainly rely on similarities and patterns [182]. Each category has its strengths and shortcomings. Yang et al. [177] investigated network fault diagnosis methods and discussed their advantages and disadvantages. They have classified the fault diagnosis techniques into model-based methods, processing-based methods, and knowledge-based methods. Ahamed et al. [1] and Du et al. [183] have also classified FDD techniques into a rule-based method that does not need any model and highly relies on expert knowledge to extract the rules, a model-based method, and data-driven methods. Zhao et al. [182] have also classified FDD techniques into data-driven, and knowledge-driven approaches and have mentioned the strengths and shortcomings of each category specifically. For example, data-driven methods demand a high amount of training data and knowledge-driven based methods highly depend on expert knowledge and have no automatic capabilities to improve diagnostic efficiency. Knowledge-driven methods diagnose different faults based on their severities and are more understandable whereas data-driven methods are black-box methods with low understandability regarding the approach and results. Zhao mentioned that hybrid methods combine two or more approaches and are thus able to obtain the advantages of both types of methods [182]. All available FDD categorizations have been summarized in Table 4 , including their techniques and associated advantages and disadvantages.

Table 4. FDD method categorization with advantages and disadvantages

FDD methods categorizations	**Adopted FDD techniques**	**Advantages**	**Disadvantages**
Signal-processing based method [177, 179, 181]	Symptom extraction	Achieved easily Avoids human mistakes	Misreporting of the false alarms
Data-driven methods [1, 25, 179, 182, 184]	Unsupervised-learning based	High accuracy	Massive data is required for training
	Classification-based		Automation
	Neural Network		Low understandability
Model-based method [1, 177, 179, 181]	State estimation by mathematical statistics	Close to the truth	
	Analytic functions		Application limitation
	Qualitative model-based (based on rules)[185–187]		Expert knowledge dependency
	Quantitative model-based (based on physics)[185–187]		

Knowledge-based method [25, 177, 179, 181, 182, 184]	Diagnostic rule-based [1] Expert system, e.g., symptom extraction by an expert [118] Fault tree [188] Inference-based Bayesian-inference Network (BN) [189–191] Fuzzy-inference theory Grey theory Petri networks	High accuracy Intelligent	Supported-theory limitation Expert knowledge dependency More time Consuming Lack of automation Simpler implementation Cost effective design
Hybrid Methods [179]	BN and machine learning BN and fuzzy logic [62, 192–195] BN and signed directed graphs [196] BN, fuzzy logic, and classifiers Fuzzy logic and petri nets [197]	High Accuracy Intelligent	Automation High understandability Time efficiency

Due to the complexity and characteristics of the HVAC processes, sometimes it is challenging to perform FDD techniques without knowledge of the processes. Therefore, it is required to consider the rules and system's process data, e.g., signals or sensor measurements, when using FDD techniques. Examples of these techniques are the cause-effect analysis approach, Neural Network (NN) approach by specifying of the faults and process variables relationship, and the combination of NN and fuzzy logic [18, 179]. Therefore, in this thesis, a hybrid diagnosis technique has been developed to combine the advantages of different techniques and to obtain more accurate and efficient diagnostic results. For this aim, the applied techniques and their state-of-the-art including knowledge-based and hybrid diagnosis techniques have been investigated in detail.

3.5.1 Knowledge-Based Fault Detection and Diagnosis Techniques

Different knowledge-based approaches in HVAC systems were designed based on the models of the thermal dynamics of the environment. For example, Behravan et al. [118] established a Diagnostic-Directed Acyclic Graph (DDAG) based on explicit knowledge of HVAC systems. This technique was tested based on DCV and heating models designed using MATLAB/Simulink. The model consists of components, including CO_2 sensors, damper actuators, temperature sensors, and heater actuators. A fault injection

framework was also designed to inject faults into the system artificially. The system behavior in different faulty conditions and non-faulty conditions can be monitored. This fault diagnosis technique uses explicit knowledge with expert rules. According to the knowledge based DDAG, once a symptom is detected, a series of diagnostic tasks, such as a combination of plausibility checks and component health evaluations based on the input signals and measurements, finds the failure cause. Shiazoki et al. [198] have also developed a FDD method using a Signed Directed Graph (SDG) with low efforts to detect the symptoms of the faults and root causes. The accuracy of the diagnosis depends on setting right thresholds, which is demanding and time-consuming.

Fuzzy logic is a common knowledge-based solution to overcome uncertainties inspired by human behavior for reasoning about imprecise problems [1, 199] and was introduced by Lotfi Zadeh in 1965 [200, 201]. A fuzzy logic system formalizes approximate reasoning. It means that fuzzy models represent vagueness information with a degree number between 0 and 1 (i.e., probability) using a reasoning mechanism [202]. This value is calculated by a Membership Function (MF) known as the Membership Degree (MD). Fuzzy models are usually constructed by fuzzification, an inference engine, and defuzzification phases [1]. There are different types of membership functions, such as triangular, trapezoidal, piecewise linear, gaussian, and singleton. MF calculates the probability and MFs can be defined based on the system requirements and characteristics. Figure 6 illustrates the fuzzy logic system and its procedure in detail.

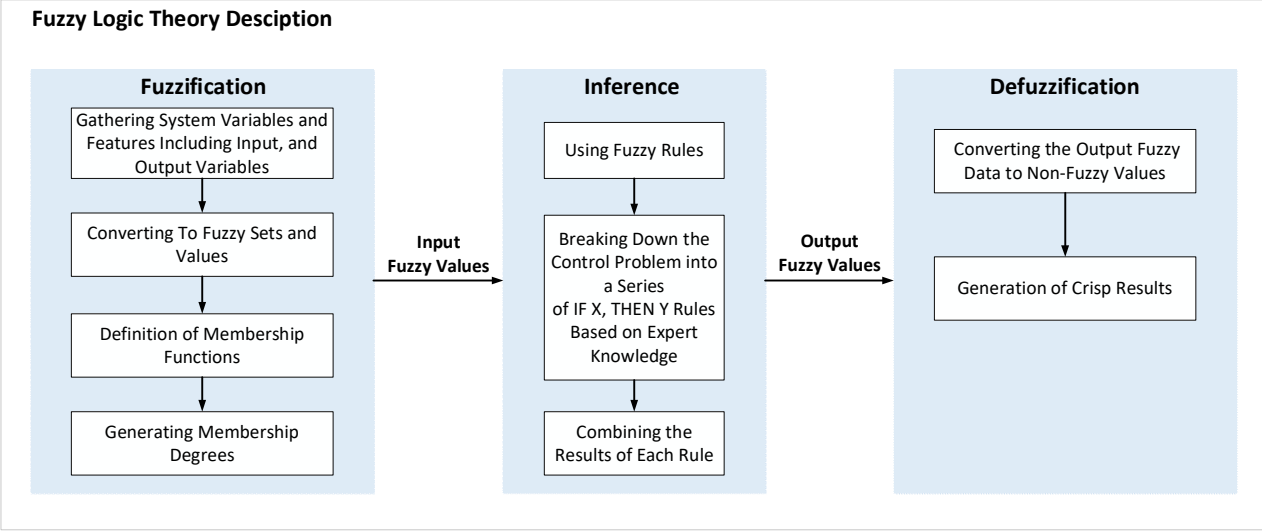

Figure 6. Fuzzy logic system with three phases of fuzzification, inference, and defuzzification [203]

Fuzzy logic has a wide application in FDD techniques due to low design cost and easier implementation [204]. Fuzzy systems are rule-based, and the output of the system is not a simple binary decision of *"Fault"* or *"No-Fault"*; rather, the severity of the fault will be provided as a fuzzy output value [181, 204]. Kolokotsa [203] provides a complete overview of fuzzy logic applications for building technology in the case of (1) indoor comfort such as modeling the thermal sensation, control of the thermal environment, indoor air quality, and (2) energy management such as energy planning, energy load prediction, optimization, and fault detection.

Some authors deploy a pure fuzzy approach in HVAC systems. For example, Dexter et al. [205] have utilized fuzzy logic to model the errors and ambiguities and reduce false alarms. Eftekhari et al. [206] have presented a fuzzy control strategy for natural ventilation in a test room. Sulaiman et al. [204] have

introduced a fault detection method based on fuzzy logic in air supply dampers of air handling units in MATLAB/Simulink. They have used three indicators for fuzzy logic labels: “No Fault” which represents the expected behavior or near-to-normal behavior of the system, ”Almost Fault” which represents 30% to 50% difference from normal behavior, and “Fault” that represents a problem in the system behavior. Allen et al. [207] demonstrated a health monitor strategy for the cooling mode of an actual variable air volume (VAV) unit in a commercial building to improve the HVAC and Build Automation System (BAS) load. Their method uses a fuzzy neural network for fault classifications. They first defined the system's inputs and labeled them based on their characteristics. For example, the damper position is labeled with “Closed”, “Open” with 0% and 100% range that specifies the damper situation, and air flow is labeled with “Min” and ”Max”.

The Bayesian Belief Network (BBN) is one of the important knowledge-based approaches in fault diagnosis methods based on probability theory for modeling uncertain knowledge and reasoning based on conditions of uncertainty, probabilities, and graph theory [193]. BBN was introduced by J. Pearl in the 1980s [208]. A Dynamic Bayesian Network (DBN) is a graphical model based on the probabilistic where nodes represent random variables and directed arcs/edges represent conditional dependencies. DBNs are an adequate formalism for representing and reasoning under uncertain conditions. However, they do not scale well for complex systems. For example, Vlachopoulou et al. [191] developed a dynamic Bayesian model for HVAC systems. They derived and trained a DBN to model aggregated loads of HVAC systems. DBNs are able to change the behavior of the model over time which is an essential feature for loading the model. Because load varies highly over time. Hector [190] proposed an algorithm for sensor validation by representing the relationships between the variables using a DBN. The validation process is based on probabilistic propagation. However, this work does not consider the model complexity and the distributed nature of components. To overcome these limitations, Garcia et al. [189] proposed a distributed probabilistic model for fault diagnosis, which is an extension of the DBNs for representing large domains and complex systems using Multiple Sectioned Bayesian Networks (MSBN). DBNs can be created locally and globally for communication with adjacent sections. The final step in the construction of the model is the inference in the MSBN, which consist of two fundamental steps: the inference at the global level using the junction tree technique and the inference to guarantee the global consistency from the construction of linked cluster trees between adjacent components for the passage of messages.

BN is utilized for many FDD methods. For example, Shi et al. [195] introduced a distributed fault diagnosis method to represent the probabilistic dependencies between faults and symptoms in a VAV AHU. Their model comprises a detection agent, a diagnostic agent, and an evaluation agent. Fault detection produces symptoms by gathering more information from sensor measurements. The fault diagnosis agent gathers the fault’s related symptoms and uses a DBN for the diagnosis process. DBN are applied due to their extensibility and ease of use in distributed systems. A DBN can diagnose persistent and transient faults. Evaluation agents determine the faults and symptoms impacts. However, the probability calculation is manual and computed with expert knowledge, which is inappropriate for a large-scale system. Zhao et al. [209] explain a generic diagnostic BN framework for chiller fault diagnosis. They have merged all system information in a diagnostic BN to simulate expert knowledge in practice. Diagnostic tasks determine faults based on one or multiple symptom observations. Faults, symptoms, and factors are included in the DBN in three different layers. Prior probabilities of faults are computed based on frequencies of faults, and conditional probabilities represent the node relations. Faults are diagnosed in the fault layer by calculating the posterior probabilities and two rules that determine fault differences by applying thresholds. However, they calculated the conditional probabilities using historical information, fault frequencies, and

maintenance records. Xiao et al. [210] introduced a diagnostic BN strategy in VAV. The entire structure of the BN depends on expert knowledge and the rules for defining the system states. Zhao et al. [211, 212] developed four DBNs to diagnose faults in air handling units (AHUs) in buildings. In this FDD method, establishing the BBNs and their nodes highly depends on expert rules. Therefore, it is desirable to decrease dependency on experts by applying automation.

3.5.2 State-of-the-art in Hybrid Single-Fault Detection and Diagnosis Techniques

Large-scaled HVAC systems comprise numerous components and units that need accurate and effective FDD methods to manage probable faults that degrade the system functionality over time, e.g., permanent stuck-at faults that lead to energy dissipation [213]. The most effective FDD solutions are developed in a hybrid manner [182]. Due to the characteristics of the HVAC system processes, expert knowledge may assist in developing the FDD methods, e.g., fuzzy logic as a knowledge-based method can facilitate this requirement (extracting the system information, e.g., labels and rules by experts). BBN allow modeling nonlinear dynamics and discrete systems appropriately and they are an effective solution for HVAC systems with nonlinear processes. However, defining a suitable conditional likelihood density function is critical in systems with discrete and continuous variables [18, 75]. BBN is also an effective method for modeling probabilistic relationships between symptoms and failures [214]. However, probabilistic modeling in case of multiple or independent symptoms is also challenging [214]. Combined diagnostic methods can use the advantages of each method to improve efficiency and accuracy [182], e.g., several examples admit the effectiveness of Fuzzy Bayesian Belief Networks (FBBNs) in solving uncertain problems by applying fuzzy sets to calculate BBN parameters (such as conditional table and nodes probabilities) [199].

There are several hybrid FDD methods with a combination of knowledge-driven and data-driven approaches, such as fuzzy theory and colony system [215], BN and ML [216], BN and Signed Directed Graph (SDG) [196], BN and hidden Markov [217], BN, fault tree, and fuzzy theory [218], BN and Genetic Algorithm (GA) [219], BN and fuzzy theory [178, 192, 193, 199, 214, 220]. For instance, Kuo et al. [221] explain a hybrid diagnostic method by integrating the fuzzy theory and Ant System-based Clustering Algorithm (ASCA). Their method is case-based, and cases are fuzzified. The same cases are grouped into different clusters. New cases should find the closest group. The fuzzy theory has been applied to find the similarities among new cases and other groups. Antnet is applicable for finding the routes in the communication network. During the seeking food procedure, ants use a specific fragrance to save the route, which is named pheromone. Therefore, other ants follow the paths with a higher density of pheromones, leading to finding the shorter path. They also used the Antnet method to choose the most similar groups and classify them. Hence the time for finding a similar case decreases.

Chiu et al. have also introduced a fuzzy-Bayesian classifier with Case-Based Reasoning (CBR) to solve diagnosis problems [192]. They have used fuzzy theory to define conditional density functions of BBNs to overcome the problem of continuous attributes. Fuzzy theory enables defining the desired conditional functions based on the system specifications. Many research studies have proven the accuracy and efficiency of integrating the fuzzy logic theory and Bayesian networks for decision-making applications, uncertain knowledge representation, and reasoning [200, 215, 222].

Hu et al. [216] have introduced an intelligent fault diagnosis Bayesian network for refrigerant flow air conditioning systems. The diagnosis network is constructed by BN, including two main elements of structure and parameters. The structure is obtained by ML and expert knowledge to map the relations, and the relations are obtained by parameters which are prior probabilities and conditional tables. However, this

method of capturing sufficient training data for data-driven methods is not cost-effective for any faulty condition [154].

SDGs are not entirely suitable for modeling complex logical relations. Therefore, Di Peng et al. [168] have proposed a Multilogic Probabilistic SDG (MPSDG) by integrating SDG and BN. The probabilities and conditional tables are calculated by historical malfunction and failure frequencies. They consider offline modeling and online diagnosis phases. In the offline mode, they utilize the historical failure frequencies to evaluate the prior probabilities of each reason node and directed edges. All system variables are monitored consistently. Once the measured value is out-of-thresholds, an alarm signal (i.e., symptom) starts the diagnosis process in the SDG to find the fault reason node. An MPSDG includes more accurate information about the system. The diagnosis process in MPSDG is divided into two parts: finding the candidate's fault reasons and probability computation. Afterward, faults with high probabilities are ranked to determine the most probable fault. However, constructing the MPSDG without historical data is impossible.

A dynamic FDD method is introduced by Don et al. [217] by integrating the BN and Hidden Markov Model (HMM). The HMM serves for anomaly detection and the BN diagnoses the root cause of faults. HMM should be trained by historical operations and process. BN also uses a log-likelihoods (LL) probability distribution to calculate the conditional probability table using historical data. When the HMM detect anomaly and BN probabilities are updated at the same time can be evidence for diagnosis. However, probability computation and graphs construction and training demand historical data.

Qiu et al. [214] demonstrate a diagnostic method for remote print defects (symptoms) by integrating fuzzy theory and BN methods. They have mentioned a common problem for diagnostic systems: mapping the exact failures to exact symptoms is challenging because a failure is usually the reason for several symptoms. They have suggested probabilistic methods to link symptoms and failures. BN is an effective method for modeling probability relations. However, probabilistic modeling is demanding in the case of multiple or independent symptoms. The BN’s prior probabilities can be calculated from operating data, repairing data over sufficient time, or consulting expert knowledge. When prior probabilities are not possible, the fuzzy theory is an appropriate solution to find the probabilities of fuzzy variables. However, they have only used a single BBN for the FDD process without using any data-driven method.

D’Angelo et al. [178] explain a fuzzy-Bayesian method for fault detection in the machine stator winding. They have used fuzzy theory for processing input uncertainties of the BN. However, they have not applied any data-driven method. Bi et al. [218] have combined fuzzy theory, fault trees, and BN for fault diagnosis of a rotor in a pumping station. Their proposed T-S fuzzy gate fault tree has solved the problem of the logical relationship between events and probabilities but not in complex reasoning. They have used BN and fuzzy theory combinations to solve the calculation of the conditional probability table. However, they have not used any data-driven method.

Zhao et al. [220] have presented a fault diagnosis method by combining fuzzy theory and BN methods in train control systems. The logical relationship of events in the fault tree is related to the conditional probability table of the BN. Fuzzy theory is used to convert the expert knowledge to the probability rates for each failure divided into seven categories including “impossibility”, “less likely”, “small possibility”, “medium possibility”, “more likely”, “most likely”, “must happened”. They have used sample data to calculate node probabilities collected by the expert experience.

Tang et al. [193] have the same strategy for constructing the BN with a fault tree and combining the method with fuzzy logic for machinery fault diagnosis. They have used expert knowledge to define

fuzzy rules. They have fuzzified the system variables with the labels “large”, “small” and “medium”. However, their approach is limited to expert knowledge and uses a single BN for the FDD process.

Yao et al. [199] have proposed a fault diagnosis method and reliability prediction. Their method models relationships among the system components with high complexity. They have used a Fuzzy Dynamic Bayesian Network (FDBN) method for combining various test information for modeling the system reliability assessment. The BN is constructed with system failures and their corresponding failure rates. The conditional table describes the relationship among components, and the BN handles the fuzzy information. A DBN is applied to capture dynamic variables over time and model dynamic systems. In addition, the fuzzy theory has been applied to evaluate the system’s reliability with different language variables, expert knowledge, and scoring fuzzy values for root nodes. The quantitative analysis of an FDBN can proceed with forwarding (or predictive) analysis and backward (or diagnostic) analysis. Then, one finds the order of failures based on their rates according to the expert assessment. However, in this method, the BN is not constructed by system variable correlations and fault statistics and historical data are used for calculating the prior probabilities.

In this thesis, a fuzzy Bayesian belief network has been developed based on the method proposed by Intan et al. [62]. Intan has used a Fuzzy-Bayesian method to track and analyze the medical records to find the relations between different diseases and other patient factors, e.g., education and the related diseases. Intan extends the MI concept by applying fuzzy theory for BBN construction [194]. They have used patients’ data records for probability calculation of the BN and have fuzzified patient information. The FBBN is constructed by measuring the dependency and casual relations between pairs of nodes for data analysis. However, they have not used the FBBN for diagnosis.

In this thesis, the FBBN is developed by integrating the fuzzy theory and BBN, where a classification algorithm helps the FBBN for fault detection and diagnosis. System attributes such as sensory measurements serve as continuous attributes and actuator measurements as discrete attributes. These attributes are fuzzified to facilitate the FBBN probabilities. Expert knowledge should help appropriately for the fuzzy sets’ introduction. The BBN construction is based on the system attribute correlations when the MI indicator has a positive value. MI determines the dependency degree and is calculated for finding the similarities of measured system signals or signal variations to detect real-world anomalies. The diagnostic method comprises two modes: offline training mode and online diagnosis mode. In each online diagnosis process, all relations, directions, and probabilities of fuzzified system attributes should be computed and saved in a table. Then, in the online mode, the classification algorithm must compare the online table with the corresponding tables in the offline library. The offline library includes all possible fault conditions. To validate its accuracy, the introduced and generic FBBN method has been applied to DCV and heating systems. The results show higher accuracy than related work [18].

3.6 Description of Research Gaps

The related work section explains various research fields and the state-of-the-art. The contributions are discussed in the following with respect to the research gaps.

Research Gap 1: Component-Based System model with Multiple-Fault Injection Framework for DCV and Heating Systems to Ensure Scalability and Universality: In prior research [21], a component-based simulation model has been introduced to develop the HVAC system of multi-floor buildings.

However, the fault injection process was static, manual and activated for few types of faults trough a dashboard without automation for FI and scenario generation. This thesis provides contributions beyond the state-of-the-art by developing a simulation-based composable model to activate multiple faults in a multi-floor building through a comprehensive fault model with extensive attributes such as type, time, duration, interarrival time, persistence, and fault occurrence rate in realistic scenarios. For each fault the address is defined (fault target including floor number, room number and component number) using indexing and fault injection matrices. When generating the component-based model, each floor, room and component gets a specific index that is an appropriate factor for the multiple-fault injection procedure. Multiple fault injection has been developed based on the multi-dimensional matrices for initializing the system attributes, e.g., intermittent fault injection times. Indexes are used to access each element of the fault injection matrices. An automatic fault injection algorithm cooperates with the component-based model to activate different fault scenarios. Fault scenarios can be designed or initialized randomly. Injected fault data is saved as objects containing all fault attributes in a library. The Fault library can be used for the FDD techniques. The component-based system model can interact with social systems (e.g., system designers, users) through a command panel to get the system configuration values to generate the system structure based on system requirements, e.g., with different floor and room numbers or nominal values.

Research Gap 2: Modeling Patterns of Multiple Faults in DCV and Heating Systems Based on Data from Field Failure Rates and Maintenance Records: In prior research [223–227], FDD techniques were introduced with fault attributes derived from maintenance records of HVAC systems. However, only individual faults were addressed, whereas the consideration of combinations of faults is essential for large-scale electronic systems. This thesis provides contributions beyond the state-of-the-art by introducing fault models and patterns for combinations of multiple faults, which consider fault attributes (e.g., occurrence rates, locality, persistence) from maintenance records and serve for FI and FDD in HVAC systems. Each fault combination has a specific occurrence rate based on the fault attributes, such as fault types. The fault model and occurrence rate are compatible with different environmental conditions by mapping the fault occurrence to real-world maintenance records [16].

Research Gap 3: Injecting Multiple Faults into a DCV and Heating System: In previous works, individual faults were injected into HVAC systems [11, 89, 90, 96, 117]. The injection of multiple faults was considered in other domains, such as semiconductor technology [135, 149, 152, 153, 156, 159, 161]. Hence, injecting multiple faults with corresponding attributes is a research gap for DCV and heating systems. This thesis goes beyond state-of-the-art by introducing a framework for injecting multiple faults with corresponding fault attributes while observing the propagation of the faults from the component level to the system level and the manifestation of system-level properties (e.g., energy efficiency and occupant comfort). The introduced framework is generic and scalable and can be instantiated for different building structures and fault combinations. The fault attributes are expressed using matrices, which are extended in size and dimensions to support more complex structures with additional components, zones, and buildings. The FI occurs using an HVAC simulation framework with realistic physical models of thermodynamics, heat/air flow transfer, and environmental conditions [16].

Research Gap 4: Experimentally Evaluating the Effects of Multiple Faults on the Behavior of DCV and Heating Systems: Experimental evaluations of HVAC systems were carried out in [164, 165, 168, 169] to monitor the system behavior in the presence of faults. However, in the field of DCV and heating

systems, the experimental evaluations of multiple faults in combination with different environmental conditions have not been published, and no such experimental data is available. This research gap is a barrier to developing fault-tolerance techniques and the dependability evaluation of HVAC systems. The FI framework introduced in this thesis monitors the system behavior for different fault patterns and multiple fault combinations defined by the user. The FI framework is generic and enables the evaluation of quality attributes such as heating cost, energy consumption, occupant comfort, indoor temperature, and air quality [16].

Research Gap 5: Single-Fault Detection and Diagnosis Service: Single-Fault Detection and Diagnosis (SFDD) methods have been accomplished in [118, 178, 189, 191, 193, 195, 198, 199, 203–207, 209–212, 214, 215, 217, 218, 220, 221] to diagnose the faulty conditions in different application domains. However, some methods are purely knowledge-based. Expert knowledge and experience have a significant role in developing the methods (e.g., BN construction and probability table calculation), which should be improved by applying suitable data-driven approaches or automation strategies. BN is an effective method for modeling uncertainties of HVAC systems that can be extended for complex structures. However, the BN construction should be cost-effective regarding operation time and independence on historical data and expert knowledge. Therefore, this thesis goes beyond state-of-the-art by introducing a fuzzy Bayesian belief network that performs the network construction based on fuzzified system attributes (knowledge-driven approach) and finding correlations between them using MI indicators with less expert effort. Finally, an automatic classifier algorithm (data-driven approach) enables fault diagnosis by classifying faults based on their similarities with online, actual system execution, and offline libraries of various faults.

4 System Model of Simulation Environment of HVAC System

This chapter presents the DCV and heating system model representing contemporary HVAC systems with manifold components as a foundation for the FI framework and the FDD techniques. In this model, embedded processing units coordinate the sensor and actuator nodes to control the air quality and thermal conditions of a multi-zone office building. This chapter describes the physical model of a multi-zone building, including the DCV and heating systems as composable system models generated in a flexible manner for different numbers of zones. Behravan [32][25] has introduced the physical model for the multi-zone building and composable structure that is applied to evaluate the proposed techniques in this thesis.

4.1 Physical Model of Multi-Zone Target System

HVAC systems are macroscale-distributed embedded systems and are among the largest energy consumers in buildings since they must maintain comfortable thermal conditions. HVAC systems consist of different kinds of sensors, actuators, and controllers, which are interconnected with various wire-bound and wireless networks. This section elucidates the system model of a DCV and heating system [25] and its embedded subsystems.

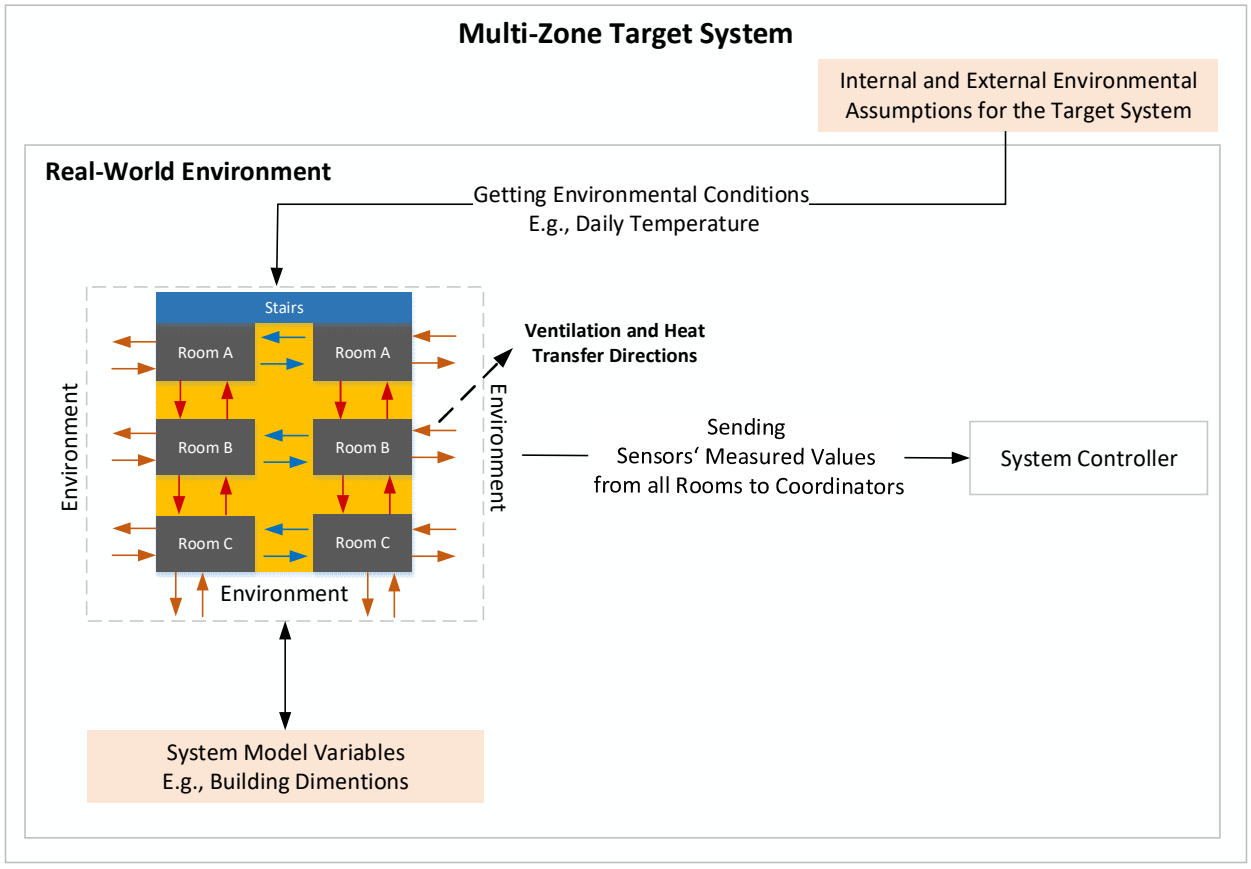

Figure 7. The overall scheme of the multi-zone target system model used to validate this thesis techniques [25].

The utilized system model of the HVAC system comprises a typical building with several rooms on different floors, e.g., an office building with six rooms and a corridor as part of a floor. Each room is typically equipped with multiple electronic components, such as sensors and actuators. Figure 7 illustrates an overall scheme for a multi-zone target system model used to validate the techniques in this thesis. The system model is based on the thermal dependencies among distinct zones. The arrows in Figure 7 demonstrate the thermal dependencies among rooms, outdoor and indoor environments, and how the system and building assumptions are applied to achieve measured outputs. Figure 8 shows the interrelation and external interactions of the system components in a room. Each room consists of different subsystems, such as the heating (thermal) subsystem and demand-controlled ventilation subsystem, including airflow subsystem, sensors, and actuators. The model's assumptions are based on natural environmental heating and ventilation. For each zone, the heat transfer differential balance formulas have been modeled.

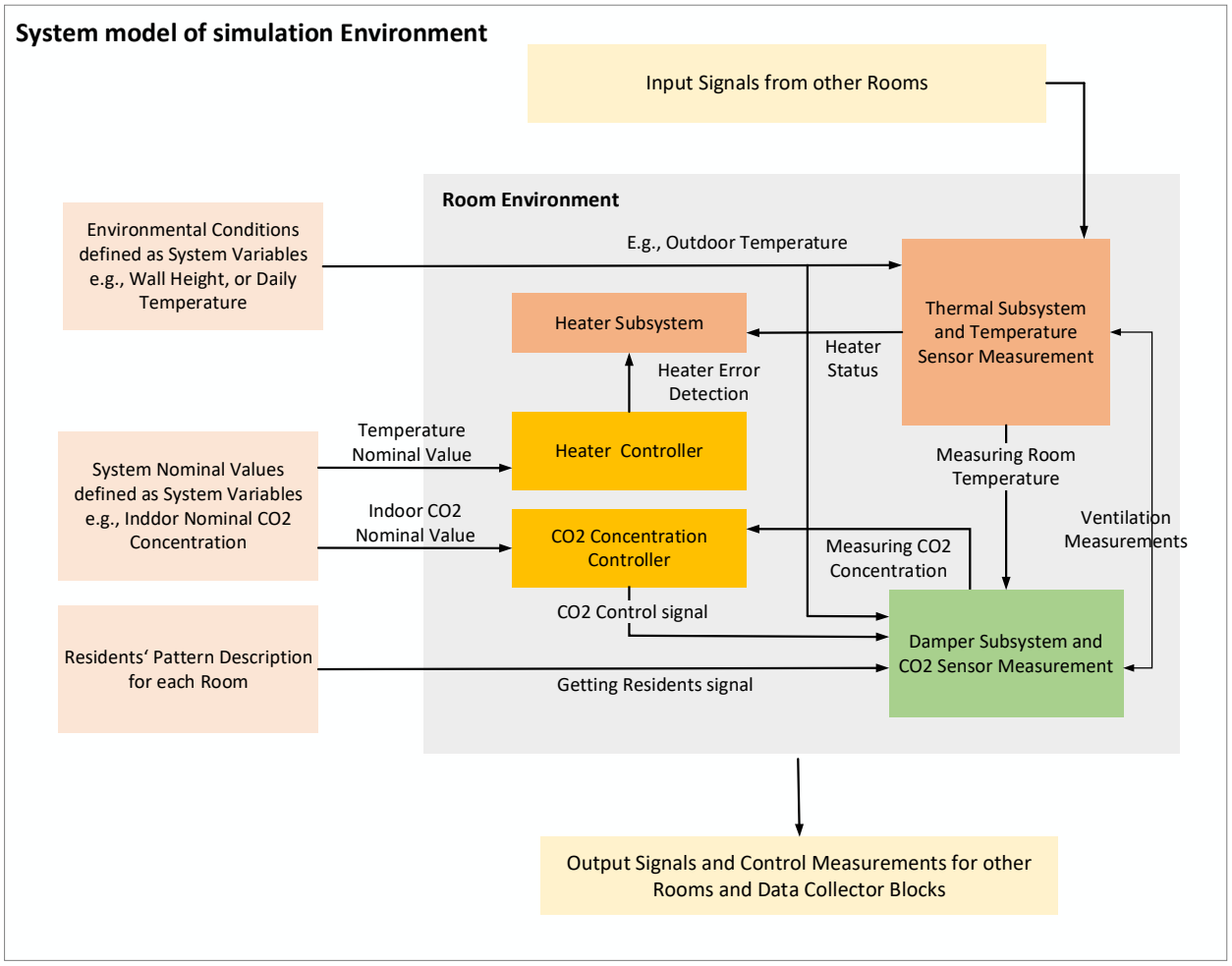

Figure 8. The system model of the simulation environment description illustrates system components and their interrelations with the room's environment.

System configurations and heat transfer computations in different subsystems are based on the thermal and building assumptions e.g., daily temperature . In addition, sensor nodes send their measured values to the controller via a coordinator. Afterward, the controller processes the received measurements, specifies the commands, and exerts them on the actuators, e.g., heater and damper actuator, to perform appropriate response actions [25]. For example, in case of a high indoor CO_2 concentration for high occupancy numbers

in a zone, the damper actuator should become open to bring fresh air into the zone. The heater control ensures thermal comfort for the occupants. Furthermore, in critical infrastructures, such as airports and hospitals, HVAC systems serve an essential role in emergencies. For example, in the case of a fire, HVAC systems need to remove toxic gases while slowing down the expansion of the fire. The heating subsystem supplies heat and thermal energy for the entire indoor space of the multi-zone building to balance the internal thermal and air conditions and keep it at a comfort level for residents. Designing the thermal model depends on several factors, including physical and thermodynamic characteristics of the building (e.g., walls, ceilings, floors, indoor air, and internal heat transfer), environmental conditions (e.g., temperature, pressure, and wind speed), heating system type, control strategies, user requirements (e.g., desired temperature, and indoor air quality) and occupant's behavior (e.g., the number of occupants over time) [25]. "Heat transfer" refers to the transmitted thermal energy due to the spatial temperature difference that can be stored in thermal heat capacities and transmitted through these elements [25]. Parameters to be calculated for internal heat transfer are conduction, convection, radiation, and ventilation [25].

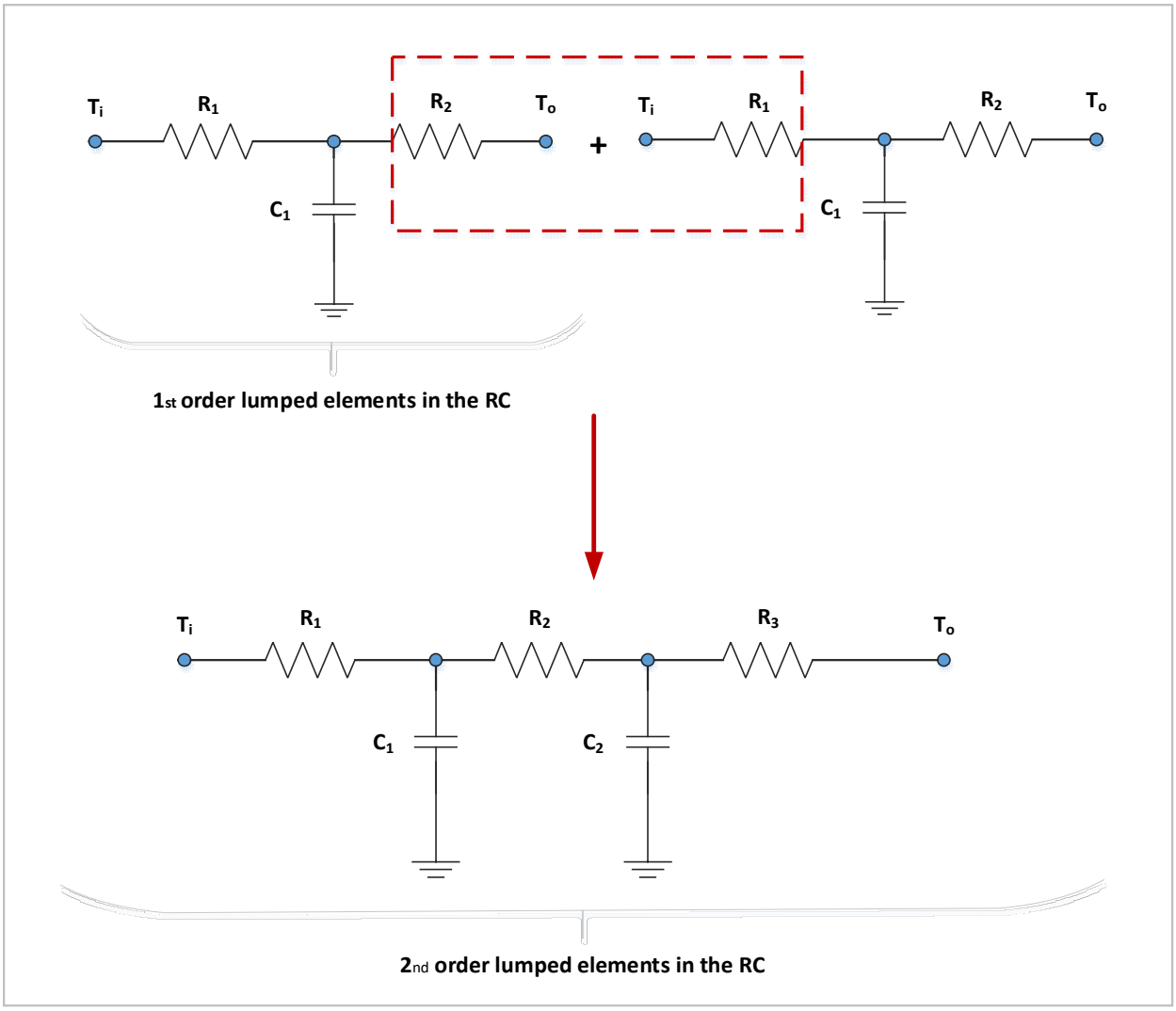

Figure 9. Lumped elements in the RC approach in two different orders [25].

The thermal system is modeled based on the lumped capacitance method introduced by Hudson and Underwood [228] based on a topology between thermal systems and the RC electric circuit model [25, 229, 230]. It means that the physical descriptions of the system model are simplified to discrete numbers of temperature elements named “lumps” that construct an energy balance equation to show the overall thermal behavior of building zones. The model is constructed with resistors and capacitors as heat storage included in an electrical network. Each resistor includes lumped thermal resistances (R_i), and each capacitor includes thermal capacitances (C_i), as illustrated in Figure 9. Figure 9 presents the order of the lumped elements as electrical networks. This network will be extended into higher orders by increasing the number of elements to build the required electrical structure [25]. Thermal nodes in the thermal model of the DCV and heating system are constructed from thermal lumped elements and are connected to each other. A central node in each zone is connected to other central nodes in other zones via thermal paths across the walls and windows shown in Figure 10.

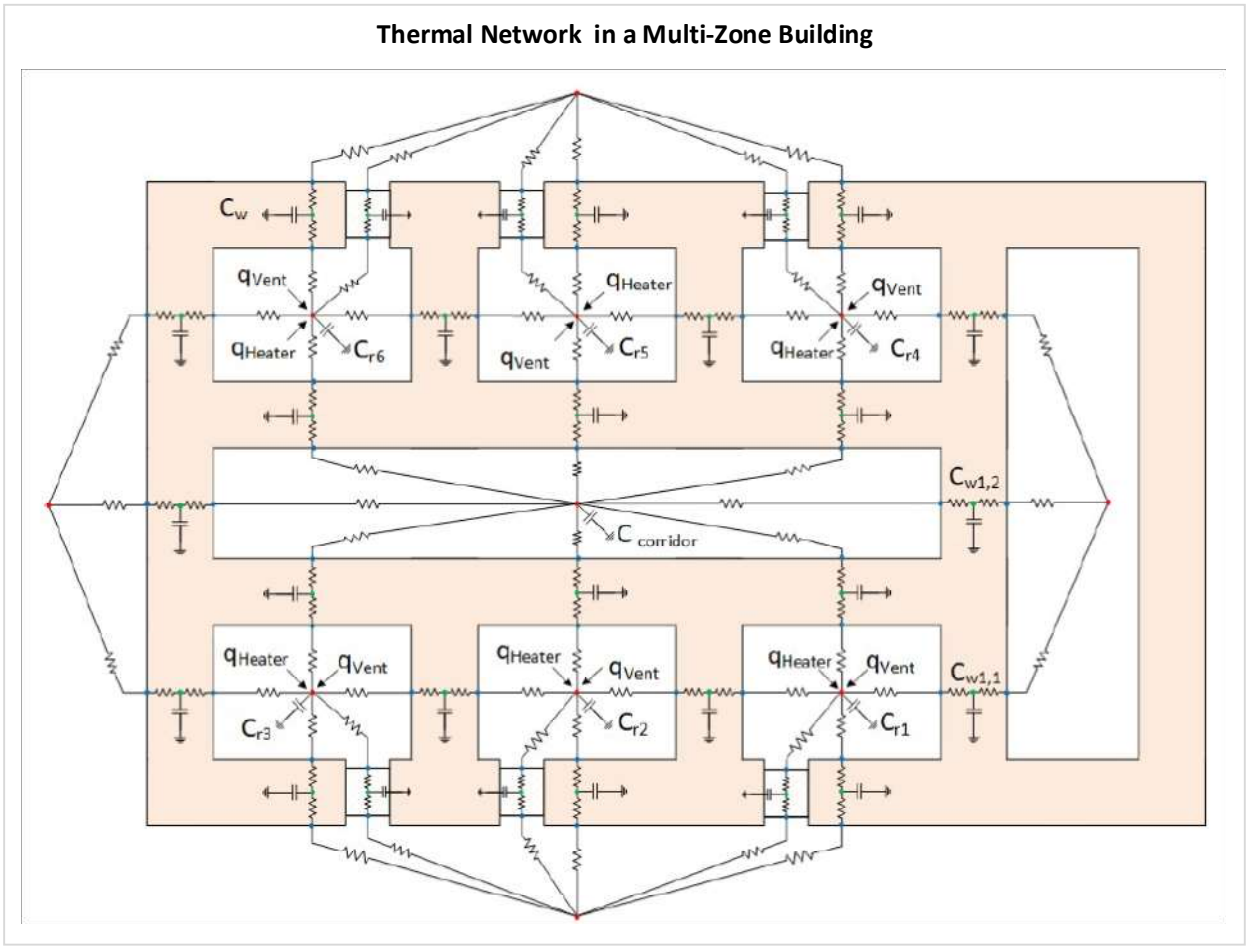

Figure 10. Thermal network (thermal paths across the walls and windows) in a multi-zone building with six zones and one corridor [25].

DCV involves a control strategy for ventilation to moderate the amount of fresh air. It also optimizes air quality in terms of CO_2 concentration and temperature. It balances energy consumption by automatically adjusting the volume of the air exchange. It uses damper actuators according to the captured sensor measurements and values from air quality sensors and the environment. This strategy enhances the quality

of the indoor air. It obtains energy saving by the automatic adjustment of damper actuators based on the sensor values that are obtained from the environment. The ventilation is represented by the internal and external linked airflows. The ventilation design should determine the amount of ventilated air to reach the best indoor air quality, e.g., an amount of 15 cubic feet per minute (cfm) of ventilation in winter and summer [25] as required for each person based on the ASHRAE standards [231, 232]. Pollutants (e.g., CO and CO_2 produced by humans and fuel gas burning) are emitted from occupants and building equipment and trapped inside the zones resulting in health consequences and discomfort of occupants [25]. Therefore, a control strategy for natural ventilation is essential, e.g., exchanging the air with the outside environment by opening the window using sensor technologies. For example, CO_2 sensors measure the amount of carbon dioxide based on the CO_2 concentration computation. A typical CO_2 sensor ranges from 0 to 9999 ppm with an accuracy of 50 ppm $\mp$ 5% [25]. A DCV subsystem includes several components, such as the airflow subsystem, CO_2 concentration sensor, temperature sensor, damper actuator, and occupancy sensor modeled by an occupancy pattern.

4.2 Component-based Development

Component-based development reduces effort and improves scalability by generating DCV and heating system models. Different component models are defined previously, stored in a repository, and integrated to build a system model based on the system requirements.

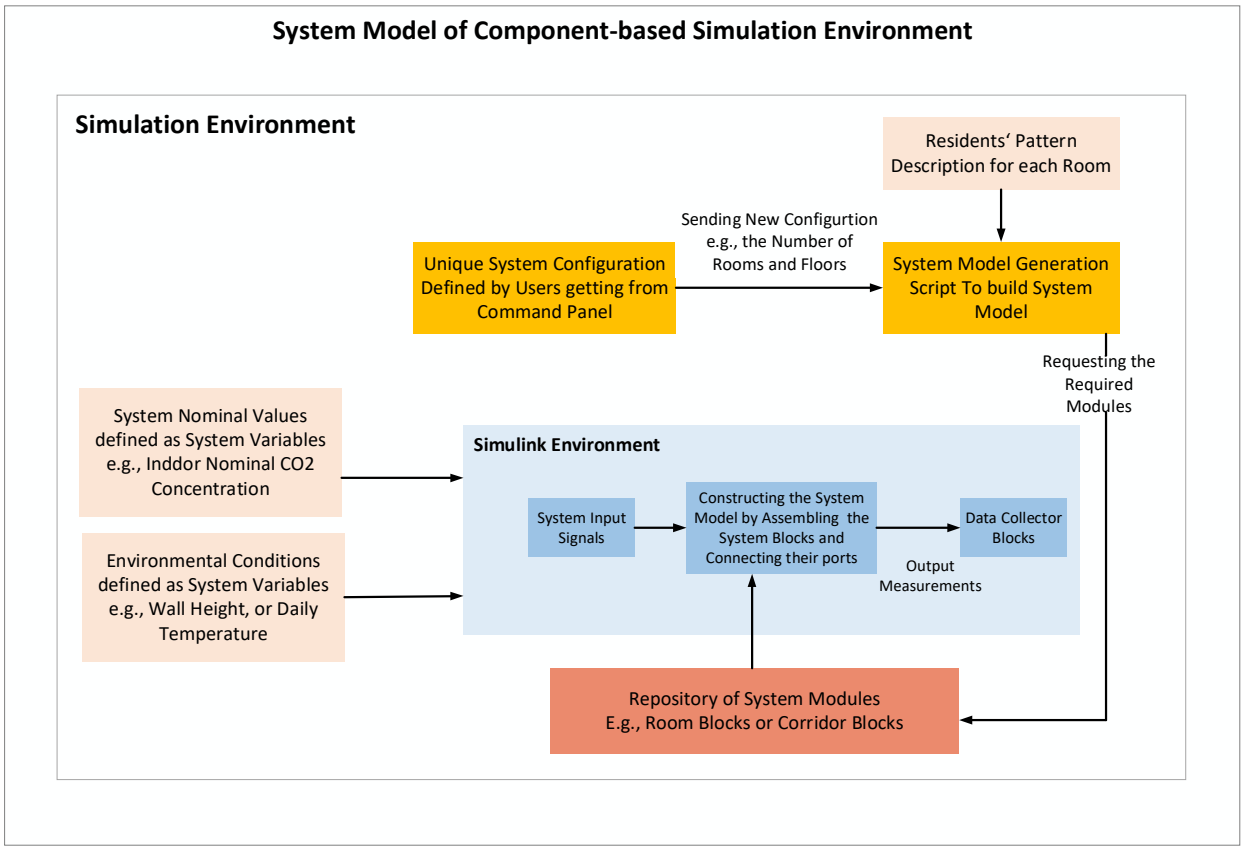

Figure 11. System model of component-based environment description.

Figure 11 describes the component-based system model, including the required specifications, interactions, and system elements such as system model generation script, user command panel, required system variables and assumptions, components of the repository, e.g., room, corridor, and thermal blocks, and how they interact to attain system goals. A generation script is responsible for module replacement, system configuration, and linking different components via their connection ports. System designers can introduce different system configurations through a high-level specification or a graphical dashboard, e.g., specifying the number of floors and rooms. Afterward, according to the number of rooms and floors, the generation script creates the building layout based on the room types. In the generated system model, there are three types of rooms consisting of “Room Type A” or “Room 1”, “Room Type B” or “Room 2”, and “Room Type C” or “Room 3”. Rooms are differed based on their locality and heat transfer issues. Room type A is placed in front of the stairs and is affected by the stair's temperature and other thermal and ventilation conditions. Room type B is located between room type A and room type C, affected by their thermal conditions. Moreover, Room type C is located beside room type B and is affected by its corresponding thermal conditions. In addition, all types of rooms are affected by the outside environment (refer to Figure 7 and Figure 11) [25].

4.3 Fault Injection

Fault injection deliberately introduces various failure modes in the target system for testing the software or hardware in the design phase to validate the system's robustness and harden the system’s resilience, stability, and performance over time. Errors and failures are inevitable in critical and modern infrastructures constructed with numerous and intelligent components [233]. In modern applications, a remarkable dependency on system infrastructure such as components, networks, and software increases the fault occurrence rate and propagation, leading to more product and system output disruptions, energy consumption, and reduced equipment life [233, 234]. To detect and diagnose faults, it is necessary to define the fault classifications of the HVAC system. FI is a method to introduce the various failure modes to the target systems to study the systems under different fault conditions. Naughton et al. [235] have mentioned four important criteria for FI systems, including (1) simplicity meaning the FI and experimental evaluation activities should be easy to setup, (2) versatility, (3) reproducibility meaning the FI should be able to activate reproducible tests, and (4) distributed environments meaning the FI should be applicable to distributed environments.

In this thesis, fault injection testing contains seven primary steps, including (1) defining the steady (healthy) states of the system parameters by modeling the DCV and heating system, (2) defining the faults hypothesis according to the system requirements (fault modeling), (3) defining the realistic faulty events to the system (fault-scenarios), (4) measuring the system parameters under fault-events by activating them via automated fault injection algorithm, (5) documenting the system observations under fault-events, (6) analyzing the system observations, and impacts by comparing them with steady states of the system, and (7) interpreting the fault scenarios results. This section contains the introduction of the automated fault injection framework in the case of single-fault injection, multiple-fault injection, and fault injection framework integrated with the composable model.

4.3.1 Automated Fault Injection in Simulation of HVAC Systems

In HVAC systems, due to high complexity, several types of faults can arise, including hardware faults, design faults, communication faults, and interaction faults, affecting the system's function. These faults in HVAC systems not only cause a waste of energy and occupant discomfort under normal conditions but also lead to hazards that impact safety in emergency scenarios. Therefore, considering the dependability evaluation abilities of the HVAC system in early implementation phases is necessary for high system performance and reliability in the systems under test.

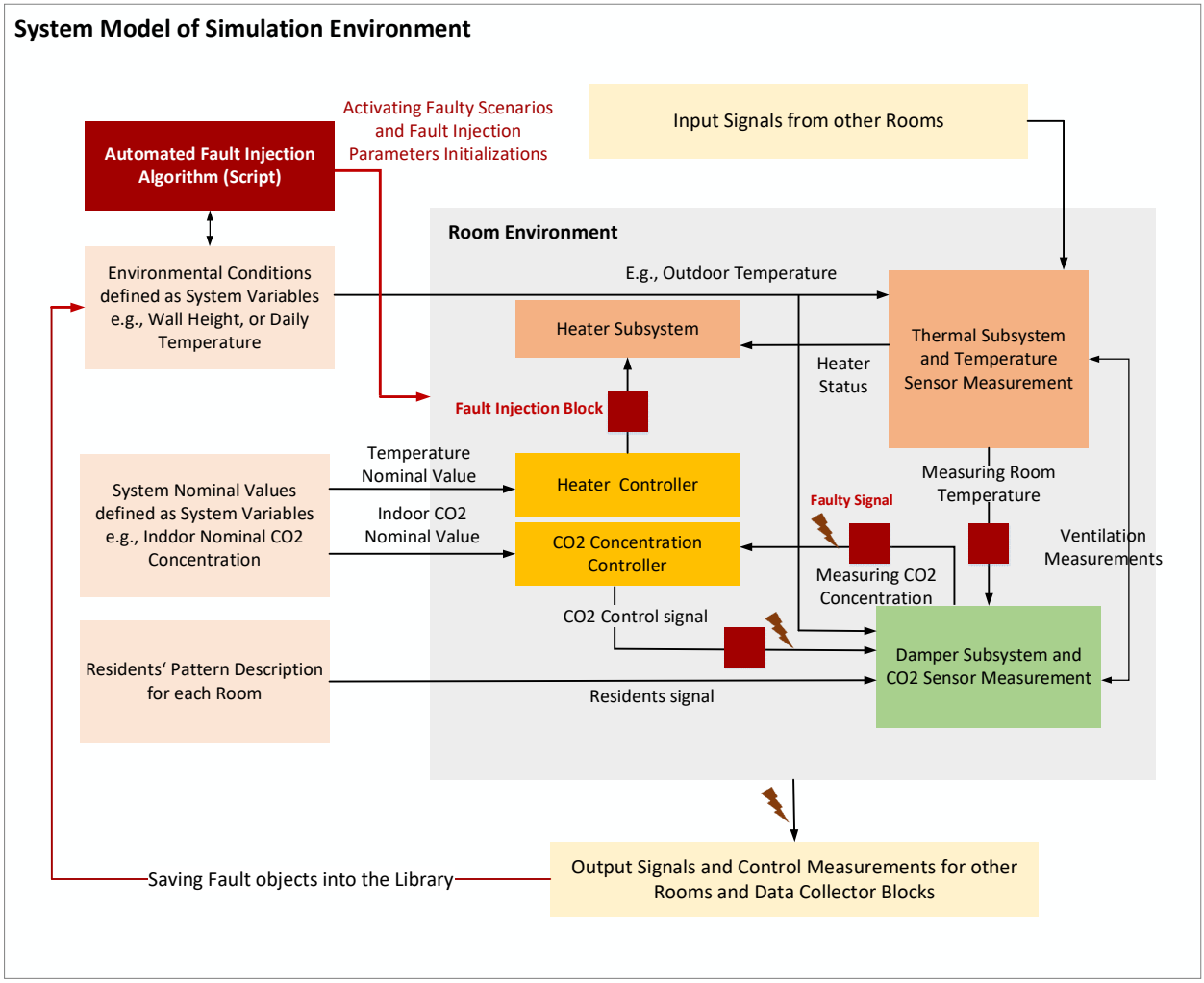

Figure 12. System model of simulation environment including fault injection framework and fault injection blocks and their interrelations.

Figure 12 gives an overview of the system model of a simulated HVAC system, including the embedded fault injection consisting of fault injection blocks with interrelations to other system subsystems and components. Faults are introduced in the system model by fault injection saboteurs. Each fault injection saboteur has been placed before the main system components or subsystems, such as temperature sensor, CO_2 concentration sensor, damper actuator, and heater actuator, to introduce the different failure modes deliberately and to analyze the system behavior and the fault impacts. The fault injector changes the steady

signal behavior as defined in fault scenarios by the system designer. When one fault occurs in one component, that fault may affect the behavior of other components. For example, in a DCV and heating system, a permanent stuck-at-value fault in a temperature sensor may cause a heater actuator to be permanently in the "off" position, resulting in rising CO_2 concentrations and a damper actuator "open" position. Due to the system parameter values, the system model reacts to balance the system thermal condition and to maintain the indoor air quality for more occupant comfort. Therefore, a single or multiple fault occurrence may cause fault propagation into other components and interrelated subsystems.

In addition, Figure 12 show how the automated fault injection enables fault injection scenarios by providing the fault model attributes to the corresponding fault injection saboteur, e.g., fault location, type, injection time, persistence, and duration. The faulty measured signal and system outputs are documented as faulty objects in a library. The library of faults can be initialized as system model variables for other objectives such as experimental evaluation activities and FDD technique development. In single-fault injection, one fault scenario is activated at one fault injection saboteur in one room. This fault may be intermittent causing a repeated fault injection with the fault attributes of the component for the defined repetition times. In contrast, in multiple-fault injection, multiple fault scenarios are activated at different fault injection saboteurs in different components, rooms, and floors. Therefore, each fault scenario should include the fault address to show the fault's target location by determining the faulty floor (floor number), faulty room (room number), and faulty component (component number).

4.3.2 Automated Fault Injection in HVAC Composable Model

In the modern world, during the construction of hospitals, airports, and office buildings, numerous HVAC systems are used as large-scale distributed systems with thousands of components, including sensors and actuators, which are vulnerable and susceptible to various and multiple failure modes. Compared to smaller-scale systems where the single fault hypothesis is common, in critical infrastructures and large-scale systems, multiple faults are more probable. As a result, multiple-fault injection is a requirement for large-scale DCV and heating systems. Modeling and simulation are cost-effective, time-efficient, and risk-free alternatives to experimental setups for system design, monitoring, and testing [21]. In system design, composition describes how components can be selected and combined in various configurations and levels to meet user requirements with significant savings in development expenses and runtime. The component-based development is challenging in the modeling and simulation discipline [236, 237]. It is convenient to express high-level models with the system requirements and to generate the simulation model in an automated manner with support for injecting various faults to test and evaluate the outputs and behavior of the system.

