Informatics

AI-gestützte Regelgenerierung für automatisierte Fehlerdetektion und -diagnose

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Software Engineering und Internet Computing

eingereicht von

Jonas Simon Paul Fischer
Matrikelnummer 12202378

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Projektass. Dipl.-Ing. Dr.techn. Thomas Frühwirth, BSc
Mitwirkung: Projektass. Dipl.-Ing. Jürgen Pannosch, BSc

Wien, 14. Jänner 2025

Jonas Simon Paul Fischer	Thomas Frühwirth

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Informatics

AI-Powered Rule Generation for Automated Fault Detection and Diagnostics

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Software Engineering and Internet Computing

by

Jonas Simon Paul Fischer
Registration Number 12202378

to the Faculty of Informatics

at the TU Wien

Advisor: Projektass. Dipl.-Ing. Dr.techn. Thomas Frühwirth, BSc
Assistance: Projektass. Dipl.-Ing. Jürgen Pannosch, BSc

Vienna, January 14, 2025

Jonas Simon Paul Fischer

Thomas Frühwirth

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
Bibliothek
Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Erklärung zur Verfassung der Arbeit

Jonas Simon Paul Fischer

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.
Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, haben ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT- Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 14. Jänner 2025

Jonas Simon Paul Fischer

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Acknowledgements

I want to express my gratitude to all those who supported me throughout my thesis journey. A special thank you goes to Thomas Frühwirth and Jürgen Pannosch for their invaluable advice and guidance during the entire research process. I am also thankful to TU Wien, particularly the Automation Systems Group, for enabling this work.
Further, I would like to gratefully acknowledge the support by Siemens Austria as well as Siemens Switzerland. A big thank you to Konrad Diwold, Lukas Krammer, and all other members of their research group for their support and advice.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Kurzfassung

Diese Arbeit untersucht das Potenzial von Künstliche Intelligenz (KI)-gestützten Methoden die Regelgenerierung in regelbasierten Fehlererkennungssystemen zu automatisieren, welche ein wichtiger Teil von Building Automation and Control Systems (BACS) sind. Durch die Integration von Techniken aus dem Bereich der Large Language Models (LLMs) versucht die Arbeit, Beschreibungen in natürlicher Sprache in strukturierte, maschinenlesbare Regeln umzuwandeln, die den Fehlererkennungsprozess steuern. Die Arbeit nutzt die Design Science Research (DSR)-Methodologie in Form eines Relevancezyklus, Rigorzyklus und Designzyklus, um die zentralen Forschungsfragen zu adressieren. Zu diesen Herausforderungen gehört die Gewährleistung der syntaktischen und semantischen Genauigkeit der generierten Regeln. Zu den Schlüsselkomponenten der Systemarchitektur gehören Semantic-Chunking, die Generierung von Embeddings und die Nutzung von Vektordatenbanken, um effizientes Informationsretrieval zu ermöglichen. Eine quantitative Evaluierung zeigt, dass das System eine hohe syntaktische Genauigkeit erreicht und in 32 von 33 Fällen korrekte JSON-Regeln generiert. Die qualitative Evaluierung zeigt jedoch Einschränkungen bei der semantischen Genauigkeit. Zusätzlich, analysiert diese Arbeit mögliche Verbesserungen durch verfeinerte menschliche Eingaben und zeigt, dass detailliertes Nutzerfeedback die Leistung des Systems steigern kann. Trotz der Probleme, die in der qualitativen Evaluierung aufgedeckt wurden, legt die Arbeit eine Grundlage für die zukünftige Entwicklung eines praktischen KI-gestütztes Regelgenerierungssystems.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Abstract

This thesis investigates the potential of Artificial Intelligence (AI)-powered methods to automate rule generation within rule-based fault detection systems, a critical component of Building Automation and Control Systems (BACS). By integrating techniques in the field of Large Language Models (LLMs), the study seeks to transform natural language descriptions into structured, machine-readable rules that drive fault detection processes. The research employs the Design Science Research (DSR) methodology, systematically progressing through the relevance, rigor, and design cycle to address the core research challenges. These challenges include ensuring the syntactic and semantic accuracy of generated rules. Key components of the system architecture include semantic chunking, embedding generation, and the use of vector databases to facilitate efficient information retrieval. A quantitative evaluation demonstrates that the system achieves high syntactic accuracy, generating well-formed JSON rules in 32 out of 33 cases. However, the qualitative evaluation reveals limitations in semantic accuracy. Additionally, this thesis explores potential improvements through refined human input, showing that detailed user feedback can significantly increase the system's performance. Despite the problems revealed in the qualitative evaluation, the work lays a foundation for the future development of a practical AI-enhanced rule generation system.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Contents

Acknowledgements **vii**

Kurzfassung **ix**

Abstract **xi**

1 Introduction **1**
1.1 Motivation and Problem Statement 2
1.2 Methodology 4
1.3 Structure of the Thesis 6

2 Relevance Cycle **9**
2.1 Rule Generation in Different Building Life Cycle Phases 9
2.2 Use Case Narrative 10
2.3 Prerequisites 11
2.4 Stakeholder Analysis 11
2.5 Requirements 12

3 Rigor Cycle **15**
3.1 Knowledge Base 15
3.2 Related Work 20
3.3 Architecture of the AI-Powered Rule Generation System 22

4 Design Cycle **29**
4.1 Open Source Libraries 29
4.2 Data Extraction 30
4.3 Implementation of the RAG Pipeline 31
4.4 Integration of the Example Rules 34
4.5 Usage of the Rule Schema 35
4.6 Implementation of the AI Agent System 36
4.7 Few-Shot Prompting and AI Agents Execution 37

5 Evaluation **41**
5.1 Evaluation Setup 41

5.2 Quantitative Evaluation 42
5.3 Qualitative Evaluation 45
5.4 Potential Improvement Through Refined Human Input 49
5.5 Requirements Evaluation 50
5.6 Discussion of the Results 51

6 Conclusion and Future Work **55**

Overview of Generative AI Tools Used **57**

List of Figures **59**

List of Tables **61**

Listings **63**

Acronyms **65**

Bibliography **67**

Appendix **75**

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 1

Introduction

In recent years, we have witnessed significant progress in various industries due to the integration of digitalization and technology. This integration has paved the way for developing highly efficient automation solutions in diverse domains [HKJ19]. An excellent example in this context is the field of Building Automation and Control Systems (BACS). BACS enable building operators to have comprehensive control and monitoring capabilities over all the systems within a building, ensuring optimal performance and the early detection of faults or issues. These systems play a crucial role in maintaining the efficiency and comfort of buildings by collecting and analyzing data from a broad collection of sensors and building systems [Sri22]. However, the sheer volume of data generated by BACS often overwhelms building operators, leading to challenges in conducting comprehensive data analysis. Automated Fault Detection and Diagnostics (AFDD) addresses this issue by seamlessly identifying and diagnosing faults within the building systems, thereby enabling efficient monitoring even with a limited workforce [LS21] [Pac18]. A particularly effective variant of AFDD is rule-based fault detection, in which specific rules are developed to uncover system components' flaws. However, this method requires expert knowledge to devise generic rules applicable to various BACS. This can be time- and resource-intensive [LS21].

One potential solution to this problem could be integrating Artificial Intelligence (AI) into BACS. AI is believed to improve BACS by making them more efficient and effective in their energy usage, as well as enhancing their understanding of energy usage within the building [Vav24]. In the context of AFDD, the integration of AI could streamline the rule-generation process. Especially Large Language Models (LLMs) are a promising new approach to achieve remarkable progress in this task. Various companies such as *OpenAI*[1], *Google*[2] or *Anthropic*[3] offer powerful AI systems based on LLMs, that could potentially be

---

[1] https://openai.com/chatgpt

[2] https://deepmind.google/technologies/gemini

[3] https://www.anthropic.com/claude

leveraged for this purpose. However, these systems are not directly usable for generating applicable rules due to a lack of necessary context. Furthermore, concerns about data privacy and vendor lock-in arise from these systems' closed-source, cloud-based nature, all of which are owned by major tech companies. To mitigate these concerns, there is a growing need for an open-source solution that can be used locally. In addition to the increasing popularity of LLMs in the commercial sector, the open-source community has also achieved excellent results in building high-quality models. Models such as those from *MistralAI*[4] or *Llama*[5] can be used by anyone and are easily accessible through websites like *Huggingface* [WDS+20]. Furthermore, platforms like *Ollama*[6] and *Langchain*[7] allow developers to integrate models in their code easily [TA23]. Hence, creating LLM-based solutions independent of major corporations has proven quite effective.

1.1 Motivation and Problem Statement

The efficiency of building operations can be enhanced by AFDD, especially through rule-based fault detection and diagnostics. An integrated AFDD system within BACS ensures efficient and comfortable building operations. The system helps save energy and detect faults effectively [Pac18]. For example, a rule in such a system could be: „Room brightness sensor measurement reacts on light command/modulation change“ [GMMH22]. This rule ensures that the light can be controlled correctly and that there are no sensor faults. Applying such rules throughout the building helps to avoid issues in building operation.

However, initially, creating generic rules is complex and requires expert knowledge. Experts need to deeply understand how BACS functions and comprehend the logic connections within the system. They must know which sensors exist and their locations. For example, the rule mentioned earlier only works if there is access to at least one sensor inside one room and only if the light command or modulation change can be detected. For other rules like „Room temperature measurement reacts on radiator valve position change when heat is provided by associated heat group“, it is crucial to have knowledge about which heat group is responsible for which rooms and which sensors in this room are capable of measuring the temperature [GMMH22]. Additionally, BACS can vary from one building to another. While most buildings have a Heating, Ventilation and Air Conditioning (HVAC) system, the system can differ significantly from those in other buildings due to differing building needs. Furthermore, having the right sensors in place is essential for applying rules. Even if the same sensors are placed throughout the building, they may be named differently. Once again, experts are needed to recreate the applied rules.

---

[4] https://mistral.ai/

[5] https://llama.meta.com/

[6] https://ollama.com/

[7] https://www.langchain.com/

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Consequently, the development of intuitive and adaptable tools that facilitate rule creation becomes imperative to manage such complexity efficiently. To ensure that users can effectively handle these tools, it is necessary to identify the requirements needed. This leads to the first research question:

RQ1: What are the requirements for using AI within variable building environments?

The complexity and variability of building environments necessitate advanced methods for rule generation in AFDD systems. Traditional rule creation requires expert knowledge and is time-consuming. To address this, it is essential to explore the latest advancements in AI that can automate and enhance this process. Understanding the current state-of-the-art AI methods in scientific research provides insights into the most effective techniques for rule generation. The second research question is therefore defined as follows:

RQ2: What are the current state-of-the-art AI methods in scientific research that can be used for rule generation in AFDD?

The integration of state-of-the-art AI methods for rule generation in AFDD systems presents an opportunity to enhance the efficiency and comfort of building operations. While individual AI techniques may be useful in automating tasks, combining multiple methods could lead to even more powerful solutions. By leveraging the strengths of various AI methods, it may be possible to create a comprehensive system that can handle the complex rule generation process. This leads to the third research question:

RQ3: How can these state-of-the-art AI methods be combined to generate rules for AFDD?

To tackle the defined research questions, it is necessary to note that AI heavily relies on relevant data to function. Having sufficient data available is crucial to achieve a reasonable outcome for rule generation [Nag24]. All data accessible to or known by experts is required to power the AI. This means determining which data can be used to generate rules and how to deal with unstructured, non-machine-readable data is necessary. Documentation in the form of Portable Document Formats (PDFs) written in natural language can be a crucial source of information.

Significant advancements have been made in the field of Natural Language Processing (NLP) since the introduction of the Transformer architecture by Google in 2017 [VSP$^+$17] [RWC$^+$19]. Recent developments have shown that LLMs, primarily designed for language generation, can also handle complex tasks that require complex reasoning and planning capabilities [YZY$^+$23] [WXJ$^+$23]. These advancements in NLP and LLMs can be utilized to address the challenge of rule generation for fault detection and diagnostics. Instead of experts having to spend time reading documentation and learning about buildings to

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

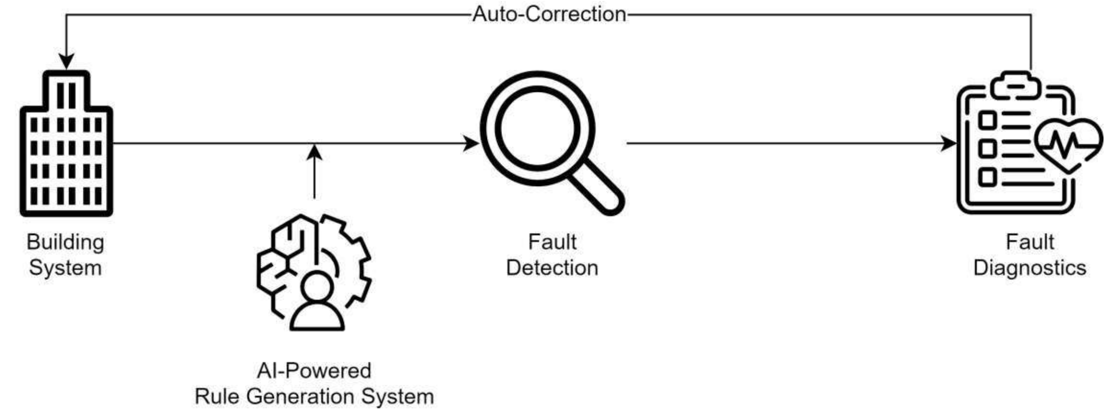

Figure 1.1: Localization of the AI-Powered Rule Generation System in BACS [Pac18]

create rules for fault detection, AI can be leveraged. The main goal of this work is to explore AI, especially NLP-based techniques that can help generate rule sets without requiring specialized skills. The proposed techniques should be compatible with arbitrary BACS. Simplifying the generation of rules for fault detection is a benefit for all the stakeholders. Stakeholders are building owners, operators, and facility managers. System operators and facility managers are able to quickly detect issues in the system, enabling them to react proactively rather than only after receiving a complaint. Additionally, residents and visitors benefit from smooth building operations and optimal conditions.

Figure 1.1 illustrates the placement of an AI-powered rule generation system within BACS. Such a system with a rule-based AFDD system normally has three parts: the building system itself, a fault detection system, and a fault diagnostics part [Pac18]. The fault detection system is equipped with rules to detect faults in data from the building system. If a fault is detected, the fault diagnostics try to analyze where the fault came from, and an auto-correction may be applied to the building system. An AI-powered rule generation system would be added to this setup right before the fault detection system. The AI-powered rule generation system would be able to add new rules to the fault detection system by leveraging data from the building system. On top of that, it can build on existing rules from the fault detection system.

1.2 Methodology

In this research, the Design Science Research (DSR) methodology is used to address the challenge of automated rule generation for AFDD. DSR is a paradigm whose goal is to solve a specific research problem. It creates an artifact, which, in most cases, is a technical solution to the problem. In this work, the artifact is a technical solution, more specifically, a software program. The DSR's findings consist of the newly designed artifact and gained know-how to understand better how the artifact performs within the

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

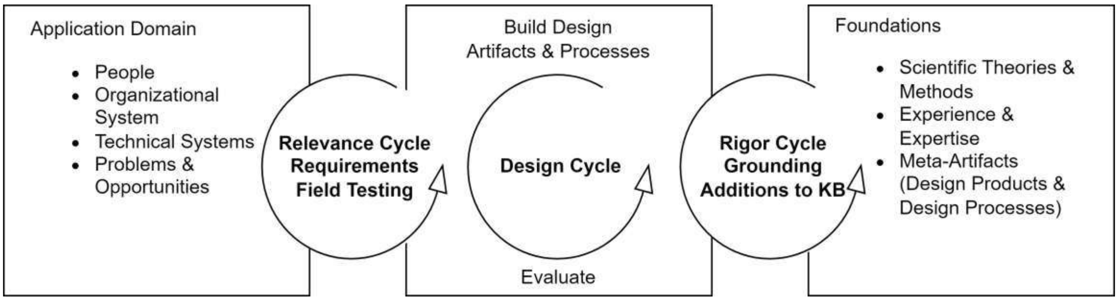

Figure 1.2: The Three-Cycle-View of DSR [Hev07]

relevant environment [BHM20]. The DSR framework by Hevner is divided into three cycles [Hev07]. The three cycles are illustrated in Figure 1.2. The role of each cycle in the context of this work is briefly discussed in the following. Each cycle aims to answer one of the three research questions.

1.2.1 Relevance Cycle

The goal of the relevance cycle is to explore the relevance of the problem and its solution. In this case, the relevance cycle focuses on identifying the specific requirements and constraints for the rule generation process in the context of BACS. First of all, the environment of the problem is defined. This includes defining the actors involved and the organizational and technological structures. From the problem and its opportunities, the requirements for the artifact are derived. As a last step, the success criteria are defined. If the artifact does not work as intended after the last cycle, the relevance cycle will be revisited to check and adjust the requirements and constraints [Hev07]. The work of the first cycle aligns with the first research question, which aims to identify the requirements for using AI within variable building environments.