In this thesis, the generated simulation models support multiple-fault injections in DCV and heating systems with the required system structure. The unique system configuration and structure, e.g., the number of floors and rooms, can be defined by users via configuration information and a command panel. Figure 13 illustrates the components of the simulation model with an embedded automated multiple fault injection framework and it also shows how the components interact. The primary components are an automated fault injection script, a simulation model generation script that supports the Simulink environment, a repository of components, e.g., room and corridor blocks equipped with automated multiple-fault injection capabilities and model assumptions as system variables (e.g., environmental conditions, building assumptions, thresholds, nominal values).

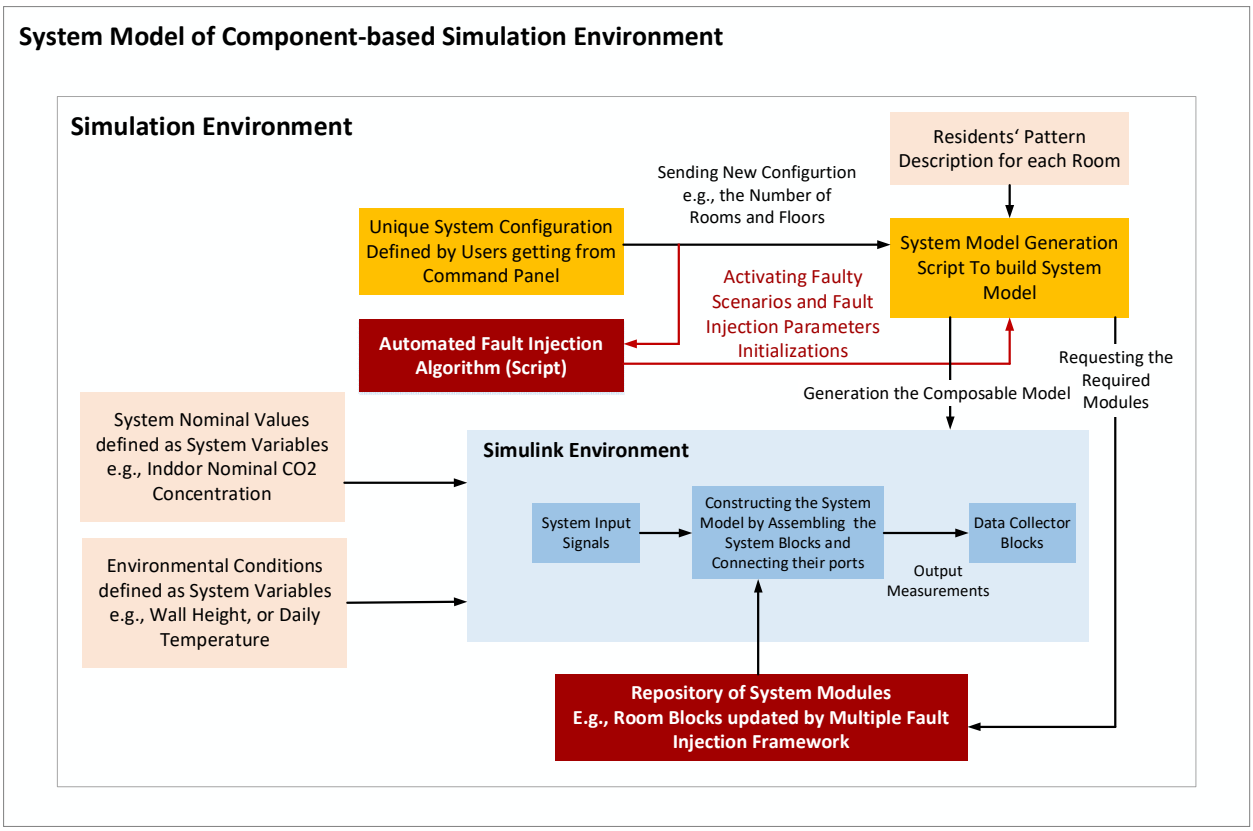

Figure 13. The generated simulation model with multiple-fault injection support and the component interrelations.

4.3.3 Automated Single-Fault Injection

Each fault injection system contains essential and primary elements such as an injector, analyzer, controller, data-collector, monitor, and target system [143, 235]. In this thesis, an automated and simulation-based method of fault injection is proposed (named ASFIF) by the combination of simulator command and simulation code modification techniques that are constructed of two main parts, including (1) the command environment and (2) the simulation environment. Simulation-based techniques simulate the system in a simulation environment with a predetermined distribution of failures via a set of inputs. The simulation code modification technique modifies the system description by adding extra components dedicated to the FI procedure called saboteurs or mutants. Saboteurs are disabled during normal system operations and enabled in case of faults activations. They can be added and enabled manually and automatically. Simulator command techniques, procedural interfaces, and command languages extend the model to speed up the simulation [53].

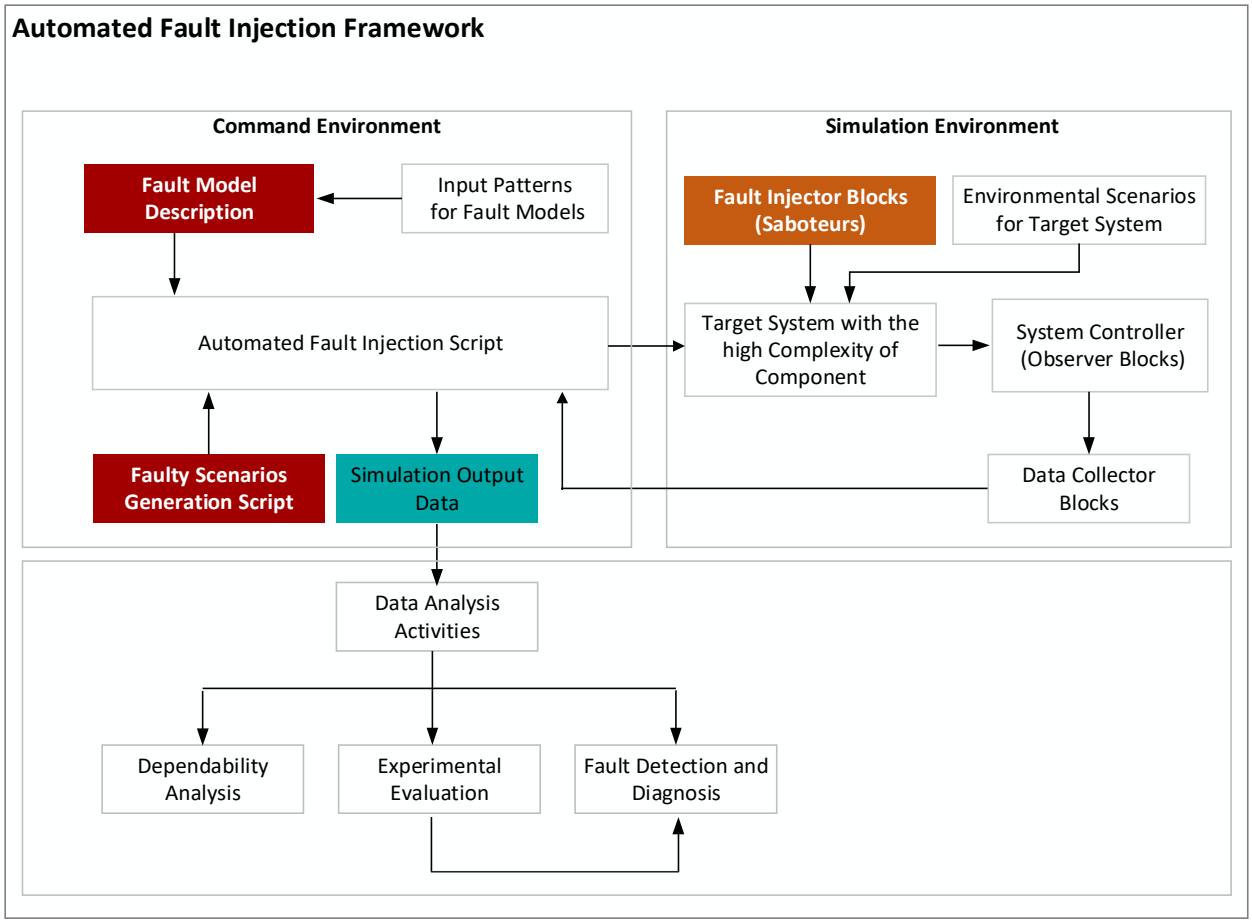

Figure 14. Automated fault injection framework and its main elements and their interrelations.

Figure 14 demonstrates the main components of the automated single fault injection framework in which the command and simulation environments interact to activate a fault case example. The environmental attributes and input patterns examine the system and provide data for the fault injection framework and the simulation model. Then, the fault injector blocks as saboteurs must be added to the simulation model (Figure 12) and initiate the corresponding attributes for each faulty mode. The fault injector blocks have been developed by Stateflow diagrams that initialize the fault attributes via an automated fault injection algorithm for each input pattern and fault scenario. After the termination of the simulation's execution time, the monitoring blocks collect the measured data. Tests are inserted by the fault injector when the test load has succeeded in the system. The simulation output is gathered and returned to the fault injection algorithm to be analyzed for data analysis activities, including the FDD methods, experimental evaluation, and impact analysis. The modules of the FI framework are the fault model description, the fault scenario generation script, the automated fault injection script, the target system model equipped by fault injection blocks, input patterns including the system assumptions and fault assumptions, data observer blocks, and data collector blocks which are described in the following.

4.3.3.1 Command Environment

The FI framework consists of two main parts, the command environment, which provides the command language space and user interface, and the simulation environment, which serves the target system simulation using Simulink blocks and tools using the MATLAB programming language. The command environment also models the fault patterns and assumption for the single and multiple fault injections. In each fault injection case, a scenario is required to activate the fault pattern. Therefore, a scenario generation script also cooperates with the fault injection process, which can be defined based on the user requirements or assign the fault attributes randomly. Different parts of the command environment are described in the following.

4.3.3.1.1 Fault Model Description

The fault model specifies the target failure modes and determines the analysis possibilities for the user. The parameters of the fault model describe the real-world environment and system characteristics. The fault model can be defined under two assumptions: single fault assumption when only one failure mode occurs in the system, and multiple fault assumption when multiple faults occur at different points in time or at the same time in the system. Each fault can be categorized using six main criteria: phase of creation (development or operational), system boundaries, domain, phenomenological causes, intents, and persistence. According to the system requirements, one or several criteria can be chosen and considered during fault modeling.

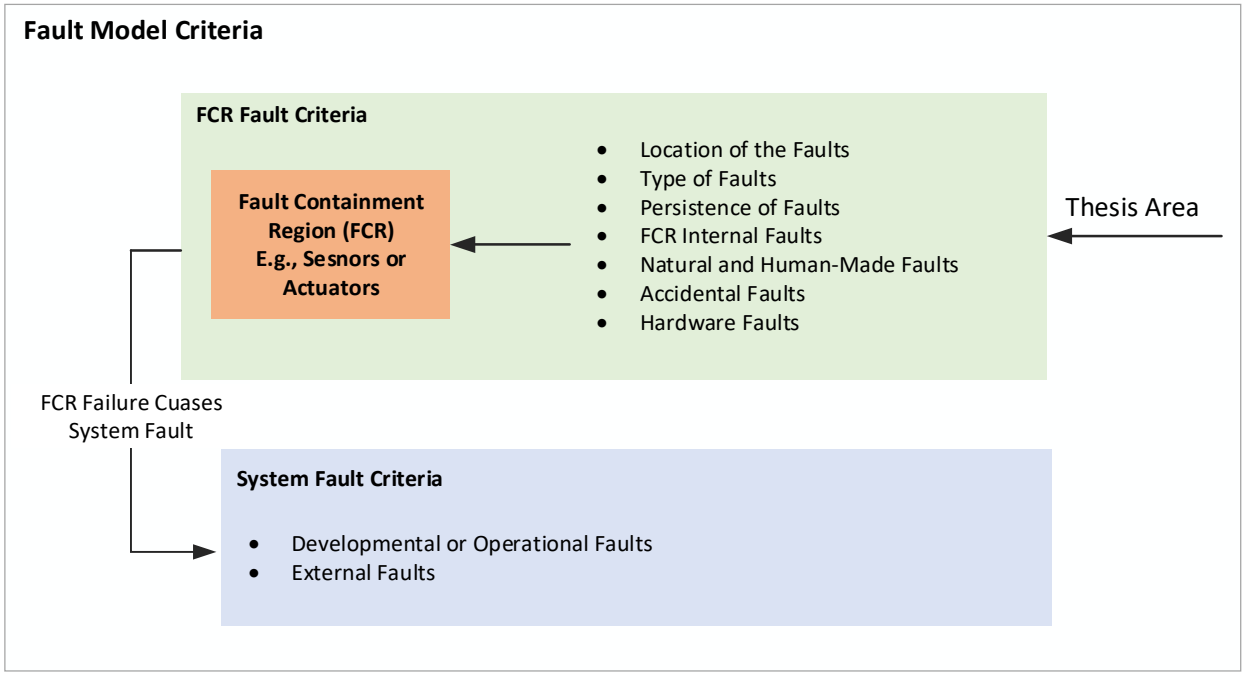

Figure 15. Fault model criteria.

Figure 15 demonstrates the applied fault criteria for FCR-level and system-level faults in our introduced fault model. In this thesis, fault criteria have been considered and modeled for FCRs, including the sensors and actuators. Their associated criteria, such as location, type, and persistence, have been

considered. These faults occur internally in each FCR as natural or human-made faults. For the criterion of the domain, only hardware faults have been considered. Hardware faults occur in communication or in devices due to various defects and malfunctions in harsh environmental conditions. In case of a permanent hardware fault, the component should be removed and replaced. An example of a hardware fault is a short circuit due to the presence of water. The other fault criteria such as developmental, operational and external faults are not considered because they are not relevant to fault containment regions in our model.

In this thesis, the fault injection framework focuses on hardware and data-centric faults of the components, including sensors and actuators with the respective persistence and fault locations.

4.3.3.1.2 Data-Centric Faults in Components

In this thesis, data-centric faults are modeled and related to the generated data from the components, including the CO_2 concentration sensor, temperature sensor, damper actuator, and heater actuator. Table 5 presents the fault attributes with their corresponding detailed information, including the fault type, persistence, duration, interarrival time, repetition, location, and value, along with their details and measurement functions. Sensor measurements are time-dependent, and they vary over time. Therefore, a time-dependent measurement function calculates the faulty values for different fault types in each faulty state of the system and each time slot for sensory components. The following equation describes the generated data for a component that can be modeled as a measurement function $x'(t)$ as shown in Equation 8. Equation 8 defines the faulty values to achieve the results of the FI for different fault types:

$$x' = \beta x + \alpha + \eta \qquad \text{Equation 8}$$

x represents healthy data, x' is the calculated faulty data, β is the coefficient for gain faults, α is the coefficient for offset faults, and η is the coefficient for white-noise uncertainties, which is a combination of the Gaussian distribution for measurements and uniform distribution of the measurement uncertainties.

Table 5. Fault attribute analysis and description of the introduced fault profile.

Nr.	Attribute in Fault Profile	Fault Details	Measurement Functions for Fault Types based on Equation 7
1	Fault type	Stuck-at-fault-value (Sensors) Stuck-at-fault (Actuators)	x′ = α+ η (Sensors) and x′ = 0 or 1 (Actuators)
		Gain fault	x′ = βx + η
		Offset fault	x′ = α + x + η
		Out-of-bound fault	x′ > θ1 or x′ < θ2
		Data-loss fault	x′ = Last measurement of actual value
		White-noise Fault	Gaussian Probability Distribution
2	Fault persistence type	Permanent fault Transient fault Intermittent fault	

3	Fault duration time	Uniform distribution of intermittent faults
4	Fault interarrival time	Uniform distribution of intermittent faults
5	Fault repetition	0 Repetition time for permanent faults 1 Repetition time for transient faults 2, 3, …, n Repetition times for intermittent faults
6	Fault location (FCR)	CO_2 sensor Damper actuator Temperature sensor Heater actuator

Table 5 shows the fault model attributes, including fault type, persistence type, duration time, interarrival time, repetitions, and location (i.e., FCR). Each fault attribute is described as follows.

4.3.3.1.3 Fault Types in HVAC Systems

Different fault types have been considered in our FI framework. There are six fault types: the stuck-at fault for the actuators and stuck-at-value, gain, offset, out-of-bound, data-loss, and white-noise faults for sensor components. Each fault type is defined as follows.

Stuck-at-Fault: A stuck-at-fault is a hardware fault in which the behavior of a component is stuck at a particular point in time, the variation of the signal is zero and it does not change over time [11, 238–240]. Stuck-at faults may happen in both actuators and sensors as stuck-at sensed values in sensor components and stuck-at statuses of actuator components, e.g., stuck-at-open and stuck-at-closed statuses in a damper actuator, and stuck-at-off and stuck-at-on statuses in a heater actuator. A stuck-at-fault occurs in actuators when x′ = 0 or 1, where 0 and 1 specify the actuator statuses. For example, in the heater actuator, one shows that the heater is "on", and zero shows that the heater is "off". A stuck-at-value fault can be modeled using Equation 9 where α is a constant sensed data, η is white noise for each measured data, and $x' \in f(t)$.

$$x' = \alpha + \eta \qquad \text{Equation 9}$$

Gain Fault: A gain fault occurs once the change rate of sensed data is different from the expected rate of data over a period of time due to the sensing unit's bias, drift, or calibration error [11, 238–240]. This fault has only been considered for the sensors. It can be injected by multiplying a constant coefficient with the actual sensed data. Equation 10 has modeled a gain fault where β is a gain coefficient with respect to the healthy measurements, and η is white noise for each measured data $x' \in f(t)$.

$$x' = \beta x + \eta \qquad \text{Equation 10}$$

Offset Fault: An offset fault occurs when a shift value is added to the actual sensed data due to the sensing unit's bias, drift, or calibration error and shows a deviation from the expected actual data [11, 238–240]. This fault has only been considered for the sensors. Equation 11 has modeled an offset fault where α is a constant value added to the healthy measurement, η is white noise for each measured data, and $x' \in f(t)$.

$$x' = x + \alpha + \eta$$ Equation 11

Out-of-bounds Fault: There are minimum ($x' > \theta_1$) and maximum ($x' < \theta_2$) bounds for each sensor, and sensor measurements should be in these ranges [11, 238–240]. An out-of-bound fault occurs when the observed values or measurement data are out of the expected ranges (bounds) where $x' \in f(t)$. θ_1 and θ_2 are required application thresholds.

Data-loss Fault: A data-loss fault occurs when a component is missing data during a specific time interval. $f(t) = \phi$, $t \succ \tau$; where ϕ is the null value and τ is the maximum required time for receiving the measured data. In case of a data loss fault, the last measurement of the sensed data indicates that the actual measurement is missing [11, 236–238]. In this thesis, the last measurement has been considered for the received data in case of the data-loss fault occurrence (i.e., x′ = Last measurement of actual value).

White-noise Fault: To develop a realistic system model, white-noise as a random uncertainty has been added to the actual sensed values considered as white-noise faults [11]. A Gaussian probability distribution or a uniform probability distribution determines these random values which are added to the actual values. The Gaussian distribution is also known as normal distribution because a random variable with a gaussian distribution is distributed normally, and it is a continuous probability distribution as defined in Equation 12, where μ is the mean of the distribution and σ is the standard deviation [241].

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$ Equation 12

4.3.3.1.4 Fault Persistence in HVAC Systems

Our proposed FI framework considers different fault persistence types, including transient, permanent, and intermittent faults. In this section, the fault persistence attributes are explained.

Single Permanent Fault: a permanent fault remains in the system for the rest of the FI system execution time. Permanent faults have been considered during the FI process for sensors and actuators. Figure 16 illustrates the generic timing diagram for the permanent fault injection.

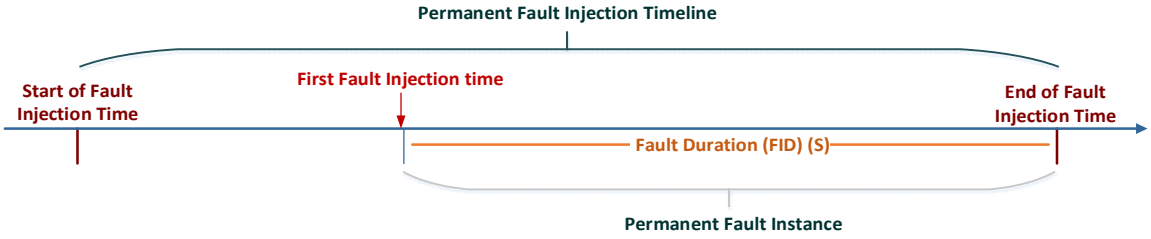

Figure 16. Generic timing diagram for a single permanent fault injection at a hardware location [11]

Single Intermittent Faults: intermittent faults are bursts of failures and emerge at the same hardware location in irregular intervals. There are different types of intermittent faults, including short intermittent faults with few failure repetitions, long intermittent faults with a greater number of repetitions that disappear, and long intermittent faults that do not disappear in the FI system execution time and become

permanent faults gradually [115]. Figure 17 shows a generic timing diagram for the intermittent faults, e.g., with three failure repetitions. This example can be considered a short intermittent fault with three repetitions. If the number of repetitions increases, the fault can be considered as a long intermittent fault. The fault injection process starts from the first fault injection time, and faulty behavior starts based on the fault type. Then, the fault injector operates according to the Fault Duration (FD) time, and the subsequent repetition starts after a Fault Interarrival Time (FIT). FIT determines the time between two repetitions. The following fault injection time can be computed by adding the fault injection time, duration, and interarrival parameters. Different fault types and faulty values (using Equation 8) can be selected for each repetition at the same hardware location.

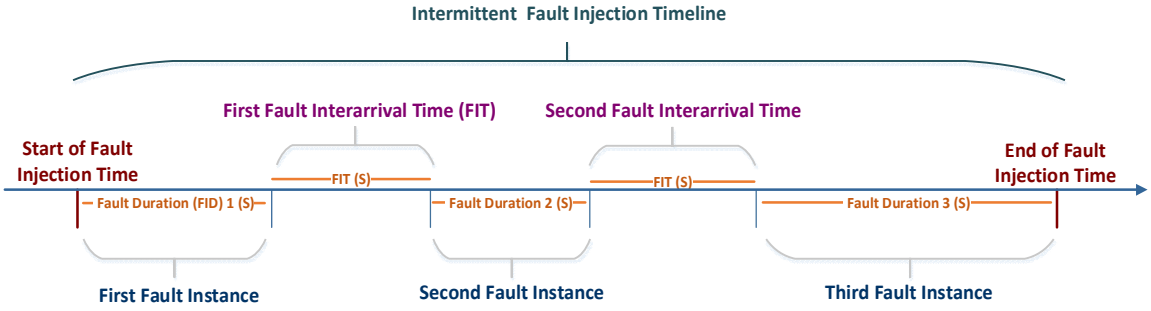

Figure 17. Generic timing diagram for a single intermittent fault injection at a hardware location [11]

Knowing the frequency of intermittent faults in different components is essential to realize the system functionalities for proper recovery actions [122]. There are no comprehensive intermittent fault studies in HVAC systems, including modeling of intermittent faults, their frequencies, and repetitions. Few fault models describe intermittent faults with their occurrence frequencies [47, 119–123]. Therefore, there is no reliable timing model for intermittent faults for the sensors in HVAC systems. Intermittent faults are more common in actuators. The timing parameters in the literature have been applied to our fault model [11, 130, 242]. In this thesis, intermittent faults have been considered only for the actuators due to the lack of proper timing models for the sensors in HVAC systems.

In this thesis, in the case of the multiple fault injection, an approximate fault occurrence probability approach based on the maintenance records and prior studies has been proposed to model the intermittent faults based on the failure probabilities with different repetitions. Both types of intermittent faults have been investigated and modeled, such as short (e.g., with two repetitions) and long intermittent faults (e.g., with N number of repetitions). Modeling of the permanent and transient faults differs from intermittent faults. In the case of long intermittent faults, repetitions can be increased based on the designer's necessities, but faults also disappear eventually. In Figure 17, a fault set with the intermittent persistence type is activated in a sequence of failures with different durations and interarrival times. Each failure can have different types and values can be measured based on the selected types. As an example, losing switch contact in measurement devices causes an intermittent fault occurrence with a sequence of multiple failures. For example, the failure cases can be a sequence of stuck-at, data-loss, or gain faults with three repetitions.

Single Transient Faults: a transient fault occurs once and then disappears till the next failure based on the Mean Time to Failure (MMTF) parameter and the system execution time. Transient faults usually occur due to environmental conditions, e.g., high-energy particles [115]. Figure 18 provides a generic timing diagram for the transient fault at a hardware location with no repetition. It occurs once during the FI system

execution time. A transient fault starts at the fault injection time and ends after its defined Fault Duration (FD). During the fault duration time, different fault types may happen. There are two types of transient faults: short intermittent fault with short fault duration and long transient faults with longer fault duration, which differ based on the system specifications [115].

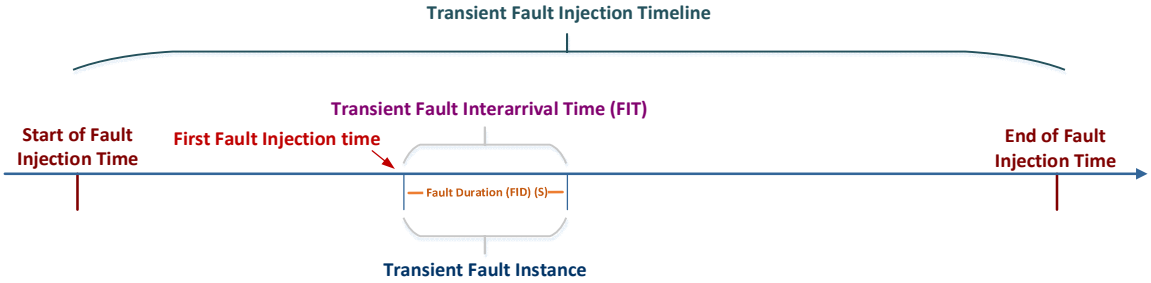

Figure 18. Generic timing diagram for single transient fault injection at a hardware location [11]

4.3.3.1.5 Multiple Fault Injection Timeline

Figure 19 shows the system timeline in case of multiple fault occurrences in which different components (FCRs) have different fault assumptions, e.g., different types of fault persistence where each repetition takes different fault types and timing parameters. In Figure 19, FCRs x, y, and z are different fault locations, e.g., different floors, rooms, and components with different fault persistence, demonstrating component-level timelines. A permanent fault has been assigned to the FCR x, an intermittent fault with two repetitions has been assigned to the FCR y, and another intermittent fault with two repetitions has been assigned to the FCR z with different timing parameters (e.g., different fault injection, duration and interarrival times). The system-level timeline specifies how fault injections at multiple locations with different fault assumptions are integrated into a unique timeline.

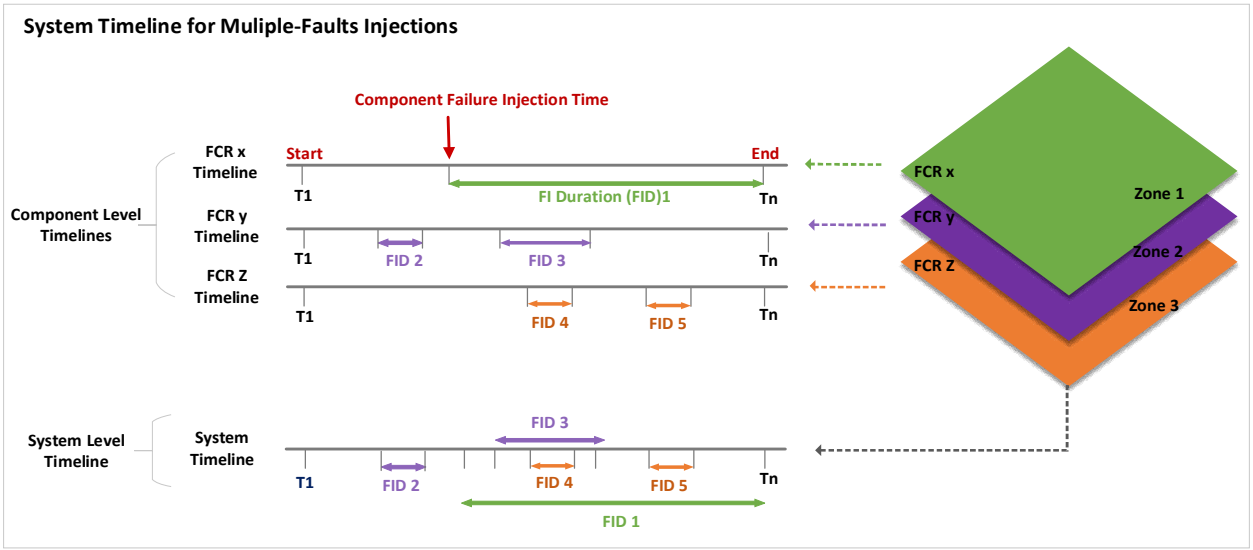

Figure 19. System timeline in case of multiple-fault occurrences.

4.3.3.1.6 Fault Occurrence Probabilities for Multiple-Fault Pattern in HVAC Systems

Two principal metrics describe the fault occurrence rate, including fault prevalence and incidence. Fault prevalence defines the fault occurrence rate of the units for a given fault at a single point in time. The fault incidence is the fault frequency in a specific period [242]. This thesis calculates the fault occurrence rates using maintenance records and field reports based on the fault type and environmental conditions, such as the season or month the system is investigated. Components in HVAC systems fail with different probabilities and rates due to various conditions, e.g., the number of components, environmental conditions, and unit failure rates. We have used available maintenance records to find the occurrence rates of HVAC system faults. For instance, Li et al. [224] used maintenance records to calculate the frequency and occurrence of incidents of various HVAC faults for one year. They calculated the average probability of occurrence around 0.0102 for each associated fault. Ebrahimifakhar [226] proposed the fault occurrence rates of several types of faults with different metric definitions calculated according to other FDD techniques. They also calculated average fault presence percentages for the various units, faults, and months. For instance, the average fault presence percentage of a stuck discharge air damper is estimated at approximately 8%, heating failure at 9%, and air temperature abnormality at 18% for HVAC and AHU in February. Faults are also listed based on their monthly presence. Hosseini Gourabpasi et al. [227] ranked HVAC-related faults and their frequencies with data-driven techniques. For example, the limit issue faults had the first rank with a rate of 15.18%. The stuck-at/partially closed faults had the second rank with a rate of 14.95%, and bias/drift/calibration faults had a probability of 10.94% and were listed in the fourth rank. Applicable unit faults in our proposed FI framework and their fault rates are listed and described in Table 6**.** The average probabilities for the associated fault types were calculated over one month and one day. In this thesis, the fault occurrence rate during each FI is the disjoint probability of both component failure rates based on Table 6 nd the application of system fault type rates. Fault type occurrence probabilities for the stuck-at fault, gain fault, offset fault, out-of-bounds, and data-loss fault can be defined as 14.95%, 10.94%, 10.94%, 10.94%, 4.46%, 4.46%, respectively [226].

Table 6. The faults and their fault occurrence incidents for the associated fault types

Nr.	Component	System Faults	Average Presence of Faults in February	Average Monthly Presence of Faults Among the Total of 28 Faults	Total Monthly Probability	Total Daily Probability
1	Temperature Faults	Temperature sensor fault	18%	8%	0.2538	0.0091
		Temperature frozen		35%		
		The mismatch between supply air temperature and its setpoint		26%		
		Supply air temperature abnormal		12%		
		Mix air temperature sensor fault		4%		
		Mix air temperature abnormal		22%		
		Return air temperature abnormal		2%		
		Setpoint fault		4%		
		Missed control optimization		28%		
2	Heater Faults	Heater abnormality	9%	2%	0.0324	0.001157
		Heating coil valve leakage		2%		
		Setpoint fault		4%		
		Missed control optimization		28%		
3	CO_2 Faults	Airflow sensors abnormalities (CO_2 sensor)	13%	10%	0.05785	0.00206

		Return airflow abnormal		1.5%		
		Return air CO_2 sensor		1%		
		Missed control optimization		28%		
		Setpoint fault		4%		
4	Damper Faults	Damper stuck	8%	11%	0.0312	0.0028
		Missed control optimization		28%		

4.3.3.1.7 Component Faults in HVAC Systems

Component faults include a fault location (i.e., FCR) such as a CO_2 concentration sensor, temperature sensor, damper actuator or heater actuator. Each FCR can take different fault characteristics, such as fault type and persistence. In this section, each fault location is described.

CO_2 Sensor Fault: The CO_2 sensor fault resembles an incorrect sensor reading. Five kinds of faults are considered for the CO_2 sensor components: the gain fault, offset fault, stuck-at-value fault, out-of-bound fault, and data-loss fault. The proposed fault injection framework is generic and compatible with different target systems. Therefore, different types of fault persistence for sensors and actuators can be activated based on the system timing parameters. There are no appropriate timing parameters for the intermittent fault types in sensors in HVAC systems. Hence, for the persistence attribute, only permanent faults have been considered for the CO_2 sensor.

Temperature Sensor Fault: The temperature sensor fault resembles an invalid sensor reading. Five kinds of faults are considered for the temperature sensor components: the gain fault, offset fault, stuck-at-value fault, out-of-bound fault, and data-loss fault. For the persistence attribute, only permanent faults have been considered for the temperature sensor.

Damper Actuator Fault: The damper actuator fault resembles a stuck-at fault when a damper is stuck at a specific position, including “Closed" equal to the binary value of 0, and “Opened” which is equal to the binary value of 1. For example, once the damper actuator is stuck to the open state, the open state of the damper actuator causes fresh air to enter the indoor environment, decreasing the temperature. Therefore, the heater actuator should constantly compensate the heat loss.

Heater Actuator (Thermostat) Fault: This fault describes a stuck-at fault when the heater sticks to a specific position, including “Off” and “On”. Suppose the heater is stuck at its “On” position. In that case, it acquires the binary value of 1, which means that the indoor temperature rises. Suppose the heater has a stuck-at fault in the ”Off” position, which equals the binary value of 0. In that case, the temperature tends to decrease.

4.3.3.2 Input Patterns of Fault Sets

The input patterns of the automated fault injection algorithm are shown in Table 7 . Each sample can be set by a specific combination of the fault inputs and variables to create fault sets for the system at operation time. A fault location (faulty component) will be selected each time for the fault-initializer algorithm which is introduced in the next section. Other aspects of faults (e.g., timing and persistence) that the system may face during the system operation time are defined in a fault model. In a random fault model, a fault set

initiates and affects a particular component in one room. The persistence, types, durations, and interarrival times are initiated in each fault set.

Table 7. Fault attributes analysis and descriptions in the introduced fault profile model.

Nr.	Properties	Realistic Example for a Fault Set in an Automated Fault Injection
1	Number of samples	The number of samples can be randomly defined or manually assigned. Each sample or system execution time equals one day or 86,400 s; 30 samples are equal to 30 days (one month), or 60 samples are equal to 60 days (two months).
2	Model of fault	Random fault happens in one component with different random fault attributes and times. A systematic fault happens in multiple components simultaneously and of the same type.
3	Fault type vector	Fault types are defined as a vector with different IDs: (1: stuck-at, 2: gain, 3: offset, 4: out-of-bound, 5: data loss)
4	Fault injection time vector	This vector includes the injection times for each FCR failure based on the fault type and its repetitions in one day. In the same way, the first injection time is randomly selected, and others are initialized based on the number of repetitions, fault duration, and fault interarrival times.
5	Fault injection persistence vector	{Permanent, transient, intermittent}
6	Repetition vector	{0, 1, 2}, where 0 is for permanent faults, 1 for transient faults, and 2 for intermittent faults.
7	Fault interarrival vector	A vector of minimum fault interarrival time (e.g., 400 s) and maximum fault interarrival time (e.g., 4000 s) that can be selected by the uniform distribution in case of intermittent faults
8	Fault duration vector	A vector of minimum fault duration (e.g., 300 s) and maximum fault duration (e.g., 3000 s) that can be selected by a uniform distribution in case of transient and intermittent faults
9	Faulty component (FCR) vector	{1: CO_2 sensor, 2: damper actuator, 3: temperature sensor, 4: heater actuator}

4.3.3.3 Automated Fault Injection Algorithm

The automated fault injection algorithm loads required variables for the system model and FI process from files as input patterns and environmental scenarios. Two types of faults can be activated in the system: systematic and random faults. In the systematic FI, some components face the same types of faults due to systematic or design problems, e.g., uncalibrated measurement devices from factories, such as sensors, which result in systematic sensor faults. In the random FI, fault attributes can be randomly selected for each fault set. Then, the location of the faults should be clarified to activate a fault set for the target fault-injector blocks (i.e., saboteurs). The room and component numbers will show the fault location in the FI process as selected by the algorithm. The persistence type of each fault set should also be determined before running the simulation file. Meanwhile, persistence presents the number of repetitions of the fault injections in each fault set. Then, the simulation runs are performed for each sample time. For example, the execution time can be one day (86,400 s). In our FI framework, a Stateflow diagram is used to model the persistence feature of the FI framework with different fault duration times and fault interarrival times. In each faulty situation, the system's state changes between a healthy state and a faulty state for each element of the fault injection vector (e.g., for an intermittent fault with two repetitions, there are two injection times in the fault injection vector). Afterward, in this process, if the fault injection time is equal to the system time, then the system's state changes. After the corresponding fault duration time, the system's state returns from the faulty state to the healthy state. Regarding the fault interarrival time, the state of the system and the signal value is healthy.

The system's fault types, and fault values are chosen in each transition of the states, according to the Stateflow model. Function 1 provides pseudo-code for the automated single-fault injection algorithm with the respective steps.

Function 1. Pseudo-code description for the automated single-fault injection algorithm.

Automated Fault Injection Algorithm

Begin

1. All system variables and inputs initialization
2. Initialization of the fault injection vector as **FIV=0.**
3. Selecting the number of days as **Num_Days.**
4. **For i=1:1: Num_Days**
5. Selecting the number of repetitions in each intermittent fault as **Num_Repetitions.**
6. Selecting the "**Systematic**" fault or "**Random**" fault.
7. If ("**Systematic**"), all fault attributes should be assigned based on a predefined scenario.
8. If ("**Random**"), all fault attributes should be assigned randomly.
9. Selecting the fault location which shows the faulty target component in each fault injection.
10. Selecting the persistence type.
11. **If** (Persistence is **Intermittent**), the following vectors should be prepared based on **Num_Repetitions.**
    a. Preparation of the fault injection time vector.
    b. Preparation of fault duration time vector.
    c. Preparation of the fault interarrival time vector.
12. **If** (Persistence is **Transient**), only fault injection time and fault duration should be assigned once.
13. **If** (Persistence is **Permanent**), fault injection time should be assigned once, and fault duration should take till the end of system model execution.
14. The system model execution file will be opened. This file is a simulated system model file.
15. The system model execution file will be run.
16. Fault types and faulty values should be assigned during the system simulation using a Stateflow diagram with a transition between faulty and healthy states for each repetition.
17. The system model should be closed after 86400 seconds.
18. An **Object** for each faulty sample is created **Fault_Object$_i$, i= {1, n}.**
19. All attributes for each faulty sample should be saved in this **Object**.
20. Output data from simulation execution should be saved in this **Object**.
21. **Fault_Object$_i$, i= {1, n}** should be stored in **FIV.**

End

End

Function 2 shows the pseudo-code for the automated multiple-fault injection algorithm. It extends the single-fault injection (Function 1) to multiple fault injection by injecting the faults at multiple locations. It means in each fault injection procedure, multiple faults in different locations are activated by their defining indexes. Indexes can distinguish each location. Each index introduces the floor number, room number, and component number. In multiple fault injections, the number of faulty samples shows the number of fault sets. Each fault set comprises the combinations of all fault attributes accordingly. With indexing, we can access the faulty component in each fault injection. Each index includes two factors including the room number, and component number. In addition, the fault attributes should be defined as matrices instead of vectors.

Function 2. Pseudo-code description for the automated multiple-fault injection algorithm.

Automated Fault Injection Algorithm

Begin

1. All system variables and inputs initialization.
2. Initialization of the fault injection vector as **FIV=0.**
3. Selecting the number of days as **Num_Days.**
4. **For i=1:1: Num_Days**
5. Selecting the number of faulty samples which introduces the number of faulty samples in each day as **Num_FaultySamples**
6. Selecting the number of repetitions in each intermittent faulty sample as **Num_Repetitions**
7. Selecting the "**Systematic**" fault or "**Random**" fault
8. If ("**Systematic**"), all fault attributes should be assigned based on a predefined scenario.
9. If ("**Random**"), all fault attributes should be assigned randomly.
10. **For j=1:1: Num_FaultySamples**
11. Selecting the fault location and indexing.. This index activates the target faulty component with **Index (#Num_Room, #Num_Component)**
12. Selecting the persistence type
13. **If** (Persistence is **Intermittent**), the following Matrices should be prepared based on the **Num_Repetitions.**
    a. Preparation of the fault injection time Matrix.
    b. Preparation of fault duration time Matrix.
    c. Preparation of the fault interarrival time Matrix.
14. **If** (Persistence is **Transient**), Only fault injection time and fault duration should be assigned once.
15. **If** (Persistence is **Permanent**), Fault injection time should be assigned once, and fault duration should take till the end of system model execution.
16. The system model execution file will be opened. This file is a simulated system model file.
17. The system model execution file will be run.
    a. Fault types and faulty values should be assigned during the system simulation using a Stateflow diagram with a transition between faulty and healthy states for each repetition.
18. The system model should be closed after 86400 seconds.
19. An **Object** for each faulty sample is created **Fault_Object_i, i= {1, n}.**
20. All attributes for each faulty sample should be saved in this **Object**.
21. Output data from simulation execution should be saved in this **Object**.

End

22. **Fault_Object_i, i= {1, n}** should be stored in **FIV.**

End

End

Function 3 describes the fault injection system model generated from the simulation components. It extends Function 2 by generating the component-based system model based on the user requirements (customized building structure). As a result, the multiple fault injection algorithm should be merged with the system model generation script. Each component of the system, including the room, corridor, monitoring, and fault injection components, should be integrated to create the system model. Different components should be linked with connections through the script that require a configuration strategy to map the faults' attributes to each fault injection component. Indexing is an appropriate solution to model the fault injection components in each room block. Each room and system components, such as sensors and actuators, have an index during the system model generation. These indices are used to access the faulty components in multiple fault injection algorithm. This function should assign the number of rooms and

floors based on the user's requirements through the command panel. Then, the generated model is created based on the individual number of floors and rooms. Each component in the generated model should get an index to be accessible during the fault injection procedure. Each index in the generated model includes three main factors: the floor number, room number, and component number.

Function 3. Pseudo-code description for the generated system model.

Automated Fault Injection Algorithm

Begin

1. All system variables and inputs initialization.
2. Initialization of the fault injection vector as **FIV=0.**
3. Selecting the Number of days as **Num_Days = n.**
4. Getting the number of floors as **Num_Floors** from user
5. Getting the number of rooms in each floor as **Num_Rooms** from user
6. **For i=1:1: Num_Days**
7. Selecting the number of faulty samples which introduces the number of faulty samples in each day as **Num_FaultySamples**
8. Selecting the number of repetitions in each intermittent faulty sample as **Num_Repetitions**
9. Selecting the "**Systematic**" fault or "**Random**" fault
10. If ("**Systematic**"), all fault attributes should be assigned based on a predefined scenario.
11. If ("**Random**"), all fault attributes should be assigned randomly.
12. **For j=1:1: Num_FaultySamples**
13. Selecting the fault location and indexing. This index activates the target faulty component with **Index (#Num_Floor, #Num_Room, #Num_Component)**
14. Selecting the persistence type
15. **If** (Persistence is **Intermittent**), the following Matrices should be prepared based on the **Num_Repetitions.**
    a. Preparation of the fault injection time Matrix.
    b. Preparation of fault duration time Matrix.
    c. Preparation of the fault interarrival time Matrix.
16. **If** (Persistence is **Transient**), Only fault injection time and fault duration should be assigned once.
17. **If** (Persistence is **Permanent**), Fault injection time should be assigned once, and fault duration should take till the end of system model execution.
18. Invoking the Composable system model generation file.
19. Generation of the composable system model based on the **Num_Floors** and **Num_Rooms.**
20. Indexing all components based on the **Index (#Floor, #Room, #Component)**.
21. The system model execution file will be run.
    a. Fault types and faulty values should be assigned during the system simulation using a Stateflow diagram with a transition between faulty and healthy states for each repetition.
22. The system model should be closed after 86400 seconds.
23. An **Object** for each faulty sample is created **Fault_Object$_i$, i= {1, n}.**
24. All attributes for each faulty sample should be saved in this **Object**.
25. Output data from simulation execution should be saved in this **Object**.

End

23. **Fault_Object$_i$, i= {1, n}** should be stored in **FIV.**

End

End

4.3.4 Simulation Environment

Simulation tools allow the establishment and execution of a simulation model. Its parameters can be set, and its simulation results can be compared with real-world scenarios. To implement the simulation model, Matlab/Simulink, as a user-friendly tool, is beneficial and is utilized to implement our FI framework. Matlab/Simulink takes advantage of the SimScape blocks to represent a schematic physical system and mathematical equations [243]. In this thesis, the simulation model has been simulated in the Matlab/Simulink environment and a Stateflow diagram has been used to realize the fault injection. The next section introduces the model flow for the fault injection using finite-state machines.

4.3.4.1 Simulation Tools and Model Flow

To model the system's behavior, a finite hierarchical state machine (HSM) is used [40]. Figure 20 presents the timeline for the sequence of actions in the FI process. Each set of actions is a sequence of states from the correct mode to the failure mode at a related FI time. At the end of each action, the failure mode returns to the correct mode and then transitions to the second set of actions. This process continues until the last failure mode, and the FI process terminates. For example, three-set actions contain three different failure modes for an intermittent fault with three repetitions.

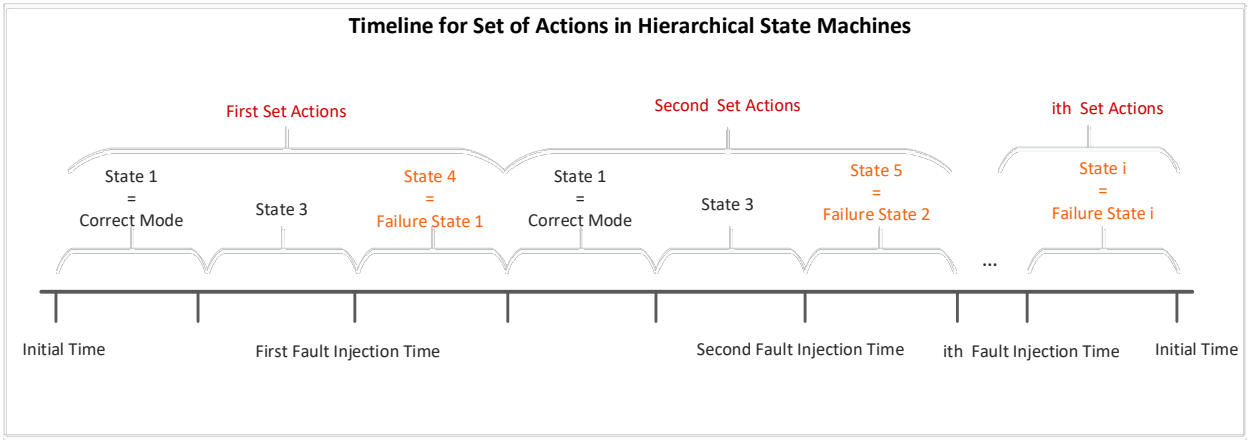

Figure 20. Timeline for actions in hierarchical state machines showing the sequence of failure modes.

Figure 21 shows a reactive finite-state machine between healthy and faulty states for the FI process. The faulty state consists of the persistence and failure states based on the number of faults (i.e., repetitions). The persistence state determines how many failures occur during the system execution time and the FI process. The model also specifies the transitions to the respective failure state based on the initial inputs, including the FI time and fault duration times that are initialized by the automated fault injection algorithm. For example, in Figure 21, the transitions with different colors define the set actions for the first failure mode, which occurs at the first FI time and the first fault duration time. A Stateflow diagram is applied to the fault injector blocks to implement this finite-state machine to produce the faulty values. Each variable of the Stateflow diagram has a fault attribute, a parameter of the FI process, or a variable of the system model, which can be defined with different types of input, output, local or global parameters.

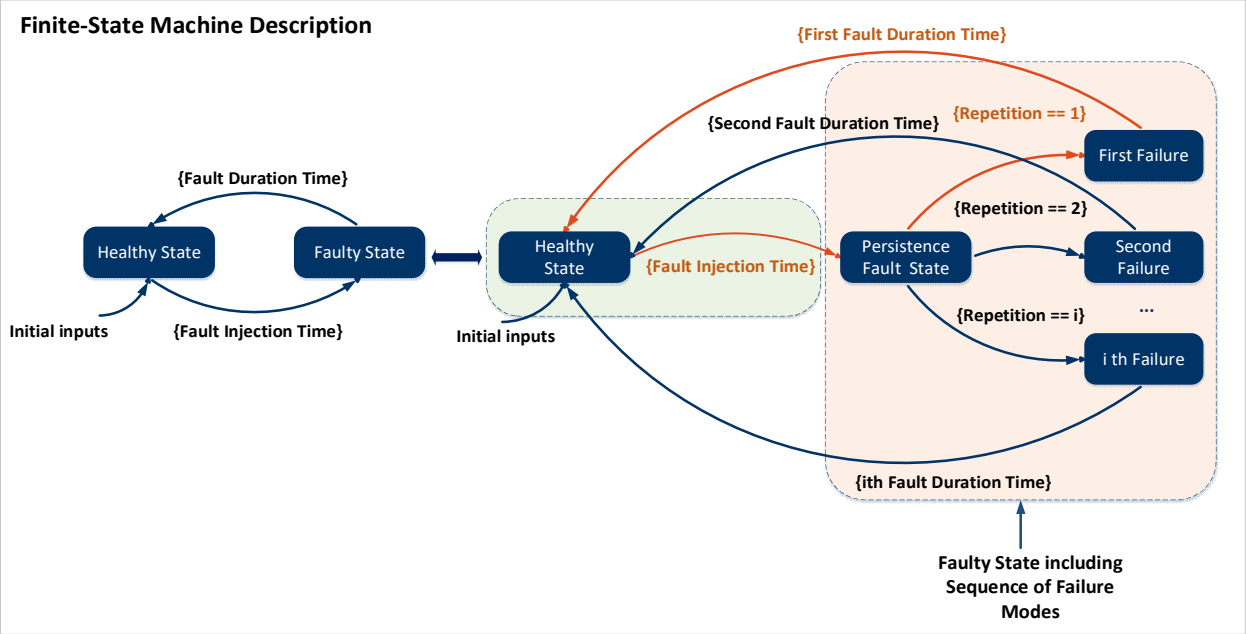

Figure 21. Finite-state machine implemented as a Stateflow diagram.

Furthermore, states change occur in the state machine during the FI process. Table 8 shows the state changes using the initial input patterns. For example, when the system meets the first fault injection and fault duration time, the state of the system changes to the faulty state, and subsequently, it changes to the healthy and faulty states in the presence of the other fault injection parameters.

Table 8. State transition table showing a Stateflow diagram for an intermittent fault with three repetitions.

Current State / Inputs	Healthy State	Faulty State		
		First Failure Mode	Second Failure Mode	Third Failure Mode
First injection time and duration time		×		
First interarrival time	×			
Second injection time and duration time			×	
Second interarrival time	×			
Third injection time and duration time				×

5 Fault Detection and Diagnosis Technique

This chapter introduces a generic and new hybrid Fault Detection and Diagnosis (FDD) technique for single fault occurrences in DCV and heating systems. The diagnostic algorithm combines two knowledge-driven and data-driven approaches. The knowledge-driven approach profoundly depends on expert knowledge. Expert knowledge is used for extracting the system attributes and their features. However, expert knowledge is not required for extracting the fuzzy rules and dependencies of system attributes. In this hybrid technique, a Bayesian Belief Network (BBN) uses statistical theories to discover hidden system correlations. For example, Mutual Information (MI) theory determines how random events or variables change when other events happen. In large-scale system structures with highly dependent system attributes, we require skillful and experienced experts that must spend significant time and energy to define the fuzzy rules to find the system attributes and their dependencies. This problem is solved with system attribute fuzzification and applying the BNN and MI theory to find the intrinsic casual relationships. The proposed hybrid FDD technique is generic and easily applicable to the different signal-based system models with numerous discrete and continuous signals and events. The fuzzy theory also provides appropriate likelihood distribution functions for calculating system attribute probabilities as membership functions to create Bayesian networks. Calculating the initial prior probabilities for the nodes of the network is challenging. This problem is solved by applying the fuzzy theory and membership functions for sample data tuples. Knowledge-driven approaches require long-term data acquisition for training the system. There is no experimental data for different types of HVAC systems with different system configurations. This problem is solved by training an extensible, offline library. This library is generated by evaluating the behavior upon various fault cases through a fault injection framework. Fault case definitions depend on the system requirements that must be precisely defined to obtain accurate fault diagnosis. Once attribute dependencies are discovered, the classifier diagnostic algorithm intervenes to map an actual fault case to the most relevant fault cases in the offline library. The hybrid FDD algorithm is explained in two specific phases including (1) the fuzzy and BBN construction and (2) the diagnostic method based on Fuzzy Bayesian Belief Network (FBBN). All steps of this FDD algorithm are modeled and explained in the following.

5.1 Fault Detection and Diagnosis Technique based on FBBN Phases

This section describes the overall steps of the FBBN fault detection and diagnosis technique that performs the network construction based on the fuzzified system attributes (knowledge-driven approach) and finds correlations between them using MI indicators with low expert effort. An automatic classifier algorithm (data-driven approach) enables fault diagnosis by classifying faults based on their similarities with online system execution and an offline library of various faults. This introduced fault detection and diagnostic technique enables an accurate diagnosis of actual permanent stuck-at faults for all system components. The FBBN diagnostic algorithm has been introduced in two specified modes: offline and online modes. Figure 22 gives a complete overview of the FBBN technique, including offline and online modes with their interrelationships and the diagnosis process.

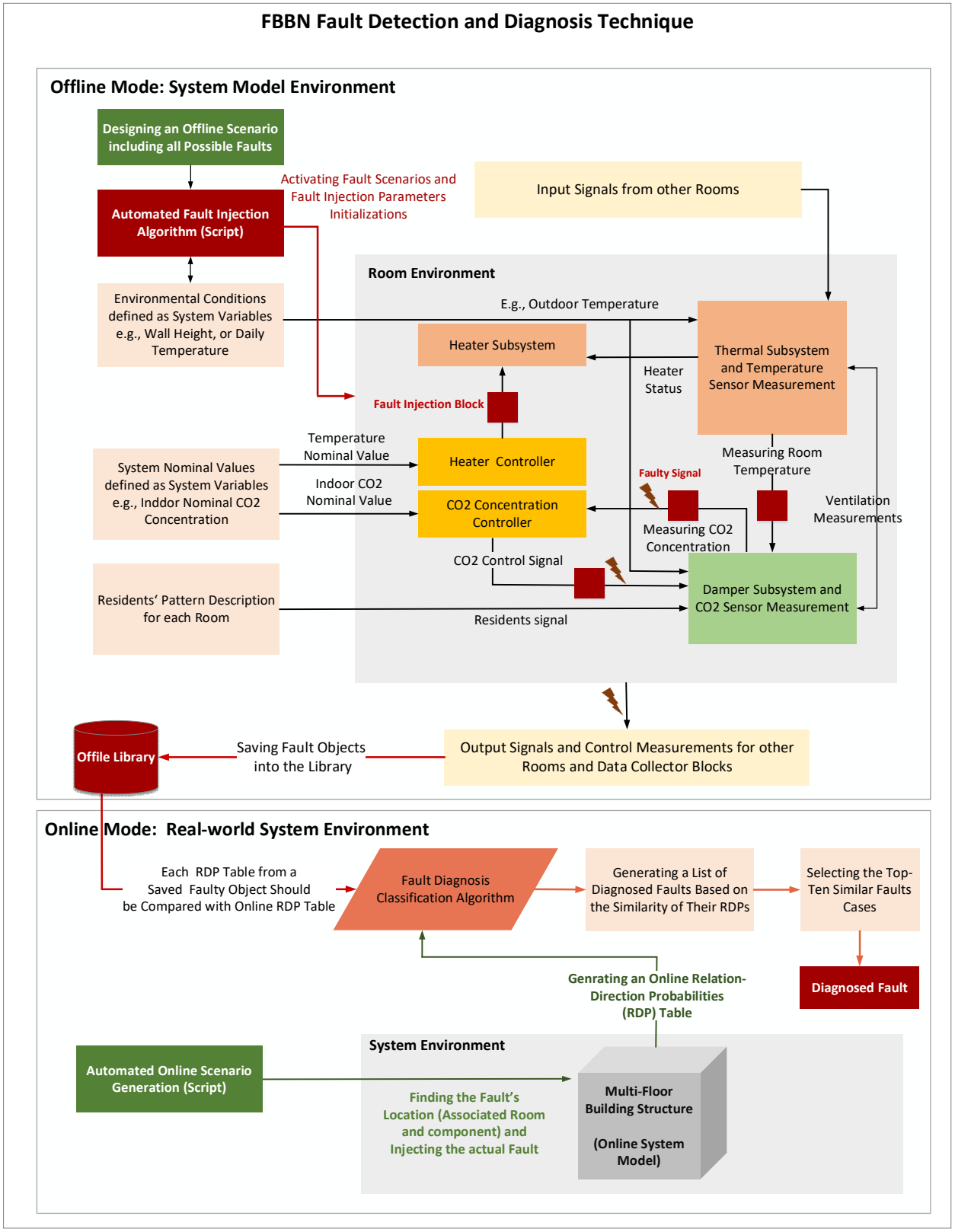

Figure 22. Overview of FBBN technique including offline and online modes and diagnosis process.

Offline mode is responsible for constructing an offline library using an automated fault injection algorithm. This offline library in the FBBN has been trained with various permanent stuck-at faults in different time domains for all system components (temperature sensor, heater actuator, CO_2 sensor, and damper actuator). The automated fault injection algorithm has injected each fault case through a predefined scenario. Each fault in the offline library is specified by an object with different properties such as time, type, location, and Relation Direction Probabilities (RDP) table. A table with the fuzzified system attributes and their causal relationships has been constructed and saved for each fault case. The causal relationships are stored in a table named RDP. RDP table can be visualized as a Bayesian Belief network with nodes (indices), edges (arcs), and values (probabilities). Each node is a fuzzified system attribute. Their probabilities have been calculated by fuzzy weights based on the associated system fuzzy rules. Fuzzy weights are assumed as the confidence factor of fuzzy system rules. Each pair node's dependency (correlations) is extracted from conditional probabilities. The higher conditional probability determines the direction from a parent to a child node. Directions (arcs) show the dependency of each pair of nodes.