1.2.2 Rigor Cycle

The main goal of the rigor cycle is to build a knowledge base. It is ensured that the research is grounded in existing knowledge and contributes to the knowledge base. This involves a continuous process of referring to the knowledge base and including relevant theories, methods, and techniques from the fields of AFDD and NLP. On top of that, existing artifacts from the field are taken into account. These can be other approaches to generating or deriving rules for a specific domain. The knowledge base is created through an in-depth literature review by systematically surveying scholarly articles, books, and other relevant sources to identify the state-of-the-art techniques in AFDD and NLP. Especially the research around LLMs is taken into account. The knowledge base is provided as a written summary of information relevant to this work. From the requirements and the collected knowledge, a software architecture containing tools and methods from the field of AI is derived. The defined architecture functions as a base for the artifact. After completing the three cycles, the gained insight is used to refine

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

the knowledge base and contribute new knowledge in the form of design principles and guidelines for AI-based rule generation in AFDD to the knowledge base [Hev07]. With the help of the knowledge base, the second research question is addressed, which was defined to identify the current state-of-the-art AI methods in scientific research that can be used for rule generation in AFDD.

1.2.3 Design Cycle

In the design cycle, the development of the artifact takes place. The knowledge is used, and a concrete solution is proposed. In this work, an AI-based system for generating rules for AFDD is the artifact. Next to the rule generation, a major topic is data processing, especially the challenge of dealing with unstructured data. This cycle heavily depends on the other two cycles. The methods, tools, and the created software architecture are converted to a technical solution. The design cycle is the core of the DSR methodology, as it is where the artifact is constructed and evaluated [Hev07]. After the development, an evaluation is conducted to assess the effectiveness of the artifact in addressing the problem. Therefore, the usefulness of the individual methods and tools and whether their combination leads to meaningful results is discussed. In order to compare the results, rules created by experts are included in the evaluation. As the last cycle involves the combination of different methods, this cycle gives answers to the third research question concerning how state-of-the-art AI methods can be combined to generate rules for AFDD.

1.3 Structure of the Thesis

In this research, the DSR methodology is used to tackle the research question. This approach follows a specific structure to ensure a systematic process for finding a solution. This structure is reflected in this work. The remaining chapters are structured as follows.

Embracing the DSR methodology, the first cycle, known as the Relevance Cycle, defines the specification of the problem's environment. The Relevance Cycle is presented in Chapter 2. To specify the problem's environment, AFDD is analyzed in different building life cycle phases. The problem's environment also includes defining the use cases for the problem and analyzing the stakeholders involved. To conclude the Relevance Cycle, prerequisites and requirements for an AI-powered rule generation system are defined.

The Rigor Cycle, outlined in Chapter 3, follows the Relevance Cycle. The Rigor Cycle seeks to build a profound knowledge base by conducting a comprehensive literature review. On top of this, other research and approaches to similar problems in the field are presented. Additionally, this cycle involves developing a software architecture for the artifact, supported by diagrams to represent the architecture visually.

Continuing with the DSR framework, the Design Cycle is executed in Chapter 4 to bring the planned artifact to realization and test the methods developed. To evaluate the quality of the work, Chapter 5 is dedicated to assessing the results and determining

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

if they are comparable to those that experts in the field would have produced. The Section 5.6 delves into a detailed discussion of the results, and finally, the thesis is concluded comprehensively in Chapter 6, summarizing all the findings and insights gathered throughout the research process.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 2

Relevance Cycle

Nowadays, it is common for commercial buildings to have a monitoring system to observe the building's operation. However, this does not necessarily mean that the data received through the monitoring system leads to insights on how the building performs. The amount of data is often too overwhelming and cannot be processed and interpreted effectively by the responsible people [KLC+20]. AI-powered rule generation can improve the monitoring system by generating rules that ease the evaluation of the data. However, the question arises of how and when to apply the rule generation process.

2.1 Rule Generation in Different Building Life Cycle Phases

It is possible to apply AI-powered rule generation at different stages of a building's life cycle, including design, construction, and operation. Each stage may have different requirements, which are described below.

During the building's design phase, rule generation can be considered. As the building's construction is planned, sensors and monitoring devices are factored in. With a detailed plan for monitoring the building and constructing the HVAC, rule generation can already considered as a part of the system. The data needed for rule generation can be collected, and rules can be prepared. Having rules prepared before the start of building construction helps to monitor the building from the beginning efficiently. However, AI-powered rule generation must adapt to changes and new information. Additionally, the rules can not be tested immediately, and a need for more or other rules may arise later on anyhow.

Furthermore, AI-powered rule generation can be used during the building's construction phase. Once the documentation is finalized and all necessary information is available, comprehensive rules can be generated and applied to the AFDD during the initial commissioning phase of a new building. The rules ensure that HVAC systems and other building components perform optimally from the start and help identify and resolve any faults or inefficiencies before the building's occupancy. This scenario is the optimal case because rules are generated just once after all information is available, and ongoing monitoring during the building's construction helps identify and fix faults before the building is used.

Moreover, rule generation can also be utilized while the building is in operation. Existing buildings that are equipped with an AFDD system can be directly improved by applying AI-powered rule generation. The newly generated rules do not affect the already existing AFDD. Available documentation and information from the building can be leveraged to be used as data sources for AI. There is no need to collect data first, as the input data consists primarily of documentation in natural language that has already been created. The rules can be generated, and the AFDD will be improved. Ongoing monitoring can detect and diagnose faults and inefficiencies as they occur, enabling early maintenance and preventing energy performance degradation. For this thesis, the last scenario is exemplary and is the only one considered, as it is the most accessible life cycle phase.

2.2 Use Case Narrative

Fault Detection and Diagnostics (FDD) systems are used to process the data and detect suboptimal performance in the building [KLC+20]. An essential technical component monitored by FDD systems are HVAC systems. They include in particular Air Handling Units (AHUs) and Variable Air Volume (VAV) terminal boxes [KLC+20]. HVAC systems therefore play an important role when using FDD systems. Estimations from the literature say that 5% - 30% of the energy that is used in commercial buildings is wasted because errors in the control system arise [GSM+17] [LKG20]. Possible reasons for errors include sensor faults, improper sensor installation, a stuck valve, or issues with pumps connected to the valve [GMMH22]. Using FDD systems saves energy and money by preventing such errors [GSM+17]. Two different studies measured the median energy saving in buildings by using FDD to 8% - 9% [LKG20] [KLC+20]. The implementation of FDD not only reduces costs and saves energy but also improves the comfort of the building [KLC+20]. AFDD systems take FDD a step further by automating the process of detecting and diagnosing faults and minimizing human intervention [LS21]. Furthermore, integrating AI-powered rule generation into this process could help to simplify the initial phase of rule-based fault detection, where the rules are created. The time-consuming task of creating initial rules by experts can be streamlined with the help of AI.

A typical use case scenario for AI-powered rule generation for AFDD in a commercial building starts with the building owner deciding to adopt the technology to reduce energy waste and improve operational efficiency for an existing building. The AI-powered rule generation is now leveraged to generate new rules. Following, information is provided on what behavior should be tested. For example, this can be done by a facility manager or system integrator familiar with common test cases in the building. The facility manager formulates in natural language which components and which behavior should be tested.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

The AI, more explicitly the LLM-powered system, is then executed and provides the rules matching the input of the facility manager. These rules can be directly integrated into the AFDD-system. The AFDD-system now continuously monitors the building with an additional set of rules, automatically detecting faults and suboptimal performance. When a fault is detected, the facility manager team is alerted. The facility manager team then uses the AFDD insights to prioritize maintenance tasks and address the identified issues promptly. Over time, the AFDD-system and the generated rules help the building achieve a lower energy consumption and a smoother operation.

2.3 Prerequisites

To be able to use AI-powered rule generation for AFDD, a number of prerequisites must be met. Those prerequisites must be met to ensure that the system can function correctly and provide the desired benefits. The prerequisites are described below.

1. Building with AFDD System

First, the building must be equipped with an AFDD-system. This implies that a system integrator has implemented an AFDD from a service provider into the building.

2. AFDD with Rule-Based Fault Detection

Second, the system must support rule-based fault detection. Without this capability, there would be no foundation for generating new rules, and the underlying technical system for applying these rules would not exist.

3. User able to provide Input

Third, one or more individuals must be available to provide input on which components and behaviors should be tested. Without this input, an LLM would not be able to generate new rules, as it is not explicitly trained for the purpose of generating rules. Instead, the LLM must be provided with clear prompts that outline the goals to be achieved. People that contain knowledge about the building, like system integrators and facility managers, provide the input for those prompts.

2.4 Stakeholder Analysis

The implementation and usage of AFDD and the necessity of AI-powered rule generation involves various stakeholders, each with their own interests and requirements. Next to the users of the building, like residents or workers, mainly three other stakeholders play an important role in an AI-powered rule generation system.

As vendors of the AFDD-system, service providers have an interest in incorporating AI-powered rule generation. This technology enhances their product offering, making

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

it more effective and attractive to potential customers. By using LLMs, they can develop a more sophisticated AFDD-system that can detect a broader range of faults with greater accuracy. This improvement in their product can lead to increased sales, market share, and customer satisfaction, ultimately strengthening their position in the market [KLC⁺20].

System integrators set up and run the AFDD-system. With AI-powered rule generation, they can benefit from a more robust and efficient system. The LLM's capabilities can help detect faults earlier and more accurately, leading to smoother building operations. The rule generation further reduces the need for manual intervention. However, system integrators may need to adapt to new technologies and processes, which could require additional skill development [KLC⁺20].

Facility managers play a vital role in maintaining smooth building operations. By closely monitoring the building through the AFDD-system, they can quickly identify and respond to any detected issues. Leveraging AI-powered rule generation can significantly enhance the capabilities of facility managers in this regard. AI-powered rule generation enables facility managers to detect problems more rapidly and accurately before they escalate and lead to tenant complaints or service disruptions. While facility managers may need to adapt to new technologies and processes, requiring training to use the AFDD-system effectively, AI can simplify their work processes [KLC⁺20].

2.5 Requirements

From the described use case, several requirements arise that are fundamental to the rule generation process. Those requirements were derived and are described below.

1. Input in Natural Language

First, the AI-powered rule generation system **must be able to receive input in natural language**. A natural language input is crucial for the facility manager to be able to provide input. The facility manager must be able to communicate their requirements in natural language. The AI-powered system must understand and process this information.

2. Directly Applicable Rules

Second, the AI-powered rule generation system **must generate rules that can be directly used by the existing AFDD system**. To achieve directly applicable rules, the rules must be in a format that the AFDD system can understand and apply. The rules must be structured in a way that the AFDD system can process them and use them for fault detection. This requirement is crucial to have a fully automated process from rule generation to rule application without interference from humans.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

3. External Knowledge Usage

Third, the AI-powered rule generation system **must be able to use knowledge from other sources** apart from the internal knowledge encoded in the LLM. The system must be capable of using documents written in natural language as input. It should extract relevant information from these documents to generate rules. This capability is essential because the language model may not possess specific knowledge about the building in question.

4. Consistency with existing AFDD System

Fourth, the AI-powered rule generation system **must be able to stay consistent with the existing AFDD system**. It must be able to understand the structure and content of existing rules from the AFDD system and generate new rules that are consistent with those rules. Taking existing rules into account is important to ensure that the new rules are seamlessly integrated into the existing process.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 3

Rigor Cycle

3.1 Knowledge Base

As a foundation for the artifact creation, an in-depth review of the technical background relevant to AI-powered rule generation for AFDD was conducted. The review explores AI-techniques relevant to the topic. As mentioned in Section 1.1, especially NLP and LLMs are promising in addressing the problem. However, the foundational approach lies in AFDD-systems, which detect faults by applying rules to BACS. The following section provides background information on the technologies used and the field in which this work is situated. However, a fundamental understanding of NLP and LLMs is necessary to comprehend the content of this work.

3.1.1 Rule-Based Fault Detection

Rule-based fault detection is a systematic approach to identify and diagnose faults in various systems, such as HVAC systems or AHUs. This method relies on predefined rules to monitor system performance and detect anomalies. The principles of rule-based AFDD involve several vital steps. First, rules are established based on expert knowledge, historical data, and system specifications, defining normal operating conditions and thresholds for various parameters. Sensors and monitoring tools continuously collect data from the system, including temperature, pressure, flow rates, and other relevant metrics. This collected data is then analyzed against the predefined rules, and if the data deviates from the expected range, an alert is triggered. When an anomaly is detected, the system identifies the specific fault based on the violated rule, helping to pinpoint the exact issue. The system also provides diagnostic information, suggesting possible causes and corrective actions, which aids technicians in quickly resolving the problem [SBCH06] [SB06].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

3.1.2 Natural Language Processing and Large Language Models

A foundational concept for AI-powered rule generation is the Deep Learning (DL) architecture called transformers, which Google first developed in 2017 [VSP$^+$17]. Since then, the field of NLP has made remarkable advancements. The rise of LLMs changed how the public sees AI and brought the potential for many new industry applications derived from NLP techniques [EMMR23].

At the heart of many NLP applications are LLMs. LLMs are trained on extensive amounts of text data and use DL techniques to understand and generate human-like text [DSS$^+$23]. LLMs are designed to predict the next word in a sentence, which allows them to create text that is coherent and contextually relevant based on the input provided. The training process for LLMs involves feeding the model with diverse text data, allowing it to learn patterns, grammar, and context. As a result, the model is able to perform a variety of language-related tasks with high accuracy [Dou23].

In the context of AFDD, NLP plays a crucial role in processing and understanding natural language data, including explanations of the AFDD-system as well as building and sensor descriptions. With the aid of NLP techniques, extracting valuable information from unstructured text data and utilizing it to enhance the rule generation process in AFDD systems becomes feasible. LLMs allow users to use natural language as input and enable unstructured data to serve as a data basis [ZZL$^+$23].

3.1.3 Prompting

Using LLMs requires accurate prompting, as it directly affects the quality and relevance of the generated responses. A well-constructed prompt offers clear context and specific guidance to produce accurate and valuable outputs. Various prompting strategies can significantly improve the use of LLMs [Sar22].

Few Shot Prompting

When creating a prompt for an LLM, it is possible to only use a simple question or instruction, a method known as zero-shot prompting. This approach provides no additional context or examples, and the model's response is solely based on the question [Sar22].

In contrast, one-shot and few-shot prompting involve providing one or multiple examples before the actual question. These methods facilitate in-context learning, allowing the model to learn tasks by being presented with examples. Few-shot prompting is an effective way to improve the performance of LLMs across various tasks [Sar22] [BMR$^+$20].

However, LLMs often encounter difficulties with complex reasoning. Chain-of-Tought (CoT) prompting can help enhance reasoning performance [BMR$^+$20].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Chain-of-Tought Prompting

The reasoning capabilities of LLMs are improved through CoT prompting by providing the model with reasoning steps. When combined with few-shot prompting, the examples consist of a question and an answer and include a description of how the answer can be derived using logical steps. This approach improves the performance of LLMs on more complex tasks. Additionally, it is possible to integrate CoT into zero-shot prompts by simply instructing the LLM to think step by step [Sar22] [BMR+20] [WWS+22].

This method is even integrated into some LLMs directly. An example is the *o1* model from *OpenAI*, which uses CoT when being prompted to solve complex problems. By breaking down difficult tasks into smaller, more manageable steps and learning to recognize and correct its mistakes, *o1* can effectively reason through challenges and provide more accurate and informative responses [Ope24].

Decomposed Prompting

Another technique for solving more complex tasks is to use iterative methods and decompose tasks into simpler subtasks. These simpler subtasks can then be handled by different LLM models or solved using various prompts. Additionally, decomposed prompting can help reduce the length of inputs and outputs, preventing the token limits of LLMs from being exceeded. LLMs have a maximum number of tokens they can process at once, restricting the amount of information they can consider during a single interaction [KTF+22].

3.1.4 Dealing with Unstructured Data

When using LLMs, the assumption is made that the information needed was included during the model's training. However, this assumption is not valid for specific use cases. In such cases, the LLM tends to provide false information [AM23]. This phenomenon is known as „hallucination“, where the model generates information that is plausible-sounding but incorrect. To use LLMs effectively, data is needed to help the model understand its task. The challenge is that the available data is often not machine-readable and, therefore, cannot be used by LLMs as context. While the data does not need to be in a specific format, it must at least be in text form to be passed to the LLM [YJT+23].

A widely used format for storing data are PDFs. They often contain structured information that can significantly improve the capabilities of LLMs. By extracting and using content from PDFs, LLMs can be given valuable data, enhancing their ability to understand and generate human-like responses. However, using PDFs for this purpose poses several challenges. First, PDFs can be created from scanned documents, which may contain images of text rather than actual text, making it difficult for LLMs to process information accurately. Second, the layout and structure of PDFs can vary widely, with some documents having complex formatting, multiple columns, or embedded multimedia elements, which complicates data extraction [dLBBdFC+24]. Lastly, it is crucial to

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

ensure that the extracted content is valuable and accurate, as irrelevant or erroneous data can negatively impact the performance of LLMs. Despite these challenges, effectively utilizing PDF documentation can provide LLMs with meaningful data, enhancing their overall performance and capabilities [CD15].

3.1.5 Retrieval Augmented Generation

As previously mentioned, LLMs require extensive and precise knowledge to handle complex tasks. Without this, „hallucinations“ can occur. Older LLMs, such as *ChatGPT 3.5* with an information cutoff in January 2022 (as of March 26, 2024), are limited by their training data. While some systems strive to stay updated, they may still need more specific information crucial for generating relevant responses. Even when the necessary data is available, it cannot be directly forwarded to the LLM due to token limits. For tasks requiring vast amounts of knowledge, including everything within the token limit is impractical [GXG+23].

To deal with this issue, Retrieval-Augmented Generation (RAG) comes into play. RAG is an innovative approach designed to enhance LLM performance by integrating external knowledge retrieval mechanisms. Instead of relying only on the pre-trained knowledge within the model, RAG dynamically retrieves relevant information from external sources during the generation process. This method significantly improves the reliability and accuracy of the responses generated by the model [LPP+20].

The RAG framework operates in two main stages: retrieval and generation. First, it retrieves relevant documents from a large corpus using a retriever model. Then, it uses a generator model to produce coherent responses based on the retrieved information. Through this technique, „hallucinations“ can be reduced. A retrieval model can be any available LLM, such as *ChatGPT*, *Mistral*, or *Llama* models [SPC+21].

The whole process involves multiple steps, starting with dividing the data into chunks. Then, embeddings are used to represent the data chunks as vectors, which are stored in a vector database. Lastly, a query is used to fetch relevant data.

Chunking

Chunking is the process of splitting text into shorter sub-texts, which are later fetched as context for the LLM. The most straightforward way to create chunks from a text is to create fixed-sized segments by using a sliding window. Fixed-sized chunking is quick, easy, and effective for text that does not have a strong semantic structure. However, the fixed-size window can result in cutting off paragraphs or sentences and splitting up text that belongs together [Mic24].

As an alternative, semantic chunking can be done by including information like header structure, tables, chapters, and paragraphs in the chunking strategy. One way to achieve this is to convert data into a Markdown format and then split the text based on Markdown elements like headers. Using Markdown text format for RAG ensures more accurate and

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

relevant results because it supplies richer data structures and more relevant data chunk loads to your LLM [Lan24b] [Mic24].

Embeddings

Once the available data is chunked, it can be converted into embeddings. Embeddings are numerical representations of text that capture the semantic meaning of the text. Each text chunk is transformed into an embedding using a pre-trained language model, such as BERT, GPT, or others. These models convert the text into high-dimensional vectors, which encapsulate the semantic meaning of the text. As a result, similar texts will have similar embeddings [Ip24] [GXG+23].

Vector Databases

A vector database is designed to store and manage these high-dimensional vectors efficiently. Each embedding is indexed with a unique identifier. The ID helps retrieve the embedding quickly when needed. Inside a vector database, a vector search can be performed. Typically, a similarity search based on the distance between vectors is performed, such as cosine similarity or the Euclidean distance [Rac17]. By calculating these distances, the database can identify which vectors are most similar to a given query vector. This capability is beneficial for LLMs, where finding a similar context to a query improves the outcome. The efficiency and speed of vector databases make them indispensable in handling large-scale data and complex queries [JSH+24].

3.1.6 Agents

Multi-agent tools are a promising technique that can further enhance the capabilities of LLM applications. One example is the open-source framework AutoGen, developed by Microsoft [WBZ+23]. AutoGen provides a high-level abstraction for using LLMs, enabling the creation of multiple agents powered by an LLM that can interact in a chat. These agents can solve tasks autonomously, improving their capabilities by iterating through multiple steps, including action plans and reasoning. This approach, described by Yao et al. in the ReAct schema, demonstrates increased task-solving efficiency compared to using a single prompt [YZY+23]. Another technique described by Shinn et al. is Reflexion. This method involves agents verbally reflecting on their results and producing better outputs by maintaining their reflections in memory [SLG23]. Other available frameworks that provide agents are CrewAI [cre24] and Agents from *Langchain* [Lan24a].

Another reason why agents can enhance the capabilities of LLMs is that they can be equipped with tools. These tools enable them to perform web searches or execute code, thereby extending the knowledge of LLMs beyond their training data. Cognition-AI exemplifies this capability with their framework Devin [Wu24].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

3.2 Related Work

This section reviews scientific literature to identify and discuss related work in the field. By examining various research papers, a comprehensive overview of existing methodologies, advancements, and applications relevant to this work is created. This analysis helps to contextualize the work within the broader academic landscape and highlight the contributions of previous studies.

3.2.1 Rule-Based Fault Detection

Rule-based fault detection is a well-researched field. In many domains, like the building sector, rule-based systems exist to detect faults. A significant amount of research has been conducted to develop rules applicable to AHUs and HVAC-systems. Schein et al. developed a rule-based fault detection method for AHUs and HVAC-systems [SBCH06] [SB06]. It consists of a set of expert rules capable of detecting faults.

Another example is the work of Guo et al., who developed a rule-based fault diagnosis strategy for air conditioning systems. Through expert knowledge and characteristics of the system, they were able to derive 22 expert rules [GWC$^+$19]. Similar to this, Wang et al. created a fault detection and diagnosis strategy for AHUs [WC16].

Bruton et al. also explored an expert rule-based AFDD tool for AHUs [BCO$^+$13]. Also, the AHU InFO tool developed by Bruton et al., a rule-based expert system for fault detection and diagnostics for AHUs, shows that rule-based approaches lead to good results [BCR$^+$15].

Furthermore, Ahmed et al. developed a rule-based fault detection tool for AHUs [AAED$^+$15]. This approach utilizes different sensor data and analyzes the time series data of the sensors using expert rules, thresholds, or statistical features to detect faults.

Additionally, Gwerder et al. developed a system for running integrity checks on data from BACS [GMMH22], which contains several rules applicable to AFDD.

3.2.2 AI for rule and test generation

Dragoni et al. [DVRG16] show that NLP can enable the generation of rules by combining NLP-based semantic analysis with Machine Learning (ML) techniques. This approach uses the strengths of both NLP and ML, enabling the extraction of meaningful patterns and dependencies from complex legal texts, ultimately contributing to the development of machine-readable rules that can enhance the automation and accuracy of legal document processing [DVRG16].

Similarly, Koziolek et al. [KABC24] developed an approach for automated test generation for control logic. They use LLMs to generate test cases, achieving high statement coverage for low-to-medium complexity programs, though manual adjustments are often needed due to erroneous assertions [KABC24].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

3.2.3 AI in BACS

Furthermore, the latest developments show that the field is moving towards AI-based fault detection systems. Recent studies have demonstrated that AI, including ML and DL models, can be effectively adopted for FDD in BACS. These AI-based methods offer several advantages over traditional approaches, particularly in HVAC systems.

Bi et al. illustrate how AI is utilized in HVAC fault detection. Their work is a comprehensive and systematic literature review covering the period from 2013 to 2023. This review aims to synthesize existing research on FDD methods for HVAC systems, categorizing them into four main types: traditional ML-based methods, DL based methods, hybrid AI model-based methods, and physical model-based methods [BWY+24].

Furthermore, research in the field indicates that LLMs are capable of generating rules. Zhu et al. state that LLMs can learn rules by being prompted with examples and intermediate steps, which allows them to perform various reasoning tasks more effectively [ZXC+23]. This capability is crucial for applications that require logical reasoning and decision-making. Luo et al. introduce a technique to extract rules from Knowledge Graphs (KGs), utilizing the semantic and structural information within KGs to prompt LLMs to generate logical rules [LJX+23]. This method, known as ChatRule, involves an LLM-based rule generator, a rule ranking module, and a rule validator to ensure the logical correctness of the generated rules.

These advancements illustrate the potential of LLMs and NLP in automating complex reasoning processes and generating interpretable rules from large datasets. However, research needs to be done that combines rule-based fault detection methods with LLMs to generate rules for AFDD. Using LLMs for the rule generation is a novel approach [BWY+24].

3.2.4 RAG Pipelines

Multiple papers provide insight into how to best incorporate RAG into an LLM system. While Lewis et al. establish the foundation for RAG [LPP+20], subsequent research explores advanced techniques.

Gao et al. explore various approaches to implementing RAG. They begin with a naive RAG pipeline and progress to advanced RAG and modular RAG pipelines. However, the fundamental steps of embedding, retrieval, and generation remain central to all the pipelines presented [GXG+23].

Joshi et al. introduce a method for handling text, tables, and images within a RAG pipeline. They address the challenge of maintaining information from images throughout the pipeline [JGKS24].

3.2.5 AI Agents

The use of LLM agents is a promising approach to enhance the capabilities of LLMs. Several researchers showed that using multiple agents together helps solve complex tasks.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Talebirad et al. [TN23] show that by leveraging a collaborative environment where multiple intelligent agent components, each with distinctive attributes and roles, work together, complex tasks can be handled more efficiently and effectively. By modeling various domains such as courtroom simulations and software development scenarios, they showcase the potential applications and benefits of their proposed multi-agent system [TN23].

Xi et al. [XCG+23] demonstrate that LLM-based agents can be effectively utilized in various applications by leveraging a general framework comprising three main components: brain, perception, and action. They explore the extensive applications of these agents in single-agent scenarios, multi-agent scenarios, and human-agent cooperation [XCG+23].

Also, specific complex problems like challenging math problems can be solved better by using agents. Wu et al. [WJZ+23] demonstrate that employing a conversational problem-solving framework called MathChat, which consists of an LLM agent and a user proxy agent, can effectively address mathematical problems. The collaboration initiates a problem-solving process, allowing agents to communicate effectively to address the issues at hand. [WJZ+23].

3.3 Architecture of the AI-Powered Rule Generation System

Using the use case and the knowledge base, an architecture is created for the AI-powered rule generation system. Figure 3.1 visualizes the created architecture. It shows the existing rule-based AFDD system as well as all necessary components for the rule generation. The architecture shows all necessary input data. Further, it includes a RAG pipeline similar to what was done by Lewis et al. in their paper about RAG for knowledge-intensive NLP tasks [LPP+20]. On top, the architecture contains an AI-agent system. This system is based on systems like AutoGen, which was presented by Wu et al. in a paper about LLM applications via Multi-Agent Conversation [WBZ+23].

Existing Rule-Based AFDD-System

At the top of Figure 3.1, the existing rule-based AFDD system is shown, which was introduced earlier. It consists of a building system on the left, a rule-based fault detection part in the middle, and a fault diagnostics part on the right. The AFDD system may also incorporate auto-correction to enhance the building system after identifying and diagnosing errors. However, this work focuses on the rule-detection part. New rules are generated for this existing system and added to the rule-detection step.

Input Data

Below the AFDD system, Figure 3.1 shows a set of necessary inputs for the AI-powered rule generation system. The inputs include a semantic description of the AFDD system,

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

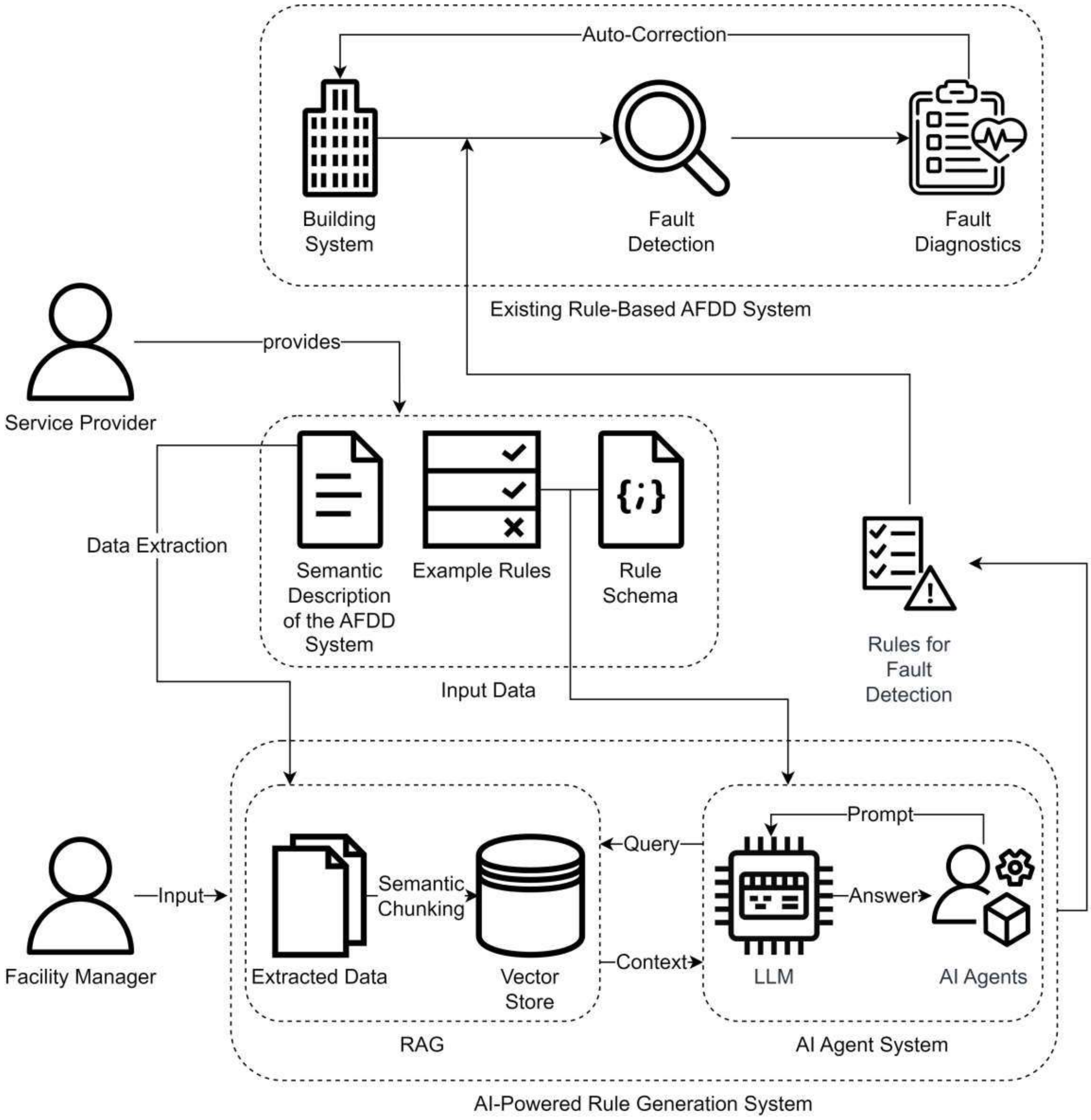

Figure 3.1: The Architecture of the AI-Powered Rule Generation System

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

example rules, and the schema for building the rules. The example rules and the rule schema, as well as the description of the AFDD system, can be obtained from the existing building system with the assistance of a service provider.

The first input is the semantic description of the AFDD system. This description contains one or more documents explaining in natural language how the AFDD system operates. These documents should explain how rules are applied to detect and diagnose faults automatically. The description is essential to aid the LLM in understanding how the system functions and what the rules look like. This is essential because, as mentioned in Section 3.1, AI only works with the help of data. This is in line with the third requirement, which specified that the use of external knowledge is fundamental to the system.

The next input data consists of example rules. These rules can be derived from the documentation or directly from the system itself. The rules must be provided because the capabilities of the AI are enhanced when using a few-shot prompt, where examples are explicitly shown to the LLM. Therefore, a set of existing rules is created and presented to the LLM as examples. The example rules shown to the LLM should be similar to the rule that is tried to be generated. To achieve this, examples are chosen based on semantic similarity. Using examples is the first step towards ensuring consistency with the existing AFDD system, as it was defined in the fourth requirement.

The rule schema is the final part of the input data. The service provider or facility manager provides the schema, which then serves as a description of how the rule is constructed. This description helps to define the scope in which the rule can be created. It shows the structure of the rule and includes specifying which values can exist in the rule and the possible data types for specific fields. Also, this aims to ensure consistency with the existing AFDD system, but it also is important to fulfill the second requirement, which specifies that rules should be directly applicable to the existing AFDD system.

AI-Powered Rule Generation System

The mentioned inputs are then utilized in the AI-powered rule generation system, which consists of two major parts: the RAG pipeline and the AI-agent system. Both parts are powered by an LLM. The AI-powered rule generation system is shown at the bottom of Figure 3.1.

As input for the AI-powered rule generation system, input from a facility manager or system operator is used. These people know the building and can provide input for new rules that are meaningful for the building at hand. This input triggers the system to create a new fault detection rule for the original AFDD system.

RAG Pipeline

The RAG pipeline is used to retrieve relevant information from the semantic description of the AFDD system. It ensures that the LLM has access to the necessary knowledge to generate new rules for the AFDD system. The pipeline consists of three main

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

components: data extraction, chunking, and embedding. It can be found on the left side of the AI-powered rule generation system in Figure 3.1.

To be able to use the provided data, the information must be extracted in a machine-readable format. Most of the provided documents from above are typically in PDF format. The unstructured and non-machine-readable nature presents a significant challenge for AI tools attempting to extract and utilize the information within. Therefore, it is necessary to convert the PDF into text and also to consider images and tables included in the PDF [CD15].