Online mode includes a fault diagnosis classifier algorithm and actual fault injection. In online mode, the system should be run in a real-world environment in the presence of a random fault case. The FBBN algorithm diagnoses the fault features such as time, location, and type. The RDP table should be constructed based on the causal relations of the system attributes for the random injected fault. The fault diagnosis classifier classifies the fault cases based on the similarities of the RDP tables based on their mutual information. Therefore, the actual RDP table must be compared with the RDP tables of all fault cases in the offline library. A list based on the similarities (as percentages) of the RDP tables is created by comparing each offline RDP table with the online RDP. Higher percentages denote the most similar faults for actual fault cases.

5.1.1 Construction of Fuzzy and Bayesian Belief Network (FBBN)

This section introduces the steps of the FBBN construction. We require the casual relationships of the system attributes to build the BBN. To find the casual relationships, mutual information theory has been applied to find the correlation between the fuzzified system attributes. Figure 23 demonstrates the overall scheme of the FBBN construction with its respective steps. Each step is described in detail.

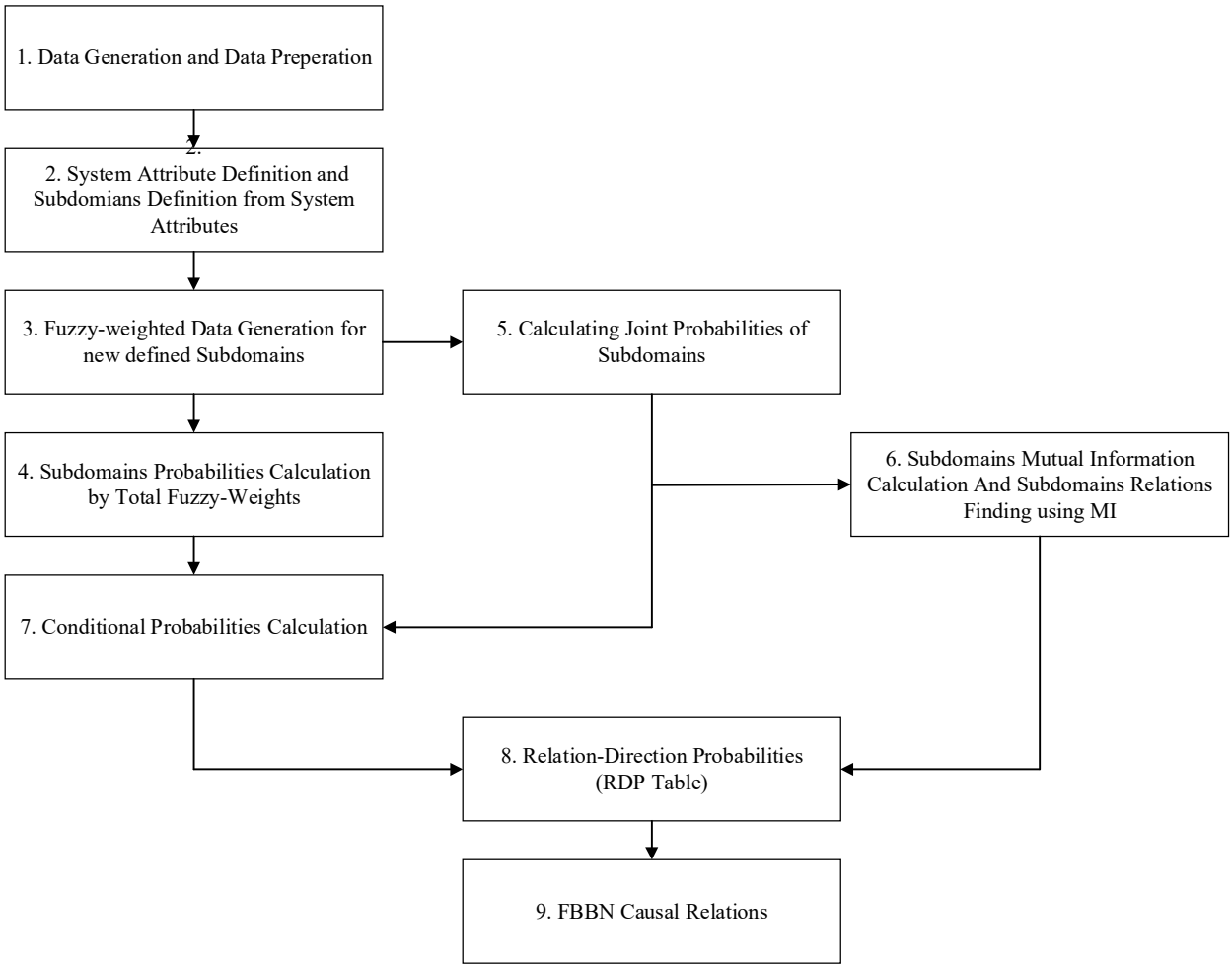

Figure 23.Fuzzy and Bayesian Belief Network (FBBN) construction steps and finding the casual relations using RDP tables.[18]

5.1.1.1 Data Generation and Data Preparation

Data Generation: The first step of the FBBN construction is data generation and data preparation. In this thesis, the automated single/multiple fault injection framework produces different types of faults to analyze the system behavior. The fault injection framework also supports fault diagnostic techniques by providing the required data for the diagnosis process. The fault injection framework studies the system behavior under faulty conditions and provides data for evaluating the fault diagnostic techniques to achieve an acceptable level of services. Therefore, the fault injection process generates the required recorded data for the FBBN.

Data Preparation: In the FBBN, a Relational Data Table (RDT) indicates all system measurements (values) at each sample time of each system execution. All values at one sample time are called a record of the system, including the measurements of all system attributes. The system can be described by random variables (attributes or domains) that obtain their values during the system runtime. The RDT table is created based on the generated data of the fault injection system, and the data samples include information on all system attributes as tuples [62]. An RDT table, RDT = $\{S_1, S_2, S_3, \ldots, S_n\}$, includes a number of data

samples, $S_i = \{Value_{i1}, Value_{i2}, \ldots, Value_{im}\}$, as a tuple of values for the i-th time instance. Table 9 shows the RDT table and the relation of samples, attributes, and their values over time.

Table 9. Relational Data Table (RDT) [18]

Samples	**Attribute$_1$**	**Attribute$_2$**	**Attribute$_3$**	**…**	**Attribute$_m$**
S_1	$Value_{11}$	$Value_{12}$	$Value_{13}$	…	$Value_{1m}$
S_2	$Value_{21}$	$Value_{22}$	$Value_{23}$	…	$Value_{2m}$
S_3	…	…	…	…	…
…	…	…	…	…	…
S_n	$Value_{n1}$	$Value_{n2}$	$Value_{n3}$	…	$Value_{nm}$

5.1.1.2 Definition of System Attributes and Subdomains

System Attributes: A system attribute is a random system variable that changes its value (e.g., as perceived by sensor measurements) over system runtime. In an FBBN, an attribute can be described by a value domain. The system has two kinds of attributes: (1) continuous attributes with continuous changes over time, e.g., temperature and CO_2 concentration sensor measurements, and (2) discrete attributes with discrete changes over time, e.g., damper and heater actuator setpoints.

Subdomains of Attributes: A domain is a set of values that ranges between thresholds. Each domain is divided into smaller ranges named subdomains. The domains of system attributes can be classified into subsets (called subdomains) of continuous or discrete values. For example, $Attribute_i = \{Subdomain_1, Subdomain_2, \ldots, Subdomain_p\}$ is the system's i-th attribute divided into a number of P subdomains. We require the calculation of system attribute probabilities to determine the correlations of system attributes. The fuzzy theory is a proper solution for calculating the probabilities of continuous attributes. Therefore, the continuous domains are classified into smaller subdomains using fuzzy functions. The probabilities of the discrete domains are calculated based on their discrete changes over time. Table 10 is Subdomain Label Table (SLT) describing all the system attributes and their associated subdomains generated and labeled newly in this step.

Table 10. Subdomain Label Table (SLT) [18]

No.	**Attributes**	**Subdomains**	**Subdomains**	**Subdomains**	**…**	**Subdomains**
1	$Attribute_1$	$Subdomain_{11}$	$Subdomain_{12}$	$Subdomain_{13}$	…	$Subdomain_{1n}$
2	$Attribute_2$	$Subdomain_{21}$	$Subdomain_{22}$	$Subdomain_{23}$		$Subdomain_{2e}$
…	…	…	…	…	…	…
n	$Attribute_n$	$Subdomain_{n1}$	$Subdomain_{n2}$	…		$Subdomain_{nf}$

5.1.1.3 Fuzzy-weighted Data Generation for Newly Defined Subdomains

In this step, we require the fuzzified system subdomains. The probability of each fuzzified subdomain is computed as a total fuzzy Weight (W) that differs in continuous and discrete system attributes:

Fuzzy-weighted data generation for continuous system attributes: The fuzzy theory provides an appropriate likelihood density function for calculating the probability of the continuous system attributes [192]. Therefore, a fuzzy weight must be computed in each subdomain as the fuzzy Membership Degree (MD) using the fuzzy Membership Function (MF) for each sample time and its corresponding value. MFs use the system measurements and produce the Membership Degrees (MDs) in the range of [0,1]. Each subdomain has its specified MF based on the system attribute features and ranges of changes. MD can be considered as the fuzzy weight or the probability of the system subdomain at the corresponding sample time. For example, the W_{11} is the MD or fuzzy weight of the $Subdomain_{11}$ of $Attribute_1$ at the first sample time. Table 11 is the Weighted Fuzzy Relational Data Table (WFRDT), which details each subdomain's total weight calculation. All weights (MD values) are extracted from the MF according to Equation 13 and summed up to compute the total fuzzy weight in each column.

Table 11. Weighted Fuzzy Relational Data Table (WFRDT) [18]

	$Attribute_1$				**$Attribute_2$**			
No. of Records	$Subdomain_{11}$	$Subdomain_{12}$	…	$Subdomain_{1m}$	$Subdomain_{21}$	$Subdomain_{22}$	…	$Subdomain_{2e}$
1	W_{11}	W_{12}	…	W_{1m}	W_{11}	W_{12}	…	W_{1e}
2	W_{21}	W_{22}	…	W_{2m}	W_{21}	W_{22}	…	W_{2e}
…	…	…	…	…	…	…	…	…
N	W_{n1}	W_{n2}	…	W_{nm}	W_{n1}	W_{n2}	…	W_{ne}
Total Weight	$W_1 = \sum_{11}^{n1} W$	$W_2 = \sum_{12}^{n2} W$	…	$W_m = \sum_{1m}^{nm} W$	$W_1 = \sum_{11}^{n1} W$	$W_2 = \sum_{12}^{n2} W$	…	$W_e = \sum_{1e}^{ne} W$

Various types of fuzzy membership functions can be chosen based on the system requirements, such as triangular, trapezoidal, Gaussian, and bell-shaped. In this thesis, the trapezoidal MF has been used to calculate the fuzzy weights. Equation 13 is the trapezoidal MF that should be initialized for each continuous fuzzified system subdomain to measure the MD of the value x. X is an actual or faulty system measurement (cf. values in Table 9).

$$Degree\ of\ Membership(x:a,b,c,d) = \begin{cases} 0 & x < a \\ \frac{(x-a)}{(b-a)} & a \le x \le b \\ 1 & b \le x \le c \\ \frac{(d-x)}{(d-c)} & c \le x \le d \\ 0 & x \ge d \end{cases} \qquad \text{Equation 13}$$

Fuzzy-weighted data generation for a discrete system attribute: In each subdomain, the fuzzy weights are equal to measured values of the system over time which is 0 or 1 based on the discrete variable statuses. These values show the status values considered as probability values. For example, the damper actuator status is equal to 0 when it is in the close status and 1 when it is in the open status. Therefore, the total weight for each discrete system attribute will be calculated based on Equation 14.

$$Total\_Weight = \sum_{i=1}^{n} Value_i = \sum_{i=1}^{n} W_i. \quad \text{Equation 14}$$

5.1.1.4 Subdomain Probability Calculation using Total Fuzzy-Weights

This thesis considers each new fuzzified subdomain (i.e., fuzzy set) as a random variable. Accordingly, the probability of subdomains can be calculated based on Equation 15**.** For example, the probability of the subdomains A and B is shown in Equation 16.

$$P(Subdomain_i) = \frac{\sum_{i=1}^{n} Weight_i}{|n|} = \frac{Total\_Weight_i}{|n|}, n = \{1,...,R\}. \quad \text{Equation 15}$$

$$P(A) = \frac{\sum_{k=1}^{|R|} A(d_{kj})}{|n|}, A(d_{kj}) \in [0,1], \quad and \quad P(B) = \frac{\sum_{k=1}^{|R|} B(d_{ki})}{|n|}, B(d_{ki}) \in [0,1]. \quad \text{Equation 16}$$

In Equation 15 to calculate the probability, all membership degrees for a subdomain in Table 11 are summed up (total weight) and divided by the number of samples (records). also describes the probability of the i-th subdomain in which n is the number of records. shows the Subdomain Probability Vector Table (SPV), where the probabilities of all system subdomains are calculated based on the total fuzzy weights.

Table 12. Subdomain Probability Vector Table (SPV) [18]

	Attribute$_1$							**Attribute$_2$**
Subdomains	Subdomain$_{11}$	Subdomain$_{12}$	…	Subdomain$_{1m}$	Subdomain$_{21}$	Subdomain$_{22}$	…	Subdomain$_{2e}$
Probability of Subdomain	$P_1 = \frac{W_1}{n}$	$P_2 = \frac{W_2}{n}$	…	$P_m = \frac{W_m}{n}$	$P_1 = \frac{W_1}{n}$	$P_2 = \frac{W_2}{n}$	…	$P_e = \frac{W_e}{n}$

5.1.1.5 Joint Probability Calculation for Subdomains

A joint probability of two events can be defined as their intersection when they coincide. In this thesis, we calculate the joint probabilities for dependent events when the probability of one subdomain changes the probabilities of the other ones. When one subdomain changes the probability of the other subdomains, they are dependent, and their intersection is not zero. Otherwise, the intersection of independent subdomains

is zero. Joint probabilities of dependent subdomains are used to calculate conditional probabilities and mutual information.

In this thesis, when the two fuzzy weights in a pair of subdomains are compared at a sample time, the minimum fuzzy weight is considered as their intersection. This action repeats for all sample times to calculate the joint probability of the subdomain pairs based on Equation 17.

$$P(A,B) = P(A \cap B) = \frac{\sum_{k=1}^{|R|} \min\left(A(d_{kj}), B(d_{ki})\right)}{|R|} = \frac{\sum_{1}^{n} \min\left(A(WSubdomain_i), B(WSubdomain_j)\right)}{n}. \quad \text{Equation 17}$$

A triangular top/down matrix is an appropriate way to show the relationships of each pair of system subdomains. The joint probability of P(A, B) equals P(B, A). Therefore, the intersection (joint) probabilities of subdomains are generated in an Intersection Triangular Top Matrix (ITTM) as shown in Table 13 using Equation 17 in which $A(WSubdomain_i)$ and $B(WSubdomain_j)$ are the corresponding fuzzy weights of subdomains A and B at the i-th sample time.

For example, $P(Subdomain_3, Subdomain_5) = P(Subdomain_5, Subdomain_3)$ shows the joint probability of the $Subdomain_3$ and $Subdomain_5$, and its value can be located in Table 13based on the table indices.

Table 13. Intersection Triangular Top Matrix (ITTM) [18]

Subdomains / Subdomains	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	…	i-1	i
1																		
2																		
3					P(3,5)													
4																		
5																		
6																		
7																		
8																		
9																		
10																		
11																		
12																		
13																		
14																		
15																		
…																		
i-1																		
i																		

5.1.1.6 Mutual Information Calculation and Relation Finding of Subdomains

MI is the information of one random variable when other random variables are observed. Ensembles can be considered random variables [71]. Therefore, in this thesis, each new fuzzified subdomain is considered a random variable. The mutual information of each subdomain is calculated when another

subdomain is observed. Multivariate mutual information is introduced in the basic concept. MI can also be calculated for only two random variables. For example, Intan et al. [62] have introduced the mutual information of two fuzzy sets of A and B in Equation 18.

$$MI(A,B) = MI(B,A) = P(A,B)\log_2\left(\frac{P(A,B)}{P(A)\times P(B)}\right), P(A) \neq 0, P(B) \neq 0. \quad \text{Equation 18}$$

In Equation 18, (A) and P(B) are the fuzzy-set probabilities. P(A, B) is the joint probability or intersection between the two fuzzy sets of A and B. Mutual information of MI(A, B) is an indicator for determining the correlation between two random variables of A and B.

One can distinguish three conditions of MI: If (MI(A, B) = 0) or (MI(A, B) < 0), there is no correlation (negative dependency) between two random variables of A and B. However, if (MI(A, B) > 0), there is a positive correlation (positive dependency) between two random variables of A and B. The MI for all pairs of new fuzzified subdomains should be calculated. The positive MI values indicate the dependent subdomains, and negative MI values indicate the independent subdomains. In FBBN, only the positive MI values are considered as "1" binary values and are located in their respective places in the Subdomain Relation Table (SRT). Otherwise, the "0" binary value is placed for the independent subdomains. Table 14 contains an SRT table showing the dependent fuzzified subdomains in the FBBN. For example, the MI($Subdomain_6$, $Subdomain_9$) = MI($Subdomain_9$, $Subdomain_6$) = 1 demonstrates a strong correlation or dependency between the two subdomains of $Subdomain_6$ and $Subdomain_9$ as shown in Table 14.

Table 14. Subdomains Relation Table (SRT) [18]

Subdomains / Subdomains	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	…	i
1																		
2																		
3																		
4																		
5																		
6									1									
7																		
8																		
9																		
10																		
11																		
12																		
13																		
14																		
15																		
16																		
…																		
i																		

5.1.1.7 Calculation of Conditional Probabilities

In an FBBN, conditional probabilities comprise directions and probabilities. In a fuzzy Bayesian belief network, two types of nodes, parent and child, are connected via arcs. The direction between two nodes (the direction of the arc) can be determined by the conditional probability calculation of two events. The conditional probability is the likelihood of one conditional event according to the occurrence of the previous event/s. The conditional probability of the two events of A and B can be denoted as P(A|B) and P(B|A). P(A|B) shows that A is the preceding event and B is the succeeding event. Although, in P(B|A), B is the preceding event, and A is the succeeding event. In an FBBN, the precedence shows a parent node and a succeeding a child node. Equation 19 calculates the conditional probability of two fuzzy sets of A and B based on the computed fuzzy weights. In Equation 19, P(A, B) is the intersection between two fuzzy sets, A and B. In an FBBN, conditional events are fuzzified subdomains. Then, P(A, B) can be described as $P(\text{subdomain}_i \mid \text{subdomainj})$ as shown in Equation 20.

$$P(A \mid B) = \frac{P(A,B)}{P(B)} = \frac{\sum_{k=1}^{|R|} \min\left(A(d_{kj}), B(d_{ki})\right)}{\sum_{k=1}^{|R|} B(d_{ki})}. \qquad \text{Equation 19}$$

$$P(subdomain_i \mid subdomain_j) = \frac{\sum_{1}^{n} \min\left(A(WSubdomain_i), B(WSubdomain_j)\right)}{\sum_{1}^{n} B(WSubdomain_j)}. \qquad \text{Equation 20}$$

Conditional probabilities must be computed for events with positive correlations in SRT generated based on the MI theory. A top/down matrix is required to store the conditional probabilities because P(A|B) and P(B|A) have different values. Therefore, the results from the above equations for all new fuzzified subdomains should be calculated and stored in a matrix called Conditional Probabilities Table (CPT) as shown in Table 15 Finally, based on the two following rules, only one conditional probability should be retained in the CPT:

Rule 1: If P(A|B) > P(B|A), it indicates that the direction of a dependency between two conditional events of A and B is from B (Parent) to A (Child). Then, P(B|A) must be eliminated, and P(A|B) must be saved in the CPT table.

Rule 2: If P(B|A) > P(A|B), it indicates that the direction of a dependency between two conditional events A and B is from A (Parent) to B (Child). Then, P(A|B) must be eliminated, and P(B|A) must be saved in the CPT table.

For example, we know that $P(\text{Subdomain}_4 \mid \text{Subdomain}_5) \neq P(\text{Subdomain}_5 \mid \text{Subdomain}_4)$; hence, their values should be compared to determine the bigger value. If $P(\text{Subdomain}_4 \mid \text{Subdomain}_5) > P(\text{Subdomain}_5 \mid \text{Subdomain}_4)$, then the $P(\text{Subdomain}_4 \mid \text{Subdomain}_5)$ should be saved in the CPT and $P(\text{Subdomain}_5 \mid \text{Subdomain}_4)$, should be eliminated.

Table 15. Conditional Probabilities Table (CPT) [18]

Subdomains / Subdomains	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1																		
2																		
3																		
4					P(4|5)													
5				~~P(5|4)~~														
6																		
7																		
8																		
9																		
10																		
11																		
12																		
13																		
14																		
15																		
16																		
17																		
18																		

5.1.1.8 Relation-Direction Probability (RDP) Table

In this thesis, a FBBN is constructed based on relations, directions and probabilities of system attributes. All this required information is computed and stored in a table called CPT. Eventually, a table called the Relation-Direction-Probability (RDP) is extracted from the CPT table and includes all system subdomain relations such as parent's nodes, children's nodes, and their corresponding conditional probabilities as demonstrated in Table 16. This table only includes the subdomains that have positive correlations and denotes how they are connected to each other based on the Bayesian network features such as nodes and their associated probabilities. For example, one record of the RDP shows that there is a positive correlation between $Subdomain_i$ and $Subdomain_j$. Then, there is a connection from $Subdomain_i$ (parent node) to $Subdomain_j$ and $Subdomain_j$ (child node) with conditional probability of $P(Subdomain_j \mid Subdomain_i)$.

Table 16. Relation Direction Probability (RDP) [18]

Number of relations	**Parents**	**Children**	**Conditional Probabilities**
1	$Subdomain_i$	$Subdomain_j$	$P(Subdomain_j \mid Subdomain_i)$
2	$Subdomain_k$	$Subdomain_w$	$P(Subdomain_w \mid Subdomain_k)$
3	…	…	…
….	…	…	…
n	$Subdomain_n$	$Subdomain_m$	$P(Subdomain_m \mid Subdomain_n)$

5.1.1.9 FBBN Causal Relations

In this thesis, a FBBN is a network comprised of a few essential elements, such as the parent node, child node, and edge (arc)**.** All causal relationships of the system attributes are classified as fuzzified subdomains. They can be figured out as a FBBN graph shown in Figure 24.

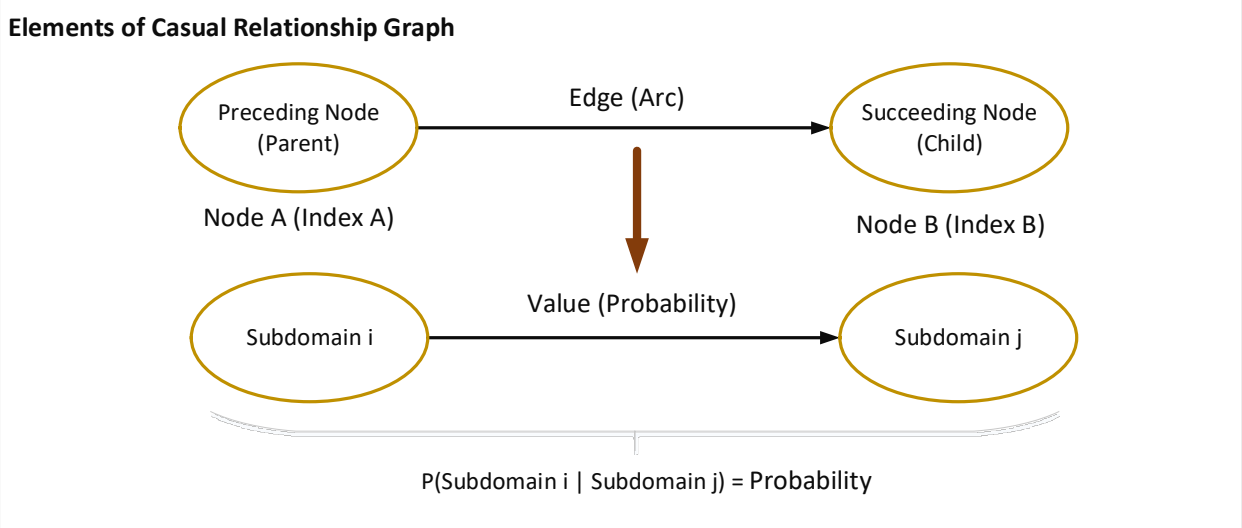

Figure 24.The causal relationship between two system attributes has been indicated as a graph.

When the RDP table is generated, the FBBN network is constructed using the relations, directions, and conditional probabilities of the subdomains. This network includes parent nodes, children nodes, and arcs which show the probability and direction of the connections between nodes. Each fault case injected by the proposed automated single/multiple fault injection algorithm includes fault features such as type, time, and output data. In addition, the generated RDP table is added as a fault feature into each fault object.

5.1.2 Classifier-based Diagnostic Algorithm using Fuzzy Bayesian Belief Network

The introduced fault detection and diagnosis method comprises two main modes: offline and online. Offline mode introduces the FBBN construction for each trained offline library fault case. Online mode uses the classifier-based diagnostic algorithm that diagnoses a real fault case. Each mode is described as in the following:

5.1.2.1 Offline Training Mode

The main procedure of the offline mode consists of offline library creation. This procedure generates and injects several stuck-at-fault types for all system components with various stuck-at values at different time instances. For each fault case, the output data and RDP tables are generated. Each fault case consists of four fault properties: fault type, time of fault injection, output data and the RDP table that must be stored in a fault object. Finally, all fault objects from different fault cases are stored in an offline library indicated in Table 17.

Table 17. Fault injection vector as offline library including the information of all fault cases [18]

No. of Faults	**1**	**2**	**3**		**n-1**	**n**
Objects for Different Fault Cases	$Fault\_Object_1$	$Fault\_Object_2$	$Fault\_Object_3$	…	$Fault\_Object_{n-1}$	$Fault\_Object_n$

Each subdomain has a representative offline library in different time intervals and fault types. The offline training mode of the FBBN diagnostic algorithm with its respective steps is introduced in Function *4*.

Function 4. Pseudo-code description for offline mode of fuzzy Bayesian belief network fault diagnosis technique

Offline mode that constructs the fuzzy Bayesian belief network to discover causal relationships

Offline Mode
Begin

1. Initialization of the fault injection vector **FIV=0** as **Offline_Library.**
2. Preparation of a **Scenario ={ $Scenario_1$, …, $Scenario_R$ }** for fault injections**. R** is the number of scenarios.
3. **For j=1:1:R**
4. Generating the **$Output\_Data_j$** for each **$Scenario_j$,** where **$Output\_Data_j = \{S_1, S_2, S_3, …, S_n\}$, n={1, Nr. Of Samples in One Execution }** and S_i is the i-th sample.
5. Data preparation as **RDT** table.
6. Defining the system **Attributes.**
7. Defining new fuzzified subdomains from system attributes, where **Attribute-i = {$Subdomain_1$, $Subdomain_2$, …, $Subdomain_p$}** as **SLT_j** table**.**
8. Fuzzification of subdomains and generation of fuzzy weights for each subdomain as **$WFRDT_j$** table.
9. Calculation of total fuzzy weight for each subdomain as $\boldsymbol{W_i = \sum_{11}^{ni} w}$ in the **$WFRDT_j$** table.
10. Calculation of subdomain probabilities using the total weight, $\boldsymbol{P_i = \frac{W_i}{n}}$ as **SPV_j** table**.**
11. Calculation of joint probabilities of subdomains using fuzzy weights for all pairs of subdomains as **$ITTM_j$** table.
12. Calculation of Mutual Information (**MI**) between each pair of subdomains as **SRT_j** table
13. Finding the relations of subdomains using **MI**.
    a. **If (MI > 0)**, there is a correlation between the two subdomains.
    b. **If (MI<=0)**, there is no correlation between the two subdomains.
14. Calculation of conditional probabilities for pairs of subdomains with positive correlation as **CPT_j** table.
15. Creating a Relation-Direction-Probability table as **RDP_j , j = {1, R}.**
16. An **Object** for each **$Scenario_j$** is created as a **$Fault\_Object_j$, j = {1, R}.**
17. Saving all fault features for **$Scenario_j$** and its calculated **RDP_j** in the **$Fault\_Object_j$**.
18. Saving the **$Fault\_Object_j$** as **Offline_Library [j] = $Fault\_Object_j$.**

End

19. Saving **Offline_Library**

End

5.1.2.2 Online Diagnostic Mode

Online mode includes a classifier-based algorithm to diagnose single stuck-at faults based on the FBBN technique using the ranking method. In online mode, a real fault case can be injected for validation purposes using the automated fault injection with a selection of random fault features. The fault object of the real fault-case scenario includes six properties, including the Type, Time, Output_Data, RDP table,

Precentage_List, and Evaluation_List. Precentage_List is a list containing all percentages of mutuality between the injected real fault case with all fault cases in the offline library. Precentage_List= {Percentage of Mutuality$_1$, …, Percentage of Mutuality$_i$ }, i={1,…, R} serves for calculating the percentage of mutuality. The real fault case RDP table must be pairwise compared with the RDP tables of all faults in the offline library. Table 18 indicates the percentage list of the fault object for the real fault-case. Each fault object in the offline library includes n rows of parent-child pairs and their respective conditional probability. Each parent-child pair in the actual RDP table is compared with each parent-child pair of all offline library fault cases to find the percentage of similarities. The percentage of similarity is considered as the mutuality indicator.

Table 18. Percentage list of a fault object for a real fault-case [18]

No. of Fault Objects in the Offline Library	**1**	**2**	**…**	**i**
Percentage of Mutuality Between the Fault Object for the Real Fault-Case and Offline Library Fault Cases	Percentage of Mutuality$_1$	Percentage of Mutuality$_2$	…	Percentage of Mutuality$_i$

In this thesis, for diagnosing the occurred fault, the ranking method has been used for classifying the most probable similar faults. The Precentage_List is sorted and the highest percentages show the most similar faults (or the faults with the highest correlation) in the offline library. An Evaluation_List is extracted from the Precentage_List and the offline library. In a way, the fault properties of the ranked faults are extracted from the offline library, such as the time, type, and the values of percentages coming from the Precentage_List. Therefore, all elements of the Precentage_List are ranked from the higher percentages of mutuality to the lower percentages of mutuality with their details in the Evaluation_List. The highest ranks are compared with the fault object of the real fault-case to diagnose the accuracy of the diagnostic algorithm. Table 19 demonstrates the evaluation list in the online diagnostic mode of the classifier-based diagnostic algorithm using the FBBN construction.

Table 19. Evaluation list of a fault object for a real fault-case [18]

No.	**Type**	**Time**	**Percentage**
1	Offline_FaultType$_1$	Offline_FaultTime$_1$	Highest_Precentage$_1$
2	Offline_FaultType$_2$	Offline_FaultTime$_2$	Highest _Precentage$_2$
3	Offline_FaultType$_3$	Offline_FaultTime$_3$	Highest _Precentage$_3$
…	…	…	…
j	Offline_FaultType$_j$	Offline_FaultTime$_j$	Highest _Precentage$_j$

Function 5 describes the pseudo-code for the online mode using the FBBN fault diagnosis technique with all respective steps. In this function, the procedure of the RDP table construction is repeated for the actual online fault case. Then, the Precentage_List and Evaluation_List are created based on the

similarities to rank all fault cases. The fault cases with higher ranks are considered as the diagnosed fault cases determining the type and time of the closest fault cases compared to the online fault case. The users of the system including the operators and the engineers can use the diagnosed fault cases for system maintenance or redesign strategies.

Function 5. Pseudo-code description for online mode of FBBN fault diagnosis technique.

Online mode and the classifier-based diagnostic algorithm

Online Mode

Begin

1. Initialization of the **Offline_Library** made in Offline Mode
2. Preparation of a FI random **Scenario** for FBBN
3. Generating the **Output_Data**
4. Data preparation as **RDT** table.
5. Defining the system **Attributes.**
6. Defining new fuzzified subdomains from system attributes, where **Attribute-i = {Subdomain$_1$, Subdomain$_2$, …, Subdomain$_p$}** as **SLT** table**.**
7. Fuzzification of subdomains and generation of the fuzzy weights for each subdomain as **WFRDT$_j$** table.
8. Calculation of total fuzzy weight for each subdomain as $\boldsymbol{W_i = \sum_{11}^{ni} w}$ in **WFRDT** table.
9. Calculation of subdomain probabilities using the total weight, $\boldsymbol{P_i = \frac{W_i}{n}}$ as a **SPV** table**.**
10. Calculation of joint probabilities of subdomains using fuzzy weights for all pairs of subdomains as **ITTM** table.
11. Calculation of Mutual Information (**MI**) between each pair of subdomains as **SRT** table
12. Finding the relations of subdomains using **MI**.
    a. **If (MI> 0)**, there is a correlation between the two subdomains.
    b. **If (MI<=0)**, there is no correlation between the two subdomains.
13. Calculation of conditional probabilities for pairs of subdomains with positive correlation as **CPT** table.
14. Creating a Relation-Direction-Probability table as **RDP$_j$, j= {1, R}.**
15. An **Object** for each **Scenario$_j$** is created as a **Fault_Object$_j$, j= {1, R}.**
16. Initializing a new vector for saving the most similar faults in **Offline_Library,** as **Similar_Faults[Num]=0** and **Num=10** as the number of similar faults.

//Classifier Algorithm Using the Fault Ranks

17. **Similar_Faults** = Function **Compare (Offline_Library, Current_RDP)** // Compare function returns the most similar faults with online injected faults by comparing the RDP tables in Offline_Library and Current_RDP table
18. Generating a list of similar faults with percentages for **Current_RDP, Precentage_List= {Percentage of Mutuality$_1$, …, Percentage of Mutuality$_i$ }, i={1,…,R}**
19. Generating an evaluation list for **Current_RDP, Evaluation_List**
20. Choosing the most similar fault in the **Similar_Faults** array as the diagnosed fault and diagnosing the **Time**, **Type,** and **Location** of the fault.

End

6 Implementation

This chapter presents the implementation of the introduced techniques. In this thesis, all techniques are implemented and simulated in the MATLAB/Simulink environment using MATLAB/Programming. Simulation imitates the real-world system behavior over time to get insights into quality attributes (e.g., quality of control), validate the system behavior in the design phase, and reduce the implementation costs. MTALB is an interactive multi-paradigm programming environment for scientific and technical approaches, e.g., data analysis, computations, matrix manipulations, and user interface creation. MATLAB provides various toolboxes constructed from libraries of functions. For example, the Simulink package in MATLAB provides graphical modeling for dynamic and embedded systems [95, 244, 245]. This chapter contains three main sections. The chapter starts by providing an example scenario of a DCV and heating system using a large-scale multi-zone building system model. A simulation model implements the fault injection framework in MATLAB/Simulink, and implementing the classifier fault diagnostic algorithm using the fuzzy Bayesian belief network. All system elements are implemented as generic simulation components, e.g., room, corridor, monitors, controllers, and fault injectors. Our system model has a component-based structure. All components must be integrated and automatically embedded in a large-scale building. The components are connected via links through an automatic script. Since the connectivity of the system components through links in a complex structure with numerous components can result in implementation errors, automation by generating generic system components that are merged with less effort is valuable. The fault injection components are completely generic and can be integrated with other component-based and non-component-based systems. Large-scale building structures are more error-prone due to their complexity and high numbers of system components. DCV and heating systems, which control and balance the air quality and thermal conditions, play a significant role in residents' comfort and emergencies, e.g., fire and toxic gas emissions. As a result, considering fault control and maintenance strategies, including detection, diagnosis, and recovery, is essential. This thesis proposes a hybrid fault diagnostic technique combining the fuzzy and Bayesian belief networks to cover permanent stuck-at faults diagnosis. This technique combines a data-driven strategy for automation and classification with a knowledge-driven strategy for adding expert knowledge for extracting the system information and characteristics to enhance the accuracy of the diagnostic algorithm. The knowledge-driven technique combines the fuzzy theory strategy to decrease the expert effort for defining the fuzzified attributes and increases the compatibility and universality of the technique for large-scale system structures with numerous component types. Bayesian belief network theory allows to find the fuzzified attribute correlations in the fault diagnostic process. Figure 25 shows how the modeling parts (including the system model of the simulation environment of HVAC systems in chapter 4 and fault detection and diagnosis service from chapter 5 are linked to different parts of the implementation in chapter 6. The implementation chapter consists of four main parts: (1) the implementation of the fault injection components, (2) the implementation of the fault injection algorithm, (3) the implementation of the large-scale component-based system model, and (4) the implementation of the diagnostic classifier algorithm based on the FBBN construction.

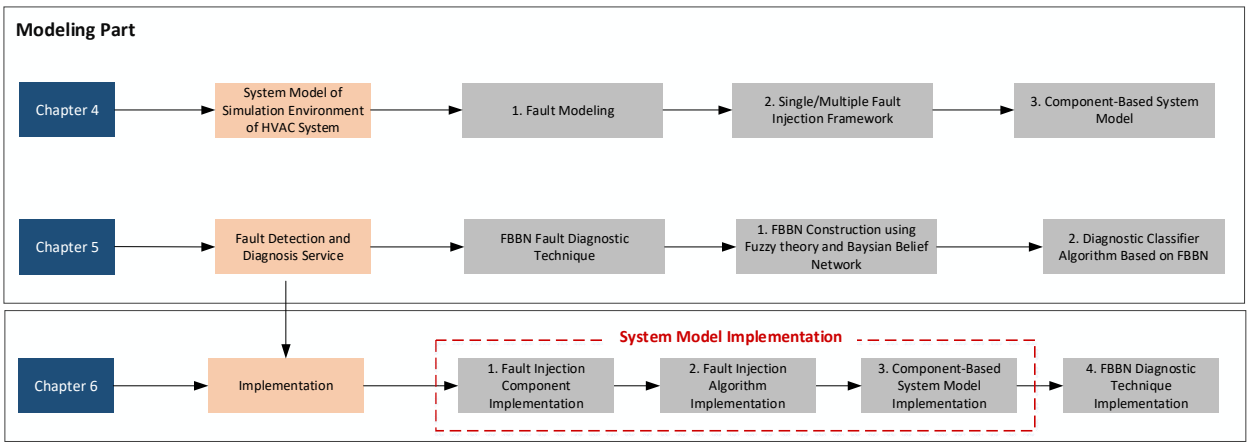

Figure 25. Implementation of simulation model, fault injection and diagnosis

6.1 Implementation of the Fault Injection Component in MATLAB/Simulink

An example scenario of a multi-zone building for DCV, and heating systems is considered for the simulated system model to implement the fault injection components. Behravan [32][25] has introduced a multi-zone and component-based building structure that is applied to evaluate and validate the proposed techniques in this thesis. This example scenario is implemented in MATLAB/Simulink for a realistic office building at the University of Siegen in Germany with six rooms and one corridor. The model and the system assumptions are explained to understand the developed techniques. For example, it is explained how the generic fault injection components must be integrated with other electronic components. In addition, the healthy system conditions are described to enable understanding the experimental evaluations results.

6.1.1 Example Scenario of a Multi-Zone Building System Model

All introduced techniques, including the single and multiple fault injection and the classifier-based fault diagnostic algorithm, are applied in an instantiated DCV and heating system model. This grey-box system model integrates physical and mathematical descriptions of the system objects to describe a real-world HVAC system. The physical model of the DCV and heating system (introduced in chapter 4) is simulated in MATLAB/Simulink, and its subsystems and the system outputs are verified to show the correctness of the system behaviors and responses [25, 32]. The system model is implemented based on the thermal dependencies among distinct zones using the SimScape library. In addition, SimScape is a practical library in MATLAB/Simulink to model the power transmission in thermal subsystems by demonstrating the physical connection lines among real-world system components modeled as Simulink blocks [25, 95]. Figure 26 shows the overall scheme of the simulated multi-zone office building with six rooms and one corridor. Room interconnections and their input and output ports are represented. Each room is considered as a system component comprising other electronics and subsystems such as thermal, damper, and heater subsystems shown in Figure 27.

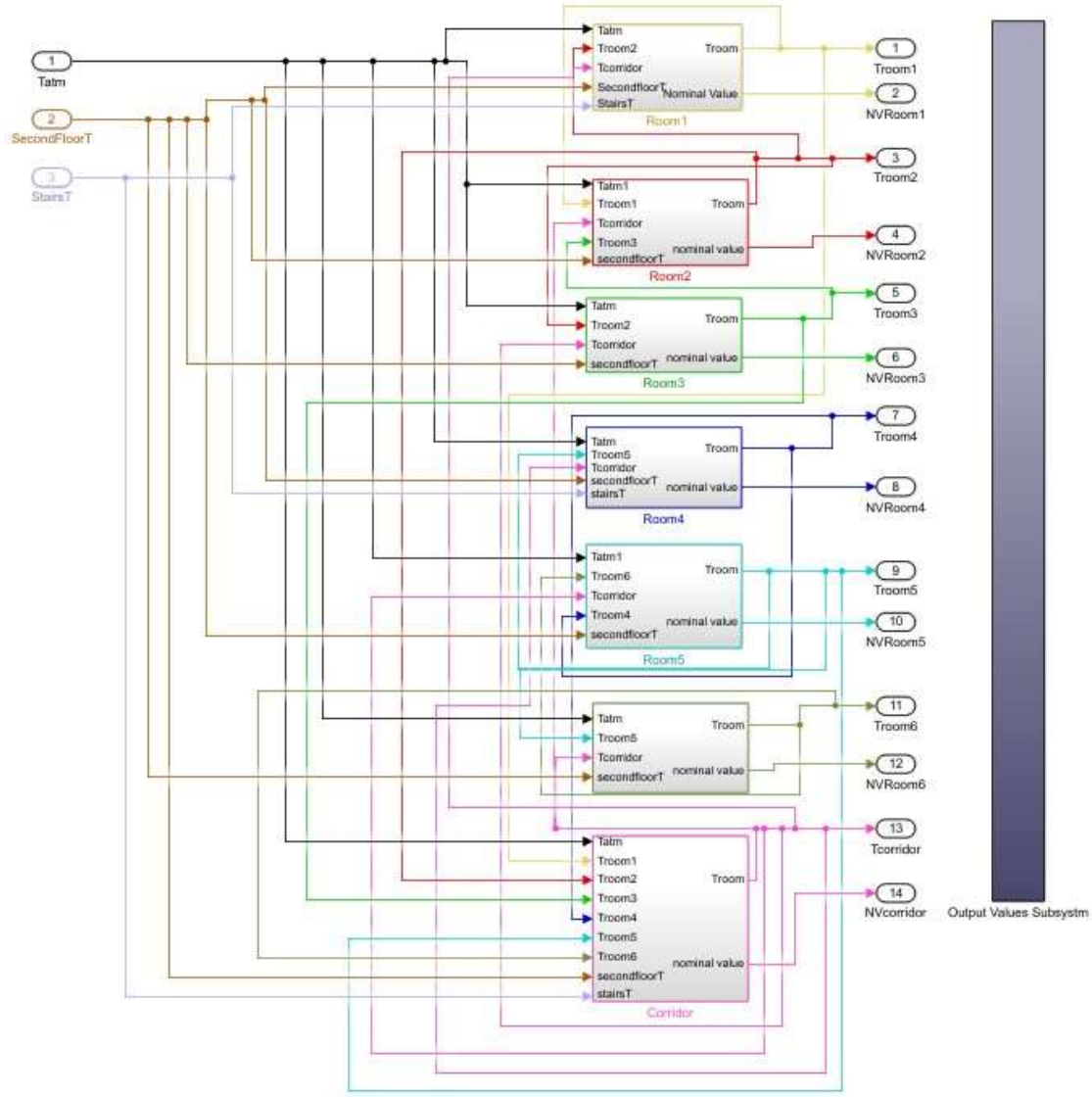

Figure 26. Overall scheme of the simulated multi-zone office building with six rooms, one corridor, and a data collector [25]

Figure 27 illustrates the interior view of one simulated room component containing the heating and DCV subsystems without fault injection components. The input parameters of the thermal subsystem (e.g., for the first room component) are outdoor temperature, the next-door room block temperature (e.g., a second room), corridor temperature, second-floor temperature (20°C), stair temperature (13.5°C) and ventilation rate from the DCV subsystem. The output of the thermal subsystem is the current room temperature. In addition, the thermal subsystem includes temperature sensors. In this example scenario, the measured temperature from each room changes between an upper threshold (22.5 ºC) and a lower threshold (17.5 ºC) and is compared with the daily temperature and the nominal value (20 ºC). Any changes over the thresholds and the healthy behavior of the thermal conditions can be considered as faulty [25]. The outdoor air temperature (daily temperature) is modeled as a sinusoidal wave. The initial daily temperature is 7°C, fluctuating between 2°C and 12°C during the day. The initial time of system execution has been considered at 6:00 a.m. with a 7°C outdoor temperature that continues for 86400 seconds (one day) [25].

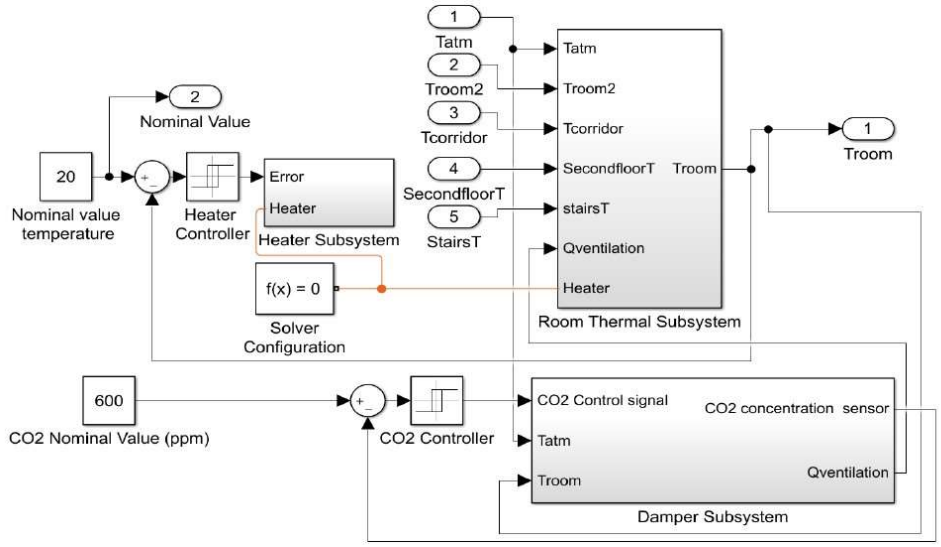

Figure 27. Interior view of a room component of the example scenario of the DCV and heating system, including the heater, thermal, and damper subsystems [25]

The damper subsystem (DCV subsystem) includes airflow, damper subsystem, and CO_2 concentration sensor.

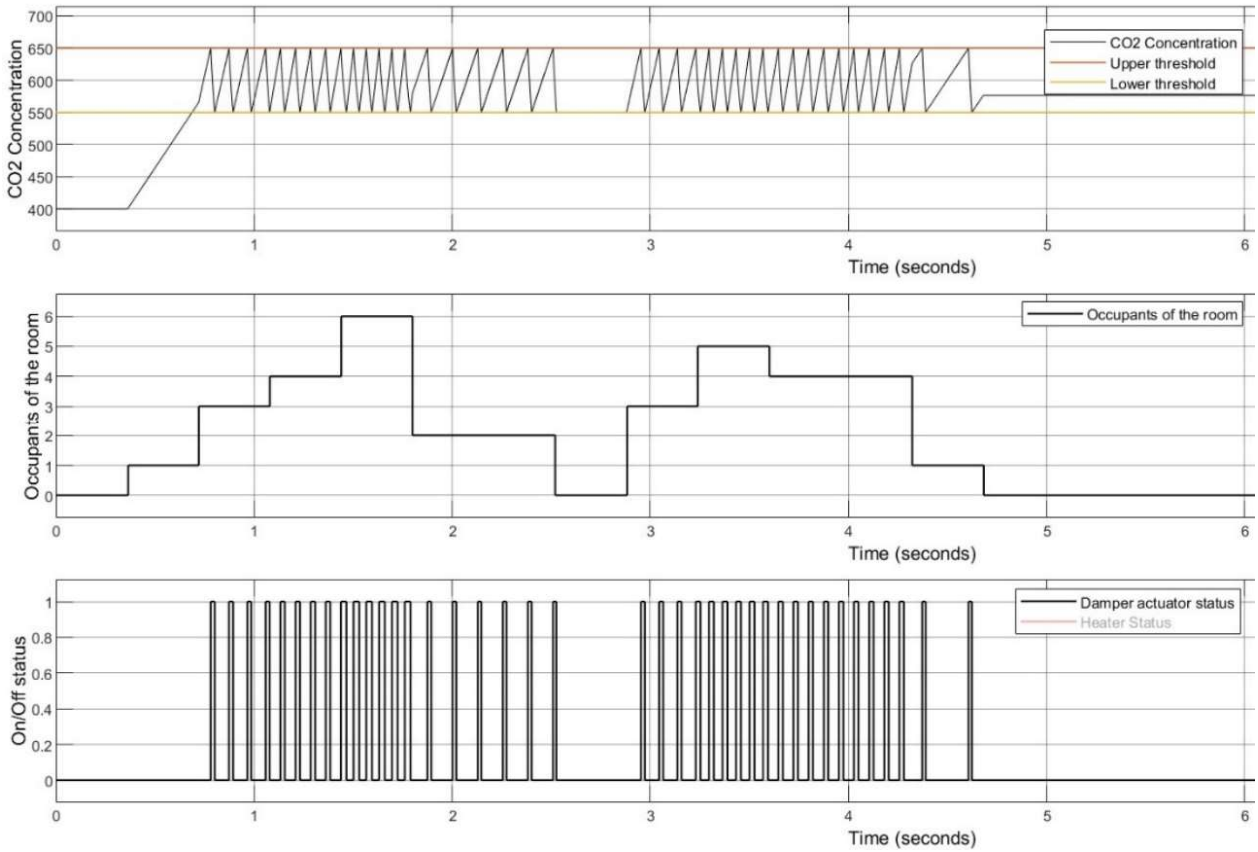

Figure 28. Healthy measured outputs in DCV subsystem including CO_2 concentration, occupancy pattern, and damper status [25]

The computed CO_2 concentration is based on the number of occupants and the outdoor CO_2 value (400 ppm). The values are provided to an embedded CO_2 controller to compare them with the desired indoor

CO_2 value (600 ppm with upper and lower thresholds of 50 ppm). Set points are provided to the DCV subsystem again as CO_2 control signals to control the damper status [25]. In addition, DCV demands knowledge about the ventilation rate and the indoor room temperature for measuring the CO_2 concentration. Figure 28 demonstrates the healthy signals of the DCV subsystem and their patterns. A higher number of occupants results in more CO_2 concentration and more frequent activation of the damper actuator. The occupancy pattern in Figure 28 represents the changes in the number of occupants in each time slot during the execution of the simulation. The figure also shows the effects on the indoor CO_2 concentration rate. A human CO_2 generation rate of 0.0052 per person has been considered [25].

6.1.2 Implementation of the Fault Injection in MATLAB/Simulink

In this section, the fault injection framework implementation is detailed for realistic single and multiple fault injection. This thesis implements the system model with a component-based strategy to integrate the system components with support for scalability of the building structure. A large-scale multi-floor and multi-room building structure is equipped with a multiple-fault injection system to validate the multiple fault injection framework extendibility and universality. The fault injection components are generic and compatible with different scenarios of any target system including the DCV and heating system model. Each fault injection component can be connected to the system components with low effort, e.g., electrical devices such as sensors and actuators. In this thesis, fault injection is a simulation-based technique, and MATLAB/Simulink and MATLAB programming are used for the composition of the simulation code modification and the simulator command techniques. Generic fault injection components are connected to other system components through input and output ports. In addition, the fault injection components include Stateflow diagram subsystems. An automatic script initializes the fault characteristics and activates the destination fault injector blocks. In this thesis, Stateflow is used to implement finite-state machines for the alterations between faulty and healthy states of the system. Stateflow is suitability because of its support for modelling hierarchical systems and parallelism. The states are mutable and can be changed based on the new input values of the system. The Stateflow diagram reads the new input values, performs the operations (e.g., using MATLAB functions), and emits a flow of new system values.

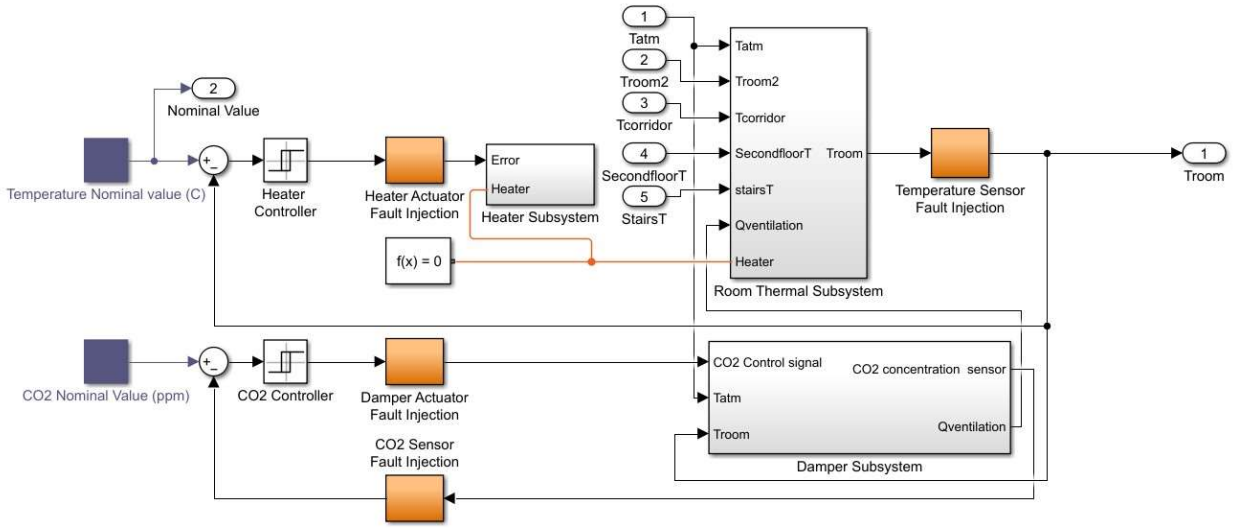

Figure 29. Interior view of a room component extended with the single/multiple fault injector components (saboteurs) [11, 16].

Figure 29 shows the exterior view of the single/multiple fault injection blocks integrated with the room components indicated in orange color. The healthy values from the system components (e.g., heater or damper subsystem) enter the fault injector blocks. If the system component is faulty, the output of the fault injector block becomes faulty. Various faults with different properties can be activated based on the system specifications. This layout is the same in both single and multiple fault injections. The elements of the generic fault injector component for the single and multiple fault injections are described in the next section.

6.1.3 Fault Injector Block (Saboteurs)

A complete overview of the DCV and heating system components and their interconnections for one room is shown in Figure 29. For each system component, one fault injector block is used to manipulate the system's behavior by changing the system measurement values under the specified fault situations. Each fault injector block is adaptable to each system component with low effort by specifying the component input values and addresses. The component address consists of the floor number, room number, and component number and it is determined by indices in Simulink blocks. The fault location can be defined once in a single fault injection. However, localization is a challenging issue for multiple fault injection, especially in large-scale structures with numerous components with high fault occurrence rates. Once the number of faults increases, the fault location should be addressed with the component and room indices in the Stateflow diagram for each fault case. The inner structure of the fault injector block consists of two levels. The first level activates the component input port, and the second level activates the addresses and the Stateflow diagram.

6.1.3.1 First Level of Inner Structure of Fault Injection Block

The fault injector block's first-level inner structure represents the input values of healthy components on the left side and the output values of faulty components on the right side. This structure differs for each system component based on the component numbers. This distinction can be seen in Figure 30 and Figure 31. For example, the component number for the CO_2 concentration sensor is 1. Therefore, the port number “1” in the “System_Monitoring” subsystem is activated when the fault injector component is connected to the CO_2 concentration sensor component. Figure *30* shows the first-level interior view of the fault injection block for the CO_2 concentration sensor with input and output blocks. A Gaussian noise subsystem is applied for each sensor component to add uncertainties to each healthy signal measurement for a more realistic system model implementation. Figure 31 shows the first-level interior view of the temperature sensor's fault injection block, including the Gaussian noise subsystem.

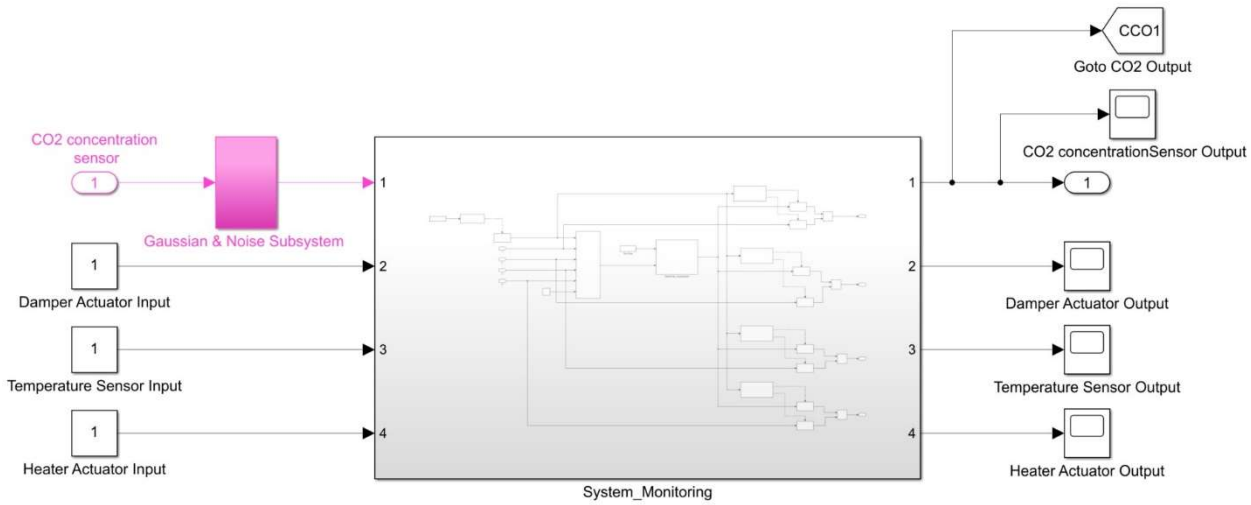

Figure 30. The first-level interior view of the fault injector component for the CO_2 concentration sensor.