As soon as the data is available in text format, it can be chunked into smaller subsets of text. The chunking component splits the input data into smaller parts based on the semantic structure of the provided text. This indicates that if a headline exists in the original file, the content below this headline and up to the next headline is utilized as a chunk, allowing for the creation of paragraphs. Those paragraphs include information that belongs together, and the information will stay together when used later. This step is shown in the figure as semantic chunking.

Now, the paragraphs, also called chunks, can be converted from text to embeddings. These embeddings are stored in a vector database, which allows for efficient retrieval of relevant information. By also embedding a query and providing it to the vector database, similar information to the query can be retrieved and used as context for the LLM.

AI Agent System

The AI agent system is responsible for generating new rules based on the input of the facility manager. On top of that, the system uses the retrieved information from the RAG pipeline, as well as the example rules and the rule schema, to generate new rules. The system consists of two agents, each with a specific role and attributes. These agents work together to generate new rules for the AFDD system. The agents are powered by an LLM. They send prompts to the LLM internally and automatically until the agent system decides that a suitable answer to the initial input can be given.

The definition of the agents is shown in Table 3.1. It consists of a role, a goal, and a backstory, which is defined in natural language. Those agents are combined and work on two tasks. The Rule Creator (Agent 1) does Task 1 and the Rule Validator (Agent 2) does Task 2. The tasks are shown in Table 3.2.

LLM

Next to the AI agents, the LLM is shown in Figure 3.1. The LLM is the core component of the system, powering the AI agent system by processing the prompts generated by the agents. It is responsible for understanding these prompts, generating responses, and providing the agents with the information needed to create and validate rules. The LLM is crucial for the rule generation process, as it comprehends the context of the input data and produces meaningful responses based on the examples and schema provided. It can

	Role	**Goal**	**Backstory**
Agent 1	Rule Creator	Your goal is to create an integrity check rule. This rule ensures that a specific function of the building operates correctly. You will be given a standardized template of the rule alongside which what specific function should be tested.	You are an expert in building automation, with knowledge of Heating, Ventilation, and Air Conditioning (HVAC) systems as well as Air Handling Units (AHUs). You have extensive experience in creating rules for integrity checks in buildings. Over the years, you have gained substantial experience in formulating effective rules for conducting integrity checks in buildings.
Agent 2	Rule Validator	Your goal is to check if a rule fulfills the syntactic and semantic requirements of the query. You will be provided with a rule and a query. Your objective is to analyze the rule and determine if it satisfies the query.	You are an expert in building automation, with knowledge of Heating, Ventilation, and Air Conditioning (HVAC) systems as well as Air Handling Units (AHUs). You have extensive experience in validating rules for integrity checks in buildings. Over the years, you have gained substantial experience in finding flaws in integrity check rules.

Table 3.1: Definition of the AI-Agents

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

	Description	**Expected Output**
Task 1	You are provided with the rule template: <ruletemplate>. Your specific task is to identify an appropriate rule that satisfy the query: <query>. Your objective is to analyze the given information, apply your understanding, and determine a rule that aligns with the requirements specified in <query>. Context: <context>. Examples: <examples>	The final result should be a rule following the rule template and satisfying the query with explanation why this rule is fitting the query.
Task 2	You are provided with a rule and an explanation. Your task is to output only the json in correct format.	The final result should be a rule in correct json format. The output should be a raw json object.

Table 3.2: Definition of the Tasks for the AI-Agents

handle multiple requests from the agents as long as they remain unsatisfied with the generated rules.

Output: Rules for Fault Detection

Finally, the output of the AI-powered rule generation system is shown in between the AI-powered rule generation system and the existing rule-based AFDD system in Figure 3.1. The output contains a new rule for the AFDD system. This new rule should follow the schema of existing rules, allowing it to be directly integrated into the AFDD system. By providing a schema to the AI agent and specifying that the output should conform to this schema, the generated rules should be syntactically correct. Those requirements are internally passed to the LLM by the agents.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 4

Design Cycle

This chapter describes the implementation of the AI-powered rule generation system. The outlined architecture serves as the foundation for creating an artifact capable of generating rules for the AFDD system. The chapter describes all methods and technologies involved, starting with an overall description of the open-source libraries that are used. Further, the realization of each part of Figure 3.1 is described. The description includes data extraction, a RAG pipeline, the usage of examples and schemas, and the implementation of an AI agent system. The whole design is based on the knowledge base and the architecture that was created earlier.

Various methods and approaches were evaluated to implement the AI-powered rule generation system. The focus of this work is on the resulting architecture and its overall functionality. While different methods for automatically generating rules were considered, this work does not present a systematic examination of all possibilities. Instead, the emphasis is on the integration and interaction of the individual components within the architecture. It is acknowledged that each component can be further optimized, and future work may explore these optimizations in more detail.

4.1 Open Source Libraries

The implementation was done entirely in Python and can be run from the command line. The program requires several open-source libraries, which are described below.

Langchain

Langchain[1] serves as a foundation for building the artifact. It is a framework that offers libraries and tools to create applications that interact with LLMs.

---

[1] https://www.langchain.com/

Huggingface

Huggingface[2] is an open-source platform that provides access to a wide range of LLMs and other AI models. It offers a variety of models that can be used for different tasks, such as text generation, text classification, text summarization, and text embedding. Huggingface also provides tools for fine-tuning models on custom datasets. The platform is a valuable resource for developing open-source projects.

Ollama

Ollama[3] is a library that provides access to LLMs and their capabilities. It allows users to interact with LLMs and generate text based on input data. Additionally, *Ollama* facilitates the use of embedding models, enabling the creation of embeddings for vector databases.

CrewAI

CrewAI[4] is used to build workflows based on AI-agents. It allows the creation of multiple agents that can work together as a crew and interact with LLMs.

4.2 Data Extraction

To use the input data effectively, it is essential to extract information from the PDF files that describe the AFDD system. Python provides tools such as pdfplumber and *PyMuPDF* for extracting raw text from PDF documents. However, this extraction process often results in the loss of the document's structure. It is important to consider structural elements, such as headings and paragraphs, to understand how different text elements relate to one another. The context of headings, paragraphs, and line breaks must be maintained to retain semantic information. The library *pymupdf4llm* is particularly suitable for this purpose, as it can convert PDF documents into Markdown format while preserving the original document's structure. In this conversion, headings from the PDF are represented as headers in Markdown, paragraphs are converted into text blocks, and line breaks are retained. This method allows the structural integrity of the semantic description of the AFDD system to be maintained, enabling the extracted data to be used for further processing in the RAG pipeline.

Another essential part of the data extraction process is the extraction of tables. Tables are a common way to present structured data in documents. Extracting tables from PDF files can be challenging because the PDF's format is optimized for rendering rather than data parsing. It only contains lines and character glyphs, which humans tend to interpret as tables. Still, *pymupdf4llm* is able to extract tables from PDF files based

---

[2] https://huggingface.co/

[3] https://ollama.com/

[4] https://crewai.com/

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

on the structure of the PDF and patterns that occur in the document. The extracted tables are then converted into Markdown table format. Another approach for extracting tables from PDFs is to use DL-models as a base for the extraction. The model can be trained on a dataset of PDF files and their corresponding tables. The model can then be used to extract tables from new PDF files. An example of this approach is the Marker library for python [Par24]. For this project, the decision was made to use *pymupdf4llm* for table extraction. This choice was based on its efficiency and the comparable quality of results to the Marker library, making it a less resource-intensive approach. DL-based approaches, on the other hand, require a lot of computational resources for inference. However, the tables were not always extracted correctly. To improve the extraction of the tables, the tables were fed into an LLM. For this, the *mistral-7b-instruct* model was used. It is an open-source model with a context length of 8192 tokens. It is a relatively lightweight model with solid capabilities. The *mistral-7b-instruct* model was a good fit for this task. The tables were corrected with the prompt „The following information was taken from a Markdown table. Please use the information and put it in a Markdown table format. Information:“. The model was able to correct the extracted tables to some extent, and the corrected tables were then used for further processing.

Furthermore, images in PDF files may hold valuable information that needs to be preserved. In such cases, LLMs can be utilized to generate textual descriptions of the images, which can then be stored alongside the document's text. *Llava*, a well-known model, excels at this task. On top of that, it is essential to ensure that the context in which the images appear is maintained.

However, when using images in documents from BACS, the image description was not very helpful because the model was unable to interpret figures as a human would. Specifically, for flow charts or graphs, the LLM struggled to understand and convey the information depicted. While the LLM could occasionally recognize the general content of an image, it frequently failed to provide accurate descriptions and, in some cases, hallucinated and generated unrelated information. For instance, when examining the flow chart „Data Check Workflow“ from the publication by Gwerder et al. [GMMH22], the LLM was unable to provide a correct description of the image. In fact, it introduced additional information that was not present in the figure. As a result, the images were excluded from further use, and only the text and tables were considered.

4.3 Implementation of the RAG Pipeline

After turning the description of the AFDD system into a structured Markdown file, the RAG pipeline was implemented. The Markdown document serves as the input for the pipeline. The pipeline consists of several stages, which are described in the following sections. Figure 4.1 illustrates how the different stages interact with one another.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

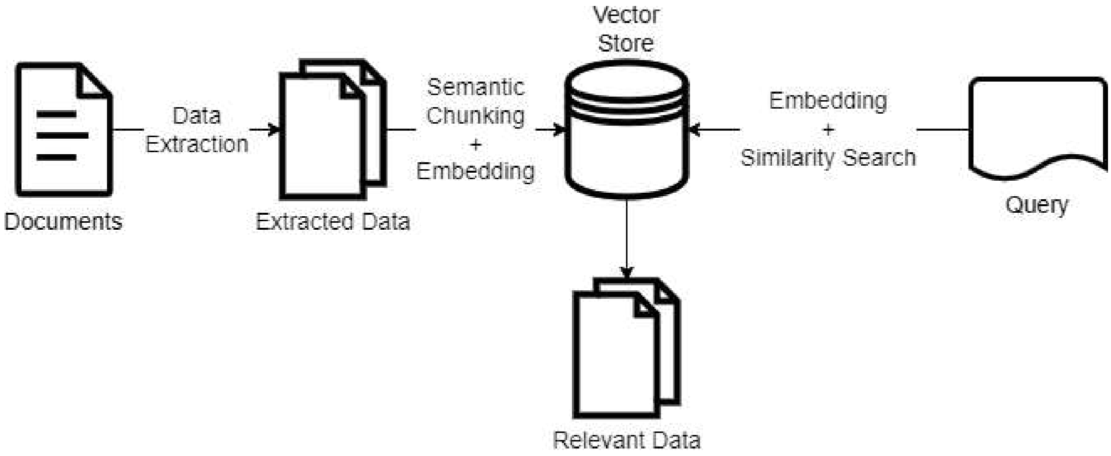

Figure 4.1: Detailed Representation of the RAG Pipeline [GXG+23]

```
markdown_splitter = MarkdownHeaderTextSplitter(
    headers_to_split_on=[("#", "h1"), ("##", "h2")],
    strip_headers=False,
    return_each_line=False,
)
chunks = markdown_splitter.split_text(full_text)
```

Listing 4.1: Definition of the Markdown Text Splitter

4.3.1 Semantic Chunking

The first step in the RAG pipeline is to split the input into chunks. Because the initial structure of the PDF document is preserved, the splitting can be done based on this structure. This is done by identifying the headings in the Markdown document and splitting the document into sections based on these headings. The headings are used as delimiters to separate the document into chunks. Each chunk represents a section of the document that contains a specific topic or information. The chunks are then processed individually in the following stages of the pipeline.

In Python, the text-splitters library from Langchain helps with this. The class MarkdownHeaderTextSplitter is beneficial for dividing the Markdown document into chunks based on the headings. By defining the headings that should be used as delimiters and giving the Markdown document as input, the library returns a list of chunks. Listing 4.1 shows this process.

For the chunking of the semantic description of the AFDD system, the headers h1 „#“ and h2 „##“ were used as delimiters. Having more detailed headers could overly fragment text that contains similar content.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

```
from langchain_community.embeddings import (
    HuggingFaceBgeEmbeddings
)
model_name = "BAAI/bge-small-en-v1.5"
model_kwargs = {"device": "cpu"}
encode_kwargs = {"normalize_embeddings": True}
embeddings = HuggingFaceBgeEmbeddings(
    model_name=model_name,
    model_kwargs=model_kwargs,
    encode_kwargs=encode_kwargs
)
```

Listing 4.2: Definition of the BGE Embedding

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

4.3.2 Embeddings

The next step in the pipeline is to generate embeddings for the chunks. Embeddings are vector representations of text that capture the semantic meaning of the text. These embeddings are used to compare the similarity between different chunks and to identify related chunks. The embeddings are generated using a pre-trained model. The model is used to encode the text into a vector representation that captures the semantic meaning of the text. The embeddings are then stored in a vector database for later use. For creating the embeddings of the semantic description of the AFDD system, the open source model *bge-small-en-v1.5*[5] from the Beijing Academy of Artificial Intelligence (BAAI) was used. Other models like bge-m3[6], e5-small[7] or gte-base[8] could be used as well. However, the *bge-small-en-v1.5* model was chosen because it is well-suited for generating embeddings for English texts in a reasonably short time without needing a lot of computational resources. As this field is moving fast, many other capable models are available, which would be similar to the described embedding task. The Huggingface-MTEB-Leaderboard[9] provides information about the latest models and their performance.

The embeddings were generated with the help of the Langchain Community Library HuggingFaceBgeEmbeddings on the CPU. Listing 4.2 shows how the embedding model was initialized.

4.3.3 Vector Store

After the embedding model is set up, the embeddings can be created and stored in a vector store. The vector store is a database that stores the embeddings of the chunks. The vector store is implemented using the *Faiss*[10] library, which is a library for efficient

---

[5] https://huggingface.co/BAAI/bge-small-en-v1.5

[6] https://huggingface.co/BAAI/bge-m3

[7] https://huggingface.co/intfloat/e5-small

[8] https://huggingface.co/thenlper/gte-base

[9] https://huggingface.co/spaces/mteb/leaderboard

[10] https://ai.meta.com/tools/faiss/

```
from langchain_community.vectorstores import FAISS
vectorstore = FAISS.from_documents(chunks, embeddings)
```

Listing 4.3: Definition of the FAISS Vectorstore

```
query_chunk = "Check if the room temperature decreases when the cooling
    coil valve is opened"
similar_chunks =
    vectorstore.similarity_search(query_chunk, k=5)
```

Listing 4.4: Exemplary Similarity Search

similarity search and clustering of dense vectors. The *Faiss* library provides a fast and memory-efficient way to store and retrieve embeddings. The *Faiss* vector store is implemented with the help of the Langchain Community Library *Faiss*[11]. Listing 4.3 shows how the embeddings are created and stored in the vector store. The chunks passed to the vector store are the chunks created earlier, and the embedding is the *bge-small-en-v1.5* model.

4.3.4 Similarity Search

With the creation of the vector store and the integration of embeddings, the vector store is ready to be used for similarity search. The similarity search is used to find chunks that are similar to a given query chunk. The similarity search is performed by comparing the embeddings of the query chunk with the embeddings of the chunks in the vector store. The similarity search is also implemented using the *Faiss* library. An example of how to perform a similarity search is shown in Listing 4.5. The variable k specifies the number of similar chunks that should be returned. In this example, the similarity search is performed with a query chunk, and the five most similar chunks are returned. Later on, the vector store is used for similarity search to improve the rule generation process. With all those steps, the RAG pipeline is implemented and ready to be used for generating rules for the AFDD system.

4.4 Integration of the Example Rules

In addition to the semantic description of the AFDD framework, it is essential to integrate example rules into the AI-powered rule generation system. For this purpose, examples are extracted from the original AFDD system. These rules are then provided to the AI-powered rule generation system in a machine-readable format, e.g., JSON or XML.

However, since it is not feasible to provide all rules as context to the LLM due to the token limit, a subset of the rules was used. This subset was chosen based on the rules that

---

[11] https://python.langchain.com/docs/integrations/vectorstores/faiss/

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

```
from langchain_core.example_selectors import (
    SemanticSimilarityExampleSelector
)
selector = SemanticSimilarityExampleSelector.from_examples(
    examples,
    embeddings=bge_small_en_v15_embeddings(),
    vectorstore_cls=FAISS,
    k=4,
)
```

Listing 4.5: Definition of the SemanticSimilarityExampleSelector

```
from pydantic import BaseModel

class Rule(BaseModel):
    model_config = ConfigDict(extra="ignore")
    test_method: Optional[str]
    test_preconditions: Optional[dict] = None
    time_start_utc: Optional[str]
    time_end_utc: Optional[str]
    violation_plausibility_code: Optional[str] = ""
```

Listing 4.6: Definition of a Pydantic Model

are most relevant to the provided input of the AI-powered rule generation system. The semantic similarity was achieved with the help of the SemanticSimilarityExampleSelector class from the Langchain Core library. The class uses embeddings of the rules and an embedding of the input to select the most similar rules. Listing 4.5 shows how the SemanticSimilarityExampleSelector class is used. For the embeddings, the *bge-small-en-v1.5* model was used, and as a vector store, *Faiss* was chosen. The variable k specifies the number of similar rules to be returned. In this example, the four most similar rules are returned.