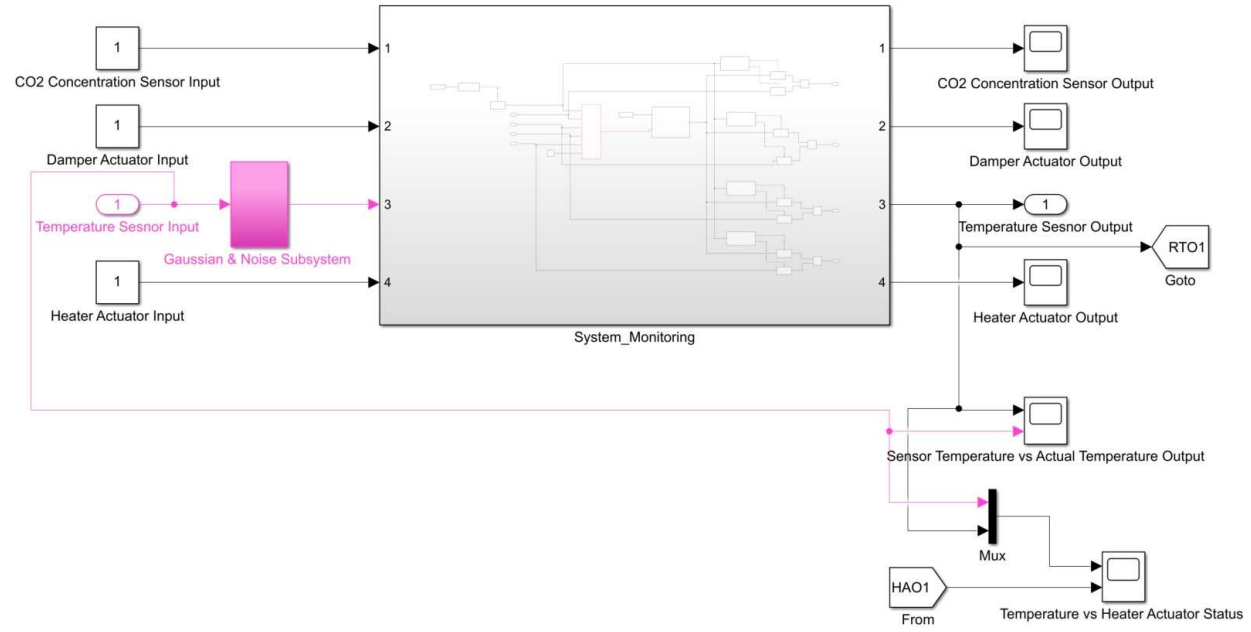

Figure 31.The first-level interior view of the fault injector component for the temperature sensor.

The "Goto" block is a data collector block in MATLAB/Simulink which transfers a signal measurement to one or more "From" blocks in other parts of the simulation environment. For example, the "CCO1" block in Figure 30 transfers room number one's faulty CO_2 concentration value to the output value subsystem. The most popular blocks to show the output values are "Scope" Simulink blocks for the generated signals and "Display" for numeric values. Input values in these blocks can be merged or combined. For example, with a "Mux" block, different input signals with the same data type are combined with a mapping in a single output [95]. In Figure 31, the mux block is used to integrate the system component outputs to show how their changes affect each other. Figure 32 illustrates the Gaussian white noise subsystem. The Gaussian distribution parameters have been calculated based on the historical system

measurements in the healthy mode. The measured uncertainties are added to the input signals and produce noisy output values.

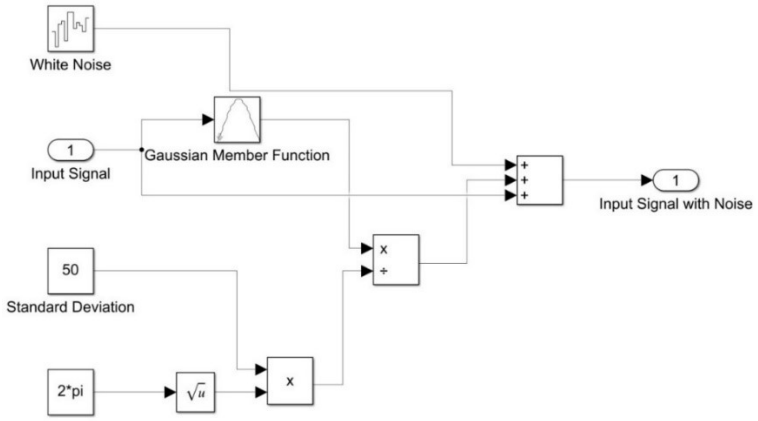

Figure 32. Gaussian white noise subsystem.

6.1.3.2 Second Level of Inner Structure of Fault Injection Block

The second-level inner structure of the fault injector block activates the fault location using its address: room number, component number and fault value as shown in Figure 33. Figure 33 consists of four main parts: (1) fault location activation, (2) distribution of the component input value using a multi-port switch, (3) Stateflow diagram subsystem, and (4) merging of faulty and healthy signals.

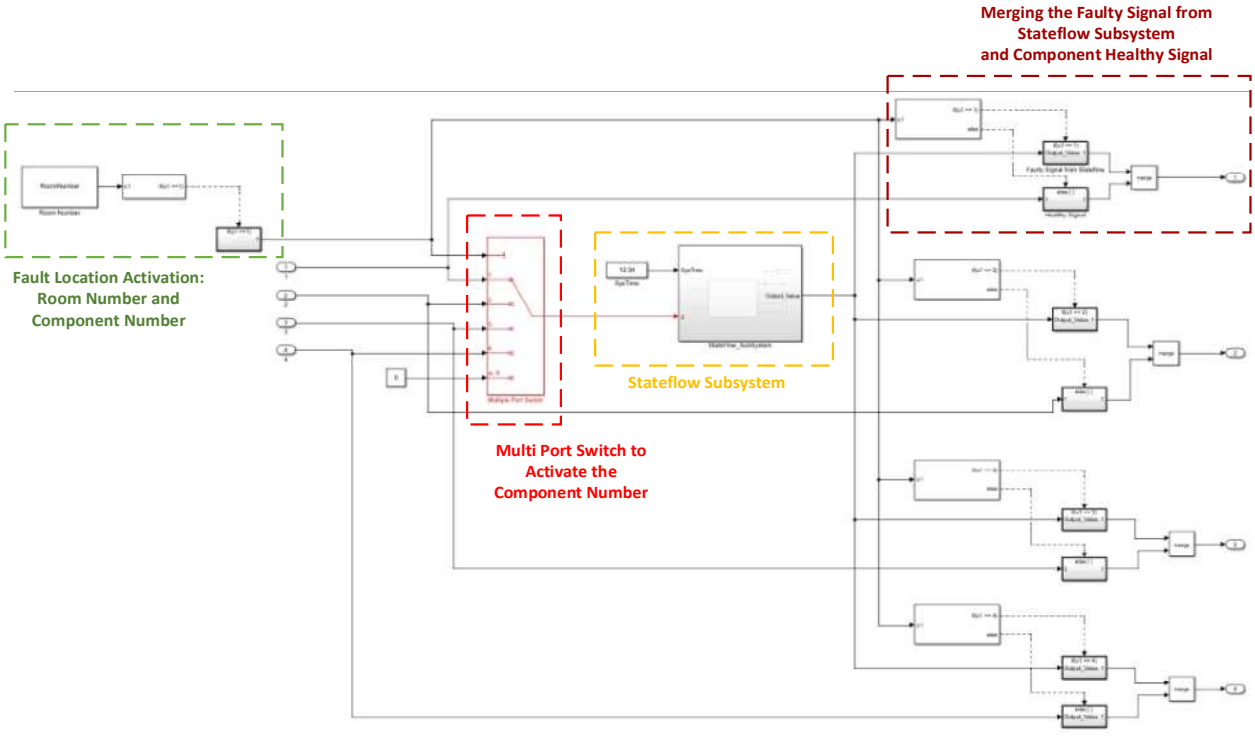

Figure 33. Second-level interior view of the fault injector component.

6.1.3.2.1 Fault Location Activation

The fault address determines the fault location. The address of each fault case is indicated with the floor number, room number, and component number. This address can be specified with constants when only one fault occurs (single-fault injection). These constant values can be implemented using constant Simulink blocks as shown in Figure 34.

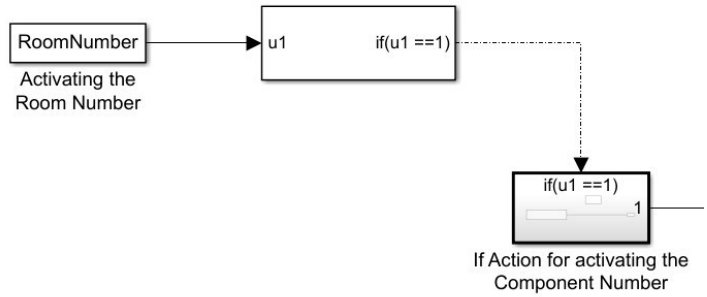

Figure 34. Fault location activation in single-fault injection using constants for room number and component number.

In the case of multiple fault injection, a constant is not applicable due to the dynamic and complex structure. Therefore, the fault location can be defined with matrices. Table 20 shows the combination of the faulty rooms and components. The Activated_Room_Component_Combination_Matrix ={$Room_1$, $Room_2$, …, $Room_m$}, represents the faulty rooms in a structure with a number of *m* rooms where $Room_i$ ={ $Component_1$, $Component_1$, …, $Component_n$}. At the initialization of the matrix, all elements are zero representing the healthy mode of all system components. $Room_iComponent_j$ in Table 20 contains binary values {0,1}. The initial values of Room-Component combinations are zero and once a fault occurs it becomes one. Each fault case's location (address) can be described with an index as $Index\_Fault_i(Room_i, Component_j)$.

Table 20. Combination of faulty rooms and components as Activated_Room_Component_Combination_Matrix.

Components / **Rooms**	**$Component_1$**	**$Component_2$**	…	**$Component_j$**	…	**$Component_n$**
$Room_1$	$Room_1Component_1$	$Room_1Component_2$	…	$Room_1Component_j$	…	$Room_1Component_n$
$Room_2$	$Room_2Component_1$	$Room_2Component_2$	…	$Room_2Component_j$	…	$Room_2Component_n$
…	…	…	…	…	…	…
$Room_i$	$Room_iComponent_1$	$Room_iComponent_2$	…	$Room_iComponent_j$	…	$Room_iComponent_n$
…	…	…	…	…	…	…
$Room_m$	$Room_mComponent_1$	$Room_mComponent_2$	…	$Room_mComponent_j$	…	$Room_mComponent_n$

The instantiated DCV and heating system model embodies six rooms and four types of components. Hence, the Activated_Room_Component_Combination_Matrix is created with six rows and four columns. Activated_Room_Component_Combination_Matrix can be initialized randomly or manually in the FI algorithm. Other fault properties (attributes) must be activated based on the faulty elements. Table 21 serves as an example for two different fault locations: $Index\_Fault_1$ ($Room_1$, $Component_3$) and $Index\_Fault_2$ ($Room_2$, $Component_3$).

Table 21. Activated_Room_Component_Combination_Matrix for the example DCV and heating system.

Components / Rooms	**Component$_1$** CO_2 Sensor	**Component$_2$** Damper Actuator	**Component$_3$** Temperature Sensor	**Component$_4$** Heater Actuator
Room$_1$	0	0	1	0
Room$_2$	0	0	1	0
Room$_3$	0	0	0	0
Room$_4$	0	0	0	0
Room$_5$	0	0	0	0
Room$_6$	0	0	0	0

Activated_Room_Component_Combination_Matrix(x,:) returns a vector including all component values in room number x. Therefore, we can use this capability to activate the fault location with a matrix of the combination of the faulty rooms and components. To activate the faults in room$_1$(zone$_1$), the Activated_Room_Component_Combination_Matrix (1,:) returns the Room$_1$ vector; where Room$_1$={ 0, 0,1,0} as shown in Figure 35.

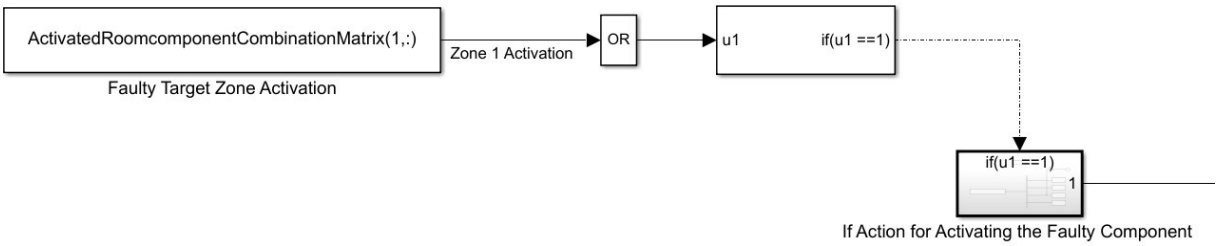

Figure 35. Fault location activation in multiple fault injection for room and component numbers.

The "Demux" Simulink block extracts the inputs to separate elements [95]. For example, the elements of Room$_1$={ 0, 0,1,0} can be extracted to different values and mapped to each component type as shown in Figure 36**,** which describes the If-Action block for activating a faulty component. The example shows that the faulty component is the temperature sensor. Figure 36 shows Room$_1$={ 0, 0,1,0} as an example and how the temperature sensor component is activated by a gain value of 3. This value differs in other fault injection blocks for other components based on the component number.

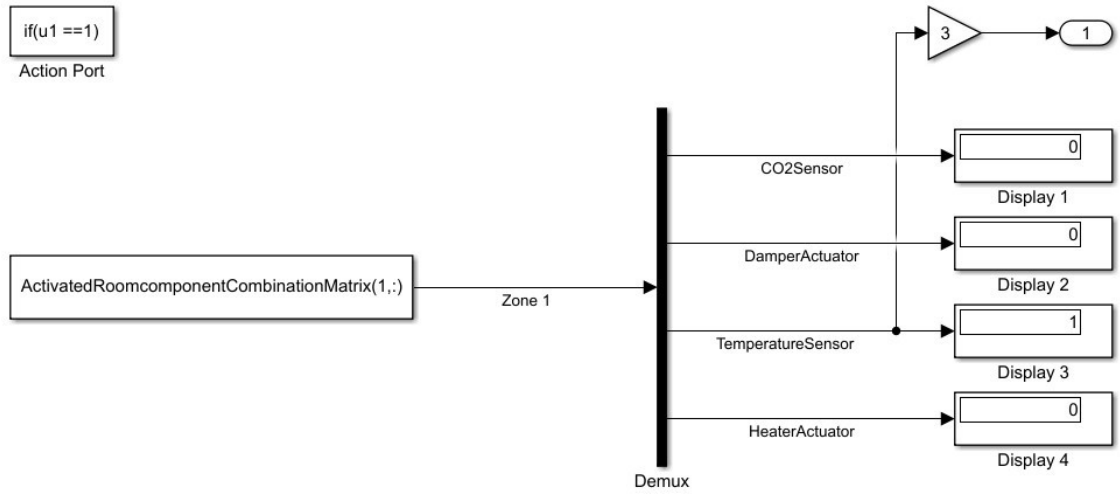

Figure 36. Component activation in multiple-fault injection using the combination matrix.

6.1.3.2.2 Distribution of the Component Input Value Using a Multi-Port Switch

The multiport switch is a Simulink block that passes an input value to the output port based on the control signal. The first input port of the multiport switch is a control signal that determines which input port must be activated. The last input data port is a default port that should certainly be valued to avoid implementation errors when the control signal is unavailable. Figure 37 demonstrates that whenever the faulty component is activated, the component number switch provides the value on the corresponding input port in the multiport switch block. Each input port conveys the generated healthy signal that must be redirected to the output port by the multiport switch. The output of the multiple switch block is an input value for the Stateflow subsystem.

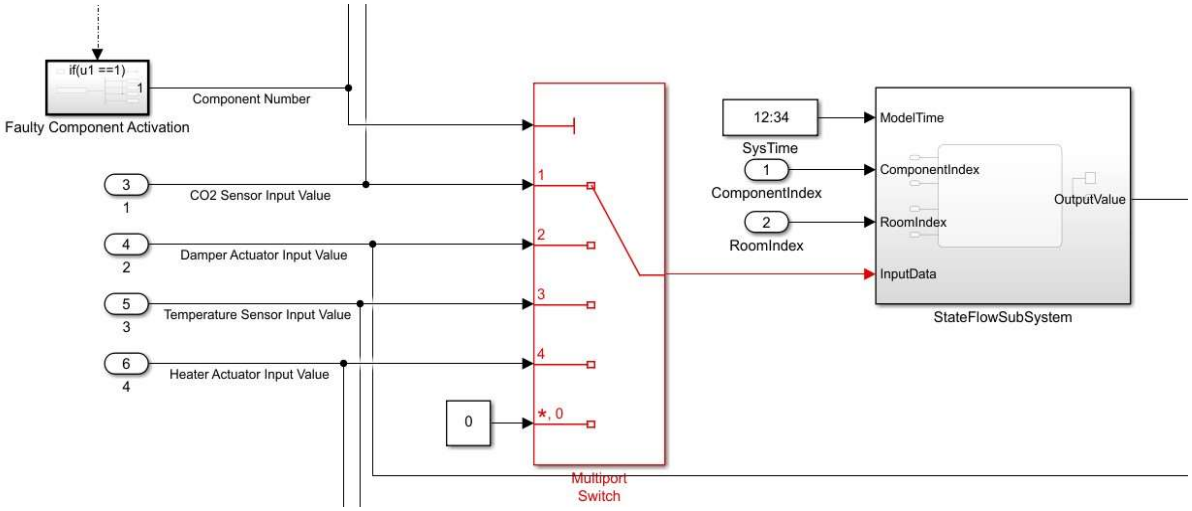

Figure 37. Multiport switch block to distribute the input values based on the component numbers.

6.1.3.2.3 Stateflow Diagram Subsystem

The operational modes in reactive systems can be represented by states describing the sequential operations and activities. In this thesis, finite-state machines are used to implement the fault injection process when the states of the system change in the presence of faults. In computation theory, there are two types of machines: Mealy and Moore state machines. A "Computing state" updates the local parameters, makes decisions based on the conditional operations, and performs the transition to change the active state to the next state. Mealy and Moore machines can be applied as embedded charts in Simulink. The output of the Moore charts is only the function of the current state, whereas the Mealy charts depend on the inputs and active states. It means that the Mealy state machine at each clock edge (time step) wakes up and computes the output and a new system configurations [95, 246, 247]. The Mealy state machines react faster and they are more suitable for machines with fewer states [246]. Therefore, the Mealy state machines are used in this thesis, and they are defined with a 5-tuple using Equation 21.

$$State\_Machine(S, S_0, Inputs, Outputs, T). \quad \text{Equation 21}$$

Where S describes a finite set of states, S_0 describes the initial state of the state machine, *Inputs* is the finite set of input values, *Outputs* is the finite set of output values, and $T: S \times Input\_Set \rightarrow S$ is a finite set of transitions that computes and maps a pair of states and input values to the next state. The three main elements of the Stateflow diagram are introduced as follows.

- **State in Stateflow diagram:** A state encompasses five main actions: entry, during, exit, condition, and transition actions. "Entry" actions are operated at the entering time. Then the action operations continue in "During" when we remain in the active state. Finally, "Exit" actions should be operated at the exiting time of the state. An initial state starts the Stateflow chart activities characterized by an initial transition.

- **Transition in Stateflow Diagram:** Each state communicates with other states through transitions. Each transition will be activated upon the satisfaction of a condition or an event. Accordingly, an action occurs based on the state and transition conditions. For example, after (Variable_Name, sec) is a default and standard function of MATLAB that activates the next state after a determined time with a second time step. Variable_Name is a value that a transition should wait for in order to perform a transition from the current active state to the following respective state. In our FI system, the "after (Fault_Duration_Time, sec)" function is used for implementing the fault duration property. It means we stay in the faulty state (source state) for a specified fault duration time. Afterward, the state changes to the healthy state (destination state). The order of the states is based on the transition status and the conditional values.

- **MATLAB Function in Stateflow Diagram:** MATLAB functions in Stateflow provide programming methods to link MATLAB/Programming and Stateflow diagrams. Each Stateflow diagram consists of three principal means: states, transitions, and functions.

In this thesis, the sequence of the actions and states of the introduced fault injections are implemented by Stateflow diagrams using the two states: healthy and faulty. The inner structure of the Stateflow diagram consists of the room number (i.e., room index), component number (i.e., component index), system time as input ports and fault values as an output port, as shown in Figure 38. For the single fault injection, indices are not required. However, in multiple fault injections, the accurate fault addresses must be specified as shown in Figure 39. The digital clock Simulink block produces the system time as an input variable that returns the system time in one day because the simulation is set to be run for one day.

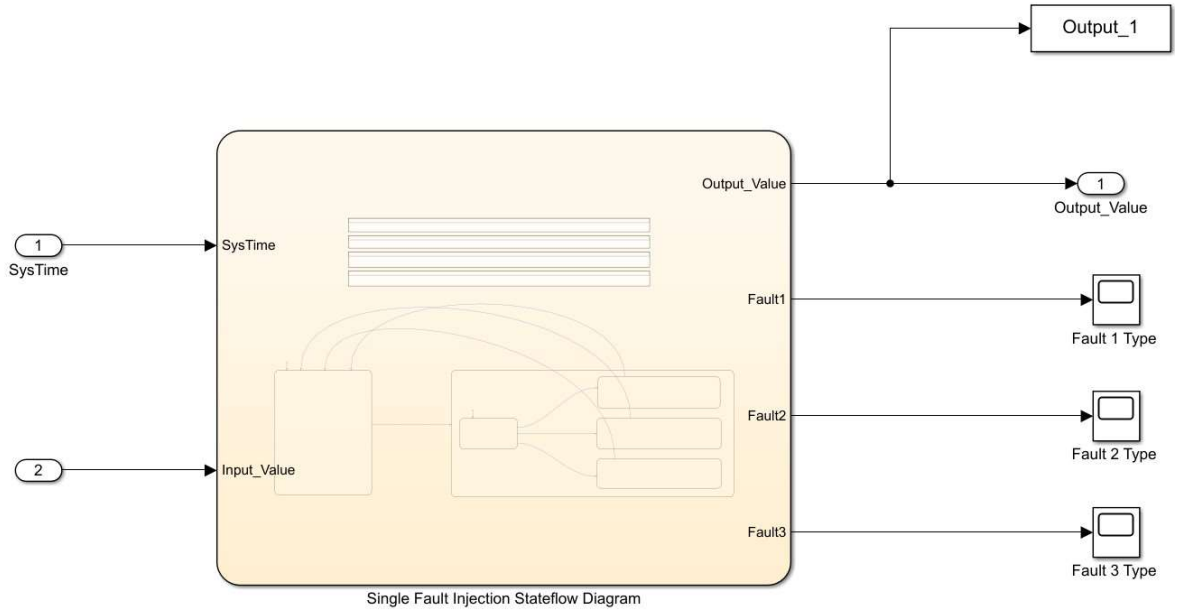

Figure 38. First level of the interior view of the Stateflow subsystem in single fault injection.

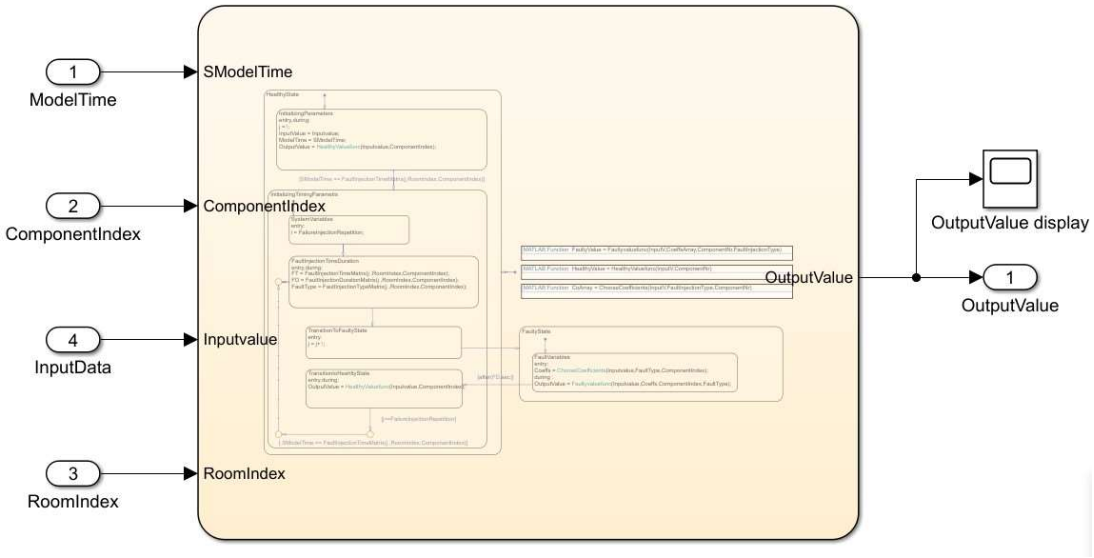

Figure 39. First level of the interior view of the Stateflow subsystem in multiple fault injection, including the room and component indices.

The second-level interior view of the Stateflow subsystem is implemented using two different methods based on the system requirements for the single and multiple fault injections. The second-level view of the Stateflow subsystem for the single fault injection, including the healthy state, faulty state, and MATLAB functions. A healthy state produces healthy values by getting the healthy value from the input port. In addition, the healthy state initializes the required parameters, such as the fault injection times. The Stateflow diagram transmits the input value to the output value by changing the parameter types.

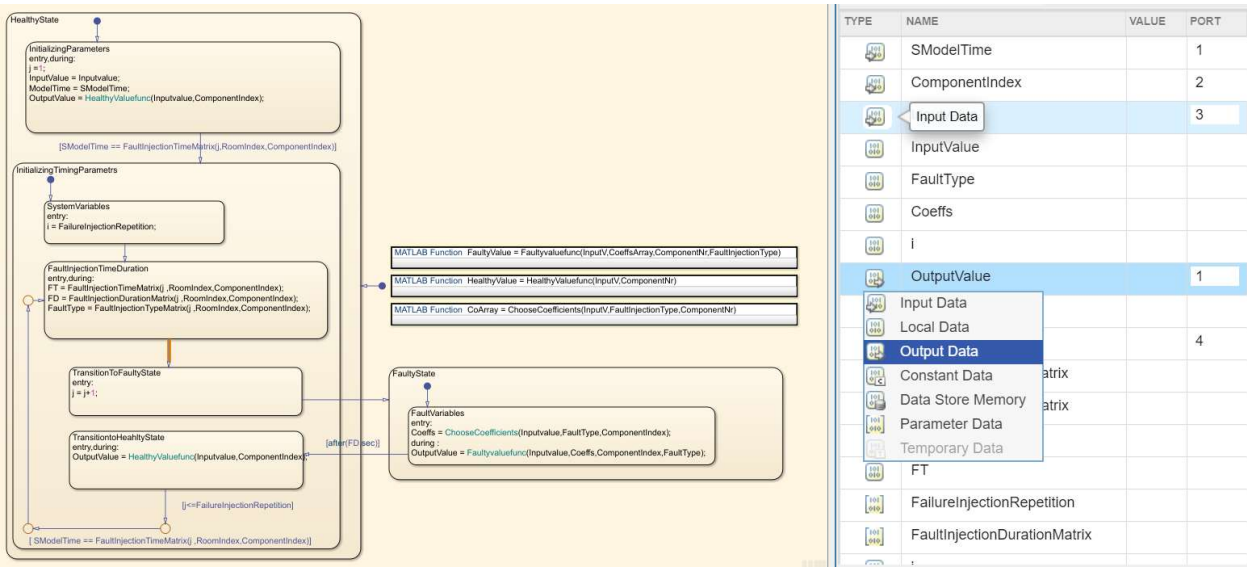

Figure 40. Symbol panel for defining the Stateflow diagram parameters, input, and output ports.

In this thesis, the automated fault injection algorithm should initialize the main fault attributes (e.g., through 1-dimensional vectors for the single-fault injection and 2 or 3-dimensional matrixes for the multiple-fault injection. A Stateflow diagram is applied to control the system's reactions and their responses under injected faults. Then the simulated component-based system model is executed to imitate the example scenario under injected fault conditions. The fault injection components are integrated with other system components, such as the heater, damper, CO_2 sensor, and temperature sensors, as explained in the high-level description in chapter 4. The designed state machine is mapped to the Stateflow diagram in the Simulink environment. In fault management, the Stateflow diagram controls system reactions. Therefore, we defined the FI framework as a finite-state machine with healthy and faulty states, as depicted in Figure 41. In addition, Figure 41 shows the second level of an interior view of the Stateflow subsystem for the multiple fault injection, the system state conditions, and interconnected transitions between healthy and faulty states using the assigned fault attributes in detail. In the interior view of the Stateflow diagram, each FI block is a collection of the functions, state diagrams, and the symbol panel to define the required interrelated parameters. Each parameter in the symbol panel attains different types of data: input data, local data, output data, constant data, data store memory, parameter data, and temporary data. We must assign the proper data type to each parameter required for the Stateflow diagram activities. For example, local fault types and faulty values in each transition of the Stateflow diagram are initialized by calling the related function. Once the FI process terminates, the data collector blocks gather all information, including faulty and healthy output signals, and return them to the automated FI algorithm. All system model variables can be saved for each execution. However, only fault attributes and the output data of each simulation are stored in a library in a MATLAB file.

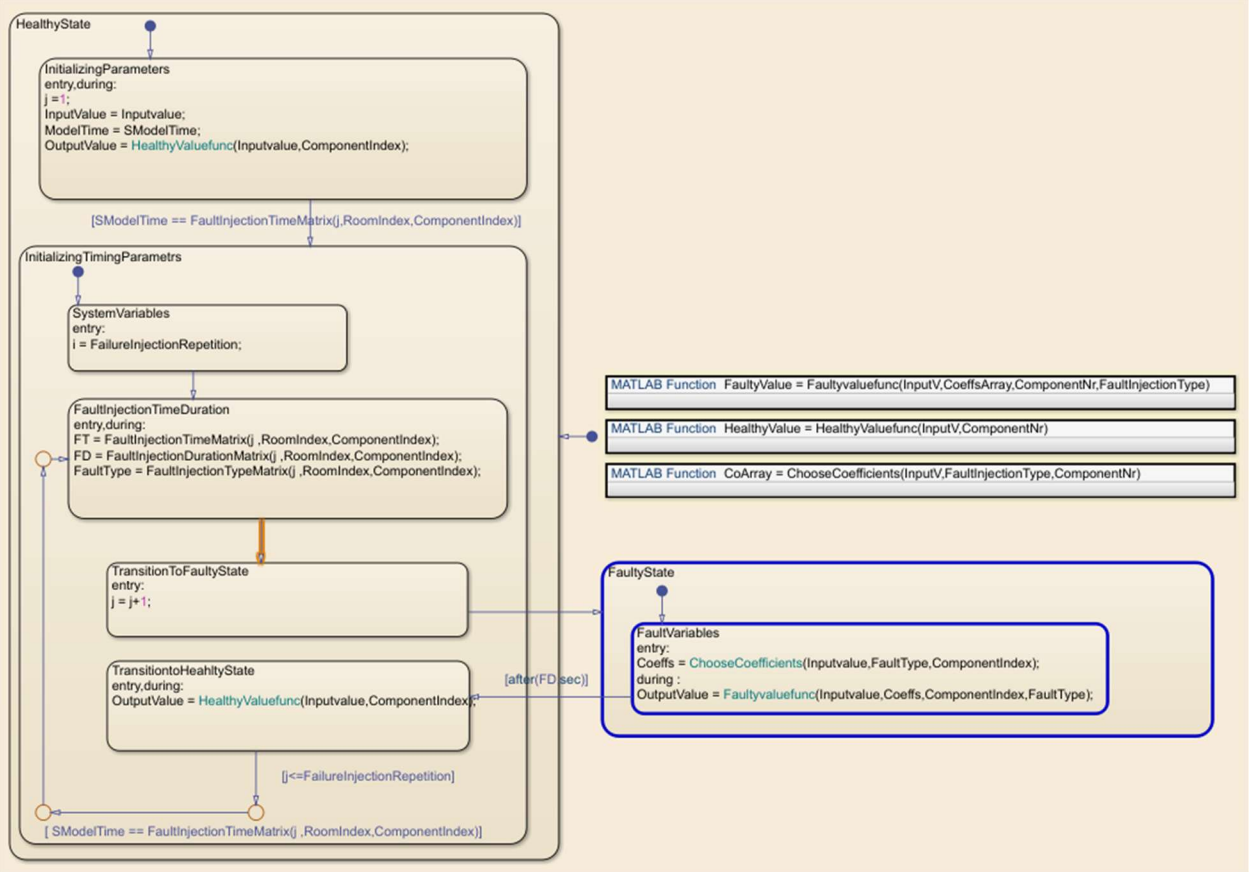

Figure 41. Second level of interior view of the Stateflow subsystem for the multiple fault injection.

6.1.3.2.4 Merging Faulty and Healthy Signals

Finally, the output signal is a merged value of a healthy signal from the input port and a faulty signal from the Stateflow subsystem to ensure an integrated signal. For example, Figure 42 depicts how the faulty temperature sensor signal from the Stateflow subsystem and its healthy signal is merged into a single output signal. The index of the temperature sensor component is three as specified by the If Simulink block. Merging blocks enables us to have a generic fault injection component that only requires addressing of fault targets. Otherwise, the healthy signals are received.

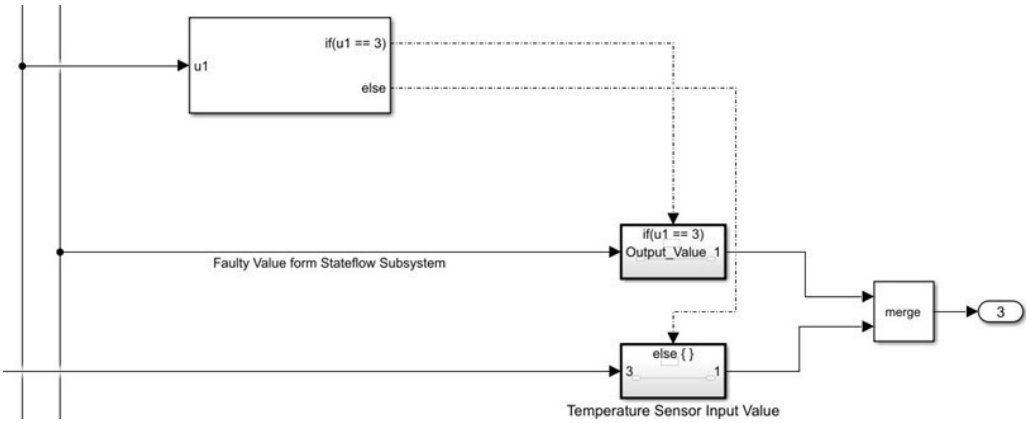

Figure 42. Merging the faulty signal from the Stateflow diagram and the healthy signal from the input port.

6.1.3.3 Data Collector Blocks and Monitoring Subsystem

The simulation output data should be collected from all subsystems from all system components, e.g., CO_2 sensor, damper actuator, temperature sensor and heater actuator. Measured data from "Goto" blocks is sent to "From" blocks and merged to one signal. The merged signals from each room are logged and saved as a single "Data_ML" variable by the "To workspace" Simulink block. The "Data_ML" variable is a collection of different measured system variables including constant, discrete, and continuous system variables. Constant system variables refer to constant system attributes which are defined during system model implementation such as outdoor daily temperature, which is a constant sinusoidal signal [25] and constant variables such as nominal indoor temperature, stair temperature, upper temperature threshold and lower temperature threshold [25]. Table 22 is part of the Data-ML time-series showing all collected and logged data of one room. The logging can be applied for all available rooms and is applicable for signal based FDD techniques.

Table 22. Output (Data_ML) time-series saved to workspace environment including the system attributes in separate columns.

Constant system Variables						**Discrete System Variables**		**Continuous System Variables**	
Nr.	**Outdoor Daily Temperature**	**Nominal Indoor Temperature**	**Stairs Temperature**	**Upper Temperature threshold**	**Lower Temperature threshold**	**Damper Status**	**Heater Status**	**CO_2 Concentration**	**Indoor Temperature**
1	7	20	13.5	22.5	17.5	0	0	400.1	20.2
2	7	20	13.5	22.5	17.5	0	0	400.7	20.4
3	7	20	13.5	22.5	17.5	0	0	399.7	20.1
4	7	20	13.5	22.5	17.5	0	0	400.4	20.3
5	7	20	13.5	22.5	17.5	0	0	400.1	20.2
6	7	20	13.5	22.5	17.5	0	0	399.9	20.2
7	7	20	13.5	22.5	17.5	0	0	399.7	20.1
8	7	20	13.5	22.5	17.5	0	0	400.0	20.21
9	7	20	13.5	22.5	17.5	0	0	400.2	20.3
10	7	20	13.5	22.5	17.5	0	0	400.0	20.2

6.2 Implementation of Automated Single and Multiple Fault Injection Script

In this framework, an automated FI activates the target saboteurs in the system model, which are inactive during the normal operation. For each FI, a fault set (sequence of failures) is injected, and for each fault set, attributes such as fault persistence (i.e., transient, intermittent, permanent), fault location, fault type, fault duration, and fault interarrival time are considered. Moreover, this framework can be evaluated for deterministic fault models (pre-defined fault scenarios) and random fault attributes for single and multiple (systematic) faults at run time. An automated algorithm was coded to inject the fault attributes randomly according to the scenario-based injection type. When our algorithm runs randomly, all variables and attributes, e.g., the number of faulty components, faulty zones, and persistence, are initiated randomly.

Figure 29 illustrates one zone of an HVAC system, including components and their interconnections, such as the thermal, damper, and heater subsystem, which are connected to fault injector blocks. The fault injection block should be linked to an automated algorithm using MATLAB/Programming. The fault injectors (saboteurs) manipulate each subsystem's output. In a single fault injection algorithm, a single fault was introduced out of a catalog of different fault types at only one target location, e.g., one intermittent fault in one temperature sensor. To implement the single fault injection, fault attributes are assigned with random vector variables, including the fault location and fault types, and faulty values are computed in Stateflow by real-time MATLAB/function invocations. To activate a single fault, each attribute is initialized randomly based on the system requirements. The automated fault injection algorithm can be executed once in a single fault injection or for more iterations based on the number of faults in a predefined scenario. Each scenario comprises several fault-sets combining different fault attributes, including fault type, time, duration, interarrival, persistence, location, and occurrence rate. The fault location is the address of the destination, i.e. the faulty component whose behavior should be changed to realize the system impacts and how they affect other subsystems in the presence of various faults. At the end of each fault injection, all system measurements should be collected from all system components of all zones. Each output data is a dataset (time-series) of different components measured at each time step of 1 second. The automatic fault injection uses an object-oriented programming approach and uses objects to preserve fault attributes. For each fault-set of the scenario, a timeseries must be kept in a fault injection vector. Each element of this vector is an object from a defined class named “FaultClass” with type, time, persistence, location, and data output properties. The initialized fault attributes and output time-series must be saved in an object. The number of objects depends on the number of fault-sets in the scenario.

The single fault injection framework was designed and extended to inject multiple faults in multiple zones modeled with varying faults and more dynamicity regarding the number of faults, their repetitions, and structures. The number of faults can be increased easily by changing the address of faulty components. Each fault injection process includes multiple injections in multiple locations with more failure repetitions in case of intermittent faults despite the single fault injection. Addressing the faulty components in multiple fault injection is a significant challenge that needs a dynamic structure for development. As a result, a multi-dimensional structure is required to access faulty components once a different fault occurs at multiple locations with different addresses. The number of dimensions differs based on the system structure Figure 43. illustrates the multi-dimensional aspect of the FI framework in a large-scale building structure where the dimensions increase in the case of system model extension and development. Figure 43 shows system specifications in different axes for multiple faults injection. Different axes relate to one aspect, such as structure, room, component, and the number of failure repetitions or other fault attributes e.g., the type in

each fault injection procedure. The number of dimensions can be extended based on the system requirements, e.g., the next-level axis can refer to other system specifications such as cluster, grid, or city.

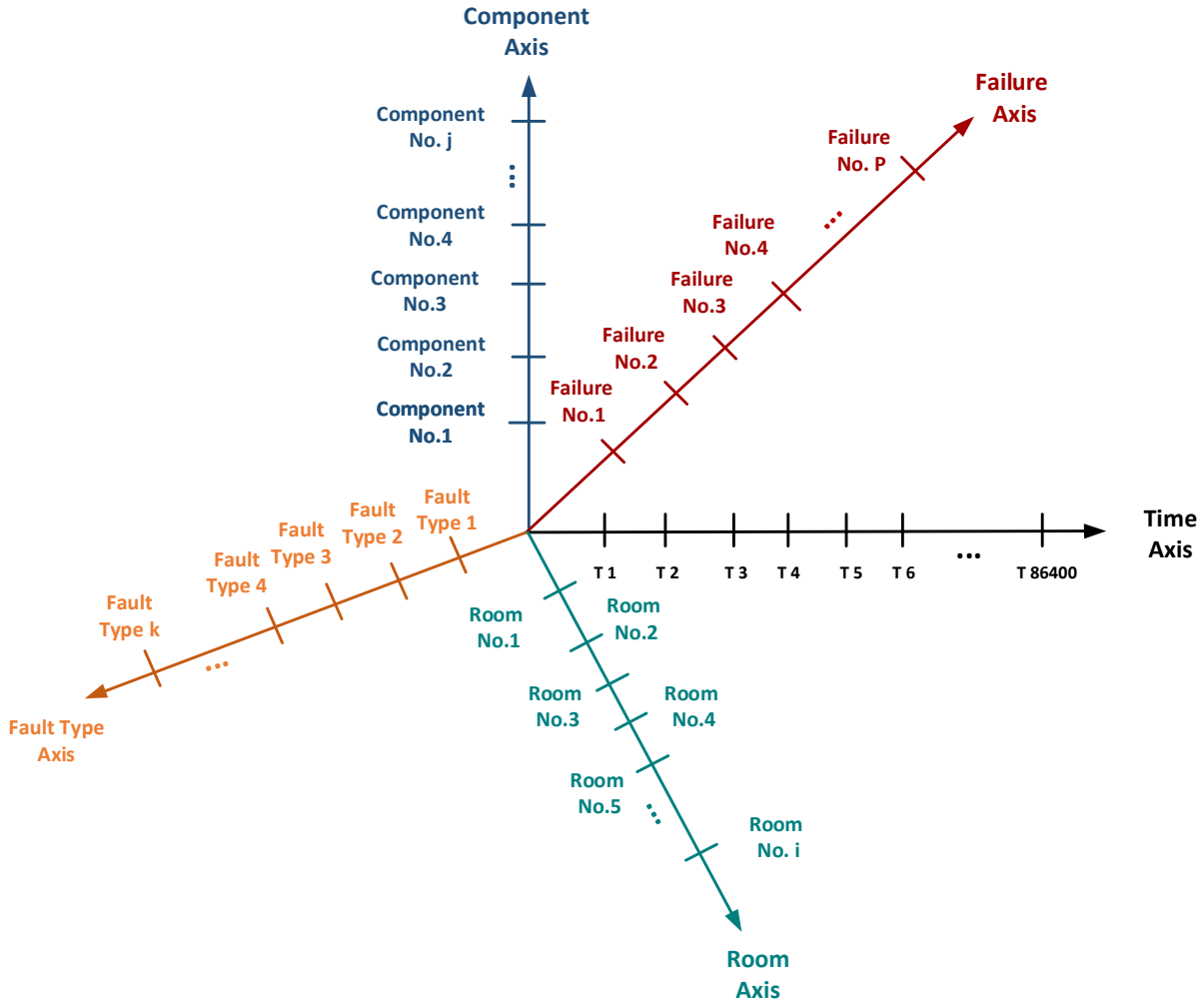

Figure 43. Dynamic multi-dimensional aspects of the FI implementation including the time axis.

This structure can be developed through a matrix for each fault attribute. In multiple fault injections, fault attributes are defined as multi-dimensional matrices. Each system specification is mapped to one dimension of a matrix. For example, Figure 44 depicts a 3- Dimensional (3-D) matrix to implement a fault injection attribute. In multiple fault injections, fault attributes are defined as multi-dimensional matrices such as FI time, fault duration, fault interarrival time, FI persistence, FI type, and fault occurrence probability. Each matrix element introduces the attribute values for each component and zone. By increasing the number of aspects of each attribute, the number of dimensions increases, providing multiple FI capabilities. The multiple fault injection algorithm starts with loading the system variables, such as building assumptions, and thermal conditions, such as daily temperature and CO_2 concentrations. Then the fault injection variables must be defined with multi-dimensional matrixes.

For example, the “Fault_Injection_Time_Matrix” is a 3-D matrix with three axes, including the number of components, number of rooms, and number of failure repetitions for each intermittent fault case. The element values of the “Fault_Injection_Time_Matrix” are assigned based on the “Activated_Room_Component_Combination_Matrix” elements specifying associated fault locations

showing all faulty room-component pairs. This matrix is a 2-dimensional (2-D) matrix introducing the combinations of faulty rooms and faulty components. Figure 44 shows an example of the 3-D matrix for the "Fault_Injection_Time_Matrix" attribute.

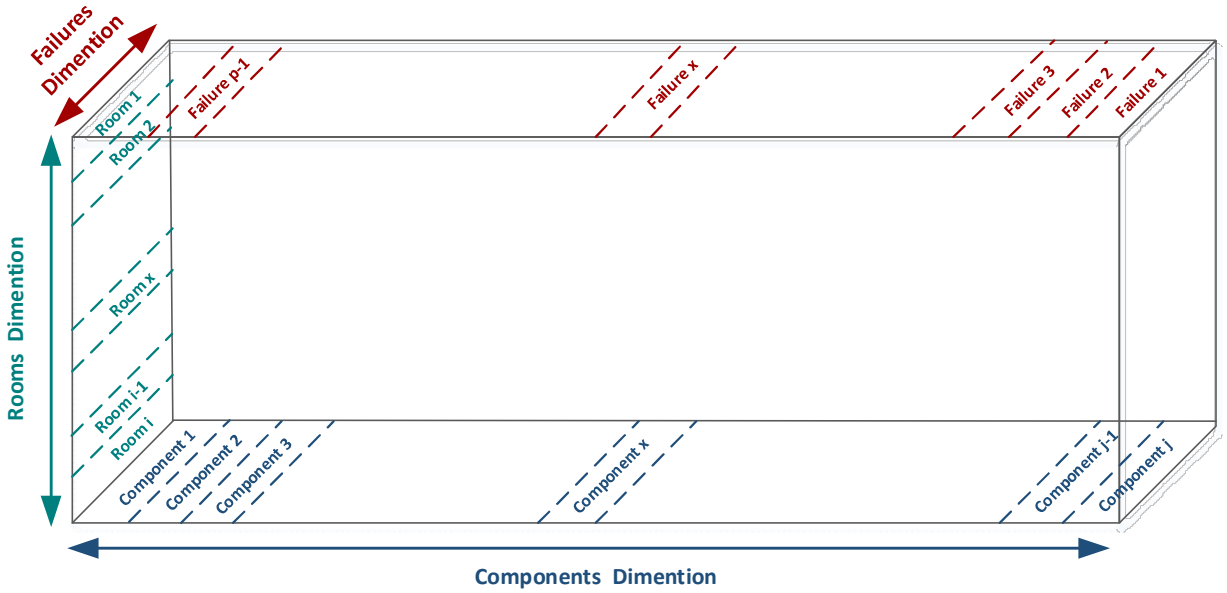

Figure 44. 3-Dimensional (3D) matrix for implementing the fault injection attributes, e.g., fault injection time matrix.

Seven multi-dimensional attributes are explained in Table 23 and Table 24 and their required parameters are defined. The behavior of the fault injection components is changed based on the values assigned to these matrixes. The first matrix is a 2-D matrix because it only shows which room-component combination should be activated. Then other matrixes must be valued randomly for a fault location.

Table 23. Multiple fault injection attributes for one building.

Multi-dimensional matrixes to define fault attributes		
No.	**Matrix Name**	**Explanations**
1.	Activated_Room_Component_Combination_Matrix	A 2-D matrix to display faulty locations, including the room and component number.
2.	Fault_Injection_Persistence_Matrix	A 2-D matrix to display the persistence type for each faulty location.
3.	Fault_Injection_Time_Matrix	A 3-D matrix displays fault injection times distinguished for each intermittent repetition.
4.	Fault_Injection_Duration_Matrix	A 3-D matrix to display fault duration times for each intermittent fault.
5.	Fault_Injection_Interarrival_Matrix	A 3-D matrix to display fault interarrival times for each intermittent fault.
6.	Fault_Injection_Type_Matrix	A 3-D matrix to display fault types for each fault injection and each repetition of an intermittent fault.
7.	Fault_Occurrence_Probability_Matrix	A 2-D matrix to display fault occurrence rates for each fault injection and each repetition of an intermittent fault.

Table 24. Multiple fault injection attributes definition.

Multi-dimensional matrixes to define fault attributes in multiple fault injection
1. **Activated_Room_Component_Combination_Matrix =** randi([0,1],RoomNumbers,ComponentNumbers);
2. **Fault_Injection_Time_Matrix** (1:FailureInjectionRepetition,1:RoomNumbers,1:ComponentNumbers) = 0;
3. **Fault_Injection_Persistence_Matrix** (1:RoomNumbers,1:ComponentNumbers) = 0.
4. **Fault_Injection_Duration_Matrix** (1:FailureInjectionRepetition,1:RoomNumbers,1:ComponentNumbers) = 0;
5. **Fault_Injection_Interarrival_Matrix** (1:FailureInjectionRepetition,1:RoomNumbers,1:ComponentNumbers) = 0;
6. **Fault_Injection_Type_Matrix** (1:FailureInjectionRepetition,1:RoomNumbers,1:ComponentNumbers) = 0;
7. **Fault_Occurrence_Probability_Matrix** (1:RoomNumbers,1:ComponentNumbers) = 0;

When the system structure increases to encompass more buildings, the fault injection algorithm and components can be compatible and matched with these kinds of alterations by increasing the dimensions to access the fault component location as shown in Equation 22.

$$\text{Matrix\_Name}(1{:}\,\text{No. Buildings},\ 1{:}\ \text{No. Rooms},\ 1{:}\ \text{No. Components}) = (1{:}\,b,\ 1{:}\,i,\ 1{:}\,j). \quad \text{Equation 22}$$

Each fault can be triggered at different locations and at the same component at different points in time when an intermittent fault occurs. Figure 45 describes a system-level timeline which is a cumulative form of all fault injection samples in different zones and components with varying fault types. In Figure 45, three fault sets (i.e., events) are triggered in different components. For example, the first injected fault is an intermittent fault with two repetitions (sub-events), and each repetition has obtained the same stuck-at faults. More repetitions with different fault types may also occur in one intermittent fault. In addition, the heater actuator in the third room has a permanent stuck-at fault.

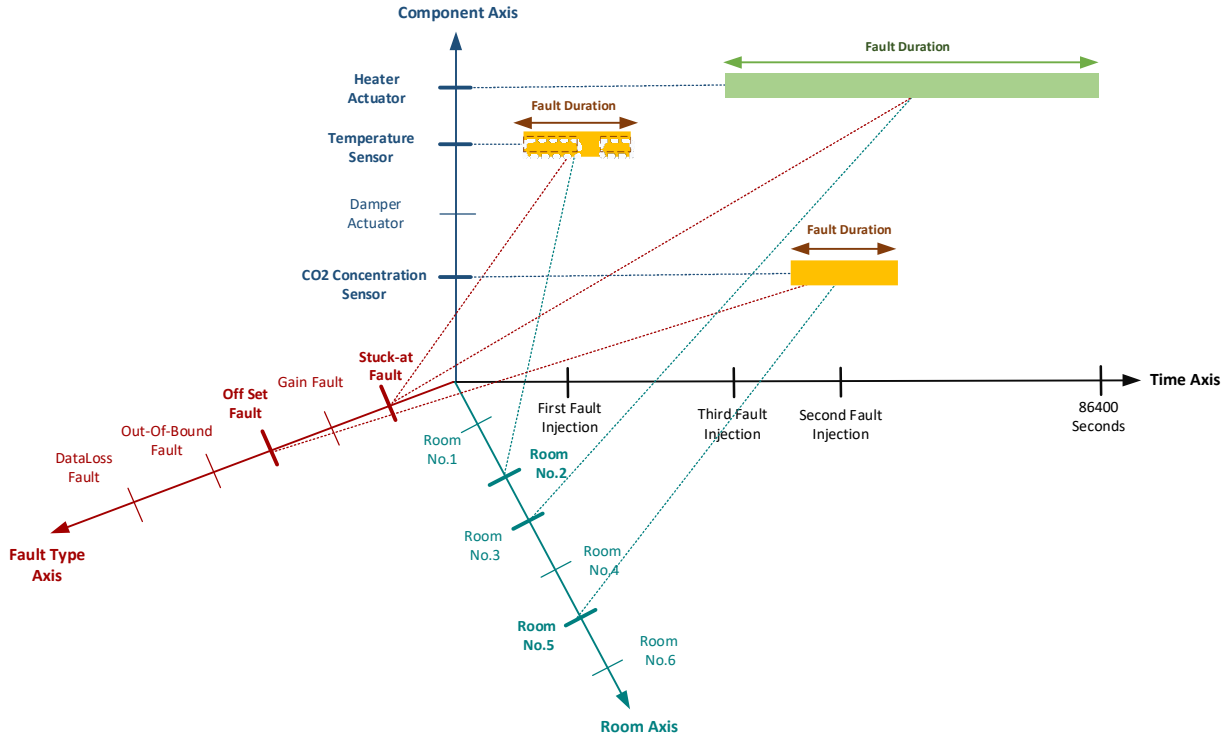

Figure 45. Example timeline for multiple fault injection framework indicating four components, six rooms, and five types of faults.

Each fault was considered as a sample event that occurred based on the probabilities of multiple faults. Each fault occurs with a specific and independent probability. The fault occurrence probability is calculated based on the probabilities of a fault (or event) that happened in one unit (i.e., FCR), saved in the "Fault_Occurrence_Probability_Matrix" and calculated according to Equation 23. Each element of this matrix is associated with one event. Each event consists of several failures (or subevents) for intermittent faults with different fault types. Fault probability occurrence rates differ based on the fault type and unit, as explained in Table 6 in Chapter 4.

$$\text{FaultEventProbability} = P(\text{Failure}_1, \ldots, \text{Failure}_i) = \text{Unit's Probability} \times \left(\prod_{1}^{i=Number\ of\ Failures} \text{Failure}_i \right). \quad \text{Equation 23}$$

6.3 Implementation of the Component-Based System Model

In this thesis, a large-scale component-based system model is developed, which expresses thermodynamics and heat transfers of DCV and heating systems with different rooms. It supports the composition of the system structure and the activation of realistic multiple faults with various specifications. The simulation output also provides proper experimental data for FDD methods. The component-based system model is explained at three primary levels: (1) a high-level system structure description, (2) generic simulation components, and (3) the methods and tools for system configuration of the system model and generic components. Eventually, an example to study the multiple fault injection in a large-scale component-based system model with four floors and four rooms at each floor is generated and shown for two fault events.

6.3.1 High-Level Specification Describing the Structure of the System

The high-level description of a realistic fault model and the system model of the DCV and heating systems and their specifications are explained comprehensively in chapter four. The DCV and heating system comprises four main components such as CO_2 concentration sensor, damper actuator, temperature sensor, and heater actuator. Behravan et al. introduced a composability structure for the DCV and heating systems for integrating the system components through a generation script that determines the system structure based on the user requirements [21, 32]. Behravan introduces electronic components, including three types of room components and one corridor component integrated with a composability structure. The system structure is extended vertically and horizontally based on the number of floors and rooms to have a multi-floor building structure. The number of rooms and floors can be obtained by user inputs via a dashboard in which the number of rooms is determined for each floor. Therefore, the total number of rooms is calculated by multiplying the number of floors and rooms [21].

The model-based construction of system models is supported for large-scale component-based system structures in which system components are integrated through generic and extendable components. The lack of standardization in modeling of DCV systems can result in inconsistencies and difficulties. Large-scale system structures, including the numerous complex embedded DCV and heating systems, are susceptible to various errors, potentially leading to failures. Faults in these systems can result in abnormal behaviors such as temperature fluctuations, discomfort for occupants, excessive ventilation, and overheating that will result in the waste of energy. Therefore, fault injection is a practical solution. The introduced multiple fault injection framework in this thesis can be integrated with different system models to provide dependability analysis in the design phase with a flexible and extendable structure to be merged with index-based structures [11, 16]. Figure 46 describes a large-scale multi-floor building that is extended based on the different room components: room types A, B, and C. Room type A is beside the stairs, Room type B is in the middle, and room type C is beside room type B [21]. To construct the system structure, room component ports should be linked to make the interconnections.

An automated fault injectional script triggers different patterns of multiple faults based on extendable matrixes [16]. An automatic generation script constructs an on-demand composability structure based on the system requirements [21]. These two algorithms should be merged to construct a large component-based building structure. The fault attributes, and fault injection components are compatible with any structural patterns by changing the dimensions and matrixes for fault injection in multi-building-floor-room-component structures.