4.5 Usage of the Rule Schema

To ensure that the LLM generates rules in the correct format, it is essential to incorporate a rule schema into the AI-powered rule generation system. The rule schema outlines the structure of the rules that the LLM should create. This schema is implemented as a Pydantic model. A Pydantic model is developed by defining a Python class that inherits from the Pydantic BaseModel class. This class contains fields that specify the structure of the rule. Listing 4.6 shows how an implementation of a rule schema could look like.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

```
rulecreator = Agent(
    role="Rule Creator",
    goal="Your goal is to ...",
    verbose=True,
    allow_delegation=False,
    llm=mistral_llm,
)
```

Listing 4.7: Agent Structure

```
task = Task(
    description=f"You are provided...",
    agent=agent1,
    output_pydantic=rule_type,
)
```

Listing 4.8: Task Structure

```
crew = Crew(
    agents=[rulecreator, validator],
    tasks=[task1, task2],
    verbose=2,
)
```

Listing 4.9: Definition of a Crew

4.6 Implementation of the AI Agent System

As the core component of the AI-powered rule generation system, the AI agent system is responsible for generating rules. It is implemented using the CrewAI library. CrewAI provides tools for creating AI agents that can interact with LLMs. The AI agent system consists of multiple agents that work together to generate rules for the AFDD system. The agents perform different tasks in the rule-generation process and communicate with each other to exchange information and coordinate their tasks. The definition of those agents and their tasks was described in Section 3.3. Again, the *mistral-7b-instruct* model was used.

In Python, various classes from the CrewAI library were used to create the agents. An agent is defined via the Agent class, as shown in Listing 4.7. A task is defined similarly to an agent and is shown in Listing 4.8.

At this point, the rule schema is integrated into the agent system. The *output_pydantic* variable specifies that the output of this task must strictly adhere to the rule schema. In total, two agents and two tasks exist. The agents and tasks are combined into a crew as shown in Listing 4.9.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

```
Question: The test checks ...
Answer: {...}
Question: The test checks ...
Answer: {...}
Question: Generate a rule to check if ...
Answer:
```

Listing 4.10: Few Shot Prompt Structure

4.7 Few-Shot Prompting and AI Agents Execution

To integrate all components and generate rules for the AFDD system, the AI agent system is combined with an LLM, a RAG pipeline, example rules, and a few-shot prompt. The rule generation process is initiated by providing an input formulated in natural language that describes the rule to be created.

First, this input is processed through the RAG pipeline. Within the semantic description of the AFDD system, the three most similar chunks of information are selected. These chunks serve as a context for the next steps.

Next, the SemanticSimilarityExampleSelector is utilized to create a few-shot prompt. This prompt includes example questions and their corresponding answers, where the question describes a rule and the answer is the rule itself. Four rules that are most similar to the input are selected using the SemanticSimilarityExampleSelector. The few-shot prompt is then constructed, concluding with another question that restates the input. The answer is left blank and is intended to be filled in by the LLM. The structure of the few-shot prompt follows the format shown in Listing 4.10.

After that, both the few-shot prompt and the derived context are passed to the AI agent system. As seen in Section 3.3, the descriptions of the agents and tasks consist of some placeholder variables. With the few-shot prompt, the context, and an empty instance of the rule schema, those placeholders are filled with concrete information. Now, the crew of agents is ready to be executed. The agents now work together to generate a rule based on the input, the context, and the few-shot prompt. The goal is to generate the answer to the last question of the few-shot prompt using the provided information. The first agent tries to do this, and the second agent validates the generated rule and puts it into the correct format. With this process, a new rule following the input can be generated and returned to the original rule-based fault detection system.

Listing 4.11 exemplifies how the output of an execution of the created AI-powered rule generation system looks. Some inputs and outputs are replaced by variables enclosed in sharp brackets to simplify reading. The listing outlines the various steps involved in creating a rule.

First, a relevant context that matches the provided input is retrieved. Next, examples that are similar to the input are selected from a list of all available examples. Finally, the AI agent system is initiated. The first component, called the „Rule Creator“, is

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

~~~
Retrieving context
Picking best examples
[2024-12-04 10:24:50][DEBUG]: == Working Agent: Rule Creator
[2024-12-04 10:24:50][INFO]: == Starting Task: <task>
    Context: <context>
    Examples: <examples>
    Question: Generate a rule that ...
    Answer:
> Entering new CrewAgentExecutor chain...
    I now can give a great answer.
    Final Answer:
    The rule that can be used in automatic fault detection
    systems to check ... is as follows: <generated-rule>
    Explanation:
    This rule is designed to test ...
> Finished chain.
[2024-12-04 10:25:07][DEBUG]: == Working Agent: Rule Validator
[2024-12-04 10:25:07][INFO]: == Starting Task:
You are provided with a rule and an explanation.
Your task is to output only the json in correct format.
> Entering new CrewAgentExecutor chain...
I now can give a great answer.
Final Answer:
```json
{ <corrected-rule> }
```
> Finished chain.
[2024-12-04 10:25:22][DEBUG]: == [Rule Validator]
Task output: { ... }
~~~

Listing 4.11: Execution of the AI-Powered Rule Generation System

activated. The agent receives its task, along with the retrieved context and the selected examples. After completing one or more steps (in this case, just one step), the agent produces an answer. In addition to the answer, the agent also provides an explanation, which the second agent later utilizes. Once the first agent has completed its task, the second agent, known as the „Rule Validator“, is activated. This agent is also given a task and additionally utilizes the input from the „Rule Creator“. Using this information, the „Rule Validator“ is able to generate a correct JSON output, which is included in the task output. The whole process is shown in Figure 4.2.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

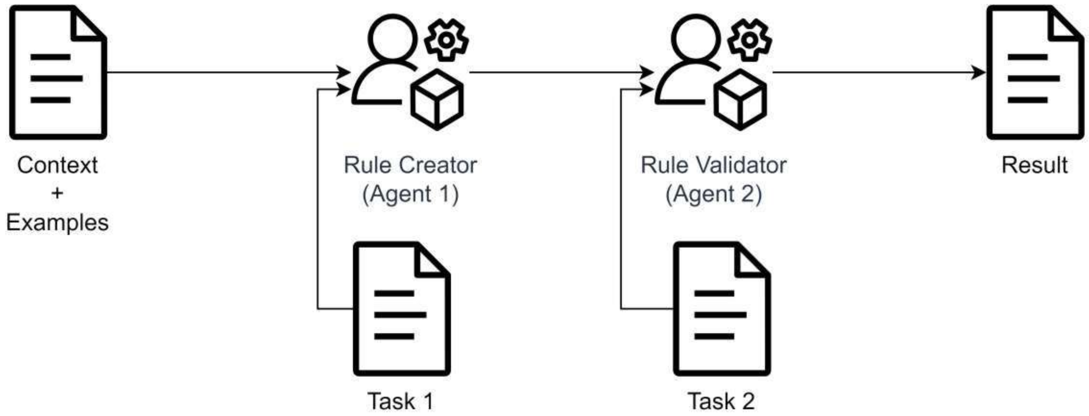

Figure 4.2: Detailed Representation of the AI-Agent System

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 5

Evaluation

The initial goal of this research is to explore how different AI approaches can be combined to generate rules for AFDD systems. By defining an architecture and implementing an artifact that should generate rules for AFDD systems, the research questions are addressed. The use of an LLM within an AI agent system, combined with a RAG pipeline, results in a functioning system. Moreover, incorporating examples and effectively using LLMs through few-shot prompting plays a crucial role in the development of the AI-powered rule generation system. Additionally, defining a rule schema is vital for producing rules in the correct format.

An evaluation was conducted to determine whether the created architecture and the implemented artifact can help detect faults in AFDD systems. A real system is used to measure how effectively the AI-powered rule generation framework works. The chosen system is the framework for data integrity checks for BACS developed by Gwerder et al. [GMMH22]. For the evaluation purpose, the framework itself, a description of the framework, a schema describing the structure of the rules, and existing rules are used.

5.1 Evaluation Setup

The first step of the evaluation setup is to use the PDF document of the integrity checking paper by Gwerder et al. [GMMH22] as a semantic description of the AFDD system. The framework described in this paper includes a set of integrity checks that ensure data integrity in BACS. These integrity checks, referred to as rules, were leveraged for the evaluation process. There are three different types of rules: „ReactionRule“, „SimilarityRule“, and „SingleSignalRule“. Those rules use time-series data to detect faults. The „ReactionRule“ uses two signals to determine if one signal influences another. For example, it checks whether the temperature in a room rises when the heating in the HVAC system is activated. The „SimilarityRule“ also uses two signals but compares them for similarity. An example of this would be when the temperature of the concrete within the wall is similar to the room temperature. The „SingleSignalRule“ evaluates a single signal to determine its validity. For instance, it ensures that the room temperature never exceeds a specified threshold. The framework uses JSON files as input, which describe the rules for detecting faults.

The data extraction process converted the PDF document of the paper into chunks. Those chunks were integrated into the RAG pipeline with the help of embeddings and a vector database. Next, the integrity-checking framework itself was used to derive rules in the form of JSON files.

The original AFDD framework used three different types of rules. For each of the rule types, a general rule schema was derived and stored as Pydantic[1] models. The AI-powered rule generation system uses those schemas in the subsequent process. Once these steps are completed, the evaluation can be started. The JSONs are used for the evaluation, rule by rule. The current rule is used for evaluation, while all other rules are used as examples. The evaluation was conducted for each existing rule in the integrity-checking framework. The evaluations are independent of each other and use different inputs and example rules. In total, 33 rules were evaluated. The rule descriptions are typically formulated like the following rule descriptions:

- "The test checks whether the minimal room air quality measurement over longer time periods is close to outside air concentration" [GMMH22]
- "The test checks whether the room temperature measurement reacts on radiator valve position change when heat is provided by associated heat group" [GMMH22]
- "The test checks whether the supply air temperature measurement reacts on heating coil valve position change when heat is provided by the associated" [GMMH22]

The evaluation consists of two parts: a quantitative evaluation and a qualitative evaluation. The quantitative evaluation uses the AI-powered rule generation framework to re-generate existing rules for the data integrity checking framework and compare the generated rules with the expected rules. The qualitative evaluation takes the generated rules and applies them to the data integrity checking framework. It is then evaluated whether the generated rules are functional and lead to the expected results.

All rules that were used for the quantitative and qualitative evaluation can be found in Section 6 in the appendix. There, the provided input, the expected output, and the actual output, which was generated by the AI-powered rule generation system, are presented.

5.2 Quantitative Evaluation

For the quantitative evaluation, all existing rules from the current rule-based AFDD system are used. In this case, the existing system refers to the integrity-checking

---

[1] https://docs.pydantic.dev/latest/

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

```
input = rule["test_description"]
del rule["test_description"]
expected_output = rule["test_description"]
```

Listing 5.1: Creation of the Input and Expected Output

```
{
  "test_method": "reaction_single_window_threshold_test",
  "test_preconditions": {...},
  "test_specs": {...},
  "time_start_utc": "start"
  "time_end_utc": "end"
}
```

Listing 5.2: Structure of the Expected Output

framework. The rules from this framework are divided into two parts: an example split and an evaluation split. The example set is provided to the AI-powered rule generation framework as examples, while the evaluation part serves as the expected output for assessing the generated rule. Only one rule is evaluated at a time. Therefore, the evaluation part always consists of exactly one rule. Each rule includes a description, which acts as the input for the AI-powered rule generation framework, simulating the input typically provided by a facility manager. The remaining fields in the rule are considered the expected output. Listing 5.1 shows how one rule is divided into input and expected output.

After the AI-powered rule generation framework generates a rule, its output is compared to the expected output by checking each field within the JSON structure individually. First, the generated output is verified to have the expected fields. If this is the case, the values are also compared. This comparison is thorough, extending to the deepest levels of the JSON structure. Consequently, string values and numerical values must be equivalent. For example, the JSON values for the statistical feature in the rule are compared: „larger“ vs. „lower“. A percentage is then calculated based on the number of correctly generated JSON fields, indicating the accuracy of the generated rule. The percentage is calculated by dividing the correct values through all values at the lowest level of the JSON. The described process is shown in Figure 5.1. The expected output for the rules mentioned above is in JSON format. Its structure is shown in Listing 5.2.

A quantitative evaluation was conducted for each of the 33 rules. Figure 5.2 presents a stacked bar for each rule. The lower part of the bar indicates the number of correct values in the generated JSON, while the upper part represents the incorrect values. The percentage of correct values for each rule is calculated by dividing the number of correct values by the total number of correct and incorrect values combined. The resulting percentages are displayed in Table 5.1. The percentages range from 0% to 95.45%, with the lowest non-zero value being 25.93%.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

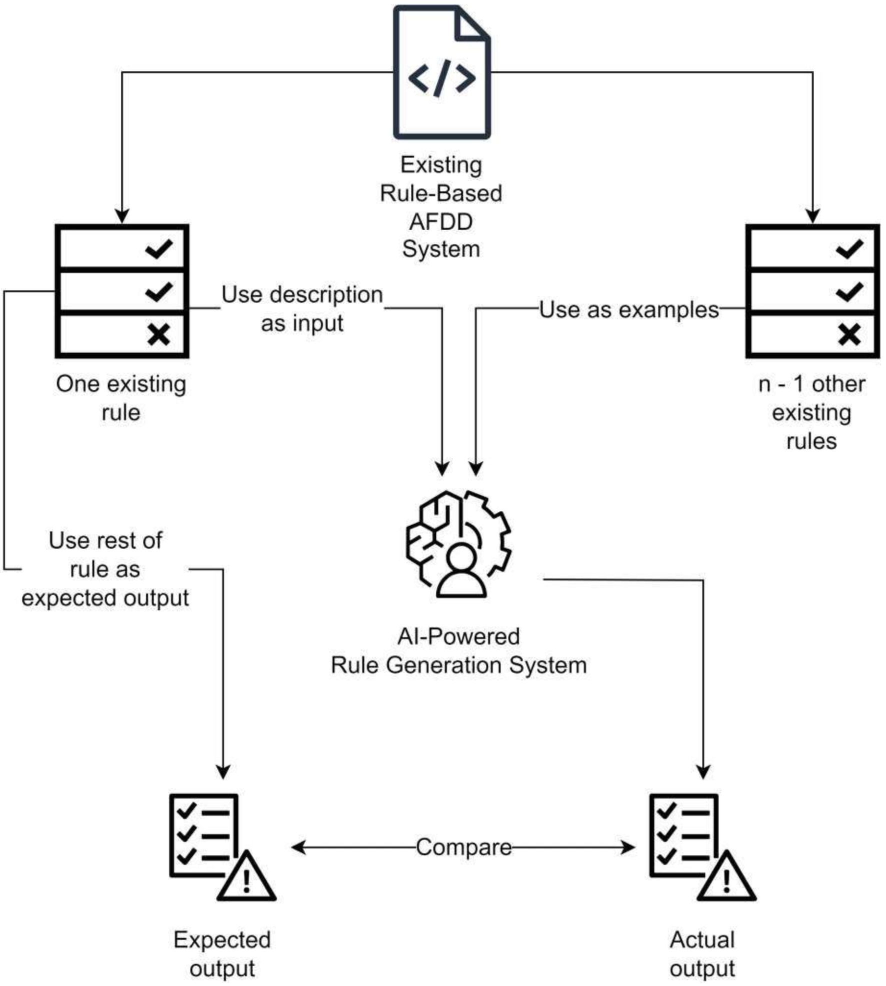

Figure 5.1: The Evaluation of the AI-Powered Rule Generation System

Rule	**Correct Values**	**Percentage**
1. Room Temp 1	21 out of 22	95.45%
2. Concrete Temp 1	7 out of 27	25.93%
3. Window Contact 1	5 out of 14	35.71%
4. Room Temp 2	21 out of 23	91.30%
5. Valve Flow 1	6 out of 18	33.33%
6. Room Temp 3	12 out of 21	57.14%
7. Concrete Temp 2	13 out of 25	52.00%
8. Room Temp 4	17 out of 22	77.27%
9. Room Temp 5	11 out of 24	45.83%
10. Room Temp 6	5 out of 19	26.32%
11. Room Temp 7	21 out of 22	95.45%

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

12. Room Air Quality 1	5 out of 16	31.25%
13. Concrete Temp 3	17 out of 25	68.00%
14. Concrete Temp 4	7 out of 27	25.93%
15. Room Air Quality 2	6 out of 17	35.29%
16. Room Temp 8	0 out of 22	0.00%
17. Air Flow 1	5 out of 18	27.78%
18. Air Flow 2	13 out of 17	76.47%
19. Room Presence 1	8 out of 11	72.73%
20. Air Flow 3	11 out of 15	73.33%
21. Room Air Quality 3	8 out of 13	61.54%
22. Air Flow 4	11 out of 19	57.89%
23. Air Flow 5	13 out of 16	81.25%
24. Air Flow 6	10 out of 18	55.56%
25. Air Flow 7	15 out of 19	78.95%
26. Room Air Quality 4	8 out of 15	53.33%
27. Air Flow 8	15 out of 17	88.24%
28. Room Presence 2	7 out of 11	63.64%
29. Room Air Quality 5	5 out of 15	33.33%
30. Air Flow 9	12 out of 16	75.00%
31. Temp Deviation 1	4 out of 14	28.57%
32. Temp Cooling 1	12 out of 16	75.00%
33. Temp Heating 1	5 out of 16	31.25%

Table 5.1: Quantitative Evaluation Results: Correctly generated JSON Values compared to all JSON Values in a Rule and the Percentage of correct Values

5.3 Qualitative Evaluation

The quantitative evaluation showed how many correct fields were generated by the AI-powered rule generation framework. However, it does not provide information on the quality of the generated rules. Therefore, a qualitative evaluation was conducted to assess whether the generated rules are functional and produce the expected results. This evaluation involved applying the rules to the integrity-checking framework and observing the outcomes. The results were categorized based on the performance of each rule within the original framework. Additionally, for any rules that did not succeed, the evaluation included reasoning for their failure. The following categories were used to evaluate the quality of the generated rules.

Score 0 is given when the generated rule cannot be applied to the integrity-checking framework at all. This indicates that the rule does not meet the minimum requirements to be recognized as a valid input, and as a result, the framework was not executed. The rule might have been in the correct JSON format, but it lacked the necessary fields to satisfy the preconditions for initiating a test run.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

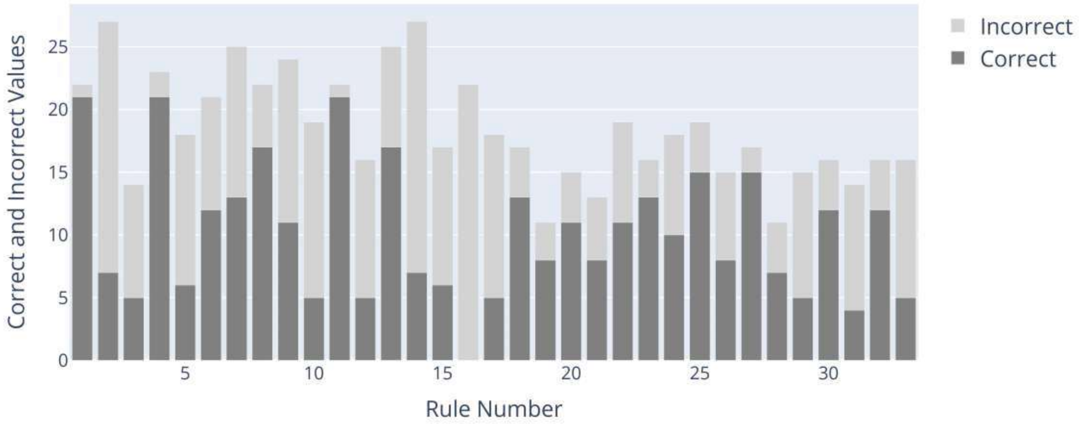

Figure 5.2: Overview over the Quantitative Evaluation showing Correct and Incorrect Values for each Rule

Score 1 indicates that the generated rule met the criteria for execution within the integrity-checking framework. However, when attempting to run the rule, those with a score of 1 failed during the test run. This failure may occur due to missing values or incorrect field parameters that are necessary for executing functions in the checking process. Therefore, the rules with this score did not lead to any results.

Score 2 is assigned when a generated rule meets all the execution criteria and produces a result. However, rules that receive a score of 2 yield an incorrect result, failing to identify the fault for which the rule was designed.

Finally, Score 3 is achieved when the generated rule can be executed successfully and produces the correct result. This can be true even if the quantitative evaluation shows that the generated rule does not match the expected output 100%. As long as the generated rule semantically represents the same meaning as in the expected outcome, it can still receive a score of 3.

Table 5.2 presents the results of the qualitative evaluation. The first column lists the same tests used in the quantitative evaluation. The second column displays the score achieved, while the last column explains the reason for that score.

Rule	**Score**	**Reason**
1. Room Temp 1	1	Test run not successful: Missing values detected and Other error than 'missing values'
2. Concrete Temp 1	1	Test run not successful: Other error than 'missing values'

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

3. Window Contact 1	1	Test run not successful: Other error than ’missing values’
4. Room Temp 2	2	EVENT Warning: no post window match. No events detected!
5. Valve Flow 1	1	Test run not successful: Other error than ’missing values’
6. Room Temp 3	1	Test run not successful: Other error than ’missing values’
7. Concrete Temp 2	1	Test run not successful: Other error than ’missing values’
8. Room Temp 4	1	No measurements found for the projects
9. Room Temp 5	1	Test run not successful: Other error than ’missing values’
10. Room Temp 6	1	Test run not successful: Other error than ’missing values’
11. Room Temp 7	1	Test run not successful: Other error than ’missing values’
12. Room Air Quality 1	1	Test run not successful: Other error than ’missing values’
13. Concrete Temp 3	1	Test run not successful: Other error than ’missing values’
14. Concrete Temp 4	1	Test run not successful: Missing values detected.
15. Room Air Quality 2	1	Test run not successful: Missing values detected.
16. Room Temp 8	0	Could not execute test run: Error in prechecks: ’tests’ not found
17. Air Flow 1	1	Test run not successful: Other error than ’missing values’
18. Air Flow 2	1	Test run not successful: Missing values detected and Other error than ’missing values’
19. Room Presence 1	1	Test run not successful: Other error than ’missing values’
20. Air Flow 3	1	Test run not successful: Other error than ’missing values’
21. Room Air Quality 3	1	Test run not successful: Other error than ’missing values’
22. Air Flow 4	1	Test run not successful: Missing values detected and Other error than ’missing values’
23. Air Flow 5	1	Test run not successful: Missing values detected and Other error than ’missing values’

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

24. Air Flow 6	1	Test run not successful: Missing values detected and Other error than ’missing values’
25. Air Flow 7	1	Test run not successful: Missing values detected and Other error than ’missing values’
26. Room Air Quality 4	1	Test run not successful: Missing values detected and Other error than ’missing values’
27. Air Flow 8	1	Test run not successful: Missing values detected and Other error than ’missing values’
28. Room Presence 2	1	Test run not successful: Missing values detected and Other error than ’missing values’
29. Room Air Quality 5	1	Test run not successful: Other error than ’missing values’
30. Air Flow 9	1	Test run not successful: Other error than ’missing values’
31. Temp Deviation 1	1	Test run not successful: Error in fetch time-series
32. Temp Cooling 1	1	Test run not successful: Other error than ’missing values’
33. Temp Heating 1	1	Test run not successful: Missing values detected.

Table 5.2: Qualitative Evaluation Results: Applying the generated Rules to the Integrity Checking Framework and adding a Score depending on the Outcome

The qualitative evaluation indicates that in nearly all cases, the prechecks of the integrity checking framework were successful, allowing the test execution to be started. Passing these prechecks means that the rule was a valid JSON and met the preconditions of the integrity-checking framework. As a result, 32 out of 33 tests were successfully started. The only exception was test case 16, which could not be executed because the AI-powered rule generation system produced an invalid JSON.

From the 32 tests that were executed in the integrity-checking framework, only one test was successful. All other 31 tests failed during the execution and therefore only reached a score of 1. The reasons for the failure of the tests were often not specified by the framework. However, some reasons were outputted. In 12 cases, missing values were detected. In those cases, the AI-powered rule generation framework was not able to fill up the rule with the necessary fields to execute the test correctly. Errors occurred, especially in fields of datatype *dict*, which allow for arbitrary information, such as the *test_specs* or the *test_preconditions*. When other errors than missing values occurred, the framework was not able to execute all the required steps to detect rules, such as applying statistical features to the data. When running test case 31, the AI-powered rule generation framework was not able to fetch the corresponding time series data necessary for the test execution. This issue occurred due to an incorrect specification of

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

the *time_start_utc* and *time_end_utc* fields by the AI-powered rule generation system. Further, one test (Rule 16.) could not be executed due to missing measurements. Test case 4 was the only one that ran successfully. However, it did not detect any events during this test run, which does not correspond with the expected outcome.

5.4 Potential Improvement Through Refined Human Input

To enhance the AI-powered rule generation system, it might be beneficial to incorporate a human correction process into the system. Currently, the system runs only once for each rule generation. However, providing feedback to users after a rule is generated could be more effective. This approach would encourage users to modify or expand their input if the generated rule does not seem appropriate. To validate this approach, a selection of previously evaluated rules are tested again with extended input.

16. Room Temp 8

Rule number 16 did not lead to any output. It reached the lowest score in the quantitative evaluation. The AI-powered rule generation system failed to meet the task and returned an empty JSON. The initial input for rule 16 was: *The test checks whether the room temperature reacts on a radiator heating valve opening.*

To improve the rule generation process, the input for rule 16 was then extended to be: *The test checks whether the room temperature reacts on a radiator heating valve opening. A precondition for the test is that the associated heat group provides heat.*

By utilizing the extended output, the AI-powered rule generation system produced a rule that had a 40.91% overlap with the expected output, which is an improvement compared to the 0% overlap of the previous run. This demonstrates that a change in the input can lead to better results.

26. Room Air Quality 4

Rule number 26 had 53.33% conformity with the expected output. Therefore, the generated rule had only around half of the variables in the generated JSON matching the expected output. The initial input for rule 26 was: *The test checks whether the estimated room air quality offset is high.*

To improve the generated rule, the following input was given to the AI-powered rule generation system: *The test checks whether the estimated room air quality offset is high. In the precondition include a weekday selection from Monday to Sunday and a cond_nan_or_larger_than where RAQual is set to 0. Make sure to use a threshold of 200 and a second threshold of 500.*

The improved input has increased the conformity of the generated rule with the expected output to 86.67% percent.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

1. Room Temp 1

Rule number one reached one of the highest scores in the quantitative evaluation. The high score was achieved because very similar examples exist in the dataset. When looking closer at the rule, one additional value is found in the preconditions, and one value is missing in the test specs. The initial input for rule 1 was: *The test checks whether the room temperature reacts on a radiant ceiling cooling valve opening*

This input was modified to include a description of the requirements for the preconditions: *The test checks whether the room temperature reacts on a radiant ceiling cooling valve opening. A precondition for the test is that the associated cold group provides cold.*

This extension to the input did not change the result of the AI-powered rule generation system. The rule still reached an overlap of 95.45% compared to the expected output. Even more specific input is required to improve the rule to match the expected output 100%. The following input was used to generate a rule that has 100% matching with the expected output: *The test checks whether the room temperature reacts on a radiant ceiling cooling valve opening. In the precondition a trimmed_consecutive_larger_than only on the values RTemp 0 - 6 and MxCrt_Pu_Cmd 0.9 - 0 should be used and a trimmed_consecutive_smaller_than only on the value MxCrt_TFl. The accepted_ratio_invalid field in the test specs dict must necessarily be included with a value of 0.2.*

Reaching 100% matching with the expected output means that this rule can directly be used in the original AFDD framework. Following this, it also reached a qualitative score of 3.

Those results indicate that it is still essential that the users have extensive domain knowledge and the ability to provide detailed, context-rich input. The case study on the selected rules demonstrates how adjusting the input can lead to significant improvements in the generated rules, underscoring the need for close collaboration between users with domain knowledge and the AI-powered rule generation system.

5.5 Requirements Evaluation

During the research, several requirements were identified that are essential for making the AI-powered rule generation system practical. In the following it is checked if those requirements are met.

As described in Chapter 4, the AI-powered rule generation system is based on an LLM, which is designed to handle natural language input. The system is triggered by inputs given in natural language, fulfilling the first requirement (1. Input in Natural Language) that was defined. Users would likely find it challenging to provide input in technical form. This capability simplifies the process for users significantly.

Chapter 5 demonstrated that the AI-powered rule generation system successfully generated the correct JSON format in 32 out of 33 cases. While LLMs are trained in processing

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

natural language, they also understand and generate machine-readable formats accurately. This feature is advantageous, as LLM-based systems can be applied in a broader range of contexts, such as industrial applications, rather than being limited to chat applications. This achievement satisfies the second requirement (2. Directly Applicable Rules), which defines that the AI-powered rule generation system must produce rules compatible with the existing AFDD system. Nevertheless, the compatibility is constrained to a syntactic understanding of the existing AFDD system. Syntactic understanding means that the AI-powered rule generation system can create rules that are in valid JSON format and meet the preconditions of the integrity-checking framework. The qualitative evaluation showed that there are limitations in semantic compatibility. This semantic limitation means that the AI-powered rule generation system is unable to produce rules that correspond to the input provided in natural language, and it cannot accurately test what the user intended.

To enhance the performance of the AI-powered rule generation system, a RAG pipeline was integrated into its design. This addition aimed to improve the system's knowledge by incorporating specific information about the building in question. The integration fulfilled the third requirement (3. External Knowledge Usage) by using data extraction, chunking, embedding, and a vector store for RAG.

Another critical aspect of the AI-powered rule generation system was the inclusion of example rules. Examples are necessary to ensure that the system's output is consistent with existing rules. By deriving various examples from the original AFDD system and integrating them into the AI-powered rule generation system using few-shot prompts and the AI agent framework, the fourth requirement (4. Consistency with existing AFDD System) was successfully met. An important aspect during this integration was the use of the SemanticSimilarityExampleSelector, which ensured that relevant rules were selected while keeping the token length of the LLM within limits.

5.6 Discussion of the Results

In this work, the design and implementation of an AI-powered rule generation system for AFDD systems was presented. The significance of the problem was established, emphasizing that the traditional process of rule creation for rule-based fault detection requires the involvement of experts with specialized domain knowledge. This highlighted the necessity for a solution that could streamline and enhance the rule-generation process. A literature review was conducted, allowing for the creation of a knowledge base that outlined all the essential background information needed for the development of a rule generation system. Subsequently, an architecture for the AI-powered rule generation system was designed. Various components, such as a RAG pipeline and a AI agent system, were incorporated into this architecture. During the implementation phase of the project, open-source libraries and open LLM models and tools were used. Finally, an evaluation was conducted to assess the effectiveness of both the proposed architecture and its implementation. This evaluation aimed to determine whether the developed system

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

could detect faults in AFDD systems, thereby validating the approach and demonstrating the potential of AI to enhance the efficiency and effectiveness of fault detection processes.

The evaluation showed that the designed system met all the requirements that were defined earlier. Moreover, the evaluation showed that the created artifact is able to generate syntactically correct rules in almost all cases. The artifact was able to generate valid JSON files in 32 out of 33 cases in the quantitative evaluation. The same number of tests also passed the prechecks of the initial rule-based AFDD system. This means that the created system is able to understand the syntactic requirements of the rules and reproduce them.

Furthermore, this work contributes to the existing research by addressing the defined research questions. The relevance cycle in Chapter 2 illustrated the requirements for effectively utilizing AI, particularly LLMs, in the context of BACS and AFDD. It highlighted the significant role of NLP in facilitating interactions with those in need of supportive tools. This cycle provided answers to the first research question regarding the requirements for employing AI in a variable building environment. Additionally, the rigor cycle in Chapter 3 reviewed existing scientific research and emphasized various state-of-the-art tools and methods that are beneficial for creating an AI-powered rule generation system. This overview reflects the current advancements in the field of LLMs and AI and answers the second research question about which current AI methods in scientific research can be used for rule generation. The third research question, which focused on how state-of-the-art AI methods can be integrated for effective rule generation in AFDD, was addressed in the design cycle, as outlined in Chapter 4. By developing and implementing an architecture for an AI-powered rule generation system, it was demonstrated how LLMs can serve as the foundation for generating rules for AFDD.

However, while meeting the requirements and addressing the research questions, the quantitative evaluation showed that the system is not able to generate rules that match the expected output fully. At best, the generated rule achieved 95.45% of the anticipated results. In other instances, the overlap was significantly lower, dropping to as little as 25.93% or even 0% in one case. Nevertheless, it could be assumed that even when the overlap is not 100%, the generated rules could still be used in the AFDD system and detect faults. A qualitative evaluation was conducted to explore this possibility.

The qualitative evaluation revealed that the results of the AI-powered rule generation system cannot be directly used in a productive environment. The system lacks a semantic understanding of the rules, which is essential for the practical deployment of the system. The system is not able to generate semantically correct rules. This is a significant limitation of the system and hinders its practical deployment in detecting faults effectively within the original AFDD framework.

Due to practical limitations, refinements were made to the input of the AI-powered rule generation system to determine if human feedback or improved human input could yield better results. These refinements were applied to rules 1, 16, and 26. Additional information concerning the requirements and objectives for each rule was included in the

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

input. As a result, there was an improvement in the percentage of correct values for the quantitative evaluation. Specifically, the performance of Rule 1 increased from 95.45

Overall, this work contributes to the research on the integration of AI technologies, particularly LLMs, in the domain of building automation and control systems. The work establishes a foundation for identifying methods that simplify work in an AFDD context. However, further efforts are needed to achieve a fully automated and reliable system.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

CHAPTER 6

Conclusion and Future Work

In conclusion, this research successfully laid the groundwork for utilizing AI approaches to generate rules for AFDD systems, addressing numerous foundational requirements for practical applications. The exploration and implementation of an AI-powered rule generation system, mainly through the integration of LLMs and a RAG pipeline, demonstrated the potential of AI in translating natural language input into machine-readable output. This capability broadens the applicability of AI-powered systems beyond traditional chat applications and into industrial contexts. The system met key requirements, such as generating compatible rules and facilitating user interaction by simplifying input processes. Moreover, the incorporation of examples and adaptive few-shot prompting ensured consistency with existing rules, enhancing the reliability of the rule-generation mechanism.