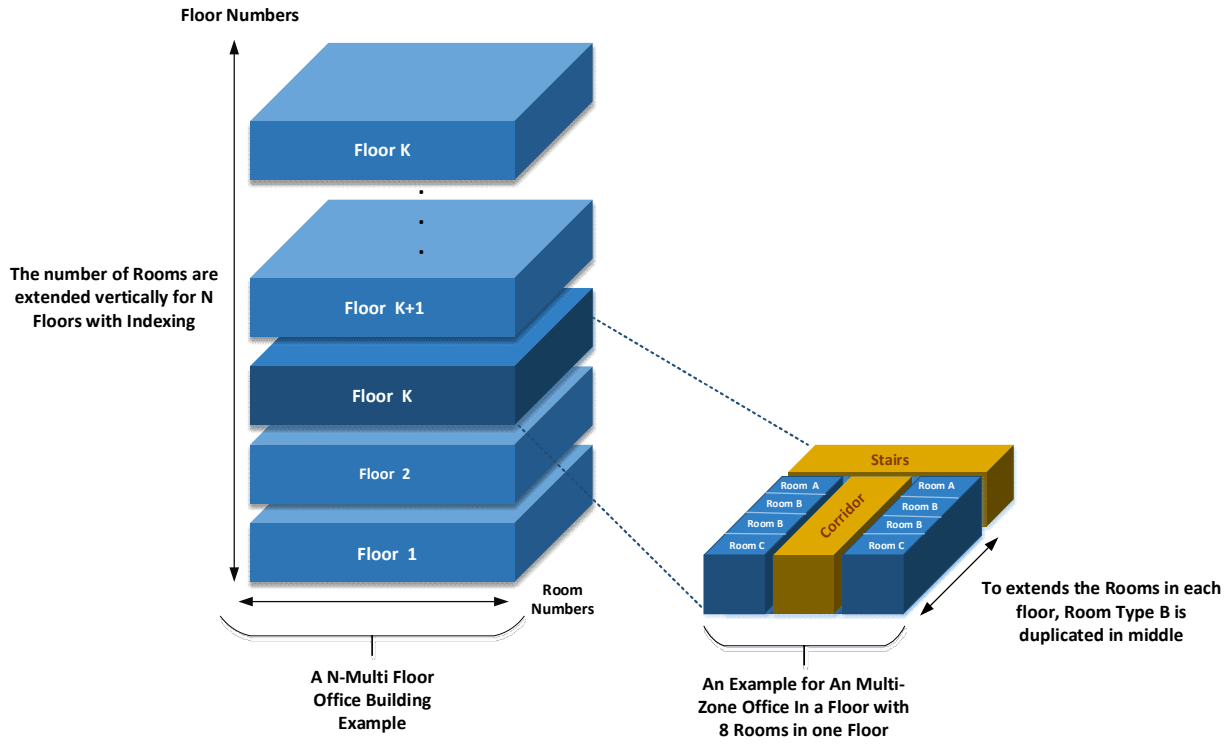

Figure 46. N-multi-floor office building describing an example office building on one floor.

6.3.2 Simulation Environment for HVAC/DCV with Generic Simulation Components

The large-scale system model is generated by merging the generic simulation components for electronics, e.g., sensors, actuators [21], and generic extendable simulation components for multiple fault injection [16]. As a result, a composable and automatic multiple-fault injection component for DCV and heating systems in complex and large buildings is introduced and implemented in this chapter. Multiple faults with associated fault attributes can be injected automatically into a building with an arbitrary structure based on the component-based methods. Therefore, FI components are automatically linked to electronic subsystems of the DCV and heating system and extend the building structure model with pre-developed components. The system specifications are defined based on the user requirements, such as the number of floors and rooms, and extend the FI blocks to the new structure accordingly. The framework presented here is generic and scalable, and it can be instantiated for various fault combinations. The fault attributes are represented by matrices, which can be expanded in dimensions to support more complex structures with extra components, zones, and buildings. The indexing method is based on the room's number increasing on each floor. It means that each room has an index number. The total number of rooms in a multi-floor building is calculated using Equation 24.

$$Total\_Rooms\_Numbers = B \times K \times N. \qquad \text{Equation 24}$$

Where B is the number of buildings, K is the floor number, and N specifies the number of rooms on each floor. To integrate the multiple fault injection components with large-scale system components, indexing is used to access the electronic components of each room. Figure 47 illustrates how a component is accessible through an index including the floor, room, and component numbers.

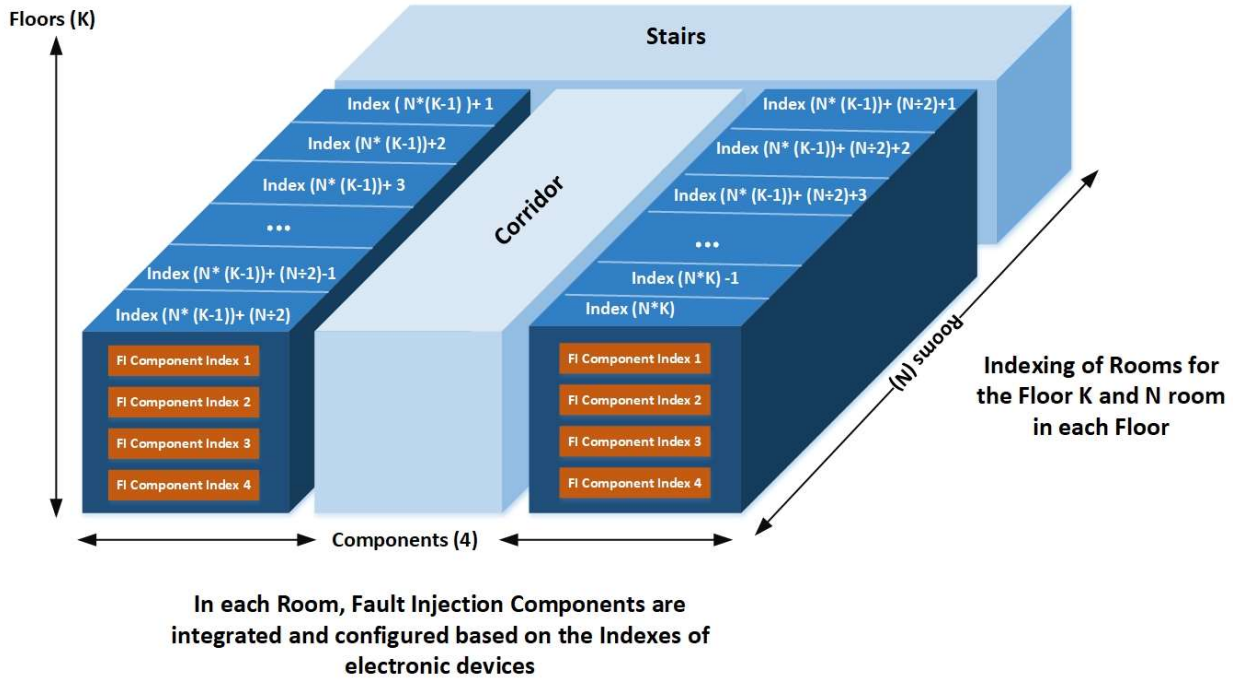

Figure 47. Components and rooms are indexed for the multiple fault injection in a large-scale building structure.

6.3.3 Methods for Configuration of Generic Simulation Components based on High-Level Specification

To integrate and configure the system structure, components are connected and linked through an indexing technique that facilitates the fault injection process. The algorithm uses indexes to manage the room components and to access the faulty components via their addresses. The FI framework injects multiple faults with different attributes at different locations and intensities, enabling the identification and analysis of different fault combinations. Function 6 describes the large-scale component-based system structure generation by integrating the composable system model generation and multiple fault injection.

Function 6. Large-scale component-based system structure generation for multiple fault injection evaluation.

Algorithm for Component-Based System Model Generation
1. System Model Requirements definition including system model specifications, assumptions, variables, fault scenarios, and fault attributes.
2. Defining the fault class and fault injection properties.
3. Generation of a repository of the room components for the DCV and heating system.
4. Integrating the different electronic components of different rooms with the multiple fault injection components.
5. Executing the automated fault injection to get input variables for the number of floors (K) and the number of rooms (N).
6. Generating the extendable and multi-dimensional fault injection matrixes based on the number of rooms and components.
7. Generating the fault scenarios for multiple fault injections.

8. Executing the generation script for creating the component-based system model using new room components capable of realistic multiple faults injection.
9. Selecting an appropriate room type based on their types and making correct and errorless connections based on the index pattern.
10. Configuration of the new room components based on the indexing method and mapping the index pattern to each room, including the floor and room number.
11. Configuration of each fault injection component and Stateflow diagram based on the indexes of electronic components in each room.
12. Integrating all system components by connecting the input and output ports in a pattern for each floor.
13. Connecting the output ports of system components to controller components to monitor the system output.

To configure the room components for multiple fault injection, the indexing should be mapped to fault injection blocks. The "Activated_Room_component_Combination_Matrix" in multiple fault injection systems is used to make a pattern for the system model indexes. Because it shows the number of buildings , floors, rooms, and components in a system structure by increasing the number of dimensions, it also determines the faulty component for multiple faults in multi-zone and multi-floor structures. During the system model generation, the room index and component indexes in the Stateflow diagram should be assigned to merge the multiple fault injection components with the room blocks. Figure 48 depicts an example cumulative timeline for the multiple fault injection in a large-scale component-based system structure with three main axes denoting the number of floors, number of rooms, and components. This timeline shows the localizations of faulty components by their addresses. Each address is an index, including the floor indicator, room indicator, and component indicator. Figure 49 depicts two fault injection examples: one intermittent temperature sensor fault with three failure repetitions with an index (floor 1, room 2) and one CO_2 concentration sensor permanent fault for an index (floor 3, room 5).

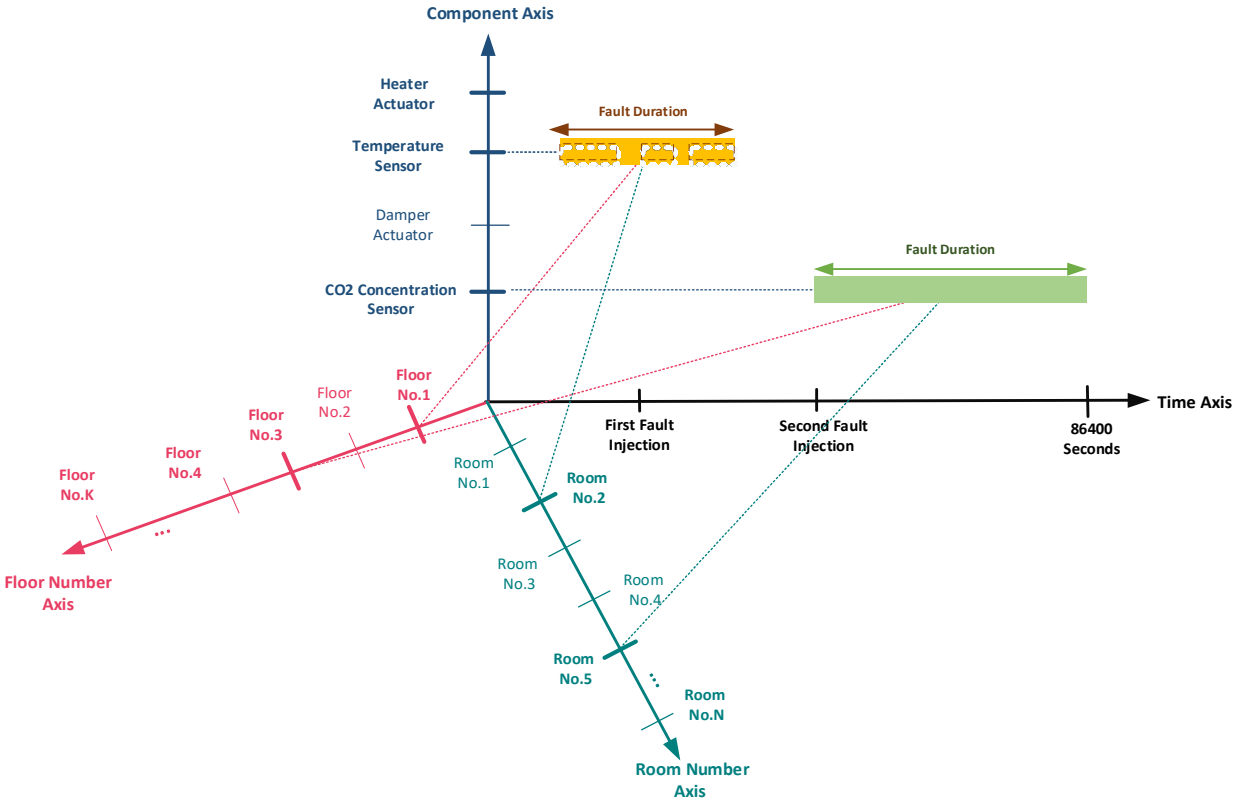

Figure 48. Multiple fault injection timelines in a component-based simulated system model, including the floors, rooms, and components axes.

6.4 Example of Multiple Fault Injection in a Component-Based System Model

An example of a component-based system model with three floors and six rooms on each floor is implemented. Consequently, the total number of rooms is 18. The minimum number of the rooms in each floor is 6 room. Therefore, Table 25 describes the "Activated_Room_Component_Combination_Matrix" for activating the multiple fault injection in an extendable component-based system model. The size of the matrix is 18 rooms × 4 components. This example specifies precisely how the multiple fault injection is mapped to a large-scale component-based system model by adapting the fault attribute matrixes with the system structure. Each element in Table 25 can obtain two binary values: 1 and 0. To address a faulty component in a specified room, the value of the associated element turns from 0 to 1. The faulty components can be determined randomly or manually for one example or with a scenario for multiple examples. In this example, the faulty values are assigned manually. There are two fault locations. The first and second faults are activated with index (2,2) and index (11,3), respectively. The number of buildings is one and the building is therefore not considered in the indexing.

Table 25. “Activated_Room_Component_Combination_Matrix“ for multiple fault injection example in the extendable component-based system model.

Floors	Components / Rooms	Component 1 (CO_2 Concentration Sensor)	Component 2 (Damper Actuator)	Component 3 (Temperature Sensor)	Component 4 (Heater Actuator)
Floor 1	Room 1	0	0	0	0
	Room 2	0	1	0	0
	Room 3	0	0	0	0
	Room 4	0	0	0	0
	Room 5	0	0	0	0
	Room 6	0	0	0	0
Floor 2	Room 7	0	0	0	0
	Room 8	0	0	0	0
	Room 9	0	0	0	0
	Room 10	0	0	0	0
	Room 11	0	0	1	0
	Room 12	0	0	0	0
Floor 3	Room 13	0	0	0	0
	Room 14	0	0	0	0
	Room 15	0	0	0	0
	Room 16	0	0	0	0
	Room 17	0	0	0	0
	Room 18	0	0	0	0

The persistence values in Table 22 must be initialized based on the fault locations as specified in Table 25. The persistence values in multiple fault injections are set as “1” for permanent fault injection and “2” for short intermittent faults. Transient faults are omitted in the multiple fault injection framework due to the lack of reliable timing parameters described thoroughly in the fault modeling in chapter 4. In addition, intermittent faults mainly occur in sensor components. Hence, the persistence value of the first and second faulty components are “1” and “2” respectively. Other fault attributes are assigned randomly in the 3-D structures for four failure repetitions with intermittent faults.

Table 26. "Fault_Injection_Persistence_Matrix" for multiple fault injection example in an extendable component-based system model.

Floors	Components / Rooms	Component 1 (CO_2 Concentration Sensor)	Component 2 (Damper Actuator)	Component 3 (Temperature Sensor)	Component 4 (Heater Actuator)
Floor 1	Room 1	0	0	0	0
	Room 2	0	1	0	0
	Room 3	0	0	0	0
	Room 4	0	0	0	0
	Room 5	0	0	0	0
	Room 6	0	0	0	0
Floor 2	Room 7	0	0	0	0
	Room 8	0	0	0	0
	Room 9	0	0	0	0
	Room 10	0	0	0	0
	Room 11	0	0	2	0
	Room 12	0	0	0	0
Floor 3	Room 13	0	0	0	0
	Room 14	0	0	0	0
	Room 15	0	0	0	0
	Room 16	0	0	0	0
	Room 17	0	0	0	0
	Room 18	0	0	0	0

6.5 Implementation of Classifier-based Fault Diagnostic Algorithm using Fuzzy Bayesian Belief Networks

The novel and generic FBBN diagnostic algorithm is implemented in a DCV and heating system scenario. Based on the presented model from chapter 5, the algorithm is implemented in three different phases: (1) fuzzification by a system expert, (2) implementation phase, (3) diagnosis phase, and (4) evaluation phase.

6.5.1 Fuzzification by System Expert

There are several requirements for implementing the FBBN. Experts use their knowledge to extract system model information and fuzzy rules. Experts must define the system attributes, subdomains (i.e., system attribute fuzzifications), and fuzzy membership functions for each fuzzified subdomain to calculate fuzzy weights (probabilities). In addition, the system model output data are raw data, including useless information. Therefore, the output data should be prepared by selecting the required system attributes in the RDT table. The RDT table is prepared for the example DCV and heating system model with three kinds of system attributes: constant, discrete, and continuous ones as shown in Table 27. Table 27 includes the system measurement values for each time step. The time sample range is between 1 and 86400 seconds.

Table 27. RDT for the DCV and heating system including constant, discrete and continuous attributes.

Constant System Attributes			**Discrete System Attributes**		**Continuous System Attributes**	
Second (Sample time)	Outdoor Daily Temperature	Occupancy Number	Damper Status	Heater Status	CO_2 Concentration	Indoor Room Temperature
1						
…						
86400						

Experts use fuzzy rules to define system subdomains. System attributes must be divided into several subdomains based on their types. Discrete attributes are divided into two different subdomains based on their status. Continuous and constant attributes are divided into three subdomains based on their ranges. Table 28 explains the seven system domains and 18 subdomains.

Table 28. SLT table to define the fuzzified subdomains based on system domains in the DCV and heating system example scenario.

No.	**Attributes (Domains)**	**Subdomains**	**Subdomains**	**Subdomains**
1	Daily Temperature	Low_Daily_Temperature	Middle_Daily_Temperature	High_Daily_Temperature
2	Number of Occupants	Low_Occupancy	Normal_Occupancy	High_Occupancy
3	Room Temperature	Lower_than_Threshold_RoomTemperature	Within_Threshold_RoomTemperature	Upper_than_Threshold_RoomTemperature
4	Heater Status	Heater_Status_On	Heater_Status_Off	---------
5	Damper Status	Damper_Status_Open	Damper_Status_Close	---------
6	Simulation Clock	Healthy_Mode	Faulty_Mode	---------
7	Room CO_2 Concentration	Lower_than_Threshold_CO_2Value	Within_Threshold_CO_2Value	Upper_than_Threshold_CO_2Value

After defining the fuzzification of system attributes, a fuzzy membership function is defined for each new subdomain based on their changes. Table 29 details the system domain, new subdomains, ranges of the new subdomains, and their units. A specific number is assigned to each new subdomain applicable to FBBN implementation tables, such as SLT (Table 10) , WFRDT (Table 11) , SPV (Table 12) , ITTM(Table 13) , SRT (Table 14), CPT (Table 15), and RDP (Table 16) tables.

Table 29. Implementation details for fuzzy membership function definitions.

No. of Fuzzified Subdomain	System Attribute (Domain)	Fuzzified Subdomain	Values and Ranges	Units
No.1	Daily Temperature	Low_Daily_Temperature	[0-5]	ºC
No.2	Daily Temperature	Middle_Daily_Temperature	[5-9]	ºC
No.3	Daily Temperature	High_Daily_Temperature	[9-14]	ºC
No.4	Occupants Number	Low_Occupancy	Less Than 3 People	Person
No.5	Occupants Number	Normal_Occupancy	3 and 4 People	Person
No.6	Occupants Number	High_Occupancy	5 and 6 People	Person
No.7	Room Temperature	Lower_than_Threshold_RoomTemperature	[0-17.5]	ºC
No.8	Room Temperature	Within_Threshold_RoomTemperature	[17.5-22.5]	ºC
No.9	Room Temperature	Upper_than_Threshold_RoomTemperature	[22.5-40]	ºC
No.10	Heater Status	Heater_Status_On	1	Binary
No.11	Heater Status	Heater_Status_Off	0	Binary
No.12	Damper Status	Damper_Status_Open	1	Binary
No.13	Damper Status	Damper_Status_Close	0	Binary
No.14	Simulation Clock	Healthy_Mode	Less Than 1800	Second
No.15	Simulation Clock	Faulty_Mode	More Than 1800	Second
No.16	Room CO_2 Concentration	Lower_than_Threshold_CO_2Value	[0-400]	ppm
No.17	Room CO_2 Concentration	Within_Threshold_CO_2Value	[400-800]	ppm
No.18	Room CO_2 Concentration	Upper_than_Threshold_CO_2Value	[800-1200]	ppm

Daily temperature is divided into three new subdomains in Table 29, including the Low_Daily_Temperature ranges, Middle_Daily_Temperature, and High_Daily_Temperature. To calculate the fuzzy membership values (W), three membership function based on the subdomains range values should be extracted as Equation 25, Equation 26, and Equation 27. All fuzzy membership functions are plotted in one single figure to understand the membership degree changes for each subdomain depicted in Figure 49.

$$Low\_Daily\_Temperature\_Fuzzy\_Membership\_Function(x) = \begin{cases} 1 & x \le 5 \\ \frac{(6.825 - x)}{(6.825 - 5)} & 5 \le x \le 6.825 \\ 0 & x \ge 6.825 \end{cases} \quad \text{Equation 25}$$

$$Middle\_Daily\_Temperature\_Fuzzy\_Membership\_Function(x) = \begin{cases} 0 & x < 5 \\ \frac{(x - 5)}{(6.825 - 5)} & 5 \le x \le 6.825 \\ 1 & 6.825 \le x \le 7.175 \\ \frac{(9 - x)}{(9 - 7.175)} & 7.175 \le x \le 9 \\ 0 & x \ge 9 \end{cases} \quad \text{Equation 26}$$

$$High\_Daily\_Temperature\_Fuzzy\_Membership\_Function(x) = \begin{cases} 0 & x < 7.175 \\ \dfrac{(x - 7.175)}{(9 - 7.175)} & 7.175 \le x \le 9 \\ 1 & x \ge 9 \end{cases} \quad \text{Equation 27}$$

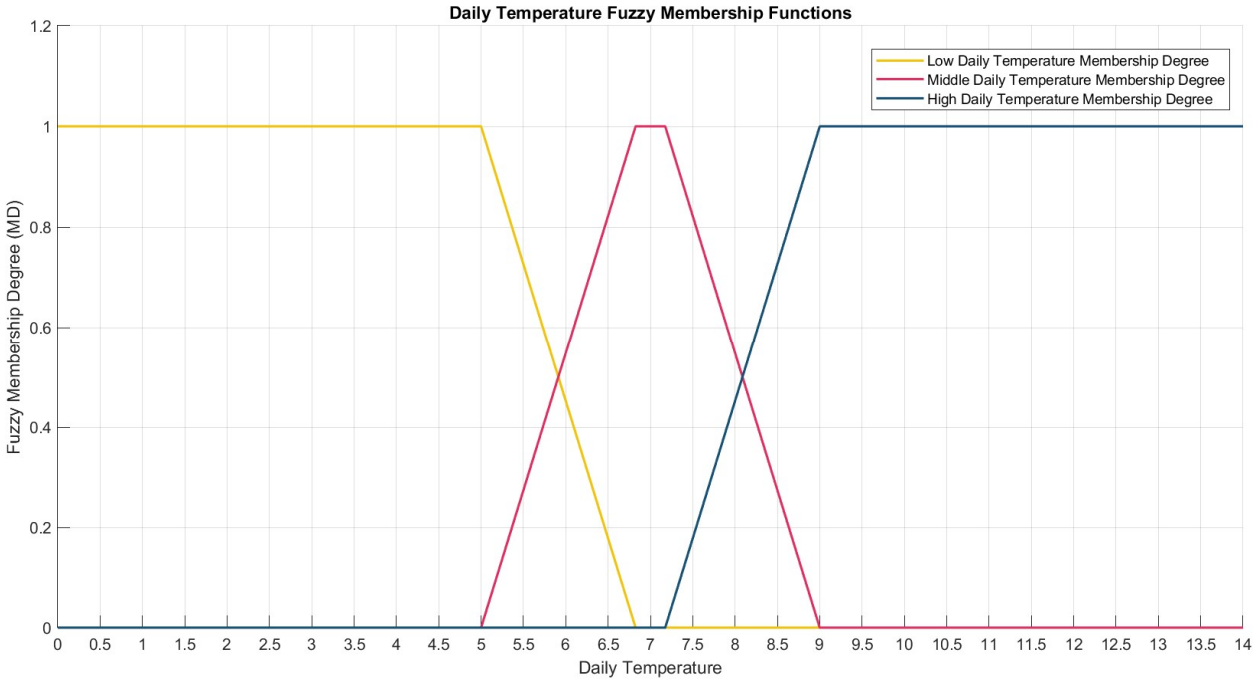

Figure 49. Plotted daily temperature fuzzy membership functions.

Three new subdomains for the occupancy numbers are Low_occupancy, Normal_Occupancy, and High_Occupancy shown in Table 29. Membership functions for the new subdomains are defined in Equation 28, Equation 29, and Equation. 30. All plotted fuzzy membership functions for the occupancy numbers is depicted in Figure 50.

$$Low\_Occupancy\_Fuzzy\_Membership\_Function(x) = \begin{cases} 1 & x \le 2 \\ \dfrac{(3-x)}{(3-2)} & 2 \le x \le 3 \\ 0 & x \ge 3 \end{cases} \quad \text{Equation 28}$$

$$Middle\_Occupancy\_Fuzzy\_Membership\_Function(x) = \begin{cases} 0 & x < 2 \\ \dfrac{(x-2)}{(3-2)} & 2 \le x \le 3 \\ 1 & 3 \le x \le 4 \\ \dfrac{(5-x)}{(5-4)} & 4 \le x \le 5 \\ 0 & x \ge 5 \end{cases} \quad \text{Equation 29}$$

$$High\_Occupancy\_Fuzzy\_Membership\_Function(x)=\begin{cases}0 & x<4\\ \dfrac{(x-4)}{(5-4)} & 4\le x\le 5\\ 1 & x\ge 5\end{cases}$$ Equation. 30

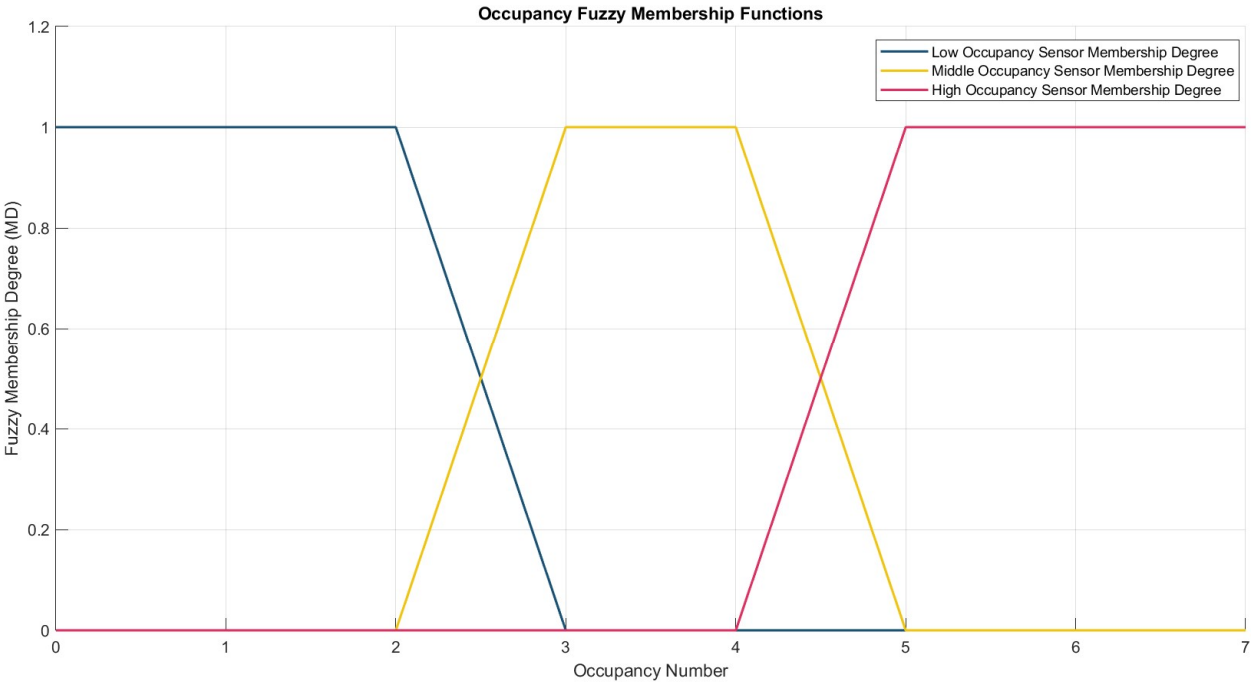

Figure 50. Plotted Occupancy fuzzy membership functions.

Subdomains for the room temperature numbers are Lower_than_Threshold_RoomTemperature, Within_Threshold_RoomTemperature, Upper_than_Threshold_RoomTemperature shown in Table 29. Fuzzy Membership functions for the room temperature are extracted based on their range values shown in Equation 31, Equation 32, and Equation 33. All room temperature's fuzzy membership functions are plotted in Figure 49to have a better understanding of the membership degree changes.

$$Low\_Room\_Temperature\_Fuzzy\_Membership\_Function(x)=\begin{cases}1 & x\le 17.5\\ \dfrac{(19.5-x)}{(19.5-17.5)} & 17.5\le x\le 19.5\\ 0 & x\ge 19.5\end{cases}$$ Equation 31

$$Middle\_Room\_Temperature\_Fuzzy\_Membership\_Function(x)=\begin{cases}0 & x<17.5\\ \dfrac{(x-17.5)}{(19.5-17.5)} & 17.5\le x\le 19.5\\ 1 & 19.5\le x\le 20.5\\ \dfrac{(22.5-x)}{(22.5-20.5)} & 20.5\le x\le 22.5\\ 0 & x\ge 22.5\end{cases}$$ Equation 32

$$High\_Room\_Temperature\_Fuzzy\_Membership\_Function(x)=\begin{cases}0 & x<20.5\\ \dfrac{(x-20.5)}{(22.5-20.5)} & 20.5\le x\le 22.5\\ 1 & x\ge 22.5\end{cases} \quad \text{Equation 33}$$

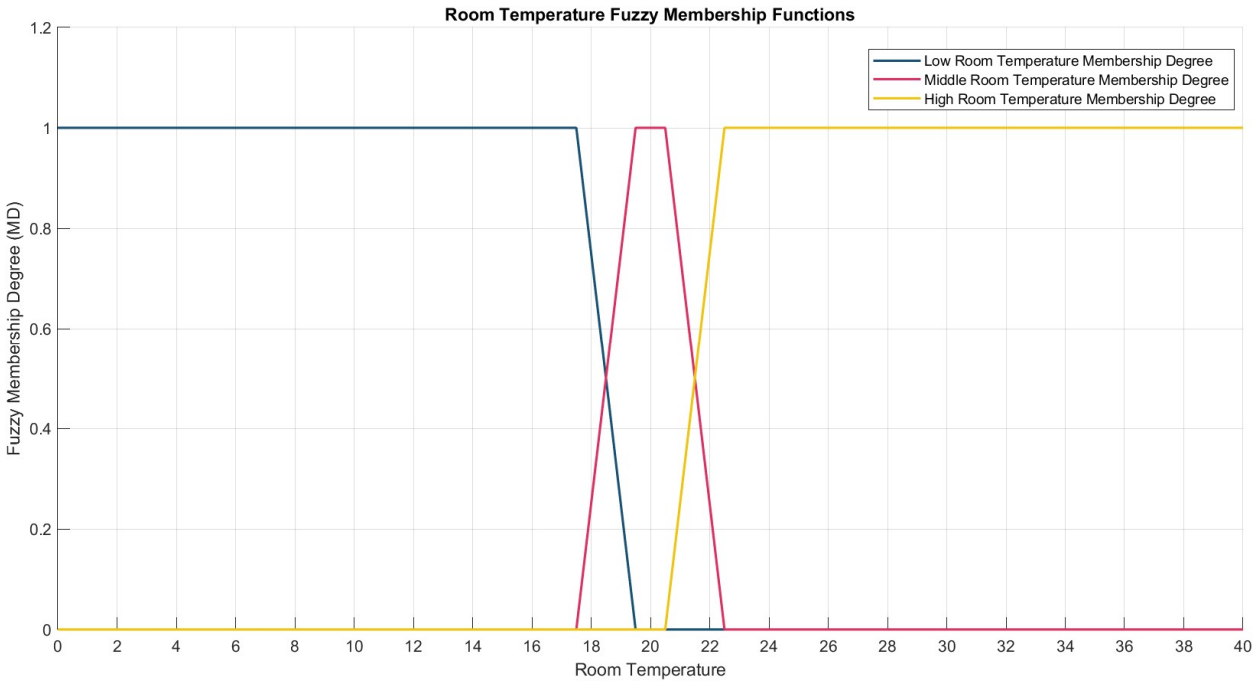

Figure 51. Plotted room temperature fuzzy membership functions.

Three new subdomains are defined for the CO_2 concentration numbers in Table 29 including Lower_than_Threshold_CO_2Value, Within_Threshold_CO_2Value, and Upper_than_Threshold_CO_2Value. A new membership function is extracted for each new subdomain based on the subdomain range values shown in Equation 34, Equation 35, and Equation 36. All fuzzy membership functions are plotted in one single figure to understand the membership degree changes for each subdomain depicted in Figure 52.

$$Low\_CO_2\_Concentration\_Fuzzy\_Membership\_Function(x)=\begin{cases}1 & x\le 400\\ \dfrac{(585-x)}{(585-400)} & 400\le x\le 585\\ 0 & x\ge 585\end{cases} \quad \text{Equation 34}$$

$$Middle\_CO_2\_Concentration\_Fuzzy\_Membership\_Function(x)=\begin{cases}0 & x<400\\ \dfrac{(x-400)}{(585-400)} & 400\le x\le 585\\ 1 & 585\le x\le 615\\ \dfrac{(800-x)}{(800-615)} & 615\le x\le 800\\ 0 & x\ge 800\end{cases} \quad \text{Equation 35}$$

$$High\_CO_2\_Concentration\_Fuzzy\_Membership\_Function(x) = \begin{cases} 0 & x < 615 \\ \dfrac{(x-615)}{(800-615)} & 615 \le x \le 800 \\ 1 & x \ge 800 \end{cases} \quad \text{Equation 36}$$

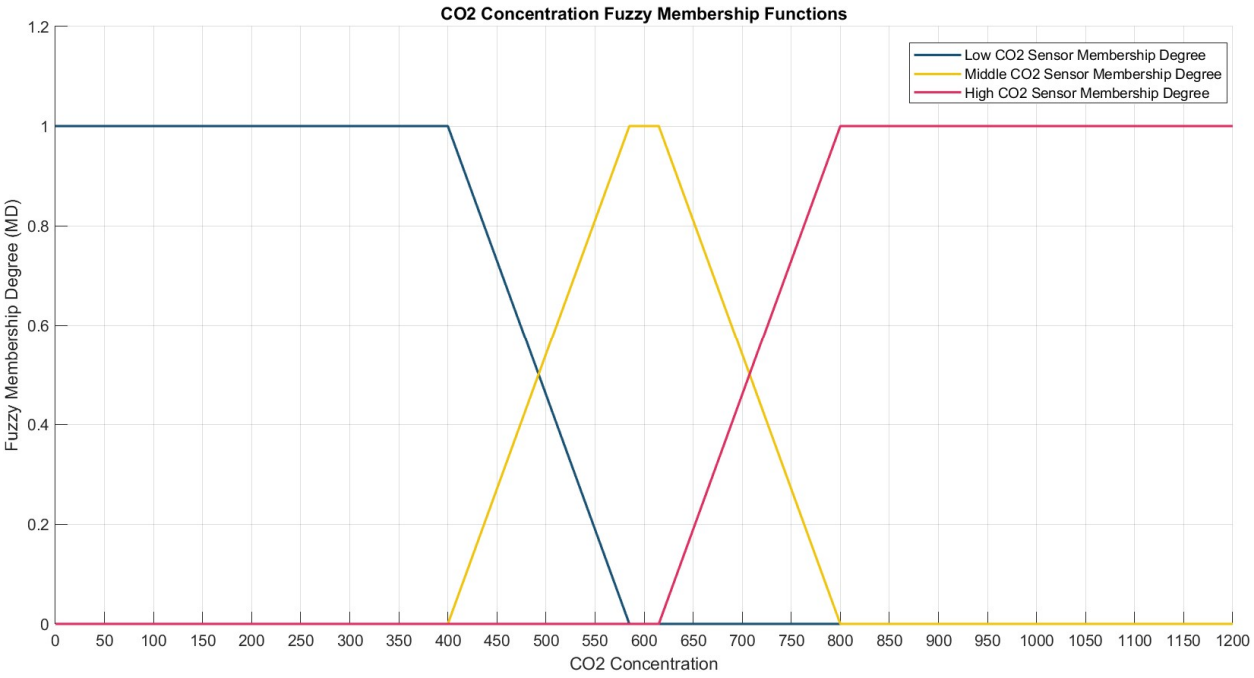

Figure 52. Plotted CO_2 concentration fuzzy membership functions.

6.5.2 Implementation Phase (Offline Training Mode)

In the second phase, the fuzzy Bayesian belief network generates a list of fault objects stored in a vector named "Offline Library." The offline library is trained with various fault case injections in this phase. Fault objects in the implementation phases include four properties: type, time, data, and RDP. Each property of one fault injection must be prepared and assigned. The offline library in the implementation phase is generated based on the different injection times (e.g., 17 points of injection times) and types (e.g., 10 points of fault types) for each system component, such as damper actuator, heater actuator, CO_2 concentration sensor, and room temperature sensor. Function 7 describes the algorithm for the implementation phase. The result of this phase is the 'ImplementationLibrary.mat' file, including the offline library, which is required for the subsequent phases. The fault injection values are selected based on the fault types and assigned in each fault injection. To facilitate the implementation, only the permanent stuck-at faults are considered for each electronic component to generate the offline library and diagnosis process.

Function 7. Algorithm with example values including the offline generation of the FBBN diagnostic technique.

Offline library implementation phase algorithm
//Requirement of implementation to initial variables
1. Load ('occupants. mat').
2. Load ('OfficeModelVaribales.mat');

```
3.  Simulation_Time = 86400;

//Choosing the fault mode, including the type and time of fault
//Fault time including 17 points of time
4.  Fault_Injection_Time_Vector= 5000:5000:86400.

//Fault type including 10 points of types
5.  Fault_Injection_Type_Vector=["CO2SensorLow","CO2SensorMiddle","CO2SensorHigh","DamperActuatorO
    ff","DamperActuatorOn","TemperatureSensorLow","TemperatureSensorMiddle","TemperatureSensorHigh",
    "HeaterActuatorOff","HeaterActuatorOn"];

//Initializing the library table
6.  [x1,y1] = size(Fault_Injection_Time_Vector);
7.  [x2,y2] = size(Fault_Injection_Type_Vector);

//Total number of injected faults to build the library table
8.  FaultCases = y1 * y2;

9.  For p =1:1:y1
10. For q =1:1:y2

11. RDP=0;
//Setting the type of fault
12. Fault_Mode= Fault_Injection_Type_Vector(q);
//Setting the time of fault
13. Delay_Time = Fault_Injection_Time_Vector(p);

//Activating the Switch cases for ten fault types and selecting the fault injection values based on the fault types in
each switch case besides FIV variables show the fault injection values.
14. Switch Fault_Mode
15. Case "CO2SensorLow"
    CO2_R1_Sensor_FIV = 350;
16. Case "CO2SensorMiddle"
    CO2_R1_Sensor_FIV = 600;
17. Case "CO2SensorHigh"
    CO2_R1_Sensor_FIV = 850;
18. Case "DamperActuatorOff"
    CO2_R1_Actuator_FIV = 0;
19. Case "DamperActuatorOn"
    CO2_R1_Actuator_FIV = 1;
20. Case "TemperatureSensorLow"
    Temp_R1_Sensor_FIV = 17;
21. Case "TemperatureSensorMiddle"
    Temp_R1_Sensor_FIV = 20;
22. Case "TemperatureSensorHigh"
    Temp_R1_Sensor_FIV = 23;
23. Case "HeaterActuatorOff"
    Temp_R1_Actuator_FIV=0;
24. Case "HeaterActuatorOn"
    Temp_R1_Actuator_FIV=1;
25. Otherwise
26. End

//Running the simulation for the system specifications and saving the results in a library file
//Open_System is a built-in function that opens the system model.
```

```
27. Open_system('Fault_Injection_Data_1.slx');
28. FaultCounter = FaultCounter + 1;

//Sim is a built-in function that runs the simulation system model
29. Sim ('Fault_Injection_Data_1.slx', Simulation_Time);

//Implementation.m MATLAB file computes the required tables for FBBN construction
30. Run ('Implementation.m');

//Creating an object from the "Fault" class
31. FaultObj= Fault;

//Saving the implementation results in fault object
32. FaultObj.Type=Fault_Mode;
33. FaultObj.Time=Delay_Time;

//The Data comes from to workspace Simulink block in the system model, which is accessible like other system
variables via the workspace panel
34. FaultObj.Data = Data;
35. FaultObj.RDP=RDP;
36. Lib(FaultCounter,1)= FaultObj;
//Close_System is a built-in function that closes the system model for each fault case
37. Close_system('Fault_Injection_Data_1.slx').
//The script continuous for other fault cases with the next iterations
38. End
39. End
//Save function uses to keep all defined fault objects in 'ImplementationLibrary.mat'
40. Save ('ImplementationLibrary.mat','Lib').
```

6.5.3 Diagnosis Phase (Online Diagnostic Mode)

The third phase is the diagnosing of the injected fault cases. Type, time, and values are selected randomly as described in Function 8. The main differences between the implementation and diagnosis phases are the fault injection value definitions which are random in the diagnosis phase despite the training mode where the offline library is generated with pre-defined fault injection attributes. Another difference relates to the diagnosis operations including the calculation of the Precentage_List referred to in Table 18. Function 8 determines the mutuality (or similarity) of the actual fault case with the entries in the offline library as parent-child pairs. The diagnosis algorithm can be iterated for any demanded fault cases, and the results are saved in 'DiagnosisLibrary.mat'.

Function 8. Algorithm for diagnosis phase including the percentage list for the FBBN diagnostic technique.

Diagnosis Phase algorithm

```
//Requirement of diagnosis phase
1. load('occupants.mat');
2. load('OfficeModelVaribales.mat');
3. load('ImplementationLibrary.mat');
4. Simulation_Time = 86400;

//The library which is generated in the implementation phase is initialed as a variable
5. ImplementationLib= Lib;
```

```
6.  Simulation_Time = 86400;

//Number of repetitions for diagnosing specifying the number of real case faults that should be diagnosed. This number is increased for the evaluation phase
7.  Real_Execuation_RepeatitionTime = 1;

//Choosing the fault mode, including the type and time of the fault.
//Fault time vector
8.  Fault_Injection_Time_Vector= 2000:2000:86400;

//Fault type vector
9.  Fault_Injection_Type_Vector = ["CO2Sensor","DamperActuator","TemperatureSensor","HeaterActuator"];

//Initializing the library table
10. [x1,y1]= size(Fault_Injection_Time_Vector);
11. [x2,y2]= size(Fault_Injection_Type_Vector);

//Running the System in an actual situation means random fault injection time and fault type.
12. For t=1:1:Real_Execuation_RepeatitionTime

13. RDP=0;
14. FaultValue=0;

//Setting the time of a fault case randomly
15. RealTime_Fault = randi(86400);
16. Delay_Time = RealTime_Fault;

//Setting the type of a fault case randomly
17. Fault_Mode = Fault _Injection_Type_Vector(randi(4));

 %Activating the fault injection type and selecting the fault injection values randomly in each switch case
18. Switch Fault_Mode
19. Case "CO2Sensor"
      CO2_R1_Sensor_FI =1;
      CO2_R1_Sensor_FIV= randi([300,850],1);
      FaultValue= CO2_R1_Sensor_FIV;

20. Case "DamperActuator"
      CO2_R1_Actuator_FI=1;
      CO2_R1_Actuator_FIV= randi([1,2],1)-1;
      FaultValue = CO2_R1_Actuator_FIV;

21. Case "TemperatureSensor"
      Temp_R1_Sensor_FI =1;
      Temp_R1_Sensor_FIV=  randi([10,40],1);
      FaultValue =Temp_R1_Sensor_FIV;

22. Case "HeaterActuatorS
      Temp_R1_Actuator_FI=1;
      Temp_R1_Actuator_FIV= randi([1,2],1)-1;
      FaultValue=Temp_R1_Actuator_FIV;
Otherwise
End

//Open_System is a built-in function that opens the system model, and it runs with the Simulink function
23. Open_system ('Fault_Injection_Data_1.slx');
```

```
24. Sim ('Fault_Injection_Data_1.slx',Simulation_Time);

//Implementation.m MATLAB file computes the required tables for FBBN construction
25. Run ('Implementation.m');

// Fault diagnosis function is not a build-in function and defined in this thesis which returns a list including the percentage of similarities

26. Precentage_List = FaultDiagnosis (RDP, ImplementationLib);

// Creating an object from the "RealCase" class to keep the property values of actual injected faults
27. RealCaseObj= RealCase;
28. RealCaseObj.Type=Fault_Mode;
29. RealCaseObj.Time=Delay_Time;
30. RealCaseObj.Value=FaultValue.
31. RealCaseObj.Precentage_List= Precentage_List;
32. DignosisLib(t,1)= RealCaseObj;

33. Close_system ('Fault_Injection_Data_1.slx');

End
34. Save ('DiagnosisLibrary.mat', 'DignosisLib');
```

The real fault objects (extracted from the 'RealCase' class) in this library are also included in the Precentage_List (Table 18) and Evaluation_List (Table 19). The diagnosis accuracy depends on how close the random fault value is to the offline library faulty values described in Function 7. For example, the low, middle, and high CO_2 concentration faulty values in the implementation phase are 350,600 and 850, respectively. The CO_2 concentration sensor in the diagnosis phases captures a random value in the range of [300,850]. Random actual fault value and time specify the closeness degree to each offline fault case. Increasing the offline fault cases and their variety will subsequently increase the accuracy of the fault diagnosis.

Function 9. Fault diagnosis function to generate the percentage list in the diagnosis phases of the FBBN diagnostic technique.

FaultDiagnosis Function

```
Function Precentage_List = FaultDiagnosis(RDP, lib)
1.  [x1,y1]= size(lib);

//Reading values from Simulink outputs
2.  RealFaultyData = RDP;
3.  PrecentageList(1:x1,1)=0;

//Finding mutuality values and differences of all pairs of subdomains
4.  For r =1:1:x1
5.  MutualPrecentage= Find_Precentage (RealFaultyData , lib(r).RDP);
6.  PrecentageList(r,1) = MutualPrecentage.
End
End
```

The evaluation list is one property of RealCase class in the diagnosis phase that returns the most probable diagnosed fault cases based on the percentage list. It maps the offline library fault properties with

diagnosed faults by adding the types and times of the offline library to the largest percentages. “Maxk” is a built-in MATLAB function that finds the *x* largest elements of an array. Function 10 describes the evaluation list algorithm in which the x-top elements of the Precentage_List are returned to the evaluation list, including the time, type, and percentage of the selected fault cases. This list shows the diagnosed faults for an actual injected fault.

Function 10. Evaluation list of the diagnosis phase for the FBBN diagnostic technique.

Evaluation_List Generation Algorithm

```
// Initializing the required libraries
1.  load('DiagnosisLibrary.mat');
2.  load ('ImplementationLibrary.mat');

3.  For p=1:1:size(DignosisLib)
// Choosing the x-top fault cases. x is a desired number  for realizing the accuracy of the algorithm
4.  x = 20;
5.  [~,B] = maxk (DignosisLib(p).Precentage_List, x);
6.  For q =1:1:size(B)
7.  Typ = Lib(B(q)).Type;
8.  time1 = Lib(B(q)).Time;
9.  pre = num2str(DignosisLib(p).Precentage_List(B(q)));
10. Evallist(q,1) = Typ;
11. Evallist(q,2) = time1;
12. Evallist(q,3) = pre;
13. End
14. DignosisLib(p).Evaluation_List = Evallist;
```

6.5.4 Evaluation Phase

The evaluation phase refers to studying the accuracy of the diagnosed faults. The number of injected faults is increased in this phase. The required values for the fault injection are defined in vectors. They can be injected randomly or based on a scenario to investigate the diagnosis results. The evaluation phase uses the RealCase class to define the fault objects, including the type, time, value, Precentage_List, and Evaluation_List properties. The results of all fault objects of the evaluation phase are stored in 'EvaluationLibrary.mat'. To evaluate the results, the algorithm uses the ranking method to distinguish which rank the diagnosed fault belongs to. Then, the results are grouped based on the ranks and components for actual injected faults. For more understandability, the results can be shown as diagrams.

7 Experimental Evaluation and Results

This chapter introduces the experimental evaluation for the single and multiple fault injection framework. Afterward, the fault injection results are discussed comprehensively. The single-fault injection has been tested and evaluated under seven fault scenarios and the results are shown in different charts for each associated and affected output signal. The impacts of each scenario on the system behavior, such as heating cost, CO_2 concentration and temperature variations, are analyzed and discussed. Furthermore, the multiple-fault injection has been evaluated under five main fault scenarios. Each scenario contains various sub-events to model the multiple-fault incidences. Each fault scenario has been evaluated and the system observations have been analyzed. After that, the component-based system model was evaluated under multiple fault injections for two different system layouts, including different numbers of floors and rooms per floor. The energy consumption and other system impacts including temperature fluctuations and CO_2 concentration changes are presented, and the results are discussed precisely. Finally, the results of the proposed fault detection and diagnosis technique based on the FBBN construction, and the classifier-based algorithm are validated under an actual fault case. The actual fault case is injected using a fault injection framework in online diagnostic mode and compared with offline library fault cases to serve as a baseline.

7.1 Single-Fault Injection Framework Validation and Results

The fundamental goal of the proposed fault injection framework is to analyze and monitor system behavior and evaluate the accuracy of the FI framework in diverse fault scenarios. Seven random fault scenarios were studied for the evaluation as described in Table 30. Hence, relevant faults were chosen and injected for each component to observe the system's behavior with its failure impacts, such as occupant discomfort and wasted energy. Therefore, scenarios were chosen according to fault attribute variations and their impacts on the system were observed to show the FI performance. Each fault case of the scenario is comprehensively explained with fault attributes. Fault parameters were initialized based on the coefficients shown in Table 5, and the faulty signals were measured according to Equation 8. The heater duty cycle and energy consumption were set using the designed system model for each scenario as shown in Table 30. To determine the heating cost, energy consumption was first measured by using the total number of working hours of the heater in one simulation execution (which was considered as one day). The heating cost was considered 0.3 EUR/kWh in the system model based on the prices in Germany at the time of writing this thesis. The impacts of the CO_2 concentration and temperature were also determined as shown in Table 30, and resembled faulty system-level behaviors. The scenarios are explained one by one as follows. System features and characteristics are depicted, such as actual and faulty CO_2 concentrations and temperature signals, damper and heater statuses, and heating costs for healthy and faulty situations for each scenario. The occurrence and timing of failures, e.g., failure start times, failure durations and failure interarrival times significantly depended on the application domain.

For example, Correcher et al. [248] and Wakil et al. [124] proposed probabilistic strategies to find failure characteristics, such as failure start times, failure durations and failure interarrival times based on experimental data. In Table 30, coefficients for each fault scenario are suggested according to the application domain of this thesis, which is a DCV and heating system with sensor and actuator components. In addition, the ranges of variables and local inputs are determined according to system thresholds.

Intermittent faults are common in actuators, e.g., damper actuators and thermostats (heater actuators) with relays. The literature suggests certain timing criteria for these intermittent faults [130, 242]. Kuflom et al. [242] investigated unstable and intermittent faults for numerical and electromechanical overcurrent relays. They examined the effect of resetting times in different fault scenarios. They used a pulse generator to generate fault signals and monitor response times. Therefore, in this thesis, the timing patterns for intermittent faults of actuators were modeled according to the timing patterns in [130, 242].

Table 30. Example fault scenarios for the evaluation of the fault injection framework.

Fault Set Nr.	Fault Injection Start Time(s)	Component	Fault Persistence	First Fault Duration(s)	Second Fault Duration(s) (In Case of the Intermittent faults)	Fault Interarrival Time (s)	Fault Type	Fault Co-efficient α	Fault Injection Co-efficient β	Heater Duty Cycle (%)	Heater Energy Consumption (KWH) Energy Consumption Change (in %)	CO_2 Concentration Impact	Temperature Impact
1	150,00	CO_2 sensor	Permanent	-	-	-	Offset fault	125 ppm	1	62.45	64.44+26.7%	√	√
2	15,000	CO_2 sensor	Permanent	-	-	-	Data loss	Last value	0	41.4	42.72−13.3%	√	×
3	15,000	CO_2 sensor	Transient	3000	-	-	Stuck at	750 ppm	0	49.51	51.1 +6.3%	√	×
4	15,000	Damper actuator	Intermittent	2700	600	2000	Stuck at	1 (on)	0	49.63	51.22+6.3%	×	×
5	15,000	Damper actuator	Permanent	-	-	-	Stuck at	1 (on)	0	89.69	92.56+80%	×	√
6	15,000	Temperature sensor	Permanent	-	-	-	Stuck at	16 °C	0	89.83	92.71+80%	×	√
7	15,000	Heater actuator	Permanent	-	-	-	Stuck at	1 (open)	0	47.25	48.76+80%	×	√

7.1.1 Scenario 1

Scenario 1 describes a permanent offset fault for the CO_2 sensor and shows the impact of a high CO_2 concentration on the system behavior, causing a high heater consumption, a clear increase in heating cost, and, subsequently, the discomfort of occupants due to lower temperature values. The CO_2 sensor has a permanent offset fault with a 125-ppm offset coefficient value in this scenario. This fault is injected at 15,000 s. In the healthy mode of the system model, whenever the CO_2 concentration increases, the damper actuator is opened due to the high number of occupants inside the room or increased CO_2 sensor concentration. Figure 53 shows the reaction of the damper subsystem to the offset fault in the CO_2 sensor, which causes an increase in CO_2 concentration values due to the faulty sensor readings and a decrease in actual CO_2 values due to an opened damper at specific times. The results include thermal discomfort and temperature decrease as shown in Figure 54, and energy waste as shown in Figure 55.

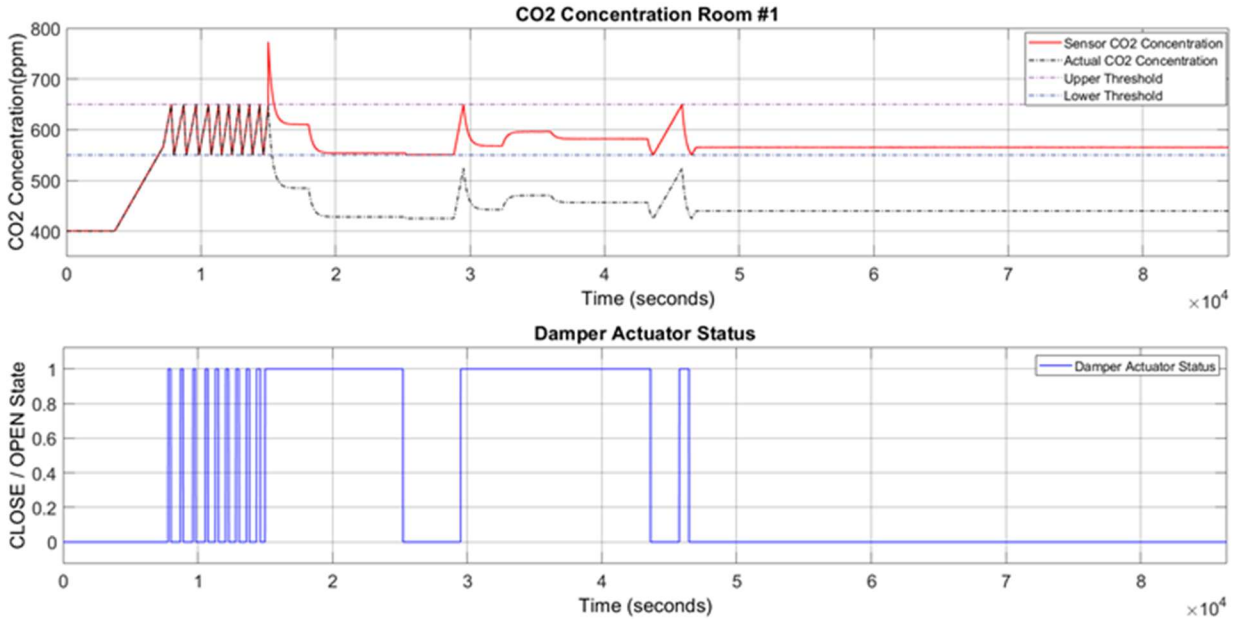

Figure 53. Permanent offset fault of CO_2 concentration sensor and damper actuator status (Scenario 1).

Figure 54 shows the signal variations of the temperature inside the room, which decreases during the fault duration because the open status of the damper actuator brings cold air from the outside to the indoor environment. In the case of permanent faults, the faulty state continues for the rest of the execution time. Since the fault injection increases the concentration value (above the upper threshold of 650 ppm), the damper actuator opens, decreasing CO_2 concentration in the room. Figure 54 shows this temperature drop, which causes occupant discomfort.

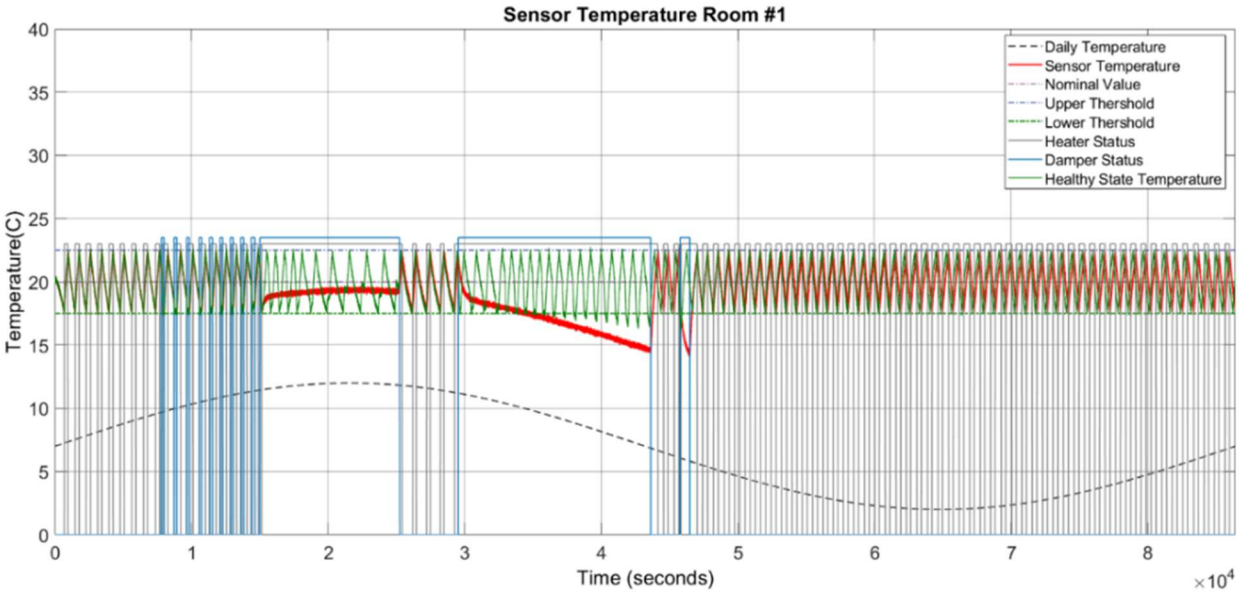

Figure 54.Temperature variation in permanent offset fault of the CO_2 concentration sensor (Scenario 1).