However, the research also highlighted significant limitations. The system depends on manual preparatory tasks, like extracting examples and defining rule schema, which restricts its efficiency. More critically, while the syntactical accuracy of generated rules was notable, their semantic accuracy was lacking, hindering the system's practical deployment in detecting faults effectively within the original AFDD framework. Therefore, while the AI-powered rule generation system offers a promising approach for augmenting AFDD systems, it currently necessitates human oversight and correction, primarily due to semantic inaccuracies.

One way of improving the AI-powered rule generation system is to evaluate the performance of different and potentially more advanced LLM models. In this work, the mistral-7b model was used in all LLM related tasks. However, more recent models could improve the overall results of the system.

Another way to enhance the AI-powered rule generation system is that the output of the original AFDD framework could be leveraged. Instead of simply running the AI-powered rule generation system once, a feedback mechanism should incorporated that utilizes the results of the generated rules when they are applied to the original framework. By then going through the rule generation process once more and refining the approach based on the feedback collected, the overall effectiveness of the rule generation process might improve. Future researchers could explore this technique.

Additionally, future research must focus on enhancing the semantic understanding and autonomous preparation capabilities of the system to move toward a more refined and deployable solution. A more general way of generating rules would increase the benefit of the developed system. A system that is more independent from a specific AFDD system is necessary. Additionally, it is important to explore the system's adaptability to different building types and contexts to ensure its scalability and applicability in diverse environments. By addressing these challenges, the AI-powered rule generation system can evolve into a valuable tool for automating rule generation in AFDD systems, thereby improving the efficiency and effectiveness of fault detection in buildings.

Next to a context-based LLM approach, with the help of RAG, fine-tuning techniques could be used to improve the system. By additional training of an LLM with domain-specific knowledge, the intrinsic knowledge about rule generation could be improved, and more accurate rules may be generated.

Based on the obtained knowledge and the suggested enhancements, the implementation of rule-based fault detection within AFDD systems can become more user-friendly and straightforward. The improvements aim to simplify the integration process for users, ensuring that they can effectively apply the necessary rules without extensive technical expertise. Therefore, the overall effectiveness of AFDD systems improves.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Overview of Generative AI Tools Used

Grammarly: Grammarly contributed to this text by responding to these AI prompts:

- „Improve it“
- „Improve my english grammar“
- „Make it inspirational“
- „Simplify it“
- „Make it more detailed“

GPT-4o: GPT-4o contributed to this text by responding to these AI prompts:

- „Improve this latex section“
- „Improve my english grammar“

DeepL: DeepL contributed to this text by translating between German and English, converting phrases and sentences from one language to the other.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Figures

1.1 Localization of the AI-Powered Rule Generation System in BACS [Pac18] 4
1.2 The Three-Cycle-View of DSR [Hev07] 5

3.1 The Architecture of the AI-Powered Rule Generation System 23

4.1 Detailed Representation of the RAG Pipeline [GXG$^+$23] 32
4.2 Detailed Representation of the AI-Agent System 39

5.1 The Evaluation of the AI-Powered Rule Generation System 44
5.2 Overview over the Quantitative Evaluation showing Correct and Incorrect Values for each Rule 46

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Tables

3.1 Definition of the AI-Agents 26
3.2 Definition of the Tasks for the AI-Agents 27

5.1 Quantitative Evaluation Results: Correctly generated JSON Values compared to all JSON Values in a Rule and the Percentage of correct Values 45
5.2 Qualitative Evaluation Results: Applying the generated Rules to the Integrity Checking Framework and adding a Score depending on the Outcome . . . 48

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Listings

4.1 Definition of the Markdown Text Splitter 32
4.2 Definition of the BGE Embedding 33
4.3 Definition of the FAISS Vectorstore 34
4.4 Exemplary Similarity Search 34
4.5 Definition of the SemanticSimilarityExampleSelector 35
4.6 Definition of a Pydantic Model 35
4.7 Agent Structure 36
4.8 Task Structure 36
4.9 Definition of a Crew 36
4.10 Few Shot Prompt Structure 37
4.11 Execution of the AI-Powered Rule Generation System 38
5.1 Creation of the Input and Expected Output 43
5.2 Structure of the Expected Output 43

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Acronyms

AFDD Automated Fault Detection and Diagnostics. 1–6, 9–13, 15, 16, 20–22, 24, 25, 27, 29–34, 36, 37, 41, 42, 50–52, 55, 56

AHU Air Handling Unit. 10, 15, 20

AI Artificial Intelligence. xi, 1, 3–6, 9–13, 15, 16, 19–25, 27, 29, 30, 34–39, 41–45, 48–53, 55, 56, 59, 63, 75

BACS Building Automation and Control Systems. ix, xi, 1, 2, 4, 5, 15, 20, 21, 31, 41, 52, 59

CoT Chain-of-Tought. 16, 17

DL Deep Learning. 16, 21, 31

DSR Design Science Research. ix, xi, 4–6, 59

FDD Fault Detection and Diagnostics. 10, 21

HVAC Heating, Ventilation and Air Conditioning. 2, 9, 10, 15, 20, 21, 41

KG Knowledge Graph. 21

KI Künstliche Intelligenz. ix

LLM Large Language Model. ix, xi, 1–3, 5, 11–13, 15–22, 24, 25, 27, 29–31, 34–37, 41, 50–53, 55, 56

ML Machine Learning. 20, 21

NLP Natural Language Processing. 3–5, 15, 16, 20–22, 52

PDF Portable Document Format. 3, 17, 18, 25, 30–32, 41, 42

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

RAG Retrieval-Augmented Generation. 18, 21, 22, 24, 25, 29–32, 34, 37, 41, 42, 51, 55, 56, 59

VAV Variable Air Volume. 10

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Bibliography

[AAED+15] Mahmoud Ahmed, Magdy Abdullah Eissa, R. Darwish, A.M. Bassiuny, and Mohamed Rady. Implementation of rule based fault detection on total fresh air handling unit. *17th International Middle-East Power System Conference*, 12 2015.

[AM23] Amos Azaria and Tom M. Mitchell. The internal state of an llm knows when its lying. In *Conference on Empirical Methods in Natural Language Processing*, 2023.

[BCO+13] Ken Bruton, Daniel Coakley, Peter O'Donovan, Marcus M. Keane, and Dominic T. J. O'Sullivan. Development of an online expert rule based automated fault detection and diagnostic (afdd) tool for air handling units: Beta test results. In *13th International Conference for Enhanced Building Operations*, 2013.

[BCR+15] Ken Bruton, Daniel Coakley, Paul Raftery, D. Og Cusack, Marcus M. Keane, and D. T. J. O'Sullivan. Comparative analysis of the ahu info fault detection and diagnostic expert tool for ahus with apar. *Energy Efficiency*, 8(2):299–322, April 2015.

[BHM20] Jan vom Brocke, Alan Hevner, and Alexander Maedche. *Introduction to Design Science Research*, pages 1–13. Springer International Publishing, 09 2020.

[BMR+20] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Ma teusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *ArXiv*, abs/2005.14165, 2020.

[BWY+24] Jian Bi, Hua Wang, Enbo Yan, Chuan Wang, Ke Yan, Liangliang Jiang, and Bin Yang. Ai in hvac fault detection and diagnosis: A systematic review. *Energy Reviews*, 3(2):100071, 2024.

[CD15] Christopher Andreas Clark and Santosh Divvala. Looking beyond text: Extracting figures, tables and captions from computer science papers. In *Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[cre24] crewAIInc. crewai. `https://github.com/crewAIInc/crewAI`, 2024. Accessed: 2024-08-13.

[dLBBdFC+24] Maria Angels de Luis Balaguer, Vinamra Benara, Renato Luiz de Freitas Cunha, Roberto de M. Estevao Filho, Todd Hendry, Daniel Holstein, Jennifer Marsman, Nick Mecklenburg, Sara Malvar, Leonardo Nunes, Rafael Padilha, Morris Sharp, Bruno Leonardo Barros Silva, Swati Sharma, Vijay Aski, and Ranveer Chandra. Rag vs fine-tuning: Pipelines, tradeoffs, and a case study on agriculture. *ArXiv*, abs/2401.08406, 2024.

[Dou23] Michael R. Douglas. Large language models. *Communications of the ACM*, 66:7 – 7, 2023.

[DSS+23] Sifatkaur Dhingra, Manmeet Singh, Vaisakh S.B., Neetiraj Malviya, and Sukhpal Singh Gill. Mind meets machine: Unravelling gpt-4's cognitive psychology. *ArXiv*, abs/2303.11436, 2023.

[DVRG16] Mauro Dragoni, Serena Villata, Williams Rizzi, and Guido Governatori. Combining nlp approaches for rule extraction from legal documents. In *AI Approaches to the Complexity of Legal Systems*, 2016.

[EMMR23] Tyna Eloundou, Sam Manning, Pamela Mishkin, and Daniel Rock. Gpts are gpts: An early look at the labor market impact potential of large language models. *ArXiv*, abs/2303.10130, 2023.

[GMMH22] Markus Gwerder, Reto Marek, Andreas Melillo, and Maria Husmann. Data integrity checks for building automation and control systems. In *CLIMA 2022*, 05 2022.

[GSM+17] Jessica Granderson, Rupam Singla, Ebony Mayhorn, Paul Ehrlich, Draguna Vrabie, and Stephen Frank. Characterization and survey of automated fault detection and diagnostics tools. Report, Lawrence Berkeley National Laboratory, 11 2017.

[GWC+19] Yabin Guo, Jiangyu Wang, Huanxin Chen, Guannan Li, Ronggeng Huang, Yue Yuan, Tanveer Ahmad, and Shaobo Sun. An expert rule-based fault diagnosis strategy for variable refrigerant flow air conditioning systems. *Applied Thermal Engineering*, 149:1223–1235, 2019.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[GXG+23] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *ArXiv*, abs/2312.10997, 2023.

[Hev07] Alan Hevner. A three cycle view of design science research. *Scandinavian Journal of Information Systems*, 19, 01 2007.

[HKJ19] Djerdj Horvat, Henning Kroll, and Angelika Jäger. Researching the effects of automation and digitalization on manufacturing companies' productivity in the early stage of industry 4.0. *Procedia Manufacturing*, 2019.

[Ip24] Jeffrey Ip. How to build a pdf qa chatbot using openai and chromadb. https://www.confident-ai.com/blog/how-to-build-a-pdf-qa-chatbot-using-openai-and-chromadb, 2024.

[JGKS24] Pankaj Joshi, Aditya Gupta, Pankaj Kumar, and Manas Sisodia. Robust multi model rag pipeline for documents containing text, table & images. In *2024 3rd International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*, pages 993–999, 2024.

[JSH+24] Zhi Jing, Yongye Su, Yikun Han, Bo Yuan, Haiyun Xu, Chunjiang Liu, Kehai Chen, and Min Zhang. When large language models meet vector databases: A survey. *ArXiv*, abs/2402.01763, 2024.

[KABC24] Heiko Koziolek, Virendra Ashiwal, Soumyadip Bandyopadhyay, and R ChandrikaK. Automated control logic test case generation using large language models. *ArXiv*, abs/2405.01874, 2024.

[KLC+20] Hannah Kramer, Guanjing Lin, Christopher Curtin, Eliot Crowe, and Jessica Granderson. Proving the business case for building analytics. Technical report, Lawrence Berkeley National Laboratory, 10 2020.

[KTF+22] Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. Decomposed prompting: A modular approach for solving complex tasks. *ArXiv*, abs/2210.02406, 2022.

[Lan24a] LangChain. Langchain documentation: Agents. `https://python.langchain.com/v0.1/docs/modules/agents/`, 2024. Accessed: 2024-08-13.

[Lan24b] LangChain. Markdownheadertextsplitter. `https://python.langchain.com/v0.1/docs/modules/data_connection/document_transformers/markdown_header_metadata/`, 2024. Accessed: 2024-08-12.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

[LJX+23] Linhao Luo, Jiaxin Ju, Bo Xiong, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. Chatrule: Mining logical rules with large language models for knowledge graph reasoning. *ArXiv*, abs/2309.01538, 2023.

[LKG20] Guanjing Lin, Hannah Kramer, and Jessica Granderson. Building fault detection and diagnostics: Achieved savings, and methods to evaluate algorithm performance. *Building and Environment*, 168:106505, 2020.

[LPP+20] Patrick Lewis, Ethan Perez, Aleksandara Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kuttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. *ArXiv*, abs/2005.11401, 2020.

[LS21] Jiajun Liao and Victor Saeh. Engineering with automated fault detection and diagnostics. *Consulting-Specifying Engineer*, 2021.

[Mic24] Microsoft. Retrieval-augmented generation with azure ai document intelligence. `https://learn.microsoft.com/en-us/azure/ai-services/document-intelligence/concept-retrieval-augmented-generation?view=doc-intel-4.0.0#semantic-chunking`, 2024. Accessed: 2024-08-12.

[Nag24] Nivedhaa Nagarajan. A comprehensive review of ai's dependence on data. In *International Journal of Artificial Intelligence and Data Science (IJADS)*, 03 2024.

[Ope24] OpenAI. Learning to reason with llms, 2024. Accessed: 2024-09-26.

[Pac18] Pacific Northwest National Laboratory. Automated fault detection and diagnostics: Affordable energy efficiency for buildings, 2018.

[Par24] Vik Paruchuri. Marker: Convert pdf to markdown quickly with high accuracy. `https://github.com/VikParuchuri/marker`, 2024. Last accessed: 2024-10-14.

[Rac17] Dmitri A. Rachkovskij. Binary vectors for fast distance and similarity estimation. *Cybernetics and Systems Analysis*, 53:138 – 156, 2017.

[RWC+19] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. In *OpenAI blog*, 2019.

[Sar22] Elvis Saravia. Prompt Engineering Guide. *https://github.com/dair-ai/Prompt-Engineering-Guide*, 12 2022.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

[SB06] Jeffrey Schein and Steven T. Bushby. A hierarchical rule-based fault detection and diagnostic method for hvac systems. *HVAC&R Research*, 12:111 – 125, 2006.

[SBCH06] Jeffrey Schein, Steven T. Bushby, Natascha S. Castro, and John M. House. A rule-based fault detection method for air handling units. *Energy and Buildings*, 38(12):1485–1492, 2006.

[SLG23] Noah Shinn, Beck Labash, and Ashwin Gopinath. Reflexion: an autonomous agent with dynamic memory and self-reflection. *ArXiv*, abs/2303.11366, 2023.

[SPC$^+$21] Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. Retrieval augmentation reduces hallucination in conversation. In *Conference on Empirical Methods in Natural Language Processing*, 2021.

[Sri22] Putty Srividya. Smart building automation system. In *Smart Buildings Digitalization*, page 10. CRC Press, 1st edition, 2022.

[TA23] Oguzhan Topsakal and Tahir Cetin Akinci. Creating large language model applications utilizing langchain: A primer on developing llm apps fast. *International Conference on Applied Engineering and Natural Sciences*, 2023.

[TN23] Yashar Talebirad and Amirhossein Nadiri. Multi-agent collaboration: Harnessing the power of intelligent llm agents. *ArXiv*, abs/2306.03314, 2023.

[Vav24] Chris Vavra. How ai can improve building automation system implementation. *Consulting-Specifying Engineer*, 2024.

[VSP$^+$17] Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Neural Information Processing Systems*, 2017.

[WBZ$^+$23] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation, 2023.

[WC16] Haitao Wang and Youming Chen. A robust fault detection and diagnosis strategy for multiple faults of vav air handling units. *Energy and Buildings*, 127:442–451, 2016.

[WDS$^+$20] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma,

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In Qun Liu and David Schlangen, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.

[WJZ⁺23] Yiran Wu, Feiran Jia, Shaokun Zhang, Han-Tai Li, Erkang Zhu, Yue Wang, Yin Tat Lee, Richard Peng, Qingyun Wu, and Chi Wang. Mathchat: Converse to tackle challenging math problems with llm agents. 2023.

[Wu24] Scott Wu. Introducing devin, the first ai software engineer. https://www.cognition-labs.com/introducing-devin, 2024. Last accessed: 2024-03-23.

[WWS⁺22] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *CoRR*, abs/2201.11903, 2022.

[WXJ⁺23] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi (Jim) Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *ArXiv*, abs/2305.16291, 2023.

[XCG⁺23] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Qin Liu, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan Dou, Rongxiang Weng, Wensen Cheng, Qi Zhang, Wenjuan Qin, Yongyan Zheng, Xipeng Qiu, Xuanjing Huan, and Tao Gui. The rise and potential of large language model based agents: A survey. *ArXiv*, abs/2309.07864, 2023.

[YJT⁺23] Jingfeng Yang, Hongye Jin, Ruixiang Tang, Xiaotian Han, Qizhang Feng, Haoming Jiang, Bing Yin, and Xia Hu. Harnessing the power of llms in practice: A survey on chatgpt and beyond. *ACM Transactions on Knowledge Discovery from Data*, 18:1 – 32, 2023.