During the whole fault duration, the heater is turned on to compensate for the heating load due to the opened damper and to increase the temperature, increasing the heater duty cycle, energy consumption, and heating costs, as shown in Figure 55.

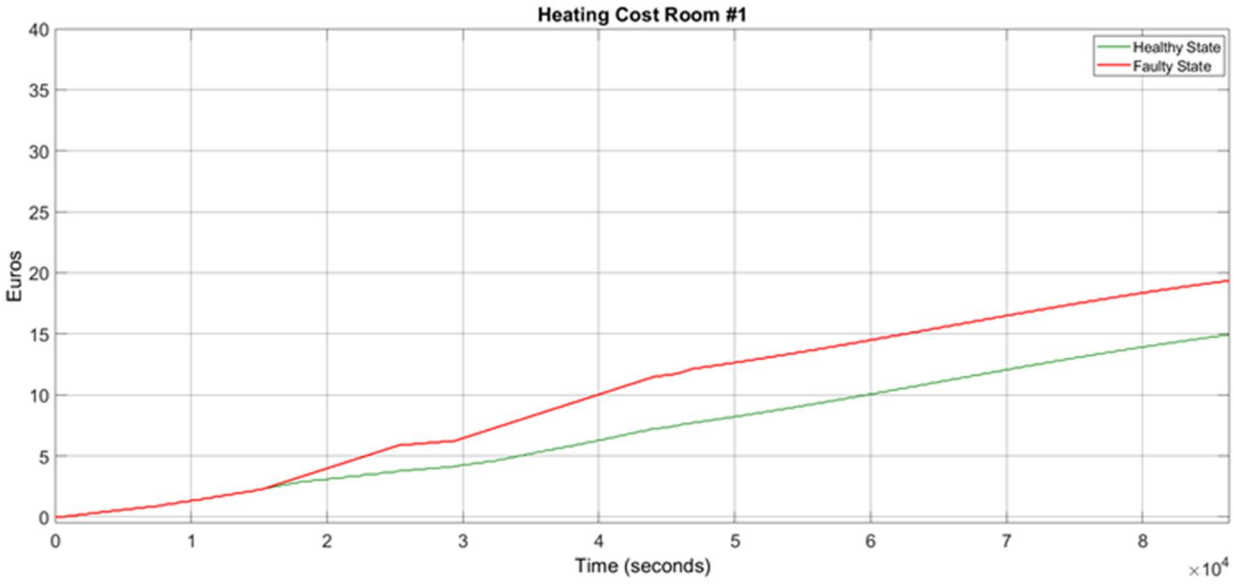

Figure 55. Heating cost determined for permanent offset fault of the CO_2 concentration (Scenario 1).

7.1.2 Scenario 2

In Scenario 2, the CO_2 concentration sensor has a permanent data loss fault. This fault is injected at 15,000 s, illustrated in Figure 56. In this scenario, the data loss fault results in the damper actuator becoming stuck at closed, diminishing the load on the heater and reducing the overall energy consumption compared to a healthy state operation.

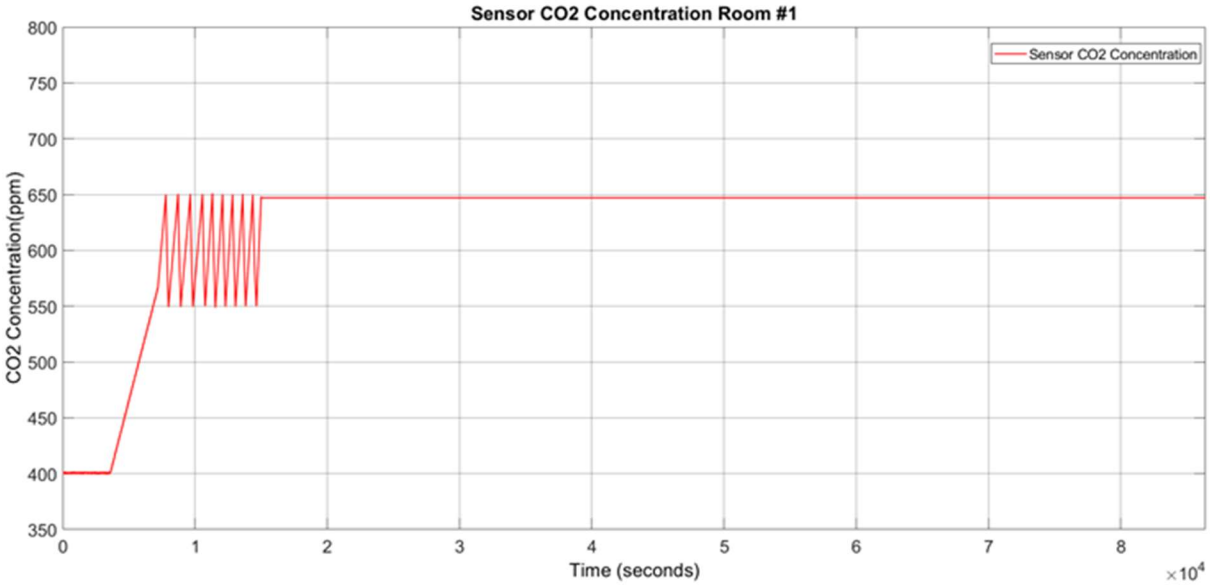

Figure 56. Zoomed view of faulty CO_2 concentration sensor reading in case of permanent data loss.

Since the CO_2 concentration value is within the threshold (650–550 ppm), the damper actuator is closed because the indoor CO_2 concentration is in the acceptable range. However, the closed damper actuator status causes an increase in CO_2 concentration, as shown in Figure 57. A high amount of CO_2 concentration causes the loss of concentration for the occupants, degradation of work efficiency, other health impacts and may even put lives in danger.

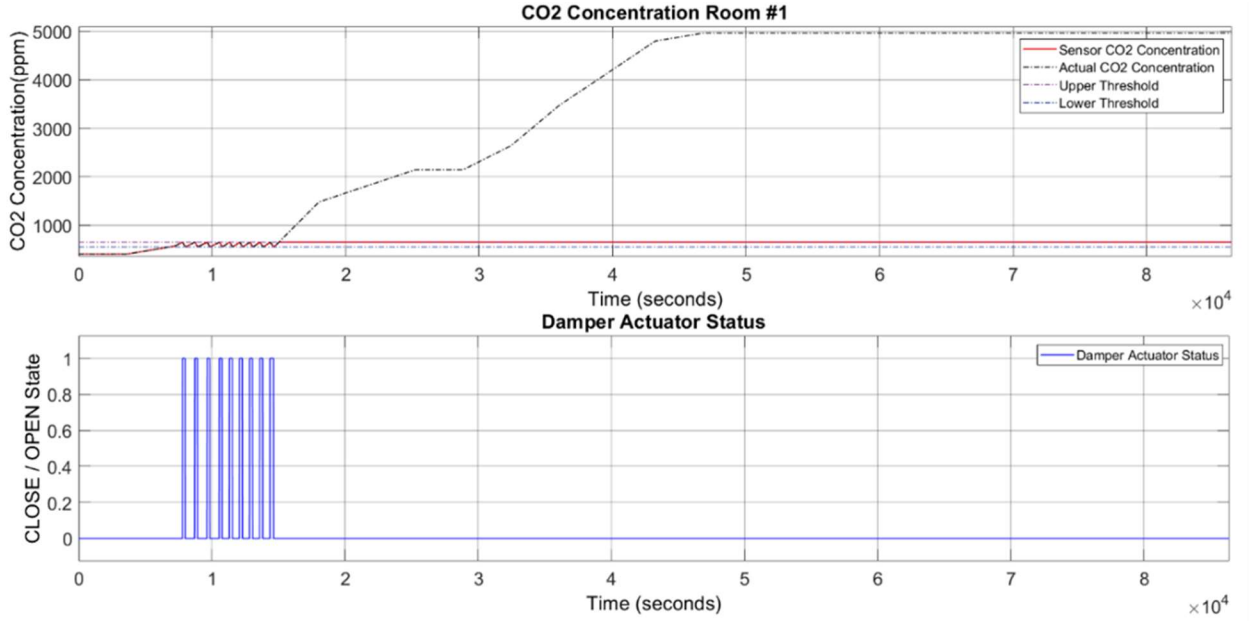

Figure 57. Actual and faulty measurements of a permanent data loss fault for the CO_2 concentration sensor vs. damper actuator status (Scenario 2).

The temperature inside the room stays in an acceptable range during the fault duration because the heater can moderately control the heating load, as shown in Figure 58.

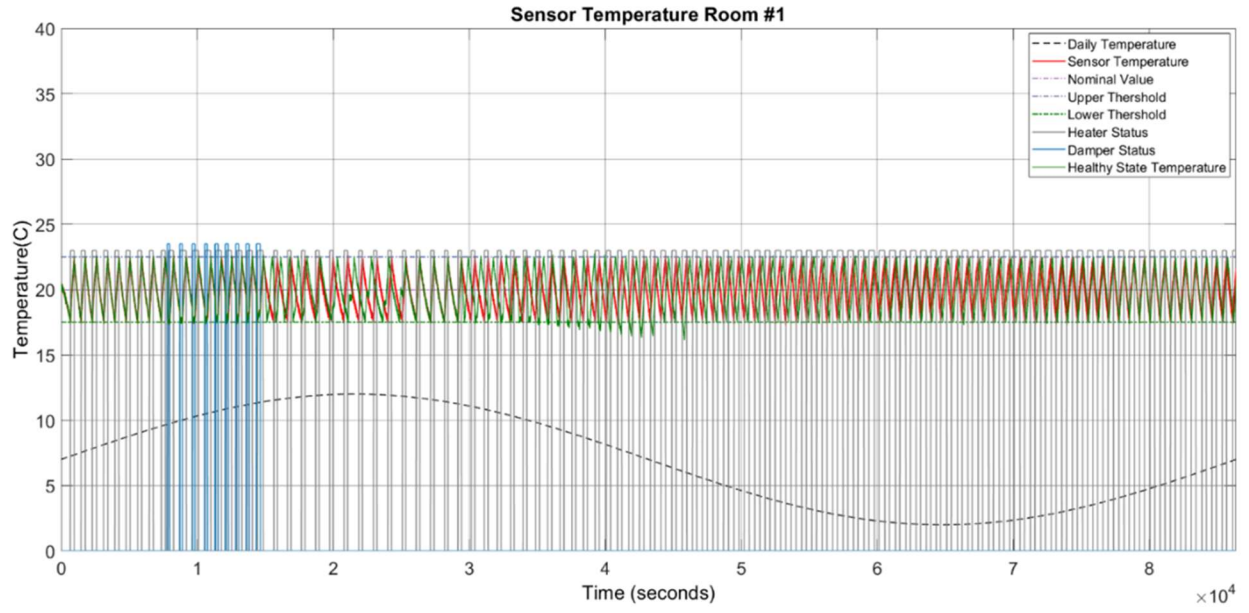

Figure 58. Temperature measurements and variations in CO_2 concentration (Scenario 2).

As the damper actuator is closed, no cold air enters the room from the outside environment. This reduces the heating load and causes a lower heater duty cycle and, accordingly, lower heating costs compared to the healthy mode, as illustrated in Figure 59.

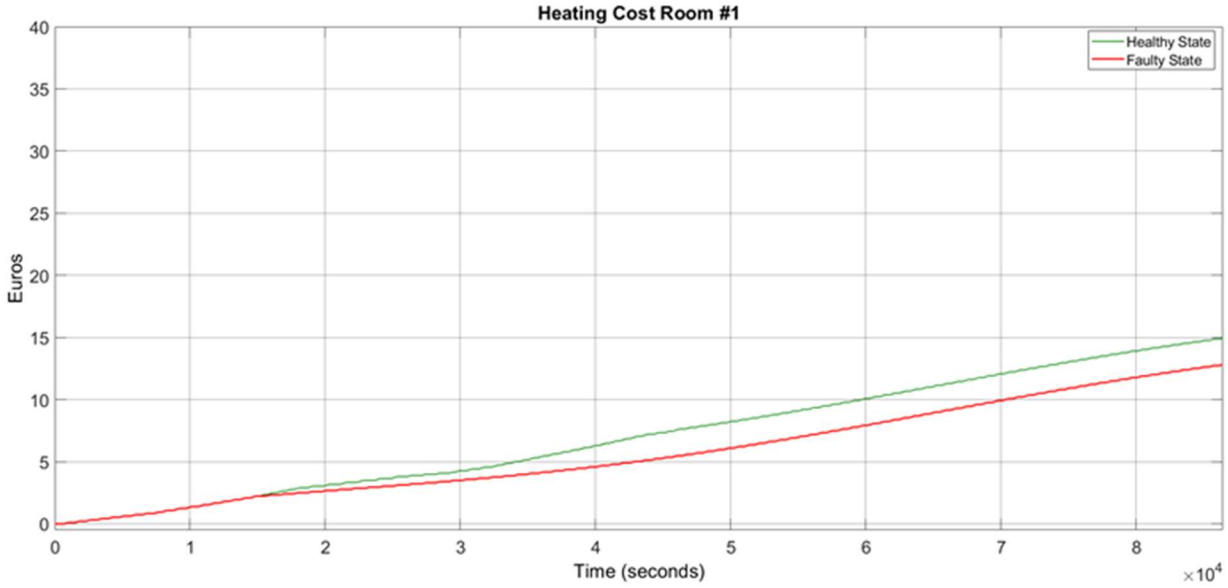

Figure 59. Heating cost and permanent data loss fault for the CO_2 concentration sensor (Scenario 2).

7.1.3 Scenario 3

Scenario 3 represents a transient stuck-at-fault for the CO_2 sensor at 750 ppm. This fault is injected at 15,000 s and lasts for the specified fault duration time, which is 3000 s.

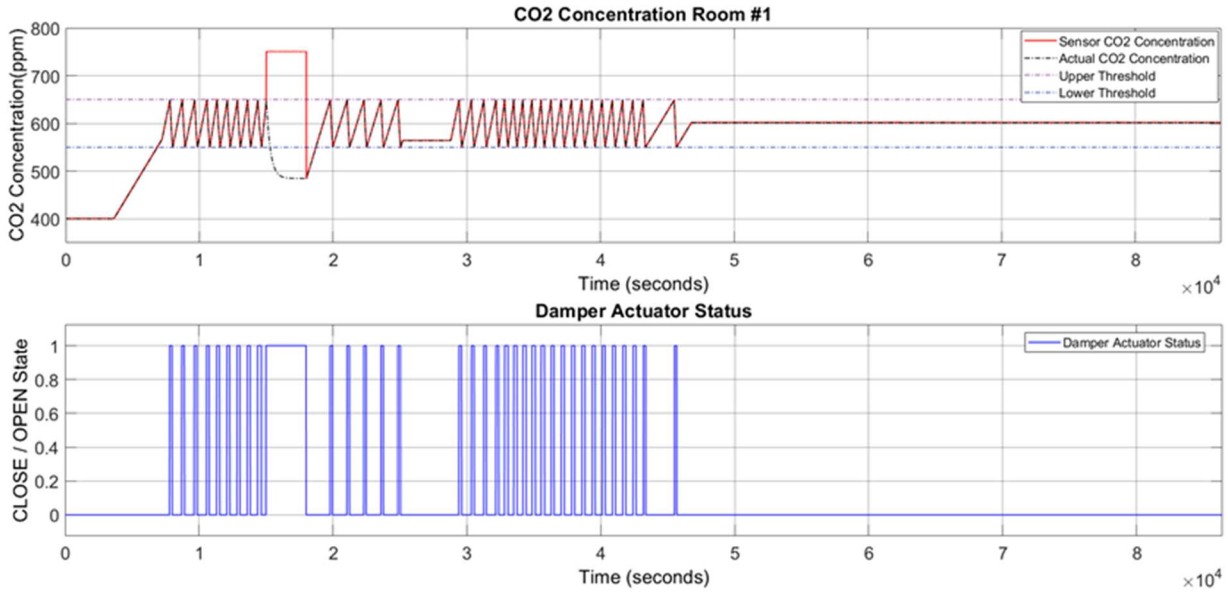

Figure 60. Actual and faulty measurements for the transient stuck-at fault for CO_2 concentration sensor vs. damper actuator states (Scenario 3).

Since the CO_2 sensor concentration is out of the nominal range of 550–650 ppm with a value greater than 650 ppm, the damper actuator should reduce the CO_2 concentration inside the room. So, the damper actuator status changes and opens at 15,000 s, remaining in this situation for a period of 3000 sec, which is clearly shown in Figure 60. The temperature inside the room decreases during the fault duration as the damper actuator state causes the entering of cold air from the environment into the system, as shown in Figure 61. To compensate for the heating load due to an opened damper during the fault duration, the heater should remain turned on for a longer time compared to the healthy mode, increasing the heater duty cycle and energy consumption. Hence, compared to the healthy state, there is a slight increase in the heating cost of the system under the faults, as shown in Figure 62.

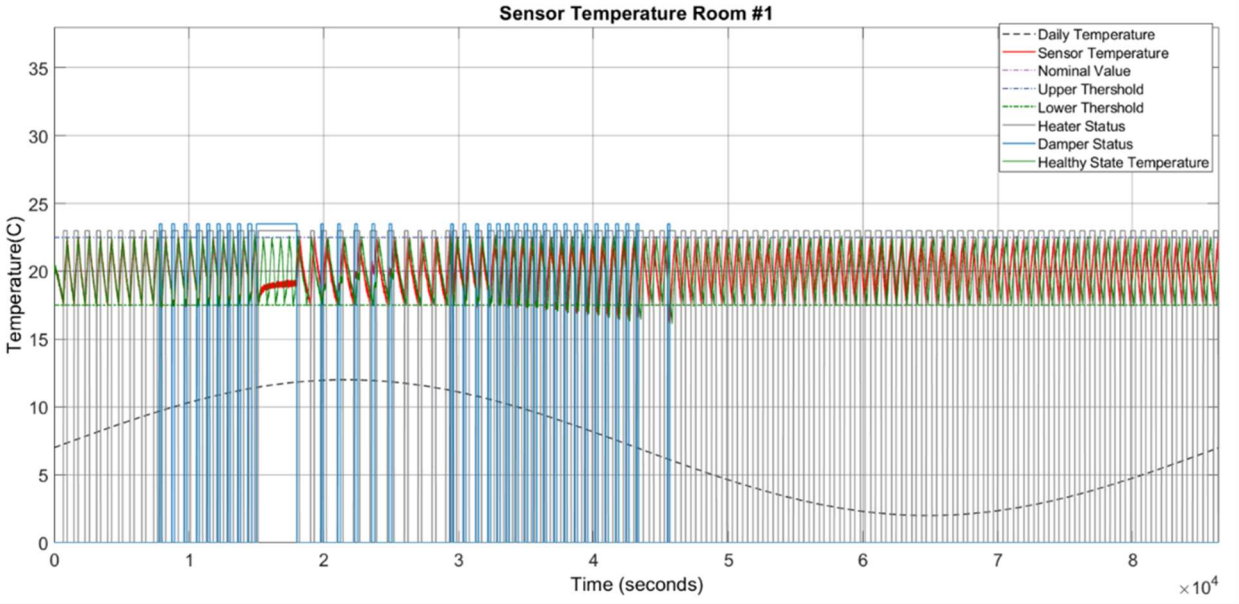

Figure 61. Temperature variation due to transient stuck-at fault of CO_2 concentration sensor (Scenario 3).

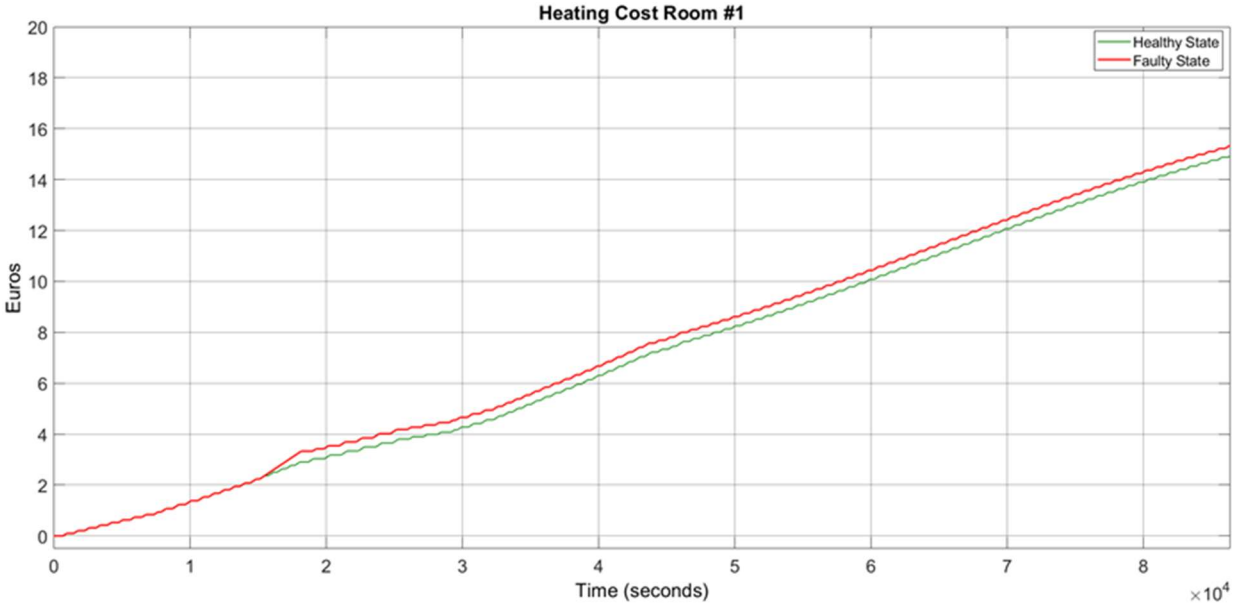

Figure 62. Heating cost due to transient stuck-at-fault of the CO_2 concentration sensor (Scenario 3).

7.1.4 Scenario 4

In scenario 4, an intermittent stuck-at fault with two repetitions is injected into the damper actuator in an open status. The first failure is injected at 15,000 s. This faulty state lasts for 2700 s. After that, the damper operation continues in a healthy mode for 2000 sec (interarrival time). Afterward, the second failure is injected into the system, it lasts for 600 s, and then the system operates normally again. Since the damper is stuck in an open state, the CO_2 concentration inside the room is reduced and reaches the minimum value of 460 ppm. The damper states and changes in the CO_2 concentration values can be seen in Figure 63.

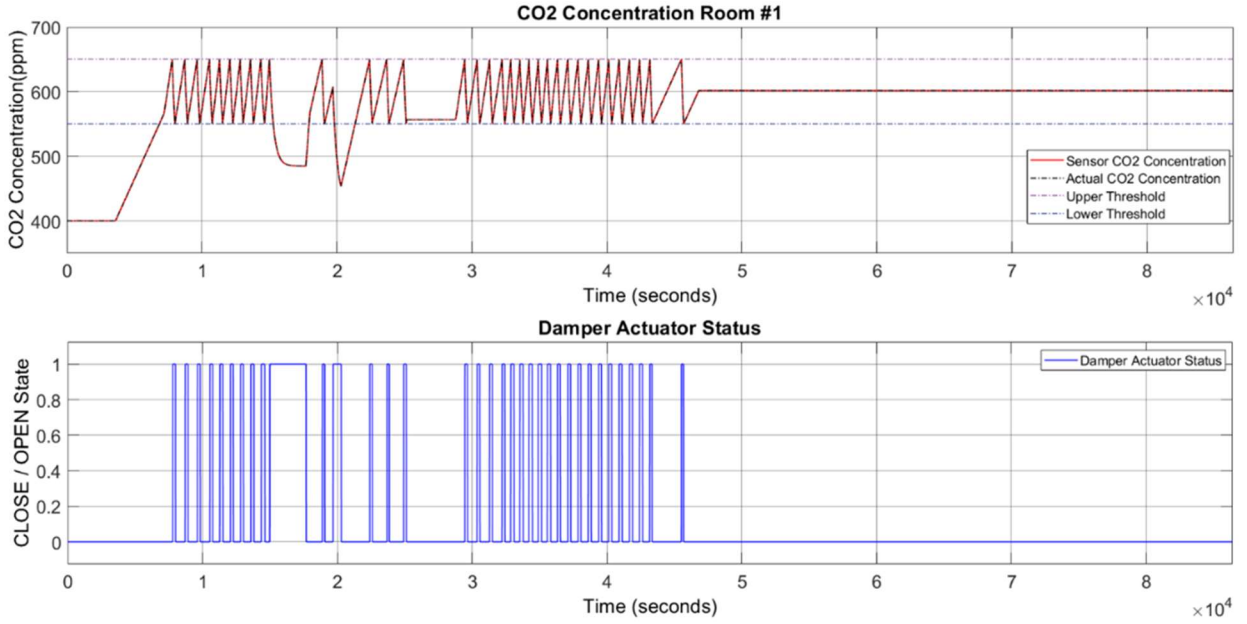

Figure 63. Actual and faulty measurements of the CO_2 concentration sensor under an intermittent stuck-at fault for the damper actuator (Scenario 4).

The damper status causes the entering of cold air from the outside environment into the room. This results in a decrease of the room temperature as depicted in Figure 64. The heater changes to an ON state after 15,000 s to increase the room temperature. However, the temperature will not stay in the acceptable range due to the damper actuator as illustrated in Figure 64. The temperature inside the room follows the trend of the environmental temperature during the fault injection time. Therefore, the heater operates at a higher duty cycle and increases the overall energy consumption and heating cost as shown in Figure 65.

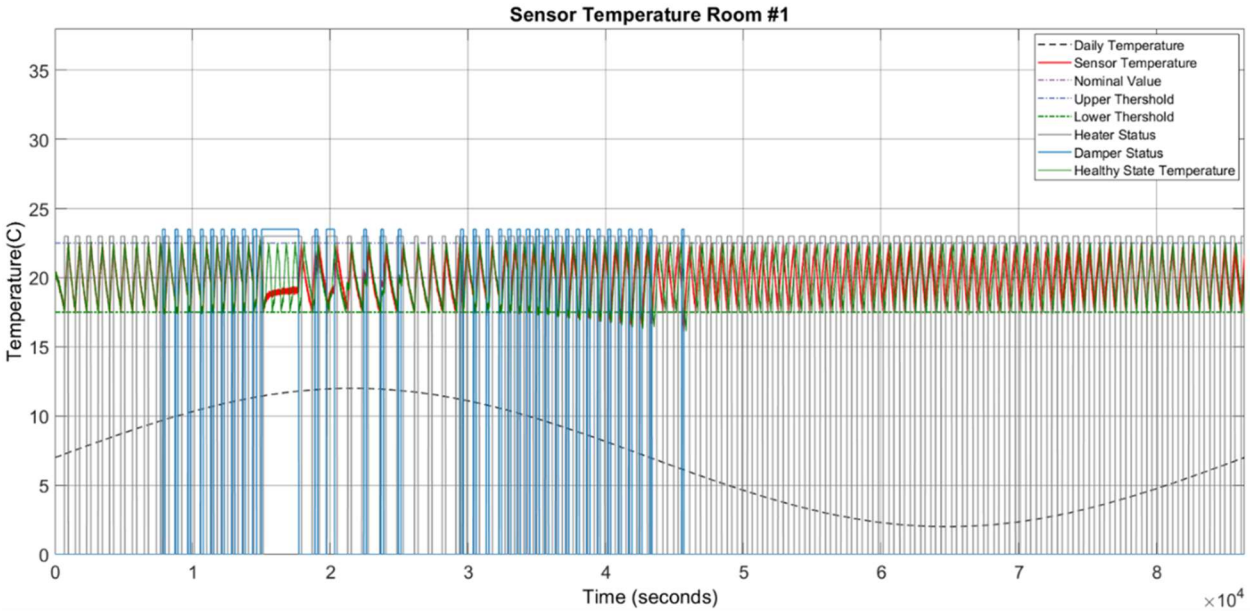

Figure 64. Temperature variations due to the intermittent stuck-at fault of the damper actuator (Scenario 4).

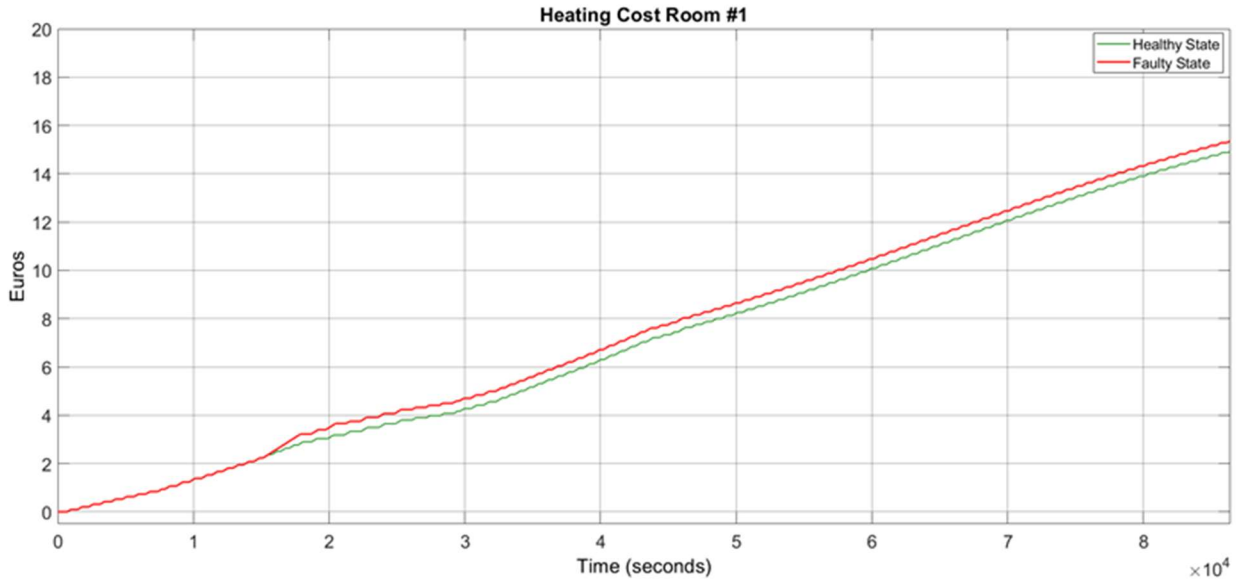

Figure 65. Heating cost due to the damper actuator's intermittent stuck-at fault (Scenario 4).

7.1.5 Scenario 5

Scenario 5 describes a permanent stuck-at-fault for the damper actuator. This fault is injected at 15,000 s. The faulty state endures until the end of the simulation. Since the damper is stuck in an open state, the CO_2 concentration inside the room decreases and reaches a minimum value of 400 ppm, equal to the outside environment's CO_2 concentration. The damper's open status and its effect on the CO_2 concentration are depicted in Figure 66.

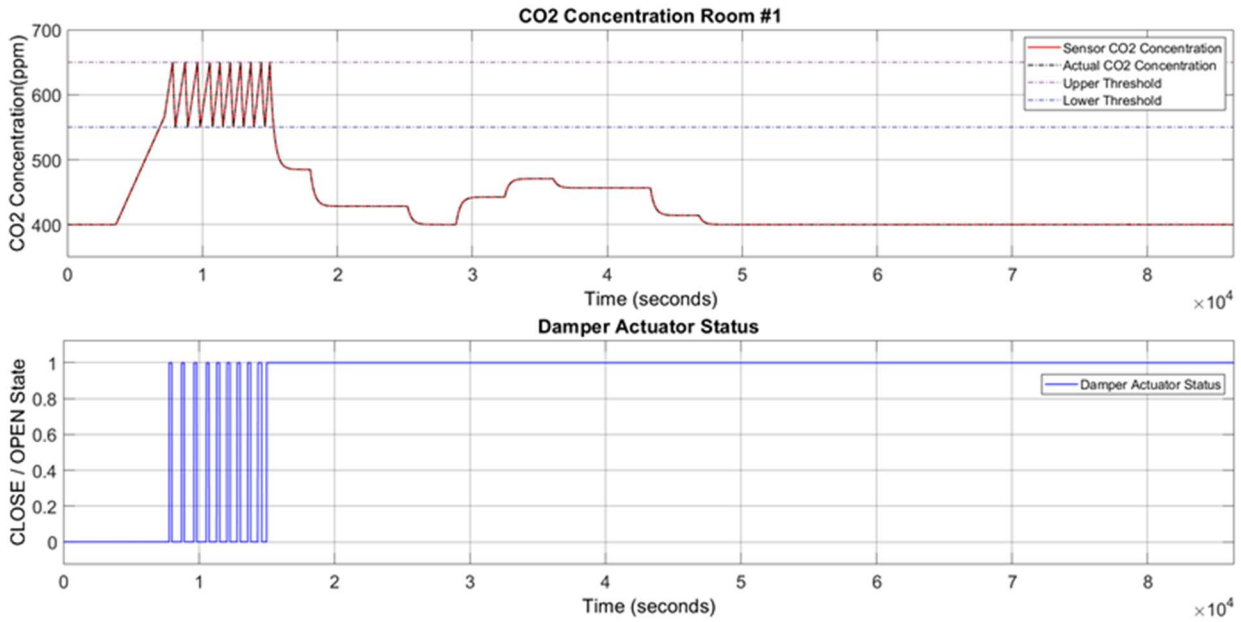

Figure 66. Actual and faulty measurements for a permanent stuck-at damper actuator fault vs. damper actuator state (Scenario 5).

However, the damper's open status also decreases the room temperature, as shown in Figure 67. The heater changes to an ON state after 15,000 s to increase the room temperature. However, the temperature will not stay in the acceptable range as the damper actuator is open. The indoor temperature follows the trend of the temperature of the outside environment. The heater operates in a high-duty cycle, increasing the overall energy consumption. Consequently, the heating cost considerably increases compared to the healthy state operation, as shown in Figure 68.

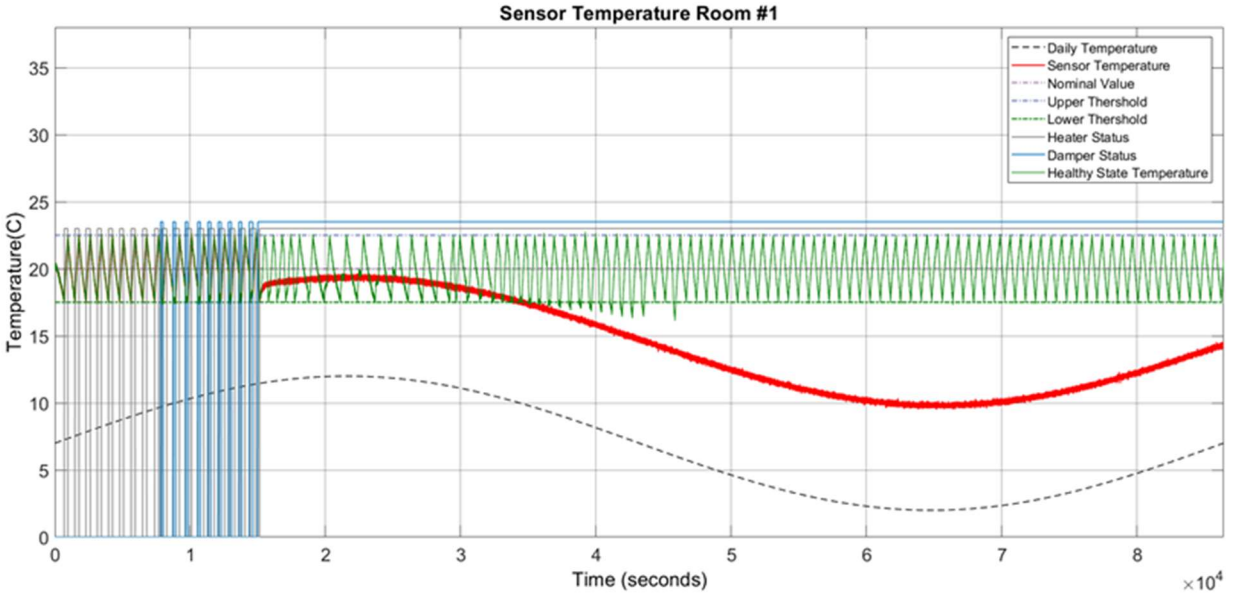

Figure 67. Temperature variations for an intermittent out-of-bound fault with two repetitions in the CO_2 concentration sensor (Scenario 5).

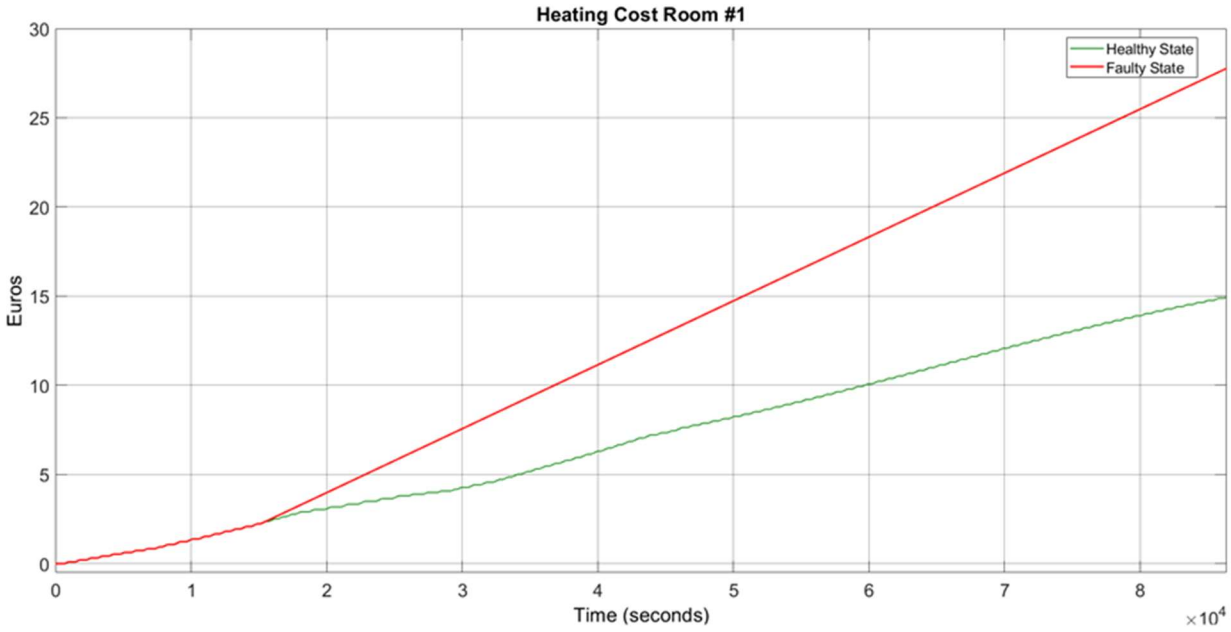

Figure 68. Heating cost for an intermittent out-of-bound fault with two repetitions for the CO_2 concentration sensor (Scenario 5).

7.1.6 Scenario 6

In this scenario, a permanent stuck-at fault is injected into the temperature sensor at 16 °C. This fault is injected at 15,000 s. The faulty state continues for the rest of the execution time until the end of the simulation. The temperature sensor is stuck below the nominal threshold (17.5–22.5 °C), depicted in Figure 69. To increase the inside temperature, the heater should be turned on. However, the damper still functions as intended while the heater is on. Subsequently, the inside temperature of the room increases, as shown in Figure 70.

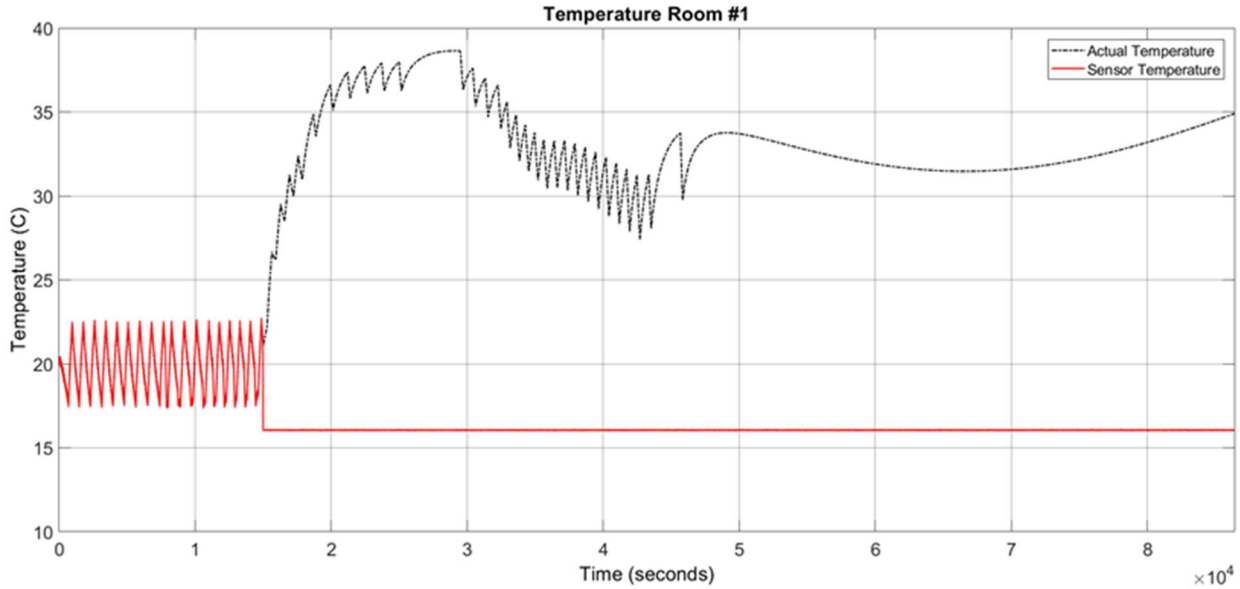

Figure 69. Actual and faulty measurements for a permanent stuck-at fault at 16 °C for the temperature sensor (Scenario 6).

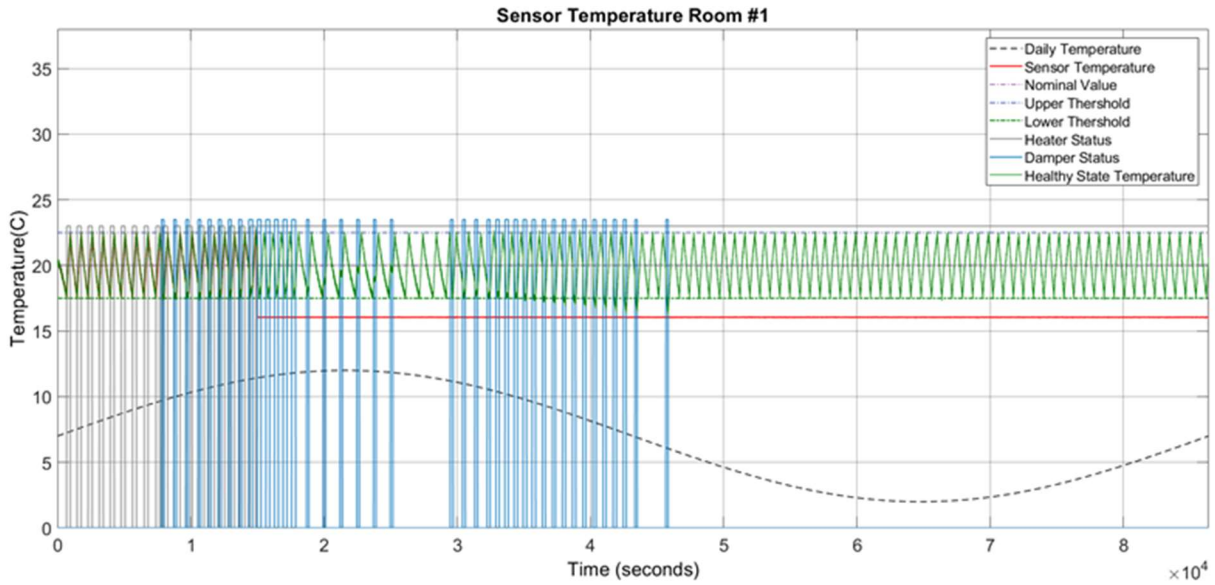

Figure 70. Temperature signal under a permanent stuck-at fault at 16 °C for the temperature sensor (Scenario 6).

Once the fault is injected, the heater is turned on. Therefore, the heater duty cycle and the overall energy consumption increases for the whole fault duration time. Figure 71 shows that the heating cost is considerably increased in comparison with the healthy mode of the system.

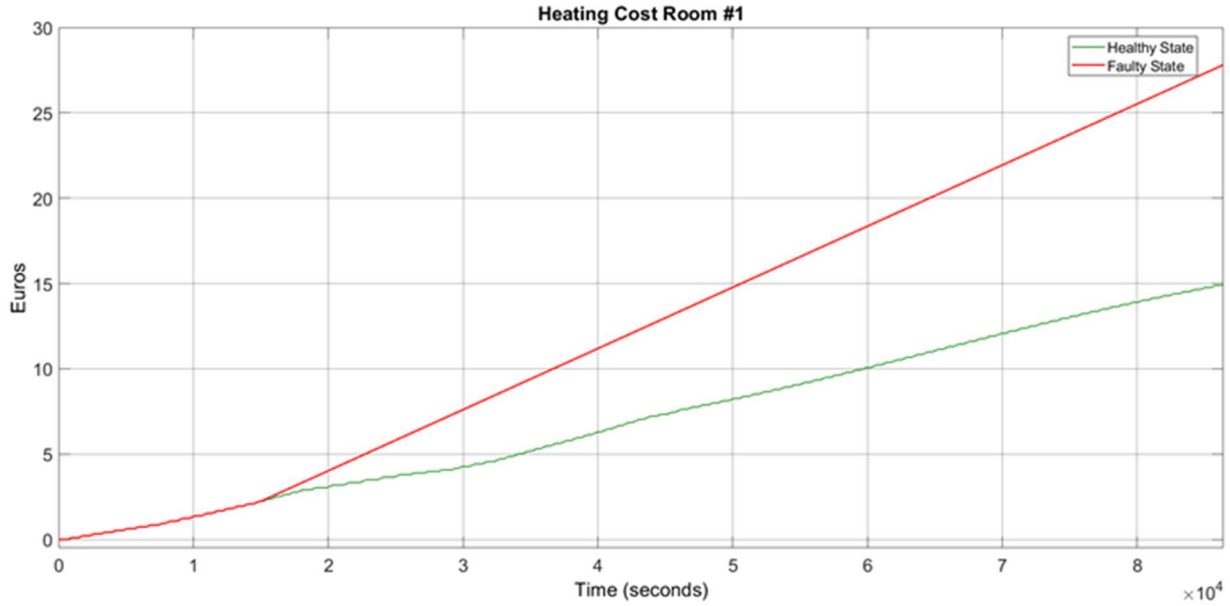

Figure 71. Heating cost for the permanent stuck-at fault at 16 °C for the temperature sensor (Scenario 6).

7.1.7 Scenario 7

Scenario 7 describes a permanent stuck-at fault for the heater actuator. This fault is injected at 15,000 s and continues until the end of the simulation. When the heater actuator is stuck in the ON state after 15,000 s, the temperature inside the room increases. When the damper status opens, the room temperature

decreases, as represented in Figure 72. The ON state of the heater results in a higher-duty cycle and increases the system's energy consumption. Figure 73 shows that the heating cost substantially increases compared to the healthy mode of the system.

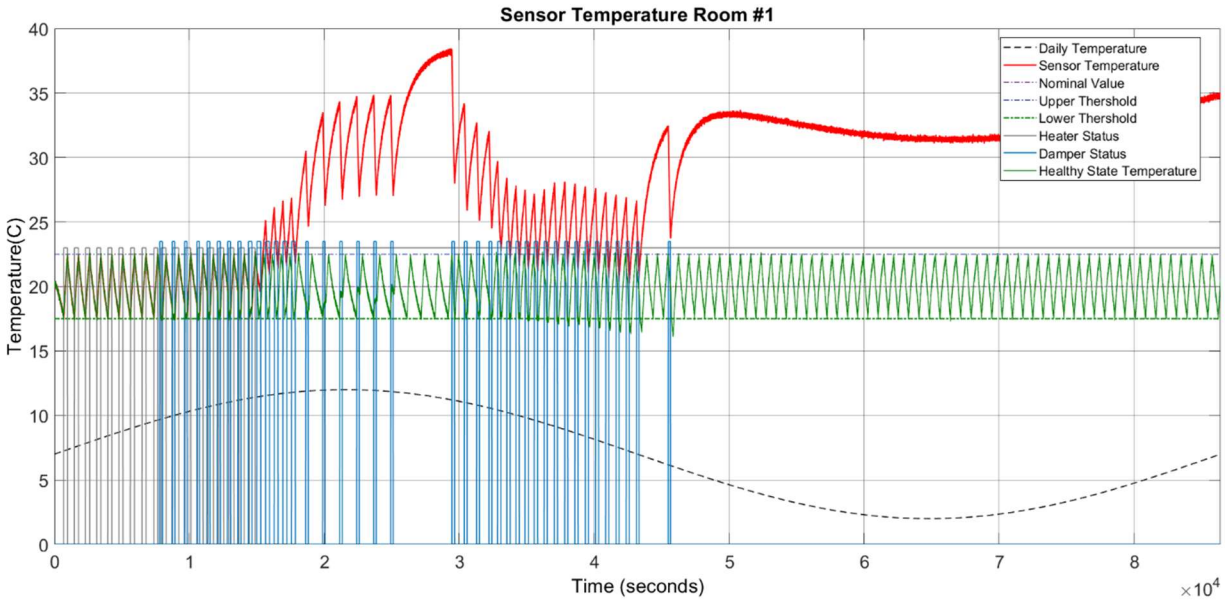

Figure 72. Permanent stuck-at open-status fault of the heater actuator (Scenario 15).

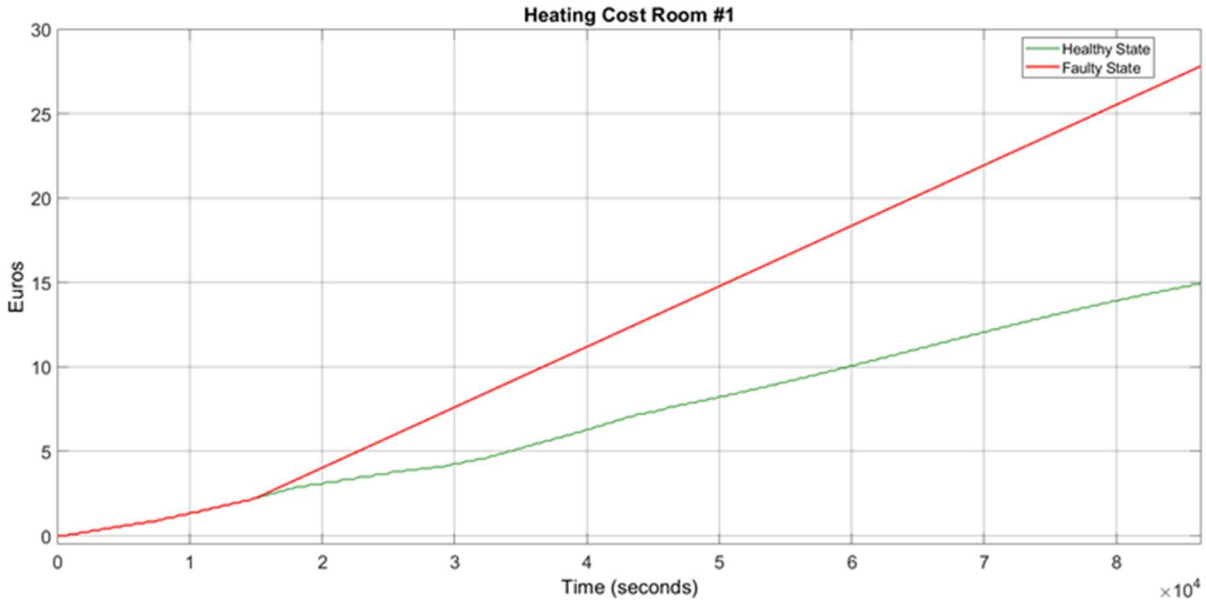

Figure 73. Heating cost due to a permanent stuck-at-open status fault of the heater actuator (Scenario 15).

7.2 Multiple Fault Injection Framework Validation and Results

This thesis introduces a set of scenarios to evaluate the multiple fault injection framework. Each fault scenario was considered as a sample event consisting of other sub-scenarios (or sub-events). Each sub-scenario includes multiple faults (failure repetitions) with different attribute descriptions, e.g., occurrence probabilities and fault type. To illustrate the results, two scenarios for the multiple FI and one case for more than two fault repetitions are shown in the results section. A scenario-based approach is considered to evaluate the multiple fault injection framework. To define the evaluation scenario, a Fault Injection Vector (FIV) including fault case objects, was introduced and is described in Table 31. Each fault object consists of fault case attributes and faulty output data generated by the introduced automated FI algorithm. Each fault case in the considered scenario is an object generated from the "Fault_Object_Generator" class described in Function 11. The evaluation scenario is detailed including the properties such as the number of FI cases, the total number of injections, the number of faulty rooms and the number of faulty components. The faulty rooms denote the destination of the anomalies, and the faulty components define the target components in each room as detailed in Table 32. Fault attributes were assigned randomly by an automatic FI algorithm. The impact shows the consequences of each fault on the system behavior. The effect is depicted and analyzed concerning the change ratio for each subevent and event in Table 32.

Table 31. FIV consisting of fault-case objects.

Fault_Case Obj_1	Fault_Case Obj_2	Fault_Case Obj_3	..	Fault_Case Obj_i	..	Fault_Case Obj_{n-1}	Fault_Case Obj_n

Function 11. Fault object generator class.

Code Description	Explanation
Class Fault Object Generator	
Properties	
Activated_Room_Component_Combination_Matrix.	Activation of the faulty rooms and components, including subevents.
Fault_Injection_Persistence_Matrix.	Assigning the FI persistence for faulty components.
Fault_Injection_Time_Matrix.	Assigning the FI time types.
Fault_Injection_Duration_Matrix.	Assigning the FI duration times.
Fault_Injection_Interarrival_Matrix.	Assigning the FI interarrival times.
Fault_Injection_Type_Matrix.	Assigning the FI types for faulty components.
Fault_Occurrence_Probability_Matrix.	Calculating the FI types for faulty components.
Faulty_SystemOutput.	Storing faulty output for each fault case, including system signals.
Fault_Repetitions.	Assigning the number of repetitions for each subevent.

To evaluate the FI framework, a total number of 14 fault cases, including five scenarios, are defined and described in Table 32. Each scenario comprises variations of sub-scenarios that explain the details of the

fault attributes and their impacts. Each fault occurrence probability value is bounded by the locality of the component, environmental conditions, and occurrence time, resulting in different CO_2 concentrations, temperatures, and energy consumption over time. Moreover, some fault cases and their impacts are described and depicted to show the accuracy and results of the FI procedure. The results show the signal changes of the fault-case scenarios compared with the healthy situation of the system model in which the up arrowhead shows an increased impact, and the down arrowhead shows a decreased impact. For some cases with intermittent faults, it was observed that the signal first increased and then decreased. The column of fault occurrence probability in Table 32 shows the calculated values using Equation 23. The intermittent fault cases were defined with two repetitions for the scenarios.

Table 32. Scenario descriptions for the FI framework in the HVAC system.

Nr.	Scenarios (Events)	Sub Scenarios (Sub Event)	Faulty Room	Faulty Components	Fault Attributes				Impact		
					Fault Persistence	First Fault Type	Second Fault Type	Fault Occurrence Probability	CO2 Concentration Changes (ppm)	Temperature Changes (°C)	Energy Changes
1	1	1	1	Damper actuator	Intermittent	Stuck-at (open)	Stuck-at (closed)	0.6258	↑	–	↑
2		2	4	CO_2 sensor	Permanent	Gain fault	–	0.0225	↑	↓	↑
3	2	1	2	Temperature sensor	Permanent	Out of bounds	–	0.0996	–	↓	↓
4		2	5	Heater actuator	Intermittent	Stuck-at (on)	Stuck-at (off)	0.2586	–	↑↓	↑
5	3	1	4	Damper actuator	Intermittent	Stuck-at (open)	Stuck-at (closed)	0.6258	↓↑	–	↑
6		2	5	Temperature sensor	Permanent	Out Of bounds	–	0.0996	–	↑	↓
7		3	5	Heater actuator	Intermittent	Stuck-at (off)	Stuck-at (on)	0.2586			
8	4	1	4	CO_2 sensor	Permanent	Offset fault	–	0.0225	↑	↑↓	↑
9		2	4	Heater actuator	Intermittent	Stuck-at (on)	Stuck-at (off)	0.25856			
10		3	5	CO_2 sensor	Permanent	Offset fault	–	0.0225	↑	↓	↑
11		4	5	Damper actuator	Intermittent	Stuck-at (open)	Stuck-at (open)	0.6258			
12	5	1	1	CO_2 sensor	Permanent	Stuck-at	–	0.0308	↓	–	↓
13		2	5	Temperature sensor	Permanent	Out of bounds	–	0.6258	↓	↑	↓
14		3	5	Damper actuator	Intermittent	Stuck-at (open)	Stuck-at (closed)	0.0996			

Two scenarios show the thermal and energy consumption changes under fault conditions. One scenario shows multiple FI with more repetitions.

7.2.1 Multiple FI with Multiple Components in One Zone (Two Intermittent Stuck-at Faults in Heater Actuator and one Permanent Offset Fault in CO_2 Sensor)

This FI case describes two component faults triggered at different points in time in one zone. One intermittent fault was activated in the heater actuator, and one permanent offset fault was initiated at the CO_2 sensor. This scenario is related to items 8 and 9 in Table 32. Figure 74 shows the two stuck-at "on" faults and the stuck-at "off" faults in the heater actuator, resulting in changes of the thermal conditions. Figure 75 also depicts the CO_2 conditions, which had a permanent offset for the rest of the execution time as shown in Figure 75. Activating both faults simultaneously in one zone resulted in a reduction of temperature and a change to the "open" status of the damper actuator. With increasing CO_2 concentration, the damper opened to decrease the harmful impact of the CO_2. The open status of the damper actuator decreased the indoor temperature subsequently. Figure 76 illustrates the damper status, which remained open. The open status of the damper could also cause a decrease in the CO_2 concentration.

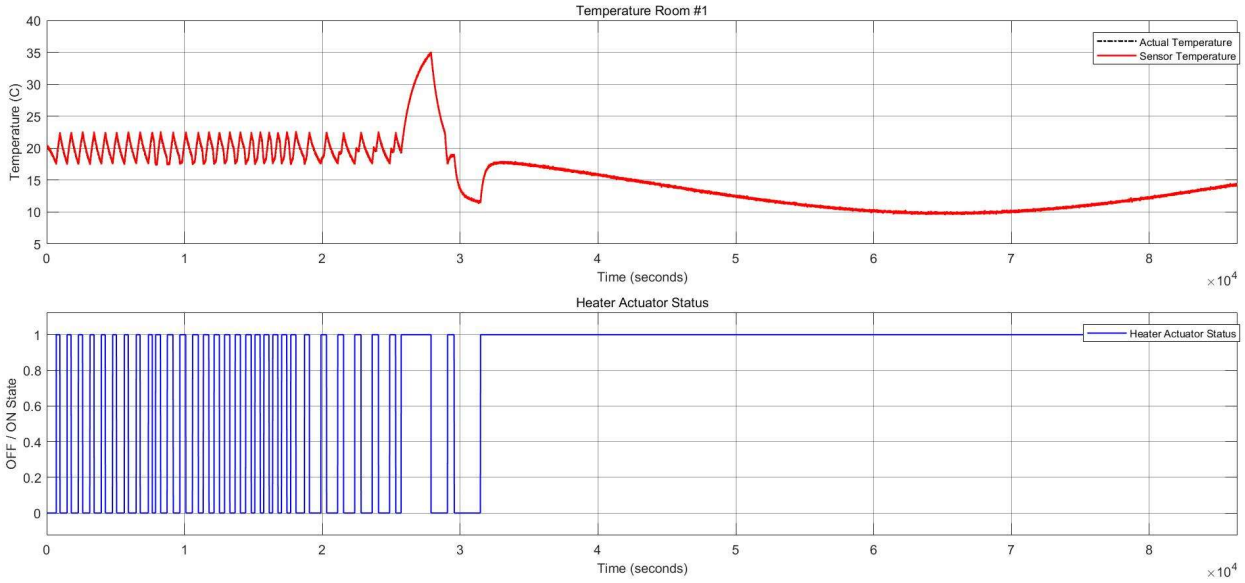

Figure 74. Thermal conditions for the heater actuator and CO_2 sensor faults.

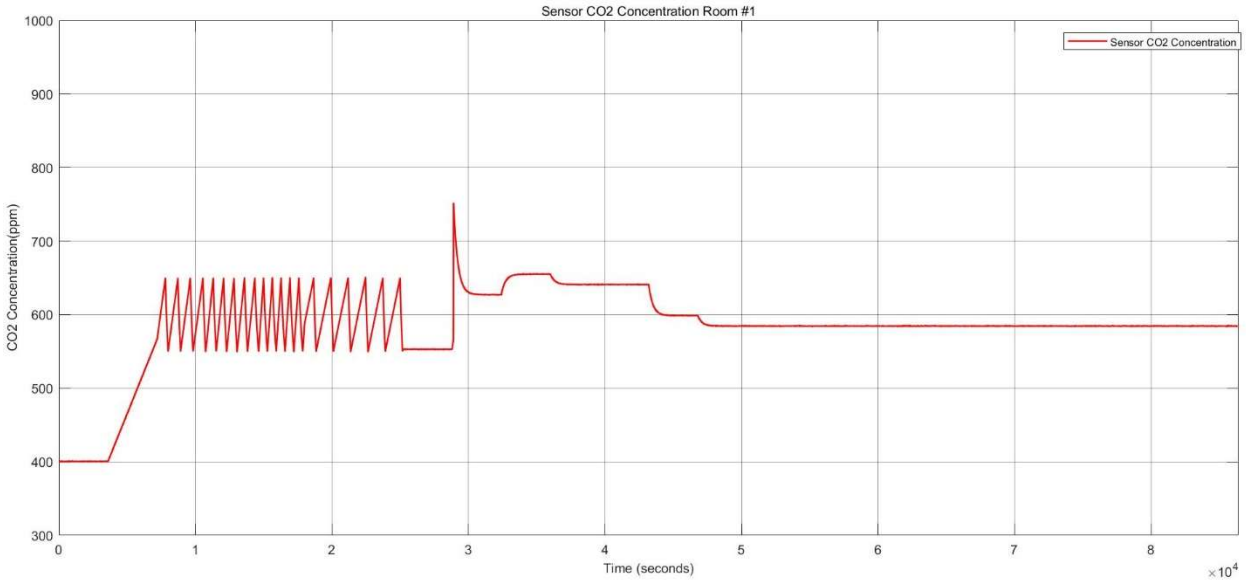

Figure 75. CO_2 concentration for the heater actuator and CO_2 sensor faults.

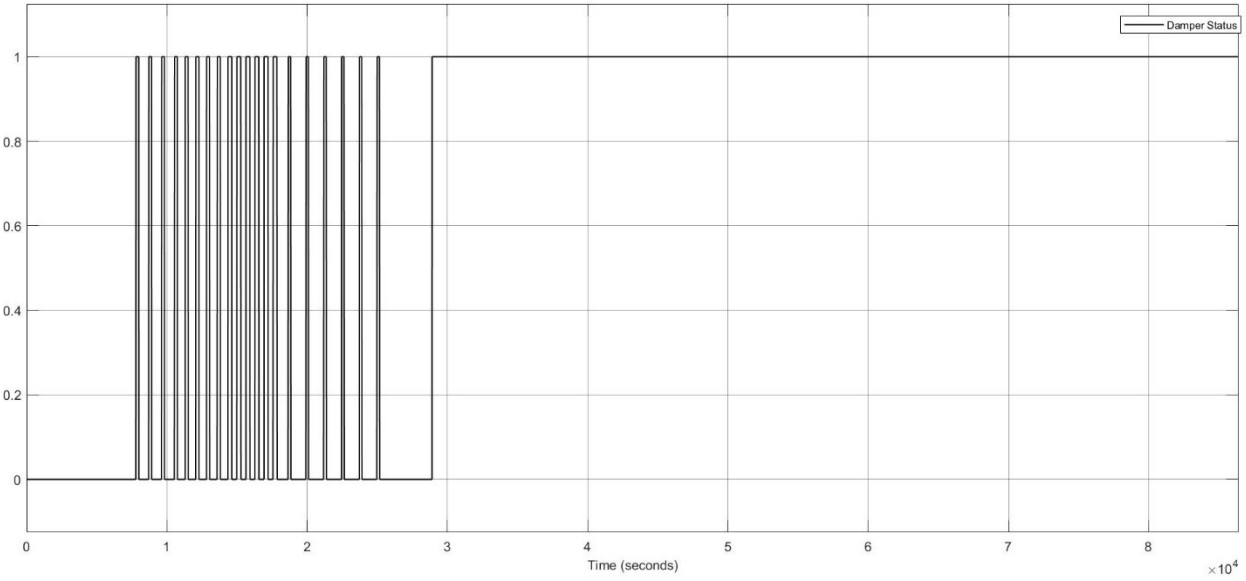

Figure 76. Damper actuator status for the heater actuator and CO_2 sensor faults.

Figure 77 shows the energy consumption condition for this FI case which represents a substantial growth of around 73.34%. Whenever the damper stays in the open status, the heater actuator should remain in "on" to mitigate the thermal consequences and balance the internal temperature.

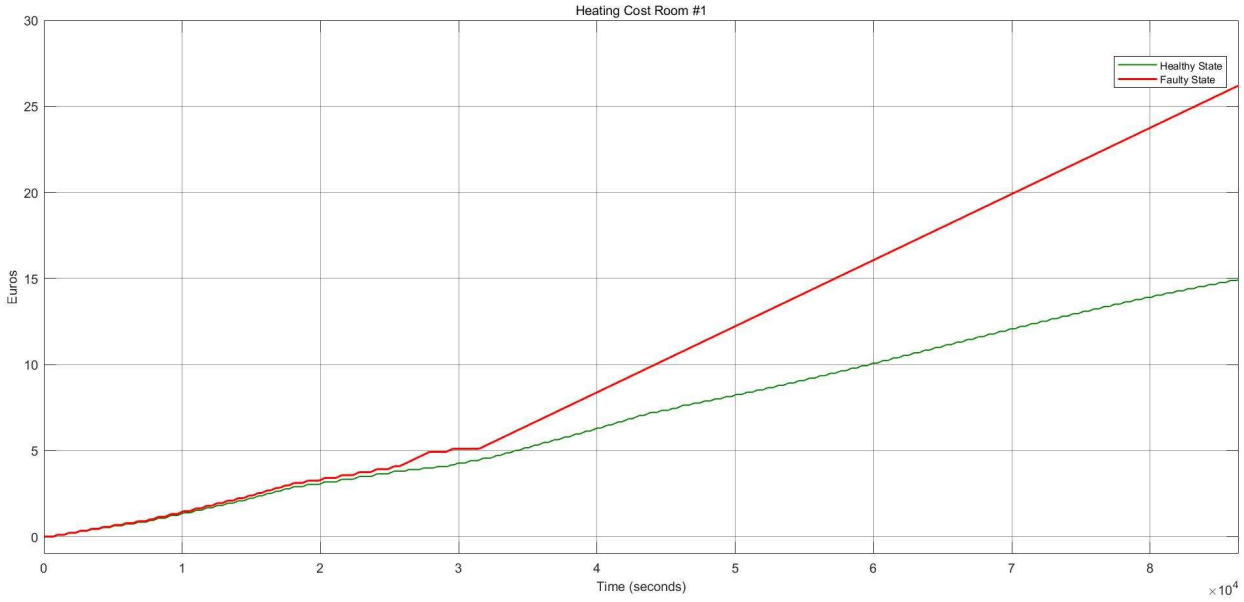

Figure 77. Heating cost for the heater actuator and CO_2 sensor faults.

7.2.2 Multiple FI with Multiple Components in One Zone (Two Intermittent Stuck-at Faults in the Damper Actuator and one Permanent Stuck-at Fault in the Temperature Sensor)

This FI case shows two component faults in the damper actuator and temperature sensor. The damper actuator had two stuck-at "open "and stuck-at "closed "faults, illustrated in Figure 78. This scenario is related to items 13 and 14 in Table 32.

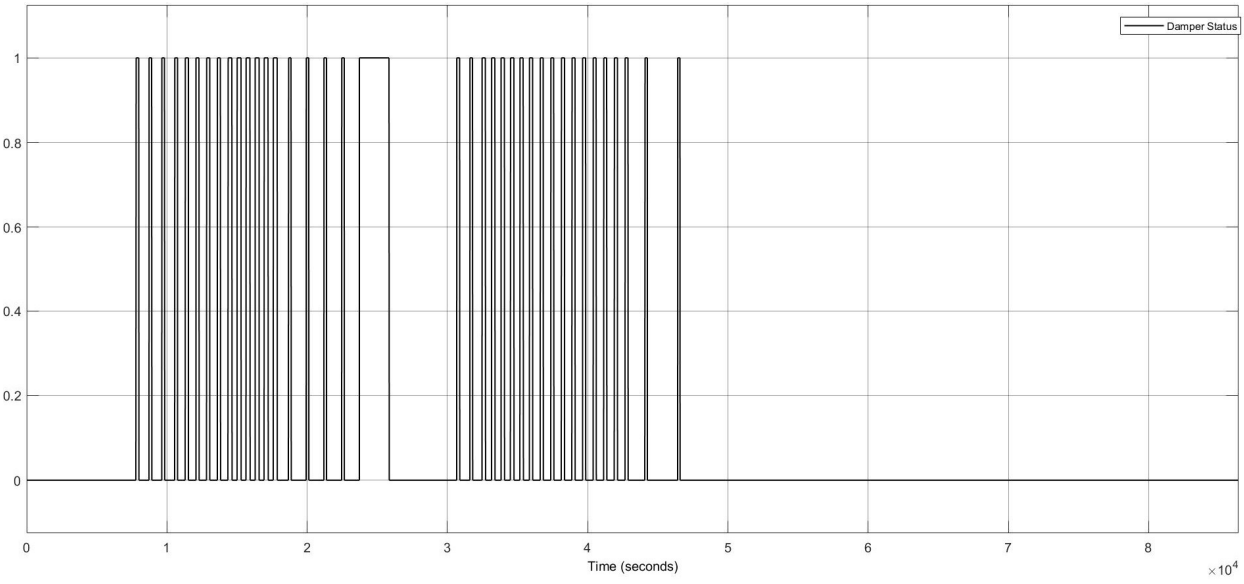

Figure 78. Damper actuator status for the damper actuator and temperature sensor faults.

The damper actuator with a stuck-at "open" fault resulted in the "on" status of the heater actuator and a reduction of the CO_2 concentration, as shown in Figure 79 and Figure 80, respectively.

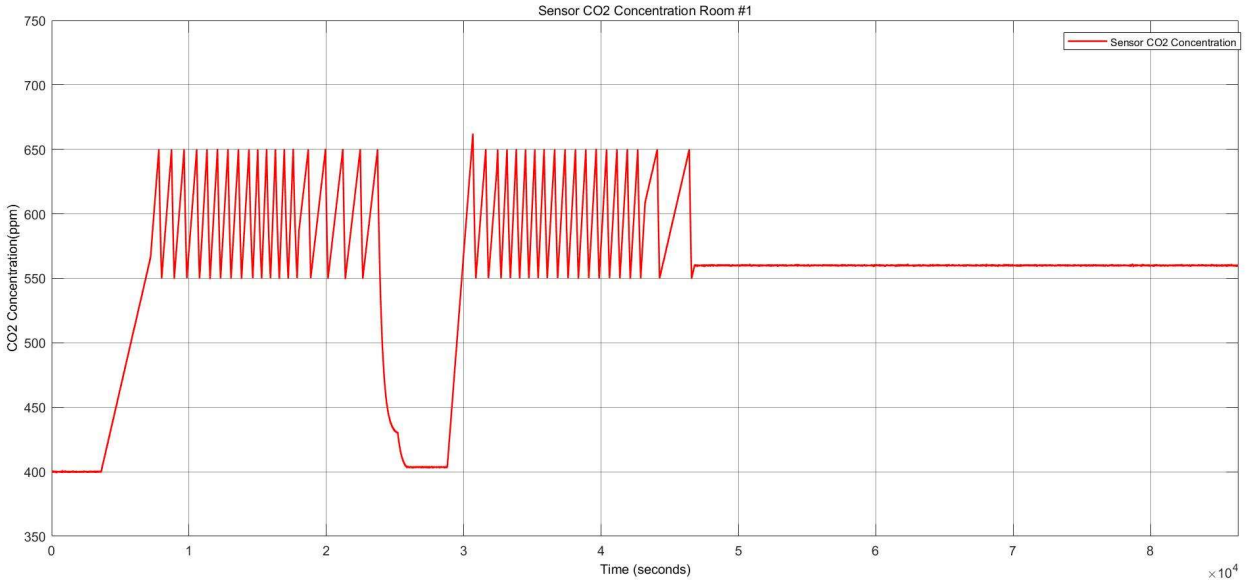

Figure 79. CO_2 concentration for the damper actuator and temperature sensor faults.

The temperature is stuck at 35 °C for the rest of the execution time, resulting in the heater actuator's permanent "off" status. These conditions are shown in Figure 80.

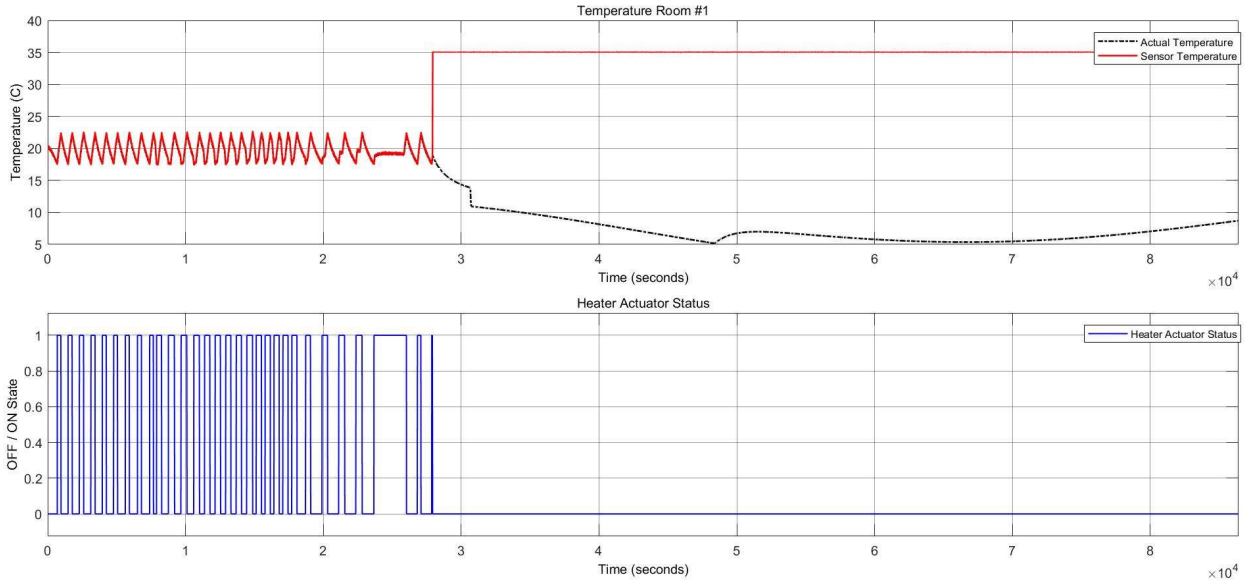

Figure 80. Thermal conditions and heater statuses for the damper actuator and temperature sensor faults.

Once the heater actuator was stuck at "off" status, this caused a remarkable reduction in energy consumption of about 67%, depicted in Figure 81. However, as the temperature decreased, it caused thermal discomfort for the occupants.

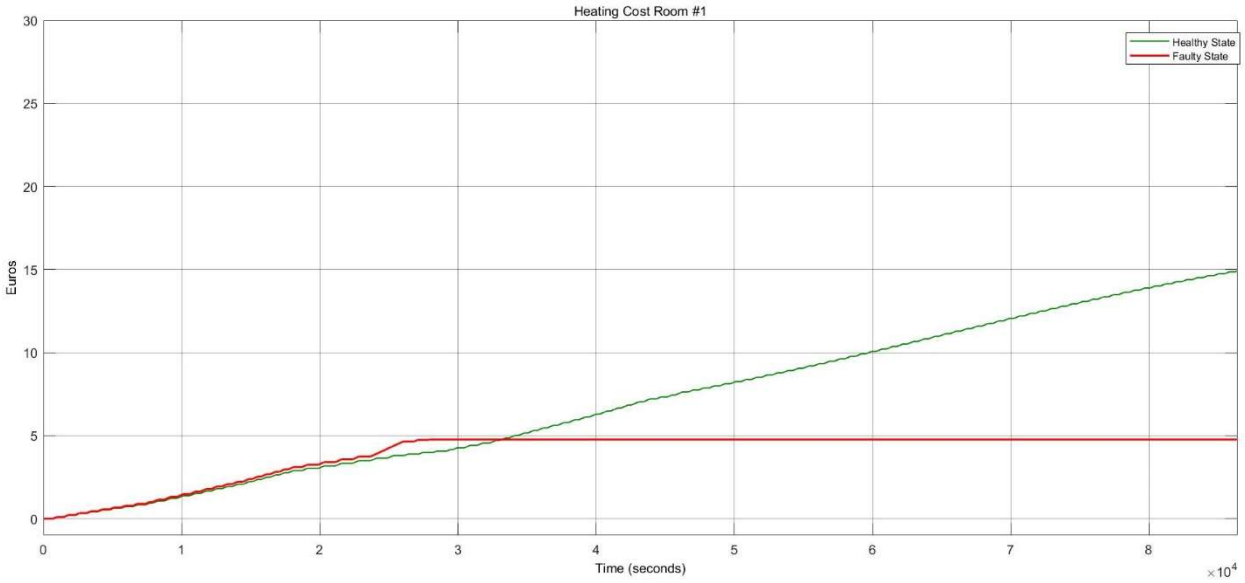

Figure 81. Heating cost variations for the damper actuator and temperature sensor faults.

7.2.3 Multiple Fault Injection in One Component (Intermittent Fault in Heater Actuator with 10 Repetitions)

This example scenario shows the effect of multiple faults in a single component. This intermittent fault was injected into the heater actuator with ten repetitions. Figure 82 shows the heater statuses and temperature sensor behavior. When the heater was stuck at “on” the temperature increased. Whenever the heater was stuck at “off” the temperature decreased, thereby closing the damper actuator. The damper and heater statuses are presented in Figure 83. The number of repetitions can be dynamic and increase according to the system requirements.

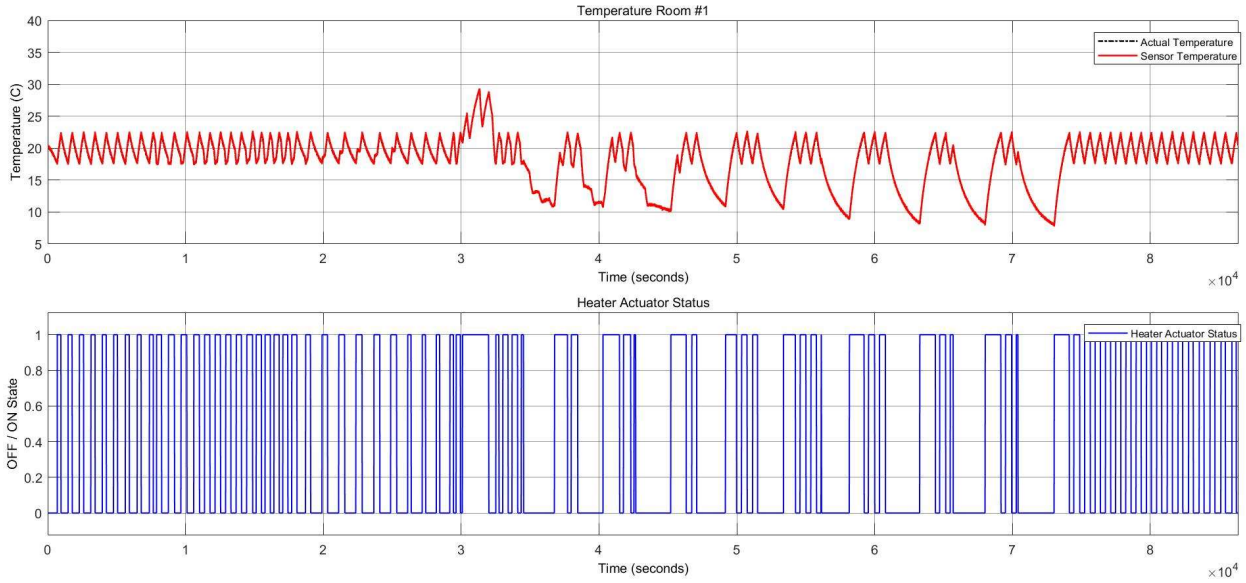

Figure 82. Heater actuator status vs. temperature sensor variations in case of an intermittent fault with ten repetitions in the HVAC system.

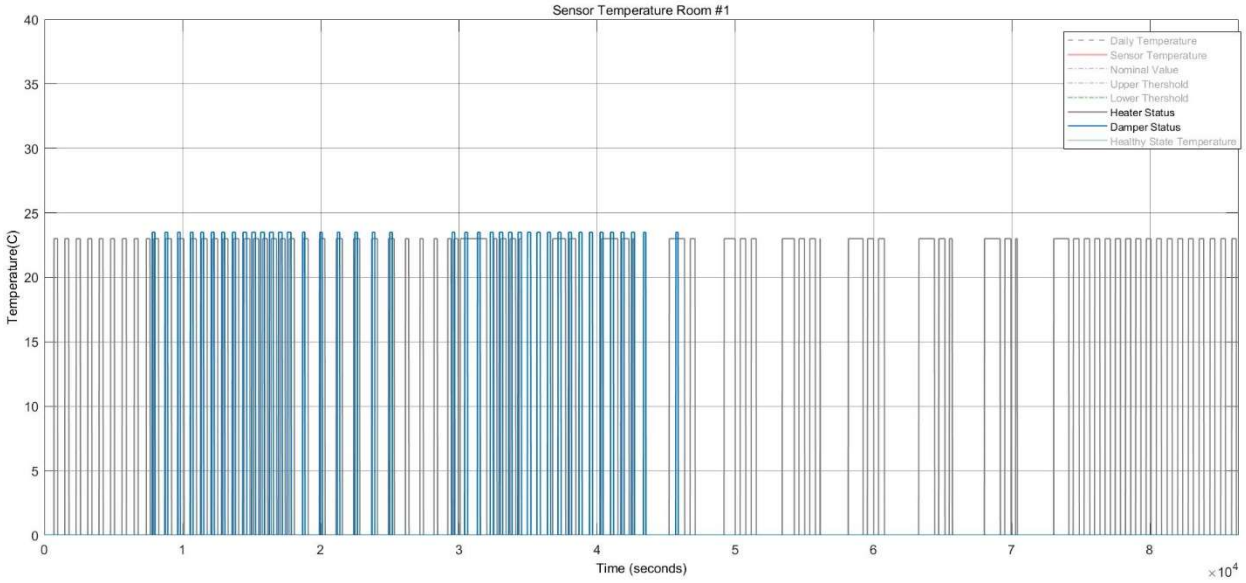

Figure 83. Heater status vs. damper status in the case of an intermittent fault with ten repetitions in the HVAC system.

Figure 84 depicts the costs due to faulty heating during the FI period. The price decreased by around 13% because the heater was stuck at closed status. It gradually decreased when the heater went to the “off” position.

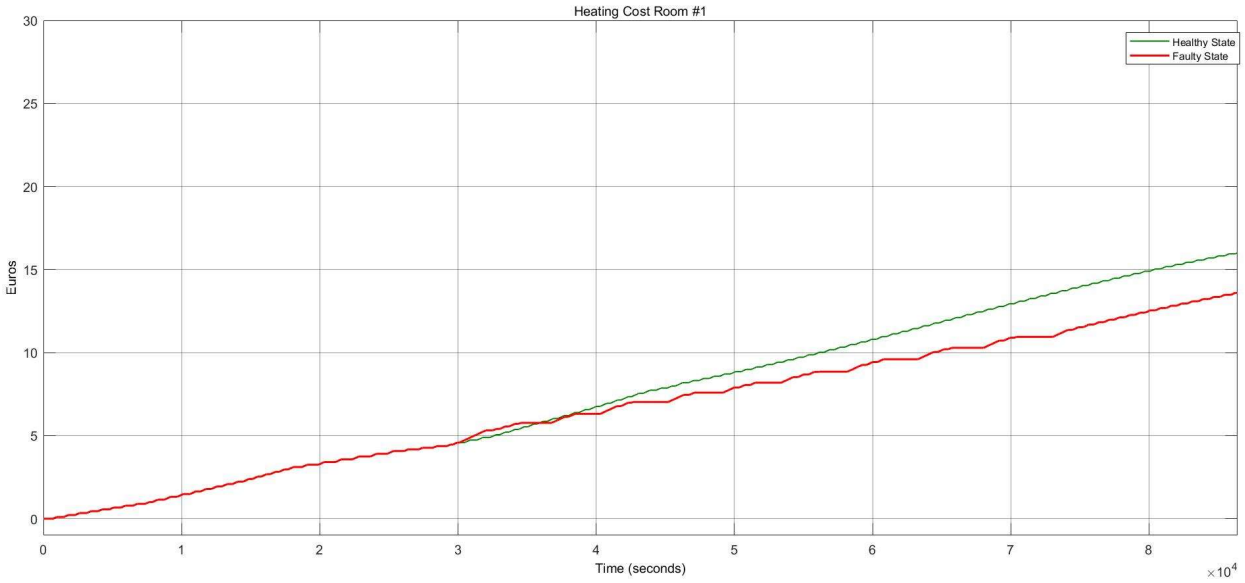

Figure 84. Heating cost variations for healthy and faulty states of the HVAC system in the case of an intermittent fault with ten repetitions.

7.3 Results for Fault Detection and Diagnosis Technique with FBBN

The fault detection and diagnosis technique with FBBN is evaluated for 200 random cases to evaluate the accuracy of the technique and showing how the detection process proceeds for different fault cases which is concluded in "Evaluation results" section. The results are described via tables, including the system properties and the list of diagnosed faults. The top 20 diagnosed fault cases show the offline library's most probable or similar fault cases compared to the actual injected fault. The list of diagnosed faults consists of time, type, and probabilities. Each fault scenario in the diagnosis phase is extracted from RealCase class, including five properties: time, type, value, Precentage_List and Evaluation_List. Each Precentage_List is a vector with 170 elements of percentage values because the offline library consists of 170 fault cases. The Evaluation_List is built based on the Precentage_List to choose the top 20 percentages. The final result of the diagnosis phase is a diagnostic library (DiagnosisLib) with four RealCase object elements. Four number of fault cases are described as follow to illustrate the fault diagnosis steps via evaluation lists and ranking method.

7.3.1 Scenario 1

The first scenario describes the heater actuator's permanent stuck-at-off position with a "0" value, starting at 70393 seconds, described in the first element of "DiagnosisLib" and detailed in Table 33. The Evaluation_List of the DiagnosisLib (1,1) is extracted and listed in Table 34.

Table 33. The first scenario with the diagnosis library for the FBBN diagnosis phase

DiagnosisLib (1,1)		**Description**
Property	**Value**	
Type	"HeaterActuator"	Injected fault type in RealCase class
Time	70393	Injected fault time in RealCase class
Value	0	Injected fault value in RealCase class
Precentage_List	170 x 1 double	A list of percentages for the RealCase class comparing the injected fault with the offline library
Evaluation_List	20 x 3 String	A list of the largest percentages for 20 top cases mapped to the offline library

To diagnose the fault of the first scenario, three properties of the actual fault case are compared with ranks of Evaluation_List and the results show the diagnosis in the first rank. The first rank of Evaluation_List in Table 34 is the most probable and closest fault case compared to an injected fault in Table 33. During the diagnosis phase, only the faulty component and the value are selected randomly. Based on the faulty value, the fault type in the offline library is mapped to the actual injected faults. For example, "HeaterActuatorOff" represents a stuck-at-off position specified by a binary value of 0. Yellow color shows the diagnosed fault rank and greens are the rest of the ranks.

Table 34. First scenario evaluation list for the FBBN diagnosis phase

DiagnosisLib (1,1). Evaluation_List				
No.	**Offline Library Fault Type**	**Offline Library Fault Time**	**Top 20 in Percentage_List**	**Ranking Process**
1	"HeaterActuatorOff"	"70000"	"51.3889"	**Diagnosed in Rank 1**
2	"HeaterActuatorOff"	"65000"	"50"	Rank 2
3	"HeaterActuatorOff"	"60000"	"49.3056"	Rank 3
4	"HeaterActuatorOff"	"75000"	"49.3056"	Rank 4
5	"HeaterActuatorOff"	"55000"	"47.9167"	Other Ranks
6	"HeaterActuatorOff"	"50000"	"45.8333"	Other Ranks
7	"HeaterActuatorOff"	"45000"	"45.1389"	Other Ranks
8	"TemperatureSensorLow"	"70000"	"45.1389"	Other Ranks
9	"HeaterActuatorOff"	"80000"	"45.1389"	Other Ranks
10	"TemperatureSensorLow"	"65000"	"43.75"	Other Ranks
11	"TemperatureSensorHigh"	"70000"	"43.75"	Other Ranks
12	"TemperatureSensorLow"	"75000"	"43.75"	Other Ranks
13	"TemperatureSensorLow"	"60000"	"43.0556"	Other Ranks
14	"TemperatureSensorHigh"	"65000"	"42.3611"	Other Ranks
15	"TemperatureSensorHigh"	"75000"	"42.3611"	Other Ranks
16	"HeaterActuatorOff"	"40000"	"41.6667"	Other Ranks
17	"TemperatureSensorLow"	"55000"	"41.6667"	Other Ranks
18	"TemperatureSensorHigh"	"60000"	"41.6667"	Other Ranks
19	"TemperatureSensorHigh"	"55000"	"40.9722"	Other Ranks
20	"TemperatureSensorLow"	"80000"	"40.9722"	Other Ranks

7.3.2 Scenario 2

For the second scenario, a "TemperatureSensor" permanent stuck-at-value is injected in online mode starting at 78916 seconds. This actual fault is described in the DiagnosisLib (2,1) and detailed in Table 35. The value of 12 ºC is limited to the lower than the thresholds of the fuzzification. The Evaluation_List of the DiagnosisLib (2,1) is extracted and listed in Table 36 showing the first rank diagnosis. To show the most probable fault cases, 20 fault cases are selected and listed, the first rank of the Evaluation_List in Table 36 is the most probable and closest fault case to an injected fault in Table 35.

Table 35. Second scenario with the diagnosis library for the FBBN diagnosis phase

DiagnosisLib (2,1)		Description
Property	**Value**	
Type	"TemperatureSensor"	Injected fault type in RealCase class
Time	78916	Injected fault time in RealCase class
Value	12	Injected fault value in RealCase class
Precentage_List	170 x 1 double	A list of percentages for the RealCase class comparing the injected fault with the offline library
Evaluation_List	20 x 3 String	A list of the largest percentages for 20 top cases mapped to the offline library

Table 36. Evaluation list for the FBBN diagnosis phase in the second scenario

DiagnosisLib (2,1). Evaluation_List				
No.	**Offline Library Fault Type**	**Offline Library Fault Time**	**Top 20 in Percentage_List**	**Ranking Process**
1	"TemperatureSensorLow"	"80000"	"50"	**Diagnosed in Rank 1**
2	"TemperatureSensorLow"	"75000"	"48.6111"	Rank 2
3	"TemperatureSensorLow"	"70000"	"47.2222"	Rank 3
4	"TemperatureSensorLow"	"85000"	"46.5278"	Rank 4
5	"TemperatureSensorLow"	"60000"	"45.8333"	Other Ranks
6	"TemperatureSensorLow"	"65000"	"45.8333"	Other Ranks
7	"DamperActuatorOff"	"75000"	"45.8333"	Other Ranks
8	"TemperatureSensorMiddle"	"75000"	"45.8333"	Other Ranks
9	"HeaterActuatorOn"	"75000"	"45.8333"	Other Ranks
10	"DamperActuatorOff"	"80000"	"45.8333"	Other Ranks
11	"TemperatureSensorMiddle"	"80000"	"45.8333"	Other Ranks
12	"HeaterActuatorOn"	"80000"	"45.8333"	Other Ranks
13	"DamperActuatorOff"	"50000"	"45.1389"	Other Ranks
14	"TemperatureSensorMiddle"	"50000"	"45.1389"	Other Ranks
15	"HeaterActuatorOn"	"50000"	"45.1389"	Other Ranks
16	"DamperActuatorOff"	"55000"	"45.1389"	Other Ranks
17	"TemperatureSensorLow"	"55000"	"45.1389"	Other Ranks
18	"TemperatureSensorMiddle"	"55000"	"45.1389"	Other Ranks
19	"HeaterActuatorOn"	"55000"	"45.1389"	Other Ranks
20	"DamperActuatorOff"	"60000"	"45.1389"	Other Ranks

7.3.3 Scenario 3

The third random actual fault is the permanent stuck-at-value for "TemperatureSensor" which occurred in the high ranges of the 30 ºC of fuzzification described in the third element of “DiagnosisLib” and detailed in Table 37. This scenario is differed in the different ranges f occurrence form the previous scenario. This scenario shows the accurate diagnosis for the same component in different fault ranges. The Evaluation_List resulted from the offline library and actual fault (shown in Table 37) is described in Table 38.

Table 37. Third scenario diagnosis library for the FBBN diagnosis phase

DiagnosisLib (3,1)		Description
Property	**Value**	
Type	"TemperatureSensor"	Injected fault type in RealCase class
Time	24063	Injected fault time in RealCase class
Value	30	Injected fault value in RealCase class

Precentage_List	170 x 1 double	A list of percentages for the RealCase class comparing the injected fault with the offline library
Evaluation_List	20 x 3 String	A list of the largest percentages for 20 top cases mapped to the offline library

Table 38. Evaluation list for the FBBN diagnosis phase in the third scenario

DiagnosisLib (3,1). Evaluation_List				
No.	**Offline Library Fault Type**	**Offline Library Fault Time**	**Top 20 in Percentage_List**	**Ranking Process**
1	"TemperatureSensorHigh"	"25000"	"46.5278"	**Diagnosed in Rank 1**
2	"TemperatureSensorHigh"	"30000"	"44.4444"	Rank 2
3	"TemperatureSensorHigh"	"40000"	"44.4444"	Rank 3
4	"TemperatureSensorHigh"	"20000"	"43.75"	Rank 4
5	"TemperatureSensorHigh"	"35000"	"42.3611"	Other Ranks
6	"TemperatureSensorHigh"	"45000"	"42.3611"	Other Ranks
7	"TemperatureSensorHigh"	"55000"	"42.3611"	Other Ranks
8	"TemperatureSensorHigh"	"50000"	"41.6667"	Other Ranks
9	"TemperatureSensorHigh"	"60000"	"41.6667"	Other Ranks
10	"TemperatureSensorHigh"	"65000"	"40.9722"	Other Ranks
11	"TemperatureSensorHigh"	"70000"	"40.2778"	Other Ranks
12	"TemperatureSensorHigh"	"15000"	"38.8889"	Other Ranks
13	"TemperatureSensorHigh"	"75000"	"38.8889"	Other Ranks
14	"TemperatureSensorMiddle"	"20000"	"38.1944"	Other Ranks
15	"TemperatureSensorHigh"	"80000"	"38.1944"	Other Ranks
16	"TemperatureSensorHigh"	"85000"	"36.8056"	Other Ranks
17	"HeaterActuatorOn"	"25000"	"36.1111"	Other Ranks
18	"TemperatureSensorMiddle"	"25000"	"35.4167"	Other Ranks
19	"HeaterActuatorOff"	"25000"	"35.4167"	Other Ranks
20	"HeaterActuatorOn"	"30000"	"35.4167"	Other Ranks

7.3.4 Scenario 4

The fourth scenario describes the "CO_2Sensor" permanent stuck-at-value of 834 ppm which is the fault occurrence in higher threshold of the CO_2 sensor values detailed in Table 39. and Table 40. This scenario shows the accuracy of the diagnosis in the other component of the system with different fault injection attributes. The Table 40 shows the first diagnosis rank with highest probability of mutuality. It means that the fault case of "CO_2SensorHigh" in "85000" time is the most similar fault case with the actual injected fault.

Table 39. Diagnosis library for the FBBN diagnosis phase in the fourth scenario

DiagnosisLib (4,1)		**Description**
Property	**Value**	
Type	"CO_2Sensor"	Injected fault type in RealCase class
Time	83367	Injected fault time in RealCase class
Value	834	Injected fault value in RealCase class
Precentage_List	170 x 1 double	A list of percentages for the RealCase class comparing the injected fault with the offline library
Evaluation_List	20 x 3 String	A list of the largest percentages for 20 top cases mapped to the offline library

To diagnose the fault of the fourth scenario, three properties of the actual fault case are compared with the ranks of Evaluation_List. The results show the diagnosis in the first rank. The first rank of Evaluation_List in Table 40 is the most probable and closest fault case to an injected fault in Table 39.

Table 40. Evaluation list for the FBBN diagnosis phase in the fourth scenario

DiagnosisLib (4,1). Evaluation_List				
No.	**Offline Library Fault Type**	**Offline Library Fault Time**	**Top 20 in Percentage_List**	**Ranking Process**
1	"CO_2SensorHigh"	"85000"	"49.3056"	**Diagnosed in Rank 1**
2	"CO_2SensorHigh"	"80000"	"45.8333"	Rank 2
3	"DamperActuatorOn"	"85000"	"45.1389"	Rank 3
4	"TemperatureSensorLow"	"85000"	"43.75"	Rank 4
5	"TemperatureSensorLow"	"80000"	"41.6667"	Other Ranks
6	"TemperatureSensorMiddle"	"85000"	"41.6667"	Other Ranks
7	"CO_2SensorHigh"	"75000"	"40.9722"	Other Ranks
8	"DamperActuatorOff"	"85000"	"40.9722"	Other Ranks
9	"HeaterActuatorOn"	"85000"	"40.9722"	Other Ranks
10	"CO_2SensorLow"	"85000"	"40.2778"	Other Ranks
11	"HeaterActuatorOff"	"85000"	"40.2778"	Other Ranks
12	"DamperActuatorOff"	"75000"	"39.5833"	Other Ranks
13	"TemperatureSensorMiddle"	"75000"	"39.5833"	Other Ranks
14	"HeaterActuatorOn"	"75000"	"39.5833"	Other Ranks
15	"DamperActuatorOff"	"80000"	"39.5833"	Other Ranks
16	"DamperActuatorOn"	"80000"	"39.5833"	Other Ranks
17	"TemperatureSensorMiddle"	"80000"	"39.5833"	Other Ranks
18	"HeaterActuatorOn"	"80000"	"39.5833"	Other Ranks
19	"HeaterActuatorOn"	"40000"	"38.8889"	Other Ranks
20	"HeaterActuatorOn"	"45000"	"38.8889"	Other Ranks

7.3.5 Evaluation Results

To evaluate the diagnosis technique based on FBBN, the diagnosis algorithm is executed for a number of 200 random actual fault cases. The fault cases are ranked based on their probabilities (or percentages in Evaluation_List). To improve the accuracy, only five groups of ranks are considered for diagnosis: rank 1, rank 2, rank 3, rank 4, and no diagnosis. These ranks can be increased if the designer wants to know the other probable fault ranges. The evaluation results are grouped based on the component type and the ranks of diagnosed faults. All fault cases are injected randomly, and the ranking process shows five cases in the second rank, 1 case in the third rank, and the rest are diagnosed in the first rank, which is a significant and reliable result for the FBBN diagnostic algorithm. The results are shown in the appendix in detail. The results specify the accuracy of the diagnosis for 100% in 200 random actual injections. Based on the ranking method, offline library cases are mapped to the actual fault cases and sorted ascendant to find and select the most probable fault case. These scenarios are selected randomly and show how the diagnosis algorithm functions in different fault cases. Because of selection of the fuzzy theory ranges, the diagnosis algorithm may not be accurate in the ranges between to close fuzzy membership functions (common areas of the range values). For instance, if the fault occurs in the common range of value of the middle and lower values, then the algorithm may diagnose this in both ranges. Therefore, ranking helps to resolve this issue when there is the same probability for two ranges.

8 Discussion and Further Research

Evaluating a system under different faults and anomalies is essential to validate fault-tolerance mechanisms and gain insights into reliability and safety. Simulation-based fault injection provides high observability and controllability of the deliberate insertion of faults and monitoring system behaviors. One advantage of our proposed automated fault injection framework is that it is extendable and compatible with different system models, which must be monitored and evaluated under fault conditions. HVAC systems are an example of such a system consisting of many sensors and actuators, resulting in a complex and error-prone critical infrastructure. The proposed FI framework was evaluated at the system level based on the component failures of the FCRs. The novelty of the proposed FI framework is that the simulator command technique and simulation code modification were merged for realistic fault scenarios, which can be automatically activated for different fault types with varying attributes. A Gaussian probability of sensor accuracy and noise with uniform distribution were modeled to reach realistic uncertainties. To implement the fault injectors, a State-flow diagram was used for the simulation-based fault injection. Numerous scenarios were considered to evaluate the system model with the fault injection framework. Each scenario allowed us to investigate and understand the system's behavior under the respective fault case. The evaluation of the framework showed us the consequences of different fault sets, which were activated for specific components, such as sensors and actuators. For each case of the scenarios in the evaluation section, there is a discussion that explains their fault attributes and parameters. Moreover, the figures represent the impact of the fault injection process on the system's behavior and the signal changes. The faults can also be randomly injected with random repetitions, which can be helpful in evaluating diagnosis techniques. In the example scenarios, we obtained insights into the impact of faults on energy consumption and heating cost. For example, there is a remarkable waste of energy of around 80% in the case of a permanent stuck-at fault in the temperature sensor, which could be avoided using diagnosis and fault tolerance.

HVAC systems in buildings are one of the most important factors for energy consumption. Due to their vulnerabilities and complexities, they have a high potential for multiple fault occurrences in reality. The experimental evaluation of HVAC systems before the operational phase of the system can help designers gain insight into them to design more reliable systems. Simulation-based FI allows the system to be evaluated under various fault conditions, especially in emergencies. Therefore, a fault model for multiple faults in HVAC systems based on field fault occurrence rates from maintenance records was described. Fault attributes were designed based on multi-dimensional matrices to be extensible for any system structure. A simulation-based multiple FI framework for DCV and heating systems was developed according to the defined fault model and implemented in MATLAB/Simulink using Stateflow diagrams. An automatic FI algorithm performed each fault scenario using the defined fault attributes. Different scenarios were defined to evaluate the system’s reliability and quality indicators, such as thermal comfort, CO_2 concentration, energy consumption, and heating cost. Each scenario consisted of other sub-scenarios to activate multiple faults in multiple components and multiple zones. The results for the scenarios show system impacts and changes in different sub-scenarios. For example, one sub-scenario showed a rise in the heating cost and energy consumption of around 70%, and another sub-scenario exhibited a decrease in the energy consumption of around 67% but a significant increase in thermal discomfort due to the low indoor temperature. Eventually, it can be concluded that multiple FI in DCV and heating systems lead to an unexpected insight into the consequences of different fault combinations. Besides, the component-based system model allows the construction of multi-floor buildings with low effort. In this system model,

different components such as rooms, corridors, controllers, and fault injection are composed and integrated. The proposed fault injection component is extendable and based on the fault attribute matrixes. Once the system structure is extended based on the user definition, the fault injection component can be easily adapted to each layout by increasing the matrix dimensions. One example of multiple fault injection is also provided and validated to show the correctness and accuracy of the fault injection component in a large-scale system with numerous components.

A generic and hybrid fault diagnostic algorithm is proposed based on the combination of data-driven and knowledge-driven approaches. Fuzzy theory and Bayesian belief networks are combined to extract the system specifications and rules and for the construction of the Bayesian network. The Bayesian network is constructed based on the correlation of system attributes. To calculate the system attribute dependencies, the algorithm uses mutual information theory to understand the system's casual relationships. The Bayesian belief network supports the fuzzy theory to increase the universality and scalability in systems with numerous system attributes and signals and to find and extract the hidden correlations between complex structures. It avoids high expert effort and expenses such as time, money, and energy. The results are explored by ranking methods. Each fault can be classified in a diagnostic rank. Eventually, the FBBN diagnosis algorithm results are demonstrated in four synthetic and actual scenarios. In each synthetic scenario, a random injected fault is studied for diagnosing the type, time, and value range by ranking the mutual percentages. Each injected fault is diagnosed in the first rank, which shows the high accuracy and precision of the FBBN fault diagnosis algorithm.

As a result, there is a complete and comprehensive generic multiple fault injection framework for complex and large-scale component-based building structures with on-demand structures providing appropriate experimental results. The framework can be used for multiple fault detection and diagnosis techniques in DCV and heating systems. This thesis covers only single-fault diagnostic techniques, which can be extended easily for multiple-fault diagnostic techniques. Mutual information can be extendable for the multivariant system structures, which can be an appropriate and applicable solution for multiple fault diagnostic techniques by grouping the components and fault classes. In previous methods, all signals differ to make an RDP table for each fault occurrence. However, in this method, the same components (e.g., all CO_2 sensors in multiple floors and rooms) are grouped to find the correlations with each existing fault class.

9 Appendix

This chapter thesis fully describes references via detailed lists, including abbreviations, figures, tables, , functions and references.

List of Abbreviations and Acronyms

No.	Full Description	Abbreviations
1	Carbon Dioxide	CO_2
2	Heating, Ventilation, and Air-Conditioning	HVAC
3	European Union	EU
4	Demand Controlled Ventilation	DCV
5	Building Management System	BMS
6	Verification and Validation	V&V
7	Fault Injections	FI
8	Air Handling Unit	AHU
9	Fault Detection and Diagnosis	FDD
10	Carbon Monoxide	CO
11	Automated Fault Detection and Diagnosis	AFDD
12	Automated Single-Fault Injection Framework	ASFIF
13	Automated Multiple-Fault Injection Framework	AMFIF
14	Fault Injection Framework	FIF
15	Fault Injection Vector	FIV
16	Fuzzy theory and Bayesian Belief Network	FBBN
17	Relation-Direction-Probability	RDP
18	Mutual Information	MI
19	Cyber-Physical Systems	CPS
20	Human-Cyber-Physical Systems	HCPS
21	Machine Learning	ML
22	Artificial Intelligent	AI
23	Autonomous Automobile Systems	AAS
24	State Machines	SMs
25	Finite State Machine	FSM
26	Concurrent State Machines	CSMs
27	Hierarchical State Machines	HSMs
28	Indoor Air Quality	IAQ
29	Fault Containment Regions	FCRs
30	Multivariate Mutual Information	MMI
31	Bayesian Network	BN
32	Greenhouse Gas	GHG
33	Model Predictive Control	MPC
34	Artificial Neural Networks	ANN
35	Model-Based Real-Time Optimization	MRTO

36	Quality of Service	QoS
37	Department of Defense	DoD
38	Missing Gate Faults	MGF
39	Automatic Test Pattern Generation	ATPG
40	Advanced Driver Assistance System	ADAS
41	Time-Dependent Dielectric Breakdown	TDDB
42	Rooftop Units	RTU
43	Building Management Systems	BMS
44	Hardware-Based Fault Injection	HaFI
45	Software-Based Fault Injection	SoFI
46	Simulation-Based Fault Injection	SiFI
47	Emulation-Based Fault Injection	EmFI
48	Hybrid Fault Injection	HyFI
49	Advanced Driver Assistance Systems	ADAS
50	Complementary Metal-Oxide-Semiconductor	CMOS
51	Single-Event Upsets	SEUs
52	Multiple-Bit Upsets	MBUs
53	Low-Level Virtual Machine	LLVM
54	Model-Based Diagnosis	MBD
55	Air Conditioning	AC
56	Ground-Coupled Heat Pump	GCH
57	Simulation-Based Fault Injection Framework	ASFIF
58	Key Performance Indicators	KPI
59	Computational Intelligence	CI
60	Intelligent Agents	IG
61	Institute of Electrical and Electronics Engineers	IEEE
62	Reinforcement Learning	RL
63	Markov Decision Process	MDP
64	Deep Deterministic Policy Gradients	DDPGs
65	Non-Intrusive Load Monitoring	NILM
66	Coupled Hidden Markov Models	CHMMs
67	Statistical Process Control	SPC
68	Bayesian Networks	BNs
69	Neural Network	NN
70	Diagnostic-Directed Acyclic Graph	DDAG
71	Signed Directed Graph	SDG
72	Membership Function	MF
73	Membership Degree	MD
74	variable air volume	VAV
75	Build Automation System	BAS
76	Bayesian Belief Network	BBN
77	Dynamic Bayesian Network	DBN
78	Multiple-Sectioned Bayesian Networks	MSBN
79	Air Handling Units	AHUs
80	Fuzzy Bayesian Belief Networks	FBBNs
81	Signed Directed Graph	SDG
82	Genetic Algorithm	GA
83	System-based Clustering Algorithm	ASCA
84	Case-Based Reasoning	CBR

85	Multilogic Probabilistic SDG	MPSDG
86	log-likelihoods	LL
87	Heat transfer	HT
88	Fault Duration	FD
89	Fault Interarrival Time	FIT
90	Mean Time to Failure	MMTF
91	hierarchical state machine	HSM
92	Relational Data Table	RDT
93	Subdomain Label Table	SLT
94	Membership Degrees	MDs
95	Weighted Fuzzy Relational Data Table	WFRDT
96	Subdomain Probability Vector Table	SPV
97	Intersection Triangular Top Matrix	ITTM
98	Subdomains Relation Table	SRT
99	Relation-Direction-Probability	RDP
100	3-Dimensional	3-D
101	2-Dimensional	2-D

List of Figures

List of Tables

List of Functions

List of bibliography

[1] M. W. Ahmad, M. Mourshed, B. Yuce, and Y. Rezgui, "Computational intelligence techniques for HVAC systems: A review," in *Building Simulation*, pp. 359–398.

[2] E. Sala Cardoso, "Advanced energy management strategies for HVAC systems in smart buildings," 2019.

[3] R. Jagpal, "Technical synthesis report Annex 34: computer aided evaluation of HVAC system performance," *Energy Conservation in Buildings and Community Systems Programme (IEA ECBCS), International Energy Agency*, 2006.

[4] M. Basarkar, "Modeling and simulation of HVAC faults in EnergyPlus," 2011.

[5] A. Vishwanath, Y.-H. Hong, and C. Blake, "Experimental evaluation of a data driven cooling optimization framework for HVAC control in commercial buildings," in *Proceedings of the Tenth ACM International Conference on Future Energy Systems*, pp. 78–88.

[6] M. Ostrý, S. Bantová, and K. Struhala, "Compatibility of phase change materials and metals: Experimental evaluation based on the corrosion rate," *Molecules*, vol. 25, no. 12, p. 2823, 2020.

[7] J. Skovajsa, P. Drabek, S. Sehnalek, and M. Zalesak, "Design and experimental evaluation of phase change material based cooling ceiling system," *Applied Thermal Engineering*, vol. 205, p. 118011, 2022.

[8] "Testing, adjusting and balancing in HVAC systems," [Online]. Available: https://www.sobieskiinc.com/blog/testing-adjusting-and-balancing-3-essentials-building-comfort/

[9] "Testing, adjusting, and balancing of HVAC and hydronic systems," [Online]. Available: https://www.gmpsop.com/testing-adjusting-and-balancing-of-hvac-systems/

[10] H. Teraoka, B. Balaji, R. Zhang, A. Nwokafor, B. Narayanaswamy, and Y. Agarwal, *Buildingsherlock: Fault management framework for hvac systems in commercial buildings*: Department of Computer Science and Engineering, University of California …, 2014.

[11] B. Kiamanesh, A. Behravan, and R. Obermaisser, "Realistic Simulation of Sensor/Actuator Faults for a Dependability Evaluation of Demand-Controlled Ventilation and Heating Systems," *Energies*, vol. 15, no. 8, p. 2878, 2022.

[12] L. Lan and Y. Chen, "Application of modeling and simulation in fault detection and diagnosis of HVAC systems," in *Proceedings of Building Simulation*, pp. 1299–1306.

[13] S. Robinson, "Simulation model verification and validation: increasing the users' confidence," in *Proceedings of the 29th conference on Winter simulation*, pp. 53–59.

[14] H. H. Ammar, S. M. Yacoub, and A. Ibrahim, "A fault model for fault injection analysis of dynamic UML specifications," in *Proceedings 12th International Symposium on Software Reliability Engineering*, pp. 74–83.

[15] B. H. Thacker, S. W. Doebling, F. M. Hemez, M. C. Anderson, J. E. Pepin, and E. A. Rodriguez, "Concepts of model verification and validation," 2004.

[16] B. Kiamanesh, A. Behravan, and R. Obermaisser, "Fault Injection with Multiple Fault Patterns for Experimental Evaluation of Demand-Controlled Ventilation and Heating Systems," *Sensors*, vol. 22, no. 21, p. 8180, 2022.

[17] C. van Stiphoudt, F. Stinner, G. Bode, A. Kümpel, and D. Müller, "Fault detection and diagnosis in building energy systems: A tool chain for the automated generation of training data," in *Journal of Physics: Conference Series*, p. 12083.

[18] A. Behravan, B. Kiamanesh, and R. Obermaisser, "Fault Diagnosis of DCV and Heating Systems Based on Causal Relation in Fuzzy Bayesian Belief Networks Using Relation Direction Probabilities," *Energies*, vol. 14, no. 20, p. 6607, 2021.

[19] M. C. Comstock, J. E. Braun, and R. Bernhard, *Development of analysis tools for the evaluation of fault detection and diagnostics in chillers*: Purdue University, 1999.

[20] J. Wen and S. Li, "Tools for evaluating fault detection and diagnostic methods for air-handling units, ASHRAE RP-1312 Final Report," *American Society of Heating, Refrigerating and Air Conditioning Engineers Inc.: Atlanta, GA, USA*, 2011.

[21] A. Behravan, N. Tabassam, O. Al-Najjar, and R. Obermaisser, "Composability modeling for the use case of demand-controlled ventilation and heating system," in *2019 6th International Conference on Control, Decision and Information Technologies (CoDIT)*, pp. 1998–2003.

[22] L. H. Chiang, E. L. Russell, and R. D. Braatz, *Fault detection and diagnosis in industrial systems*: Springer Science & Business Media, 2000.

[23] Z. Gao, C. Cecati, and S. X. Ding, "A survey of fault diagnosis and fault-tolerant techniques—Part I: Fault diagnosis with model-based and signal-based approaches," *IEEE transactions on industrial electronics*, vol. 62, no. 6, pp. 3757–3767, 2015.
[24] A. Abid, M. T. Khan, and J. Iqbal, "A review on fault detection and diagnosis techniques: basics and beyond," *Artificial Intelligence Review*, vol. 54, no. 5, pp. 3639–3664, 2021.
[25] A. Behravan, "Diagnostic classifiers based on fuzzy Bayesian belief networks and deep neural networks for demand-controlled ventilation and heating systems," 2021.
[26] S. R. West, Y. Guo, X. R. Wang, and J. Wall, "Automated fault detection and diagnosis of HVAC subsystems using statistical machine learning," in *12th International Conference of the International Building Performance Simulation Association*, pp. 2659–2665.
[27] S. Wu, *System-level monitoring and diagnosis of building HVAC system*: University of California, Merced, 2013.
[28] A. Rosato, F. Guarino, V. Filomena, S. Sibilio, and L. Maffei, "Experimental calibration and validation of a simulation model for fault detection of HVAC systems and application to a case study," *Energies*, vol. 13, no. 15, p. 3948, 2020.
[29] C. Y. Leong, "Fault detection and diagnosis of air handling unit: A review," in *MATEC Web of Conferences*, p. 6001.
[30] G. Abaei and A. Selamat, "A survey on software fault detection based on different prediction approaches," *Vietnam Journal of Computer Science*, vol. 1, no. 2, pp. 79–95, 2014.
[31] A. R. Abbasi, M. R. Mahmoudi, and Z. Avazzadeh, "Diagnosis and clustering of power transformer winding fault types by cross-correlation and clustering analysis of FRA results," *IET Generation, Transmission & Distribution*, vol. 12, no. 19, pp. 4301–4309, 2018.
[32] A. Behravan, R. Obermaisser, and A. Nasari, "Thermal dynamic modeling and simulation of a heating system for a multi-zone office building equipped with demand controlled ventilation using MATLAB/Simulink," in *2017 International Conference on Circuits, System and Simulation (ICCSS)*, pp. 103–108.
[33] H. Kopetz and R.-T. Sytems, "Design principles for distributed embedded applications," *Real-Time Systems. Springer*, 1997.
[34] J. Zhou, Y. Zhou, B. Wang, and J. Zang, "Human–cyber–physical systems (HCPSs) in the context of new-generation intelligent manufacturing," *Engineering*, vol. 5, no. 4, pp. 624–636, 2019.
[35] W. Baicun, Z. Jiyuan, Q. Xianming, D. Jingchen, and Z. Yanhong, "Research on new-generation intelligent manufacturing based on human-cyber-physical systems," *Strategic Study of Chinese Academy of Engineering*, vol. 20, no. 4, pp. 29–34, 2018.
[36] X. Yao, J. Zhou, Y. Lin, Y. Li, H. Yu, and Y. Liu, "Smart manufacturing based on cyber-physical systems and beyond," *Journal of Intelligent Manufacturing*, vol. 30, no. 8, pp. 2805–2817, 2019.
[37] Q. Li and C. Yao, *Real-time concepts for embedded systems*: CRC press, 2003. [Online]. Available: https://doi.org/10.1201/9781482280821
[38] R. Obermaisser, *Event-triggered and time-triggered control paradigms*: Springer Science & Business Media, 2004.
[39] R. Obermaisser, *Time-triggered communication*: CRC press, 2018.
[40] E. A. Lee and S. A. Seshia, *Introduction to embedded systems: A cyber-physical systems approach*: Mit Press, 2016.
[41] A. Avizienis, J.-C. Laprie, and B. Randell, "Fundamental concepts of dependability," *Department of Computing Science Technical Report Series*, 2001.
[42] J. Kohlas, J. Kohlas, B. Meyer, and A. Schiper, *Dependable systems: software, computing, networks: research results of the DICS program*: Springer Science & Business Media, 2006.
[43] K. S. Trivedi, D. S. Kim, A. Roy, and D. Medhi, "Dependability and security models," in *2009 7th International Workshop on Design of Reliable Communication Networks*, pp. 11–20.
[44] B. Parhami, "From defects to failures: a view of dependable computing," *ACM SIGARCH Computer Architecture News*, vol. 16, no. 4, pp. 157–168, 1988.
[45] J.-C. Laprie and A. Costes, "Dependability: A unifying concept for reliable computing," in *Proceedings of the 12th International Symposium on Fault-Tolerant Computing (FTCS-12)*, pp. 22–24.
[46] *Design Guide for Heating, Ventilating, and Air Conditioning Systems.* [Online]. Available: https://www.usbr.gov/tsc/techreferences/mands/mands-pdfs/HVACManl.pdf (accessed: Nov. 22 2022).
[47] R. McDowall, *Fundamentals of HVAC systems: SI edition*: Academic Press, 2007.
[48] J. W. Mitchell and J. E. Braun, *Principles of heating, ventilation, and air conditioning in buildings*: John Wiley & Sons, 2012.

[49] R. W. H. PE and PE, LEED AP Michael E Myers, *HVAC systems design handbook*: McGraw-Hill Education, 2010.
[50] R. Montgomery and R. McDowall, *Fundamentals of HVAC control systems*: Elsevier, 2008.
[51] *HVAC HVAC Assessment Handbook, Measurements in Mechanical Heating, Ventilation, and Air Conditioning Systems.* [Online]. Available: https://tsi.com/getmedia/ea283b1e-86d9-4078-9fc9-41d5cca9f5f6/5001019C-HVAC-Handbook-2013-A4-web?ext=.pdf (accessed: Nov. 22 2022).
[52] M. łgorzata Steinder and A. S. Sethi, "A survey of fault localization techniques in computer networks," *Science of computer programming*, vol. 53, no. 2, pp. 165–194, 2004.
[53] H. Ziade, R. A. Ayoubi, and R. Velazco, "A survey on fault injection techniques," *Int. Arab J. Inf. Technol.*, vol. 1, no. 2, pp. 171–186, 2004.
[54] J. Arlat, *Validation de la sûreté de fonctionnement par injection de fautes: méthode, mise en oeuvre, application*: Toulouse, INPT, 1990.
[55] F. Guarino, V. Filomena, L. Maffei, S. Sibilio, and A. Rosato, "A Review of Fault Detection and Diagnosis Methodologies for Air-Handling Units," *Global Journal of Energy Technology Research Updates*, vol. 6, pp. 26–40, 2019.
[56] R. B. Ash, *Basic probability theory*: Courier Corporation, 2008.
[57] E. T. Jaynes, *Probability theory: The logic of science*: Cambridge university press, 2003.
[58] "Correlation," [Online]. Available: https://en.wikipedia.org/wiki/Correlation
[59] C. E. Shannon, "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.
[60] *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*: IEEE.
[61] N. Veyrat-Charvillon and F.-X. Standaert, "Mutual information analysis: how, when and why?," in *Cryptographic Hardware and Embedded Systems-CHES 2009: 11th International Workshop Lausanne, Switzerland, September 6-9, 2009 Proceedings*, pp. 429–443.
[62] R. Intan and O. Y. Yuliana, "Fuzzy bayesian belief network for analyzing medical track record," in *Advances in Intelligent Information and Database Systems*: Springer, 2010, pp. 279–290.
[63] "Mutul Information," [Online]. Available: https://en.wikipedia.org/wiki/Mutual_information
[64] S. Srinivasa, "A review on multivariate mutual information," *Univ. of Notre Dame, Notre Dame, Indiana*, vol. 2, no. 1, 2005.
[65] L. Batina, B. Gierlichs, E. Prouff, M. Rivain, F.-X. Standaert, and N. Veyrat-Charvillon, "Mutual information analysis: a comprehensive study," *Journal of Cryptology*, vol. 24, no. 2, pp. 269–291, 2011.
[66] D. François, V. Wertz, and M. Verleysen, "The permutation test for feature selection by mutual information," in *ESANN*, pp. 239–244.
[67] J. R. Vergara and P. A. Estévez, "A review of feature selection methods based on mutual information," *Neural computing and applications*, vol. 24, no. 1, pp. 175–186, 2014.
[68] T. Ekwevugbe, N. Brown, and V. Pakka, "Realt-time building occupancy sensing for supporting demand driven hvac operations," 2013.
[69] G. Doquire and M. Verleysen, "Mutual information-based feature selection for multilabel classification," *Neurocomputing*, vol. 122, pp. 148–155, 2013.
[70] H. Liu, J. Sun, L. Liu, and H. Zhang, "Feature selection with dynamic mutual information," *Pattern Recognition*, vol. 42, no. 7, pp. 1330–1339, 2009.
[71] G. Zeng, "A unified definition of mutual information with applications in machine learning," *Mathematical Problems in Engineering*, vol. 2015, 2015.
[72] X. Qi, X. Fan, Y. Gao, and Y. Liu, "Learning Bayesian network structures using weakest mutual-information-first strategy," *International Journal of Approximate Reasoning*, vol. 114, pp. 84–98, 2019.
[73] M. Kubkowski, J. Mielniczuk, and P. Teisseyre, "How to Gain on Power: Novel Conditional Independence Tests Based on Short Expansion of Conditional Mutual Information," *J. Mach. Learn. Res.*, vol. 22, 62-1, 2021.
[74] R. He and P. A. Narayana, "Global optimization of mutual information: application to three-dimensional retrospective registration of magnetic resonance images," *Computerized medical imaging and graphics*, vol. 26, no. 4, pp. 277–292, 2002.
[75] U. Lerner, R. Parr, D. Koller, and G. Biswas, "Bayesian fault detection and diagnosis in dynamic systems," in *Aaai/iaai*, pp. 531–537.
[76] K. B. Korb and A. E. Nicholson, *Bayesian artificial intelligence*: CRC press, 2010.
[77] T. Koski and J. Noble, *Bayesian networks: an introduction*: John Wiley & Sons, 2011.
[78] R. E. Neapolitan, *Learning bayesian networks*: Pearson Prentice Hall Upper Saddle River, 2004.

[79] A. Costa, M. M. Keane, J. I. Torrens, and E. Corry, "Building operation and energy performance: Monitoring, analysis and optimisation toolkit," *Applied energy*, vol. 101, pp. 310–316, 2013.
[80] S. Wang and Z. Ma, "Supervisory and optimal control of building HVAC systems: A review," *Hvac&R Research*, vol. 14, no. 1, pp. 3–32, 2008.
[81] C. Lapusan, R. Balan, O. Hancu, and C. Rad, "Rapid Control Prototyping in the Development of Home Energy Management Systems," in *Applied Mechanics and Materials*, pp. 395–400.
[82] A. Afram and F. Janabi-Sharifi, "Review of modeling methods for HVAC systems," *Applied Thermal Engineering*, vol. 67, 1-2, pp. 507–519, 2014.
[83] R. Z. Homod, "Review on the HVAC system modeling types and the shortcomings of their application," *Journal of Energy*, vol. 2013, 2013.
[84] S. Seyam, "Types of HVAC systems," *HVAC System*, pp. 49–66, 2018.
[85] B. Tashtoush, M. Molhim, and M. Al-Rousan, "Dynamic model of an HVAC system for control analysis," *Energy*, vol. 30, no. 10, pp. 1729–1745, 2005.
[86] N. A. Gershenfeld and N. Gershenfeld, *The nature of mathematical modeling*: Cambridge university press, 1999.
[87] R. Z. Homod, K. S. M. Sahari, H. af Almurib, and F. H. Nagi, "RLF and TS fuzzy model identification of indoor thermal comfort based on PMV/PPD," *Building and Environment*, vol. 49, pp. 141–153, 2012.
[88] A. Thosar, A. Patra, and S. Bhattacharyya, "Feedback linearization based control of a variable air volume air conditioning system for cooling applications," *ISA transactions*, vol. 47, no. 3, pp. 339–349, 2008.
[89] S. Wu and J.-Q. Sun, "A physics-based linear parametric model of room temperature in office buildings," *Building and Environment*, vol. 50, pp. 1–9, 2012.
[90] A. F. Handbook, "Energy Estimating and Modeling Methods; SI edn," *American Society of Heating, Refrigerating, and Air-conditioning Engineers, Atlanta, GA*, 2009.
[91] S. Soyguder and H. Alli, "Predicting of fan speed for energy saving in HVAC system based on adaptive network based fuzzy inference system," *Expert Systems with Applications*, vol. 36, no. 4, pp. 8631–8638, 2009.
[92] G. Huang, "Model predictive control of VAV zone thermal systems concerning bi-linearity and gain nonlinearity," *Control engineering practice*, vol. 19, no. 7, pp. 700–710, 2011.
[93] J. Banks, "Introduction to simulation," in *Proceedings of the 31st conference on Winter simulation: Simulation---a bridge to the future-Volume 1*, pp. 7–13.
[94] P. Riederer, "Matlab/Simulink for building and HVAC simulation-State of the art," in *Ninth International IBPSA Conference*, pp. 1019–1026.
[95] "Mathworks," [Online]. Available: https://de.mathworks.com/
[96] S. Karmacharya, G. Putrus, C. Underwood, and K. Mahkamov, "Thermal modelling of the building and its HVAC system using Matlab/Simulink," in *2012 2nd International Symposium On Environment Friendly Energies And Applications*, pp. 202–206.
[97] M. M. Gouda, C. P. Underwood, and S. Danaher, "Modelling the robustness properties of HVAC plant under feedback control," *Building Services Engineering Research and Technology*, vol. 24, no. 4, pp. 271–280, 2003.
[98] M. Kassas, "Modeling and simulation of residential HVAC systems energy consumption," *Procedia computer science*, vol. 52, pp. 754–763, 2015.
[99] H. S. Asad, R. K. K. Yuen, J. Liu, and J. Wang, "Adaptive modeling for reliability in optimal control of complex HVAC systems," in *Building Simulation*, pp. 1095–1106.
[100] S. Kasputis and H. C. Ng, "Composable simulations," in *2000 Winter Simulation Conference Proceedings (Cat. No. 00CH37165)*, pp. 1577–1584.
[101] E. W. Weisel, *Models, composability, and validity*: Old Dominion University, 2004.
[102] D. Siegele, E. Leonardi, and F. Ochs, Eds., *A new MATLAB Simulink Toolbox for Dynamic Building Simulation with BIM and Hardware in the Loop compatibility*, 2019.
[103] P. Zhang, *Advanced industrial control technology*: William Andrew, 2010.
[104] D. F. Blumberg, "Advanced diagnostics and artificial intelligence," in *Clinical engineering handbook*: Elsevier, 2004, pp. 464–475.
[105] J. Duato, S. Yalamanchili, and L. Ni, "Chapter 6–Fault-Tolerant Routing," *Interconnection Networks, An Engineering Approach*, pp. 287–357, 2003.
[106] S. Ghosh and T. J. Chakraborty, "On behavior fault modeling for digital designs," *Journal of Electronic Testing*, vol. 2, no. 2, pp. 135–151, 1991.
[107] J. Delange and P. Feiler, "Architecture fault modeling with the AADL error-model annex," in *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications*, pp. 361–368.

[108] P. H. Feiler, C. B. Weinstock, J. B. Goodenough, J. Delange, A. Z. Klein, and N. Ernst, "Improving Quality Using Architecture Fault Analysis with Confidence Arguments," CARNEGIE-MELLON UNIV PITTSBURGH PA PITTSBURGH.

[109] I. Polian, T. Fiehn, B. Becker, and J. P. Hayes, "A family of logical fault models for reversible circuits," in *14th Asian Test Symposium (ATS'05)*, pp. 422–427.

[110] A. Joshi and M. P. E. Heimdahl, "Behavioral fault modeling for model-based safety analysis," in *10th IEEE High Assurance Systems Engineering Symposium (HASE'07)*, pp. 199–208.

[111] J. C. Da Silva, A. Saxena, E. Balaban, and K. Goebel, "A knowledge-based system approach for sensor fault modeling, detection and mitigation," *Expert Systems with Applications*, vol. 39, no. 12, pp. 10977–10989, 2012.

[112] H. Najeh, M. P. Singh, S. Ploix, K. Chabir, and M. N. Abdelkrim, "Diagnosis in buildings: New trends illustrated by an application," in *2019 International Conference on Control, Automation and Diagnosis (ICCAD)*, pp. 1–6.

[113] M. Maleki and B. Sangchoolie, "Simulation-based fault injection in advanced driver assistance systems modelled in sumo," in *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S)*, pp. 70–71.

[114] L. Song, J. Cai, and G. Li, "Research on simulation-based testability verification method of radar," in *Proceedings of the IEEE 2012 Prognostics and System Health Management Conference (PHM-2012 Beijing)*, pp. 1–5.

[115] D. Gil-Tomás, J. Gracia-Morán, J.-C. Baraza-Calvo, L.-J. Saiz-Adalid, and P.-J. Gil-Vicente, "Injecting intermittent faults for the dependability assessment of a fault-tolerant microcomputer system," *IEEE Transactions on Reliability*, vol. 65, no. 2, pp. 648–661, 2015.

[116] A. Behravan, A. Mallak, R. Obermaisser, D. H. Basavegowda, C. Weber, and M. Fathi, "Fault injection framework for fault diagnosis based on machine learning in heating and demand-controlled ventilation systems," in *2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI)*, pp. 273–279.

[117] A. Behravan, R. Obermaisser, and M. Abboush, "Fault Injection Framework for Demand-Controlled Ventilation and Heating Systems Based on Wireless Sensor and Actuator Networks," in *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pp. 525–531.

[118] A. Behravan, R. Obermaisser, D. H. Basavegowda, and S. Meckel, "Automatic model-based fault detection and diagnosis using diagnostic directed acyclic graph for a demand-controlled ventilation and heating system in Simulink," in *2018 Annual IEEE International Systems Conference (SysCon)*, pp. 1–7.

[119] A. Bondavalli, S. Chiaradonna, F. Di Giandomenico, and F. Grandoni, "Threshold-based mechanisms to discriminate transient from intermittent faults," *IEEE Transactions on Computers*, vol. 49, no. 3, pp. 230–245, 2000.

[120] W. S. Ahmad, S. Perinpanayagam, I. Jennions, and S. Khan, "Study on intermittent faults and electrical continuity," *Procedia Cirp*, vol. 22, pp. 71–75, 2014.

[121] C. Constantinescu, "Intermittent faults and effects on reliability of integrated circuits," in *2008 Annual Reliability and Maintainability Symposium*, pp. 370–374.

[122] L. Rashid, K. Pattabiraman, and S. Gopalakrishnan, "Intermittent hardware errors recovery: Modeling and evaluation," in *2012 Ninth International Conference on Quantitative Evaluation of Systems*, pp. 220–229.

[123] W. Chao, F. Zhongchuan, C. Hongsong, and C. Gang, "FSFI: A full system simulator-based fault injection tool," in *2011 First International Conference on Instrumentation, Measurement, Computer, Communication and Control*, pp. 326–329.

[124] W. A. Syed, S. Khan, P. Phillips, and S. Perinpanayagam, "Intermittent fault finding strategies," *Procedia Cirp*, vol. 11, pp. 74–79, 2013.

[125] L. V. Kirkland, *When should intermittent failure detection routines be part of the legacy re-host TPS?*: IEEE, 2011.

[126] N. Torabi, H. B. Gunay, and W. O'Brien, "A review of common human errors in design, installation, and operation of multiple-zone VAV AHU systems," in *Journal of Physics: Conference Series*, p. 12130.

[127] S. M. Frank, J. Kim, J. Cai, and J. E. Braun, "Common Faults and Their Prioritization in Small Commercial Buildings: February 2017-December 2017," National Renewable Energy Lab.(NREL), Golden, CO (United States).

[128] Y. Yan, P. B. Luh, and K. R. Pattipati, "Fault diagnosis of HVAC air-handling systems considering fault propagation impacts among components," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 2, pp. 705–717, 2017.

[129] Y. Chen, G. Lin, E. Crowe, and J. Granderson, "Development of a unified taxonomy for hvac system faults," *Energies*, vol. 14, no. 17, p. 5581, 2021.

[130] "Non-Directional Intermittent Ground Fault Protection," [Online]. Available: https://www.webgreenstation.com /non-directionalintermittent-ground-fault-protection-siprotec-5-siemens-si5034/
[131] P. Haves, "Fault modelling in component-based HVAC simulation," *Proceedings of Building Simulation'97*, 1997.
[132] K. Choi, S. M. Namburu, M. S. Azam, J. Luo, K. R. Pattipati, and A. Patterson-Hine, "Fault diagnosis in HVAC chillers," *IEEE Instrumentation & Measurement Magazine*, vol. 8, no. 3, pp. 24–32, 2005.
[133] A. Sheikh, V. Kamuni, A. Patil, S. Wagh, and N. Singh, "Cyber attack and fault identification of hvac system in building management systems," in *2019 9th International Conference on Power and Energy Systems (ICPES)*, pp. 1–6.
[134] M. Kooli and G. Di Natale, "A survey on simulation-based fault injection tools for complex systems," in *2014 9th IEEE International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS)*, pp. 1–6.
[135] J. Fernández Briones, M. A. de Miguel Cabello, J. P. Silva Gallino, and A. A. Alonso Muñoz, "Analysis of quality dependencies in the composition of software architectures," *84927575*, 2010.
[136] A. Karimi, B. Kiamanesh, F. Zarafshan, and S. A. Al-Haddad, "Markov process modeling for wireless sensor network availability with QOS constraints," in *Applied Mechanics and Materials*, pp. 1054–1058.
[137] N. Song, J. Qin, X. Pan, and Y. Deng, "Fault injection methodology and tools," in *Proceedings of 2011 International Conference on Electronics and Optoelectronics*, V1-47.
[138] J. Arlat *et al.,* "Fault injection for dependability validation: A methodology and some applications," *IEEE Transactions on software engineering*, vol. 16, no. 2, pp. 166–182, 1990.
[139] D. Lee and J. Na, "A novel simulation fault injection method for dependability analysis," *IEEE Design & Test of Computers*, vol. 26, no. 6, pp. 50–61, 2009.
[140] M. Eslami, B. Ghavami, M. Raji, and A. Mahani, "A survey on fault injection methods of digital integrated circuits," *Integration*, vol. 71, pp. 154–163, 2020.
[141] Y. S. Jeong, S. M. Lee, and S. E. Lee, "A Survey of fault-injection methodologies for soft error rate modeling in systems-on-chips," *Bulletin of Electrical Engineering and Informatics*, vol. 5, no. 2, pp. 169–177, 2016.
[142] R. K. Lenka, S. Padhi, and K. M. Nayak, "Fault Injection Techniques-A Brief Review," in *2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*, pp. 832–837.
[143] M.-C. Hsueh, T. K. Tsai, and R. K. Iyer, "Fault injection techniques and tools," *Computer*, vol. 30, no. 4, pp. 75–82, 1997.
[144] C. Evangeline and N. M. Sivamangai, "Evaluation of testability of digital circuits by fault injection technique," in *2015 2nd International Conference on Electronics and Communication Systems (ICECS)*, pp. 92–96.
[145] S. Salih and R. Olawoyin, "Fault Injection in Model-Based System Failure Analysis of Highly Automated Vehicles," *IEEE Open Journal of Intelligent Transportation Systems*, vol. 2, pp. 417–428, 2021.
[146] J. Hyvarinen and I. E. Annex, "Final Report, Volume I," *VTT, Espoo, Finland*, 1997.
[147] A. Behravan, N. Tabassam, O. Al-Najjar, and R. Obermaisser, "Composability modeling for the use case of demand-controlled ventilation and heating system," in *2019 6th International Conference on Control, Decision and Information Technologies (CoDIT)*, pp. 1998–2003.
[148] R. Obermaisser and P. Peti, "A fault hypothesis for integrated architectures," in *2006 International Workshop on Intelligent Solutions in Embedded Systems*, pp. 1–18.
[149] G. Yalcin, O. S. Unsal, A. Cristal, and M. Valero, "FIMSIM: A fault injection infrastructure for microarchitectural simulators," in *2011 IEEE 29th International Conference on Computer Design (ICCD)*, pp. 431–432.
[150] C. E. Stroud and C. A. Ryan, "Multiple fault simulation with random and clustered fault injection," in *Proceedings of Eighth International Application Specific Integrated Circuits Conference*, pp. 218–221.
[151] J. Tarrillo, J. Tonfat, L. Tambara, F. L. Kastensmidt, and R. Reis, "Multiple fault injection platform for SRAM-based FPGA based on ground-level radiation experiments," in *2015 16th Latin-American Test Symposium (LATS)*, pp. 1–6.
[152] S. Kundu, S. Chattopadhyay, I. Sengupta, and R. Kapur, "Multiple fault diagnosis based on multiple fault simulation using particle swarm optimization," in *2011 24th Internatioal Conference on VLSI Design*, pp. 364–369.
[153] J. Arlat, Y. Crouzet, J. Karlsson, P. Folkesson, E. Fuchs, and G. H. Leber, "Comparison of physical and software-implemented fault injection techniques," *IEEE Transactions on Computers*, vol. 52, no. 9, pp. 1115–1133, 2003.
[154] F. Zhong, J. K. Calautit, and Y. Wu, "Assessment of HVAC system operational fault impacts and multiple faults interactions under climate change," *Energy*, vol. 258, p. 124762, 2022.

[155] B. Sangchoolie, K. Pattabiraman, and J. Karlsson, "An Empirical Study of the Impact of Single and Multiple Bit-Flip Errors in Programs," *IEEE Transactions on Dependable and Secure Computing*, 2020.
[156] M. Tadeusiewicz and S. Halgas, "An efficient method for simulation of multiple catastrophic faults," in *2008 15th IEEE International Conference on Electronics, Circuits and Systems*, pp. 356–359.
[157] C. A. L. Lisboa and L. Carro, "Arithmetic operators robust to multiple simultaneous upsets," in *19th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, 2004. DFT 2004. Proceedings*, pp. 289–297.
[158] A. Papadimitriou, D. Hély, V. Beroulle, P. Maistri, and R. Leveugle, "A multiple fault injection methodology based on cone partitioning towards RTL modeling of laser attacks," in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–4.
[159] H. Wang, Y. Chen, C. W. H. Chan, and J. Qin, "An online fault diagnosis tool of VAV terminals for building management and control systems," *Automation in Construction*, vol. 22, pp. 203–211, 2012.
[160] A. Takakusagi, "Analytical study on preventive maintenance of air-conditioning system," *Journal of Architecture Planning and Environmental Engineering*, vol. 430, pp. 45–53, 1991.
[161] Y. C. Kim, V. D. Agrawal, and K. K. Saluja, "Multiple faults: Modeling, simulation and test," in *Proceedings of ASP-DAC/VLSI Design 2002. 7th Asia and South Pacific Design Automation Conference and 15h International Conference on VLSI Design*, pp. 592–597.
[162] M. A. Elnour, *Fault Diagnosis Of Sensor And Actuator Faults In Multi-Zone Hvac Systems*, 2019.
[163] B. Lee *et al.,* "Experimental evaluations on the outdoor air-based methods for water saving and plume abatement of cooling tower," *International Journal of Low-Carbon Technologies*, vol. 15, no. 3, pp. 421–426, 2020.
[164] Y. Lyu, Y. Pan, X. Yuan, M. Zhu, Z. Huang, and R. Kosonen, "A comprehensive evaluation method for air-conditioning system plants based on building performance simulation and experiment information," *Buildings*, vol. 11, no. 11, p. 522, 2021.
[165] M. Andrés *et al.,* "Real-Scale Experimental Evaluation of Energy and Thermal Regulation Effects of PCM-Based Mortars in Lightweight Constructions," *Applied Sciences*, vol. 12, no. 4, p. 2091, 2022.
[166] "Directive (eu) 2018/844 of the European Parliament and of the Council of 30 May 2018 Amending Directive 2010/31/eu on the Energy Performance of Buildings and Directive 2012/27/eu on energy efficiency, 2018," [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32018L0844&from=IT
[167] "Directive 2010/31/eu of the European Parliament and of the Council of 19 May 2010 on the Energy Performance of Build-ings, 2010," [Online]. Available: https://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2010:153:0013:0035:en:PDF
[168] K. A. Antonopoulos, M. Vrachopoulos, and C. Tzivanidis, "Experimental evaluation of energy savings in air-conditioning using metal ceiling panels," *Applied Thermal Engineering*, vol. 18, no. 11, pp. 1129–1138, 1998.
[169] H. N. Al-Deen and A. Al-Samari, "Experimental assessment of combining geothermal with conventional air conditioner regarding energy consumption in summer and winter," *Diyala Journal of Engineering Sciences*, vol. 14, no. 3, pp. 94–107, 2021.
[170] M. Krajčík, B. W. Olesen, and D. Petráš, "Sustainable Heating/Cooling for Low Energy Buildings: Experimental Evaluation of Indoor Environment in Residential Rooms with Different Heating/Cooling Concepts," in *E-nova International Congress 2012: Nachhaltige Gebäude*, pp. 107–114.
[171] A. Arteconi, C. Brandoni, G. Rossi, and F. Polonara, "Experimental evaluation and dynamic simulation of a ground coupled heat pump for a commercial building," *International journal of energy research*, vol. 37, no. 15, pp. 1971–1980, 2013.
[172] L. Yu *et al.,* "Multi-agent deep reinforcement learning for HVAC control in commercial buildings," *IEEE Transactions on Smart Grid*, vol. 12, no. 1, pp. 407–419, 2020.
[173] L. Yu *et al.,* "Deep reinforcement learning for smart home energy management," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 2751–2762, 2019.
[174] D. Huang, M. Thottan, and F. Feather, "Designing customized energy services based on disaggregation of heating usage," in *2013 IEEE PES Innovative Smart Grid Technologies Conference (ISGT)*, pp. 1–6.
[175] Z. Wu, Q.-S. Jia, and X. Guan, "Optimal control of multiroom HVAC system: An event-based approach," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 2, pp. 662–669, 2015.
[176] I. Shanin, S. Stupnikov, and V. Zakharov, "Application of Anomaly Detection Methods in the Housing and Utility Infrastructure Data," in *2019 Ivannikov Memorial Workshop (IVMEM)*, pp. 101–105.
[177] Z. Yang, Y. Wang, and J. Lv, "Survey of modern fault diagnosis methods in networks," in *2012 International Conference on Systems and Informatics (ICSAI2012)*, pp. 1640–1643.

[178] M. F. D'Angelo, R. M. Palhares, L. B. Cosme, L. A. Aguiar, F. S. Fonseca, and W. M. Caminhas, "Fault detection in dynamic systems by a Fuzzy/Bayesian network formulation," *Applied Soft Computing*, vol. 21, pp. 647–653, 2014.
[179] P. You-Jin, S.-K. S. Fan, and H. Chia-Yu, "A Review on Fault Detection and Process Diagnostics in Industrial Processes," *Processes*, vol. 8, no. 9, p. 1123, 2020.
[180] R. Isermann, *Fault-diagnosis applications: model-based condition monitoring: actuators, drives, machinery, plants, sensors, and fault-tolerant systems*: Springer Science & Business Media, 2011.
[181] D. Miljković, "Fault detection methods: A literature survey," in *2011 Proceedings of the 34th international convention MIPRO*, pp. 750–755.
[182] Y. Zhao, T. Li, X. Zhang, and C. Zhang, "Artificial intelligence-based fault detection and diagnosis methods for building energy systems: Advantages, challenges and the future," *Renewable and Sustainable Energy Reviews*, vol. 109, pp. 85–101, 2019.
[183] Z. Du, B. Fan, X. Jin, and J. Chi, "Fault detection and diagnosis for buildings and HVAC systems using combined neural networks and subtractive clustering analysis," *Building and Environment*, vol. 73, pp. 1–11, 2014.
[184] A. Rafati, H. R. Shaker, and S. Ghahghahzadeh, "Fault Detection and Efficiency Assessment for HVAC Systems Using Non-Intrusive Load Monitoring: A Review," *Energies*, vol. 15, no. 1, p. 341, 2022.
[185] S. P. Melgaard, K. H. Andersen, A. Marszal-Pomianowska, R. L. Jensen, and P. K. Heiselberg, *Fault Detection and Diagnosis Encyclopedia for Building Systems: A Systematic Review. Energies 2022, 15, 4366*: s Note: MDPI stays neutral with regard to jurisdictional claims in published …
[186] S. Katipamula and M. R. Brambley, "Methods for fault detection, diagnostics, and prognostics for building systems—a review, part I," *Hvac&R Research*, vol. 11, no. 1, pp. 3–25, 2005.
[187] S. Katipamula and M. R. Brambley, "Methods for fault detection, diagnostics, and prognostics for building systems—A review, part II," *Hvac&R Research*, vol. 11, no. 2, pp. 169–187, 2005.
[188] D. Hongzhi, C. Da-qing, and L. Bo, "Net work fault diagnosis technique based on fault tree analysis and XML," *Journal of Xiamen University (Natural Science)*, vol. 46, no. 2, pp. 205–208, 2007.
[189] A. L. Oña García, L. E. Sucar, and E. F. Morales, "A Distributed Probabilistic Model for Fault Diagnosis," in *Ibero-American Conference on Artificial Intelligence*, pp. 42–53.
[190] P. H. Ibargüengoytia, S. Vadera, and L. E. Sucar, "A probabilistic model for information and sensor validation," *The Computer Journal*, vol. 49, no. 1, pp. 113–126, 2006.
[191] M. Vlachopoulou, G. Chin, J. Fuller, and S. Lu, "Aggregated residential load modeling using dynamic Bayesian networks," in *2014 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pp. 818–823.
[192] C.-Y. Chiu, C.-C. Lo, and Y.-X. Hsu, "Integrating bayesian theory and fuzzy logics with case-based reasoning for car-diagnosing problems," in *Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007)*, pp. 344–348.
[193] H. Tang and S. Liu, "Basic theory of fuzzy Bayesian networks and its application in machinery fault diagnosis," in *Fourth international conference on fuzzy systems and knowledge discovery (FSKD 2007)*, pp. 132–137.
[194] J. Cheng, D. Bell, and W. Liu, "Learning Bayesian networks from data: An efficient approach based on information theory," *On World Wide Web at http://www. cs. ualberta. ca/~ jcheng/bnpc. htm*, 1998.
[195] Z. Shi, W. O'Brien, and H. B. Gunay, "Development of a distributed building fault detection, diagnostic, and evaluation system," *ASHRAE Transactions*, vol. 124, no. 2, pp. 23–37, 2018.
[196] Di Peng, Z. Geng, and Q. Zhu, "A multilogic probabilistic signed directed graph fault diagnosis approach based on Bayesian inference," *Industrial & Engineering Chemistry Research*, vol. 53, no. 23, pp. 9792–9804, 2014.
[197] J. Sun, S.-Y. Qin, and Y.-H. Song, "Fault diagnosis of electric power systems based on fuzzy Petri nets," *IEEE Transactions on Power Systems*, vol. 19, no. 4, pp. 2053–2059, 2004.
[198] J. Shiozaki and F. Miyasaka, Eds., *A fault diagnosis tool for HVAC systems using qualitative reasoning algorithms*, 1999.
[199] J. Y. Yao, J. Li, H. Li, and X. Wang, "Modeling system based on fuzzy dynamic Bayesian network for fault diagnosis and reliability prediction," in *2015 Annual Reliability and Maintainability Symposium (RAMS)*, pp. 1–6.
[200] L. A. Zadeh, "Fuzzy sets," *Information and control*, vol. 8, no. 3, pp. 338–353, 1965.
[201] L. A. Zadeh, "Fuzzy logic," *Computer*, vol. 21, no. 4, pp. 83–93, 1988.
[202] P. Hájek, *Metamathematics of fuzzy logic*: Springer Science & Business Media, 2013.
[203] D. Kolokotsa, "Artificial intelligence in buildings: A review of the application of fuzzy logic," *Advances in Building Energy Research*, vol. 1, no. 1, pp. 29–54, 2007.

[204] N. A. Sulaiman, M. F. Othman, and H. Abdullah, "Fuzzy logic control and fault detection in centralized chilled water system," in *2015 IEEE Symposium Series on Computational Intelligence*, pp. 8–13.
[205] A. L. Dexter and M. Benouarets, "Model-based fault diagnosis using fuzzy matching," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 27, no. 5, pp. 673–682, 1997.
[206] M. Eftekhari, L. Marjanovic, and P. Angelov, "Design and performance of a rule-based controller in a naturally ventilated room," *Computers in Industry*, vol. 51, no. 3, pp. 299–326, 2003.
[207] W. H. Allen, A. Rubaai, and R. Chawla, "Fuzzy neural network-based health monitoring for HVAC system variable-air-volume unit," *IEEE Transactions on Industry Applications*, vol. 52, no. 3, pp. 2513–2524, 2015.
[208] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*: Morgan kaufmann, 1988.
[209] Y. Zhao, F. Xiao, and S. Wang, "An intelligent chiller fault detection and diagnosis methodology using Bayesian belief network," *Energy and Buildings*, vol. 57, pp. 278–288, 2013.
[210] F. Xiao, Y. Zhao, J. Wen, and S. Wang, "Bayesian network based FDD strategy for variable air volume terminals," *Automation in Construction*, vol. 41, pp. 106–118, 2014.
[211] Y. Zhao, J. Wen, F. Xiao, X. Yang, and S. Wang, "Diagnostic Bayesian networks for diagnosing air handling units faults–part I: Faults in dampers, fans, filters and sensors," *Applied Thermal Engineering*, vol. 111, pp. 1272–1286, 2017.
[212] Y. Zhao, J. Wen, and S. Wang, "Diagnostic Bayesian networks for diagnosing air handling units faults–Part II: Faults in coils and sensors," *Applied Thermal Engineering*, vol. 90, pp. 145–157, 2015.
[213] F. M. Mele, *A model-based approach to HVAC fault detection and diagnosis*.
[214] S. Qiu, A. M. Agogino, S. Song, J. Wu, and S. Sitarama, "A fusion of bayesian and fuzzy analysis for print faults diagnosis," in *CATA*, pp. 229–232.
[215] R. J. Kuo, C. L. Cha, S. H. Chou, C. W. Shih, and C. Y. Chiu, "Integration of ant algorithm and case based reasoning for knowledge management," in *Proceedings of International Conference on IJIE*, pp. 10–12.
[216] M. Hu *et al.,* "A machine learning bayesian network for refrigerant charge faults of variable refrigerant flow air conditioning system," *Energy and Buildings*, vol. 158, pp. 668–676, 2018.
[217] M. G. Don and F. Khan, "Dynamic process fault detection and diagnosis based on a combined approach of hidden Markov and Bayesian network model," *Chemical Engineering Science*, vol. 201, pp. 82–96, 2019.
[218] Z. Bi, C. Li, X. Li, and H. Gao, "Research on fault diagnosis for pumping station based on TS fuzzy fault tree and Bayesian network," *Journal of Electrical and Computer Engineering*, vol. 2017, 2017.
[219] P. Larranaga, C. M. H. Kuijpers, R. H. Murga, and Y. Yurramendi, "Learning Bayesian network structures by searching for the best ordering with genetic algorithms," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 26, no. 4, pp. 487–493, 1996.
[220] J. Zhao and W. Zheng, "Study of fault diagnosis method based on fuzzy Bayesian network and application in CTCS-3 train control system," in *2013 IEEE International Conference on Intelligent Rail Transportation Proceedings*, pp. 249–254.
[221] R. J. Kuo, Y. P. Kuo, and K.-Y. Chen, "Developing a diagnostic system through integration of fuzzy case-based reasoning and fuzzy ant colony system," *Expert Systems with Applications*, vol. 28, no. 4, pp. 783–797, 2005.
[222] S. Gottwald, "An early approach toward graded identity and graded membership in set theory," *Fuzzy Sets and Systems*, vol. 161, no. 18, pp. 2369–2379, 2010.
[223] D. Dey, B. Dong, and Z. Li, "A Probabilistic Framework To Diagnose Faults in Air Handling Units," 2016.
[224] Y. Li and Z. O'Neill, "A critical review of fault modeling of HVAC systems in buildings," in *Building Simulation*, pp. 953–975.
[225] S. Myrefelt, "Reliability and functional availability of HVAC systems," 2004.
[226] A. Ebrahimifakhar, *Investigation of the Prevalence of Faults in the Heating, Ventilation, and Air-Conditioning Systems of Commercial Buildings*: The University of Nebraska-Lincoln, 2021.
[227] A. Hosseini Gourabpasi and M. Nik-Bakht, "Knowledge Discovery by Analyzing the State of the Art of Data-Driven Fault Detection and Diagnostics of Building HVAC," *CivilEng*, vol. 2, no. 4, pp. 986–1008, 2021.
[228] G. Hudson and C. Underwood, "A simple building modelling procedure for MATLAB," *SIMULINK1999*, 1999.
[229] M. G. Davies, "Optimum design of resistance and capacitance elements in modelling a sinusoidally excited building wall," *Building and Environment*, vol. 18, 1-2, pp. 19–37, 1983.
[230] L. Evangelisti, C. Guattari, and P. Gori, "Energy retrofit strategies for residential building envelopes: An Italian case study of an early-50s building," *Sustainability*, vol. 7, no. 8, pp. 10445–10460, 2015.
[231] American Society of Heating, Refrigerating, and Air Conditioning, *ANSI/ASHRAE Standard 140-2004: Standard Method of Test for the Evaluation of Building Energy Analysis Computer Programs*: ASHRAE Atlanta, GA, USA.

[232] A. ASHRAE, "Standard 62.2-2016 Ventilation for acceptable indoor air quality in residential buildings," *Atlanta, GA, USA*, 2016.
[233] S.-H. Cho, H.-C. Yang, M. Zaheer-Uddin, and B.-C. Ahn, "Transient pattern analysis for fault detection and diagnosis of HVAC systems," *Energy Conversion and Management*, vol. 46, 18-19, pp. 3103–3116, 2005.
[234] "Fault Injection Testing," [Online]. Available: https://microsoft.github.io/code-with-engineering-playbook/automated-testing/fault-injection-testing/
[235] T. Naughton, W. Bland, G. Vallee, C. Engelmann, and S. L. Scott, "Fault injection framework for system resilience evaluation: fake faults for finding future failures," in *Proceedings of the 2009 workshop on Resiliency in high performance*, pp. 23–28.
[236] O. Balci, J. D. Arthur, and W. F. Ormsby, "Achieving reusability and composability with a simulation conceptual model," *Journal of Simulation*, vol. 5, no. 3, pp. 157–165, 2011.
[237] P. K. Davis and R. H. Anderson, "Improving the composability of department of defense models and simulations," RAND CORP SANTA MONICA CA.
[238] Z. Noshad *et al.,* "Fault detection in wireless sensor networks through the random forest classifier," *Sensors*, vol. 19, no. 7, p. 1568, 2019.
[239] S. Zidi, T. Moulahi, and B. Alaya, "Fault detection in wireless sensor networks through SVM classifier," *IEEE Sensors Journal*, vol. 18, no. 1, pp. 340–347, 2017.
[240] T. Muhammed and R. A. Shaikh, "An analysis of fault detection strategies in wireless sensor networks," *Journal of Network and Computer Applications*, vol. 78, pp. 267–287, 2017.
[241] "Normal or Gaussian probability distribution," [Online]. Available: https://en.wikipedia.org/wiki/Normal_distribution
[242] M. Kuflom, P. A. Crossley, and N. Liu, "Impact of pecking faults on the operating times of numerical and electromechanical over-current relays," 2016.
[243] M. Abarkan, N. K. M'Sirdi, and F. Errahimi, "Analysis and simulation of the energy behavior of a building equipped with RESin simscape," *Energy procedia*, vol. 62, pp. 522–531, 2014.
[244] "MATLAB Introduction in scholarpedia," doi: 10.4249/scholarpedia.2929.
[245] "MATLAB introduction in Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/MATLAB
[246] "Mealy Machine," [Online]. Available: https://en.wikipedia.org/wiki/Mealy_machine
[247] "Overview of Mealy and Moore Machines.," [Online]. Available: https://de.mathworks.com/help/stateflow/ug/overview-of-mealy-and-moore-machines.html
[248] A. Correcher, E. García, F. Morant, E. Quiles, and L. Rodríguez, "Intermittent failure dynamics characterization," *IEEE Transactions on Reliability*, vol. 61, no. 3, pp. 649–658, 2012.

List of Evaluation Results of the FBBN-Fault Diagnosis Method

No.	Type	Time	Value	Diagnosed Type	Diagnosed Time	Diagnosed Ranks
1	HeaterActuator	70393	0	HeaterActuatorOff	70000	Rank 1
2	TemperatureSensor	78916	12	TemperatureSensorLow	80000	Rank 1
3	TemperatureSensor	24063	30	TemperatureSensorHigh	25000	Rank 1
4	CO_2Sensor	83367	834	CO_2SensorHigh	85000	Rank 1
5	DamperActuator	82700	1	DamperActuatorOn	85000	Rank 1
6	DamperActuator	12259	1	DamperActuatorOn	15000	Rank 1
7	HeaterActuator	68447	1	HeaterActuatorOn	65000	Rank 1
8	HeaterActuator	3086	1	HeaterActuatorOn	5000	Rank 1
9	HeaterActuator	58643	1	HeaterActuatorOn	50000	Rank 1
10	TemperatureSensor	33889	13	TemperatureSensorLow	35000	Rank 1
11	CO_2Sensor	61003	452	CO_2SensorLow	60000	Rank 1
12	CO_2Sensor	3990	753	CO_2SensorHigh	5000	Rank 1
13	DamperActuator	60034	1	DamperActuatorOn	60000	Rank 1
14	DamperActuator	2977	0	DamperActuatorOff	5000	Rank 1
15	HeaterActuator	66141	0	HeaterActuatorOff	65000	Rank 1
16	DamperActuator	42316	1	DamperActuatorOn	40000	Rank 1
17	HeaterActuator	61290	0	HeaterActuatorOff	60000	Rank 1
18	TemperatureSensor	58727	13	TemperatureSensorLow	60000	Rank 1
19	DamperActuator	10282	1	DamperActuatorOn	10000	Rank 1
20	TemperatureSensor	29410	14	TemperatureSensorLow	30000	Rank 1
21	DamperActuator	64910	1	DamperActuatorOn	65000	Rank 1
22	HeaterActuator	60401	1	HeaterActuatorOn	50000	Rank 1
23	CO_2Sensor	47280	382	CO_2SensorLow	50000	Rank 1
24	HeaterActuator	22249	0	HeaterActuatorOff	25000	Rank 1
25	CO_2Sensor	70355	812	CO_2SensorHigh	70000	Rank 1
26	CO_2Sensor	30239	438	CO_2SensorLow	30000	Rank 1
27	DamperActuator	53227	0	DamperActuatorOff	50000	Rank 1
28	TemperatureSensor	71784	21	TemperatureSensorMiddle	70000	Rank 2
29	DamperActuator	79246	1	DamperActuatorOn	80000	Rank 1
30	DamperActuator	65123	1	DamperActuatorOn	65000	Rank 1
31	CO_2Sensor	6554	592	CO_2SensorMiddle	5000	Rank 1
32	HeaterActuator	67321	0	HeaterActuatorOff	65000	Rank 1
33	DamperActuator	49147	0	DamperActuatorOff	50000	Rank 2
34	CO_2Sensor	29128	737	CO_2SensorHigh	30000	Rank 1
35	TemperatureSensor	26889	13	TemperatureSensorLow	40000	Rank 1
36	DamperActuator	52012	1	DamperActuatorOn	50000	Rank 1
37	TemperatureSensor	59549	19	TemperatureSensorLow	50000	Rank 1
38	CO_2Sensor	7243	803	CO_2SensorHigh	5000	Rank 1
39	HeaterActuator	13166	1	HeaterActuatorOn	15000	Rank 1
40	CO_2Sensor	86067	543	CO_2SensorLow	85000	Rank 1
41	HeaterActuator	9215	0	HeaterActuatorOff	10000	Rank 1
42	HeaterActuator	66953	1	DamperActuatorOff	65000	Rank 1
43	DamperActuator	7296	0	DamperActuatorOff	5000	Rank 1
44	DamperActuator	69126	1	DamperActuatorOn	70000	Rank 1
45	DamperActuator	15712	0	DamperActuatorOff	15000	Rank 1
46	HeaterActuator	11757	1	HeaterActuatorOn	10000	Rank 1
47	CO_2Sensor	47508	770	CO_2SensorHigh	50000	Rank 1
48	DamperActuator	53746	1	DamperActuatorOn	50000	Rank 1
49	CO_2Sensor	34717	432	CO_2SensorLow	35000	Rank 1

50	CO_2Sensor	10655	432	CO_2SensorLow	10000	Rank 1
51	CO_2Sensor	36052	797	CO_2SensorHigh	35000	Rank 1
52	DamperActuator	81630	0	DamperActuatorOff	75000	Rank 1
53	HeaterActuator	29179	0	HeaterActuatorOff	30000	Rank 1
54	HeaterActuator	9608	0	HeaterActuatorOff	10000	Rank 1
55	DamperActuator	20883	0	DamperActuatorOff	20000	Rank 1
56	HeaterActuator	11403	1	HeaterActuatorOn	10000	Rank 1
57	CO_2Sensor	49699	429	CO_2SensorLow	50000	Rank 1
58	HeaterActuator	30513	0	HeaterActuatorOff	30000	Rank 1
59	CO_2Sensor	3718	657	CO_2SensorHigh	5000	Rank 1
60	TemperatureSensor	63221	19	TemperatureSensorMiddle	60000	Rank 1
61	DamperActuator	47262	1	DamperActuatorOn	45000	Rank 1
62	TemperatureSensor	16326	13	TemperatureSensorLow	15000	Rank 1
63	TemperatureSensor	31838	26	TemperatureSensorHigh	30000	Rank 1
64	HeaterActuator	7010	1	HeaterActuatorOn	5000	Rank 1
65	DamperActuator	42059	0	DamperActuatorOff	40000	Rank 1
66	TemperatureSensor	26469	20	TemperatureSensorMiddle	25000	Rank 1
67	HeaterActuator	70644	1	HeaterActuatorOn	65000	Rank 1
68	HeaterActuator	32712	1	HeaterActuatorOn	35000	Rank 1
69	HeaterActuator	30303	1	HeaterActuatorOn	30000	Rank 1
70	TemperatureSensor	47534	22	HeaterActuatorOn	45000	Rank 1
71	DamperActuator	17949	0	DamperActuatorOff	20000	Rank 1
72	HeaterActuator	19915	0	HeaterActuatorOff	20000	Rank 1
73	CO_2Sensor	19520	425	CO_2SensorLow	20000	Rank 1
74	DamperActuator	37645	1	DamperActuatorOn	40000	Rank 1
75	CO_2Sensor	37170	798	CO_2SensorHigh	40000	Rank 1
76	DamperActuator	84651	0	DamperActuatorOff	85000	Rank 1
77	DamperActuator	22297	1	DamperActuatorOn	20000	Rank 1
78	TemperatureSensor	22656	24	TemperatureSensorHigh	25000	Rank 1
79	CO_2Sensor	19159	463	CO_2SensorLow	20000	Rank 1
80	DamperActuator	27543	1	DamperActuatorOn	25000	Rank 1
81	DamperActuator	7389	1	DamperActuatorOn	5000	Rank 1
82	HeaterActuator	2525	1	HeaterActuatorOn	5000	Rank 1
83	TemperatureSensor	42216	14	TemperatureSensorLow	40000	Rank 1
84	HeaterActuator	39645	1	HeaterActuatorOn	40000	Rank 1
85	CO_2Sensor	45027	569	CO_2SensorMiddle	45000	Rank 1
86	TemperatureSensor	53919	18	TemperatureSensorLow	60000	Rank 1
87	HeaterActuator	31747	0	HeaterActuatorOff	30000	Rank 1
88	HeaterActuator	76479	1	DamperActuatorOff	75000	Rank 1
89	DamperActuator	8529	0	DamperActuatorOff	10000	Rank 1
90	CO_2Sensor	58729	697	CO_2SensorHigh	55000	Rank 1
91	TemperatureSensor	9225	20	TemperatureSensorMiddle	5000	Rank 1
92	TemperatureSensor	67311	28	TemperatureSensorHigh	70000	Rank 1
93	DamperActuator	76976	1	DamperActuatorOn	75000	Rank 1
94	CO_2Sensor	17091	709	CO_2SensorMiddle	15000	Rank 1
95	DamperActuator	43202	1	DamperActuatorOn	45000	Rank 1
96	TemperatureSensor	52693	28	TemperatureSensorHigh	55000	Rank 1
97	TemperatureSensor	69595	13	TemperatureSensorLow	70000	Rank 1
98	HeaterActuator	20731	0	HeaterActuatorOff	20000	Rank 1
99	CO_2Sensor	42328	839	CO_2SensorHigh	40000	Rank 1
100	TemperatureSensor	61577	19	TemperatureSensorLow	80000	Rank 1
101	TemperatureSensor	5152	10	TemperatureSensorLow	5000	Rank 1
102	TemperatureSensor	6173	12	TemperatureSensorLow	5000	Rank 1
103	HeaterActuator	70689	1	HeaterActuatorOn	65000	Rank 1
104	TemperatureSensor	12949	20	TemperatureSensorMiddle	10000	Rank 1
105	TemperatureSensor	84066	26	TemperatureSensorHigh	85000	Rank 1

106	DamperActuator	39209	1	DamperActuatorOn	40000	Rank 1
107	CO_2Sensor	7212	395	CO_2SensorLow	5000	Rank 1
108	HeaterActuator	33778	1	HeaterActuatorOn	35000	Rank 1
109	DamperActuator	5225	1	DamperActuatorOn	5000	Rank 1
110	TemperatureSensor	36012	23	TemperatureSensorHigh	35000	Rank 1
111	DamperActuator	25228	0	DamperActuatorOff	25000	Rank 1
112	CO_2Sensor	85024	358	CO_2SensorLow	85000	Rank 1
113	CO_2Sensor	32177	569	CO_2SensorMiddle	30000	Rank 1
114	HeaterActuator	29333	1	HeaterActuatorOn	30000	Rank 1
115	TemperatureSensor	4552	15	TemperatureSensorLow	5000	Rank 1
116	TemperatureSensor	36533	29	TemperatureSensorHigh	35000	Rank 1
117	HeaterActuator	36094	0	HeaterActuatorOff	35000	Rank 1
118	TemperatureSensor	60575	21	TemperatureSensorMiddle	60000	Rank 2
119	TemperatureSensor	60317	13	TemperatureSensorLow	60000	Rank 1
120	HeaterActuator	11061	0	HeaterActuatorOff	10000	Rank 1
121	TemperatureSensor	2817	28	TemperatureSensorHigh	5000	Rank 1
122	CO_2Sensor	57817	503	CO_2SensorLow	60000	Rank 1
123	HeaterActuator	39807	0	HeaterActuatorOff	40000	Rank 1
124	TemperatureSensor	73918	17	TemperatureSensorLow	70000	Rank 1
125	DamperActuator	16496	0	DamperActuatorOff	15000	Rank 1
126	TemperatureSensor	10421	14	TemperatureSensorLow	10000	Rank 1
127	TemperatureSensor	33232	15	TemperatureSensorLow	35000	Rank 1
128	TemperatureSensor	25095	15	TemperatureSensorLow	25000	Rank 1
129	HeaterActuator	71227	1	HeaterActuatorOn	65000	Rank 1
130	TemperatureSensor	29711	12	TemperatureSensorLow	30000	Rank 1
131	HeaterActuator	78306	1	DamperActuatorOff	75000	Rank 1
132	TemperatureSensor	22527	10	TemperatureSensorLow	25000	Rank 1
133	DamperActuator	36743	0	DamperActuatorOff	35000	Rank 1
134	DamperActuator	15446	0	DamperActuatorOff	15000	Rank 1
135	DamperActuator	51713	1	DamperActuatorOn	50000	Rank 1
136	TemperatureSensor	60471	10	TemperatureSensorLow	60000	Rank 1
137	DamperActuator	5945	1	DamperActuatorOn	5000	Rank 1
138	DamperActuator	56545	1	DamperActuatorOn	55000	Rank 1
139	HeaterActuator	62067	1	HeaterActuatorOn	50000	Rank 1
140	CO_2Sensor	28093	636	HeaterActuatorOn	25000	Rank 1
141	DamperActuator	67289	0	DamperActuatorOff	65000	Rank 1
142	CO_2Sensor	23024	454	CO_2SensorLow	25000	Rank 1
143	TemperatureSensor	38024	19	TemperatureSensorMiddle	40000	Rank 1
144	TemperatureSensor	75633	29	TemperatureSensorHigh	75000	Rank 1
145	HeaterActuator	55099	0	HeaterActuatorOff	55000	Rank 1
146	DamperActuator	58417	1	DamperActuatorOn	55000	Rank 1
147	CO_2Sensor	60061	440	CO_2SensorLow	60000	Rank 1
148	TemperatureSensor	19358	27	TemperatureSensorHigh	20000	Rank 1
149	HeaterActuator	29762	1	HeaterActuatorOn	30000	Rank 1
150	TemperatureSensor	581	18	TemperatureSensorLow	5000	Rank 1
151	CO_2Sensor	79142	554	CO_2SensorLow	85000	Rank 1
152	DamperActuator	36664	1	DamperActuatorOn	35000	Rank 1
153	HeaterActuator	27862	0	HeaterActuatorOff	30000	Rank 1
154	CO_2Sensor	3090	697	CO_2SensorHigh	5000	Rank 1
155	CO_2Sensor	40910	487	CO_2SensorLow	40000	Rank 1
156	CO_2Sensor	52479	706	CO_2SensorHigh	55000	Rank 1
157	HeaterActuator	20983	0	HeaterActuatorOff	20000	Rank 1
158	CO_2Sensor	66140	458	CO_2SensorLow	65000	Rank 1
159	TemperatureSensor	7873	24	TemperatureSensorHigh	5000	Rank 1
160	DamperActuator	47226	1	DamperActuatorOn	45000	Rank 1
161	TemperatureSensor	55955	23	TemperatureSensorHigh	55000	Rank 1

162	CO_2Sensor	81664	690	CO_2SensorHigh	80000	Rank 1
163	CO_2Sensor	20411	634	CO_2SensorMiddle	20000	Rank 1
164	DamperActuator	38892	1	DamperActuatorOn	40000	Rank 1
165	DamperActuator	66553	1	DamperActuatorOn	65000	Rank 1
166	HeaterActuator	35957	1	HeaterActuatorOn	35000	Rank 1
167	TemperatureSensor	22157	22	TemperatureSensorHigh	50000	Rank 2
168	HeaterActuator	46720	0	HeaterActuatorOff	45000	Rank 1
169	CO_2Sensor	27482	817	CO_2SensorHigh	25000	Rank 1
170	DamperActuator	55776	1	DamperActuatorOn	55000	Rank 1
171	TemperatureSensor	47064	21	TemperatureSensorMiddle	50000	Rank 3
172	TemperatureSensor	62299	30	TemperatureSensorHigh	60000	Rank 1
173	CO_2Sensor	18894	360	CO_2SensorLow	20000	Rank 1
174	DamperActuator	5495	0	DamperActuatorOff	5000	Rank 1
175	HeaterActuator	31607	1	HeaterActuatorOn	30000	Rank 1
176	HeaterActuator	66700	1	HeaterActuatorOn	65000	Rank 1
177	CO_2Sensor	16592	683	CO_2SensorMiddle	15000	Rank 1
178	TemperatureSensor	8107	21	TemperatureSensorMiddle	5000	Rank 2
179	DamperActuator	74403	0	DamperActuatorOff	75000	Rank 1
180	TemperatureSensor	58012	20	TemperatureSensorMiddle	50000	Rank 1
181	CO_2Sensor	30043	622	CO_2SensorMiddle	30000	Rank 1
182	CO_2Sensor	22650	715	CO_2SensorHigh	25000	Rank 1
183	DamperActuator	20977	1	DamperActuatorOn	20000	Rank 1
184	TemperatureSensor	31038	18	TemperatureSensorLow	30000	Rank 1
185	TemperatureSensor	59048	19	TemperatureSensorMiddle	80000	Rank 1
186	DamperActuator	1692	0	DamperActuatorOff	5000	Rank 1
187	CO_2Sensor	23352	752	CO_2SensorHigh	25000	Rank 1
188	HeaterActuator	37146	0	HeaterActuatorOff	40000	Rank 1
189	DamperActuator	66452	1	DamperActuatorOn	65000	Rank 1
190	DamperActuator	65239	0	DamperActuatorOff	65000	Rank 1
191	HeaterActuator	68292	0	HeaterActuatorOff	70000	Rank 1
192	DamperActuator	57998	1	DamperActuatorOn	55000	Rank 1
193	CO_2Sensor	66430	774	CO_2SensorHigh	70000	Rank 1
194	TemperatureSensor	85525	28	TemperatureSensorHigh	85000	Rank 1
195	CO_2Sensor	50806	410	CO_2SensorLow	50000	Rank 1
196	TemperatureSensor	35161	27	TemperatureSensorHigh	30000	Rank 1
197	DamperActuator	68253	1	DamperActuatorOn	70000	Rank 1
198	CO_2Sensor	7772	375	CO_2SensorLow	10000	Rank 1
199	DamperActuator	58636	0	DamperActuatorOff	50000	Rank 1
200	CO_2Sensor	42769	330	CO_2SensorLow	40000	Rank 1