[YZY⁺23] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023.

[ZXC⁺23] Zhaocheng Zhu, Yuan Xue, Xinyun Chen, Denny Zhou, Jian Tang, Dale Schuurmans, and Hanjun Dai. Large language models can learn rules. *ArXiv*, abs/2310.07064, 2023.

[ZZL+23] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jianyun Nie, and Ji rong Wen. A survey of large language models. *ArXiv*, abs/2303.18223, 2023.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Appendix

Used Test Cases for the Evaluation

In the following, all test cases used for the evaluation are listed. Each test case, referred to as a rule, includes the given input, the expected output, and the actual output generated by the AI-powered rule generation system. The source code used for the AI-powered rule generation system, along with the evaluation code, can be found at `https://github.com/Yleisnero/ai-powered-rule-generation`.

1. Room Temp 1

Input

The test checks whether the room temperature reacts on a radiant ceiling cooling valve opening

Expected Output

```
{"test_method": "reaction_single_window_threshold_test", "test_preconditions": {"
    trimmed_consecutive_larger_than": {"RTemp": [0, 6], "MxCrt_Pu_Cmd": [0.9, 0]}, "
    trimmed_consecutive_smaller_than": {"MxCrt_TFl": [19, 2]}}, "event_dict": {"
    conditional_poschange_event_end_monotonous": {"RcgVlvPos": {"lower_threshold": 1, "
    upper_threshold": 20, "pre_window": 12, "change_window": 5, "post_window": 16, "
    end_threshold": 20, "max_length_after_condition": 60}}}, "test_specs": {"
    reaction_signal": "RTemp", "window_shift_event_start": 0, "window_shift_event_end":
     3, "feature": "diffEndMax", "kind": "larger", "threshold": −0.1, "second_threshold
    ": 0.0, "accepted_ratio_invalid": 0.2}, "time_start_utc": "now() − 31 days", "
    time_end_utc": "now()", "violation_plausibility_code": "suspicious"}
```

Output

```
{"test_method": "reaction_single_window_threshold_test", "test_preconditions": {"
    trimmed_consecutive_larger_than": {"RTemp": [0, 6], "Cho2PipeModCmd": [0.99, −6], "
    MxCrt_Pu_Cmd": [0.9, 0]}, "trimmed_consecutive_smaller_than": {"MxCrt_TFl": [19,
    2]}}, "time_start_utc": "now() − 31 days", "time_end_utc": "now()", "
    violation_plausibility_code": "suspicious", "test_specs": {"reaction_signal": "
    RTemp", "window_shift_event_start": 0, "window_shift_event_end": 3, "feature": "
    diffEndMax", "kind": "larger", "threshold": −0.1, "second_threshold": 0.0}, "
    event_dict": {"conditional_poschange_event_end_monotonous": {"RcgVlvPos": {"
    lower_threshold": 1, "upper_threshold": 20, "pre_window": 12, "change_window": 5, "
    post_window": 16, "end_threshold": 20, "max_length_after_condition": 60}}}}
```

2. Concrete Temp 1

Input

The test checks whether the concrete temperature reacts on a TABS heating valve opening

Expected Output

```
{"test_method": "alternative_test", "test_method_sub": "
    reaction_single_window_threshold_test", "test_virtual_timeseries": {"DTFlTCon": "
    TFlH - ConTemp"}, "test_preconditions": {"rolling_feature_threshold": {"DTFlTCon":
    {"feature": "mean", "window_size": 12, "kind": "larger", "threshold": 3}}, "
    trimmed_consecutive_larger_than": {"PrOpModH": [0.01, 0]}, "
    trimmed_consecutive_smaller_than": {"CCmd": [0.01, 12]}}, "event_dict": {"
    conditional_poschange_event_end": {"HCmd": {"lower_threshold": 0.1, "
    upper_threshold": 0.9, "pre_window": 6, "change_window": 2, "post_window": 12, "
    end_threshold": 0.9, "max_length_after_condition": 36}}}, "test_specs": {"
    reaction_signal": "ConTemp", "window_shift_event_start": 2, "window_shift_event_end
    ": 0, "feature": "diffEndMin", "kind": "lower", "threshold": 0.05, "
    second_threshold": 0.0, "accepted_ratio_invalid": 0.2}, "time_start_utc":
    "2022-01-01 00:00:00", "time_end_utc": "2022-02-01 00:00:00", "
    violation_plausibility_code": "suspicious"}
```

Output

```
{"test_method": "reaction_single_window_threshold_test", "test_preconditions": {"
    rolling_feature_threshold": {"DTFlHCon": {"feature": "mean", "window_size": 12, "
    kind": "larger", "threshold": 3}}, "trimmed_consecutive_smaller_than": {"HCmd":
    [0.01, 12]}}, "time_start_utc": "now() - 31 days", "time_end_utc": "now()", "
    violation_plausibility_code": "suspicious", "test_specs": {"reaction_signal": "
    ConTemp", "window_shift_event_start": 0, "window_shift_event_end": 0, "feature": "
    diffEndMin", "kind": "lower", "threshold": 0.05, "second_threshold": 0.0}, "
    event_dict": {"conditional_poschange_event_end": {"CCmd": {"lower_threshold": 0.1,
    "upper_threshold": 0.9, "pre_window": 12, "change_window": 2, "post_window": 5, "
    end_threshold": 0.9, "max_length_after_condition": 60}}}}
```

3. Window Contact 1

Input

The test checks whether there is presence detected when the window contact state changes

Expected Output

```
{"test_method": "reaction_single_window_threshold_test", "test_preconditions": {}, "
    event_dict": {"variable_conditional_poschange": {"RWndSta": {"change": 0.1, "
    time_period": 1, "pre_window": 0, "post_window": 0}}}, "test_specs": {"
    reaction_signal": "RPscDet", "window_shift_event_start": -1, "
    window_shift_event_end": 1, "feature": "sum", "kind": "lower", "threshold": 0.01},
    "time_start_utc": "now() - 31 days", "time_end_utc": "now()", "
    violation_plausibility_code": "implausible"}
```

Output

```
{"test_method": "reaction_single_window_presence_test", "test_preconditions": {"
    trimmed_consecutive_larger_than": {"WindowContactState": [0, 3]}}, "time_start_utc
    ": "now() - 31 days", "time_end_utc": "now()", "violation_plausibility_code": "
    implausible", "test_specs": {"reaction_signal": "PresenceDetected", "
    window_shift_event_start": -12, "window_shift_event_end": 0, "feature": "sum", "
    kind": "lower", "threshold": 0.1, "second_threshold": 0.0}, "event_dict": {"
    variable_conditional_poschange": {"PresenceDetected": {"change": 1, "time_period":
    12, "pre_window": 3, "post_window": 3}}}}
```

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

4. Room Temp 2

Input

The test checks whether the room temperature reacts on a radiant ceiling cooling valve opening

Expected Output

```
{"test_method": "reaction_single_window_threshold_test", "test_preconditions": {"
    trimmed_consecutive_larger_than": {"RTemp": [0, 6], "Cho2PipeModCmd": [0.99, -6], "
    MxCrt_Pu_Cmd": [0.9, 0]}, "trimmed_consecutive_smaller_than": {"MxCrt_TFl": [19,
    2]}}, "event_dict": {"conditional_poschange_event_end_monotonous": {"RcgVlvPos": {"
    lower_threshold": 1, "upper_threshold": 20, "pre_window": 12, "change_window": 5, "
    post_window": 16, "end_threshold": 20, "max_length_after_condition": 60}}}, "
    test_specs": {"reaction_signal": "RTemp", "window_shift_event_start": 0, "
    window_shift_event_end": 3, "feature": "diffEndMax", "kind": "larger", "threshold":
     -0.1, "second_threshold": 0.0, "accepted_ratio_invalid": 0.2}, "time_start_utc": "
    now() - 31 days", "time_end_utc": "now()", "violation_plausibility_code": "
    suspicious"}
```

Output

```
{"test_method": "reaction_single_window_threshold_test", "test_preconditions": {"
    trimmed_consecutive_larger_than": {"RTemp": [0, 6], "MxCrt_Pu_Cmd": [0.9, 0]}, "
    trimmed_consecutive_smaller_than": {"MxCrt_TFl": [19, 2]}}, "time_start_utc": "now
    () - 31 days", "time_end_utc": "now()", "violation_plausibility_code": "suspicious
    ", "test_specs": {"reaction_signal": "RTemp", "window_shift_event_start": 0, "
    window_shift_event_end": 3, "feature": "diffEndMax", "kind": "larger", "threshold":
     -0.1, "second_threshold": 0.0}, "event_dict": {"
    conditional_poschange_event_end_monotonous": {"RcgVlvPos": {"lower_threshold": 1, "
    upper_threshold": 20, "pre_window": 12, "change_window": 5, "post_window": 16, "
    end_threshold": 20, "max_length_after_condition": 60}}}}
```

5. Valve Flow 1

Input

The test checks whether the increase in valve opening causes an increase in flow

Expected Output

```
{"test_method": "reaction_single_window_threshold_test", "test_virtual_variables": {"
    tshFl": "max(0.1,0.05*FlMtr.quantile(0.95))"}, "test_preconditions": {"
    weekday_selection": {"week_day_list": [0, 1, 2, 3, 4, 5, 6]}}, "event_dict": {"
    conditional_poschange": {"Vlv": {"lower_threshold": 1, "upper_threshold": 10, "
    pre_window": 3, "change_window": 8, "post_window": 7}}}, "test_specs": {"
    reaction_signal": "FlMtr", "window_shift_event_start": -2, "window_shift_event_end
    ": 5, "feature": "diff", "kind": "lower", "threshold": "tshFl", "second_threshold":
     "tshFl"}, "time_start_utc": "now() - 31 days", "time_end_utc": "now()", "
    violation_plausibility_code": "implausible"}
```

Output

```
{"test_method": "reaction_single_window_threshold_test", "test_preconditions": {"
    delayed_consecutive_larger_than": {"ValvePos": [0.9, -6]}}, "time_start_utc": "now
    () - 31 days", "time_end_utc": "now()", "violation_plausibility_code": "implausible
    ", "test_specs": {"reaction_signal": "FlowRate", "window_shift_event_start": -2, "
    window_shift_event_end": 6, "feature": "diff", "kind": "larger", "threshold": "
    tshFlow", "second_threshold": 0}, "event_dict": {"conditional_poschange": {"
    ValvePos": {"lower_threshold": 10, "upper_threshold": 30, "pre_window": 3, "
    change_window": 5, "post_window": 7}}}}
```

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

6. Room Temp 3

Input

The test checks whether the room temperature reacts on a cooling coil valve opening

Expected Output

```
{"test_method": "reaction_single_window_threshold_test", "test_preconditions": {"
    trimmed_consecutive_larger_than": {"RTemp": [0, 6]}, "cond_equal_value": {"PltCtl":
     2}, "trimmed_consecutive_smaller_than": {"MxCrt_TFl": [19, 2]}}, "event_dict": {"
    conditional_poschange_event_end": {"CclVlvPos": {"lower_threshold": 5, "
    upper_threshold": 30, "pre_window": 12, "change_window": 5, "post_window": 12, "
    end_threshold": 30, "max_length_after_condition": 60}}}, "test_specs": {"
    reaction_signal": "RTemp", "window_shift_event_start": 0, "window_shift_event_end":
     0, "feature": "diffEndMax", "kind": "larger", "threshold": −0.1, "second_threshold
    ": 0.0}, "time_start_utc": "now() − 31 days", "time_end_utc": "now()", "
    violation_plausibility_code": "suspicious"}
```

Output

```
{"test_method": "reaction_single_window_threshold_test", "test_preconditions": {"
    trimmed_consecutive_larger_than": {"RTemp": [0, 6], "Cho2PipeModCmd": [0.99, −6], "
    MxCrt_Pu_Cmd": [0.9, 0]}, "trimmed_consecutive_smaller_than": {"MxCrt_TFl": [19,
    2]}}, "time_start_utc": "now() − 31 days", "time_end_utc": "now()", "
    violation_plausibility_code": "suspicious", "test_specs": {"reaction_signal": "
    RTemp", "window_shift_event_start": 0, "window_shift_event_end": 3, "feature": "
    diffEndMax", "kind": "larger", "threshold": −0.1, "second_threshold": 0.0}, "
    event_dict": {"conditional_poschange_event_end_monotonous": {"CoolCoilVlvPos": {"
    lower_threshold": 1, "upper_threshold": 20, "pre_window": 12, "change_window": 5, "
    post_window": 16, "end_threshold": 20, "max_length_after_condition": 60}}}}
```

7. Concrete Temp 2

Input

The test checks whether the concrete temperature reacts on a TABS heating valve opening

Expected Output

```
{"test_method": "reaction_single_window_threshold_test", "test_virtual_timeseries": {"
    DTFlTCon": "TFlH − ConTemp"}, "test_preconditions": {"rolling_feature_threshold":
    {"DTFlTCon": {"feature": "mean", "window_size": 12, "kind": "larger", "threshold":
    3}}, "trimmed_consecutive_smaller_than": {"CCmd": [0.01, 12]}}, "event_dict": {"
    conditional_poschange_event_end": {"HCmd": {"lower_threshold": 0.1, "
    upper_threshold": 0.9, "pre_window": 12, "change_window": 2, "post_window": 5, "
    end_threshold": 0.9, "max_length_after_condition": 60}}}, "test_specs": {"
    reaction_signal": "ConTemp", "window_shift_event_start": 0, "window_shift_event_end
    ": 0, "feature": "diffEndMin", "kind": "lower", "threshold": 0.05, "
    second_threshold": 0.0, "accepted_ratio_invalid": 0.2}, "time_start_utc": "now() −
    31 days", "time_end_utc": "now()", "violation_plausibility_code": "suspicious"}
```

Output

```
{"test_method": "reaction_single_window_threshold_test", "test_preconditions": {"
    rolling_feature_threshold": {"DTFlHCon": {"feature": "mean", "window_size": 12, "
    kind": "larger", "threshold": 3}}, "trimmed_consecutive_larger_than": {"PrOpModH":
    [0.01, 0]}, "trimmed_consecutive_smaller_than": {"CCmd": [0.01, 12]}}, "
    time_start_utc": "2022−01−01 00:00:00", "time_end_utc": "2022−02−01 00:00:00", "
    violation_plausibility_code": "suspicious", "test_specs": {"reaction_signal": "
    ConTemp", "window_shift_event_start": 2, "window_shift_event_end": 0, "feature": "
```

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

```
    diffEndMin", "kind": "lower", "threshold": 0.05, "second_threshold": 0.0}, "
    event_dict": {"conditional_poschange_event_end": {"HCmd": {"lower_threshold": 0.1,
    "upper_threshold": 0.9, "pre_window": 6, "change_window": 2, "post_window": 12, "
    end_threshold": 0.9, "max_length_after_condition": 36}}}}
```

8. Room Temp 4

Input

The test checks whether the room temperature reacts on a radiant ceiling heating valve opening

Expected Output

```
{"test_method": "reaction_single_window_threshold_test", "test_preconditions": {"
    trimmed_consecutive_larger_than": {"RTemp": [0, 6], "MxCrt_Pu_Cmd": [0.9, 0], "
    MxCrt_TFl": [30, 2]}}, "event_dict": {"conditional_poschange_event_end_monotonous":
     {"RcgVlvPos": {"lower_threshold": 1, "upper_threshold": 20, "pre_window": 12, "
    change_window": 5, "post_window": 16, "end_threshold": 20, "
    max_length_after_condition": 60}}}, "test_specs": {"reaction_signal": "RTemp", "
    window_shift_event_start": 0, "window_shift_event_end": 3, "feature": "diffEndMin",
     "kind": "lower", "threshold": 0.1, "second_threshold": 0.0, "
    accepted_ratio_invalid": 0.2}, "time_start_utc": "now() - 31 days", "time_end_utc":
     "now()", "violation_plausibility_code": "suspicious"}
```

Output

```
{"test_method": "reaction_single_window_threshold_test", "test_preconditions": {"
    trimmed_consecutive_larger_than": {"RTemp": [0, 6], "MxCrt_Pu_Cmd": [0.9, 0], "
    MxCrt_TFl": [19, 2]}}, "time_start_utc": "now() - 31 days", "time_end_utc": "now()
    ", "violation_plausibility_code": "suspicious", "test_specs": {"reaction_signal": "
    RTemp", "window_shift_event_start": 0, "window_shift_event_end": 3, "feature": "
    diffEndMax", "kind": "larger", "threshold": -0.1, "second_threshold": 0.0}, "
    event_dict": {"conditional_poschange_event_end_monotonous": {"RcgVlvPos": {"
    lower_threshold": 1, "upper_threshold": 20, "pre_window": 12, "change_window": 5, "
    post_window": 16, "end_threshold": 20, "max_length_after_condition": 60}}}}
```

9. Room Temp 5

Input

The test checks whether the room temperature reacts on a window opening

Expected Output

```
{"test_method": "reaction_single_window_threshold_test", "test_virtual_timeseries": {"
    DTOaTR": "RTemp - TOa"}, "test_preconditions": {"rolling_feature_threshold": {"
    DTOaTR": {"feature": "mean", "window_size": 12, "kind": "larger", "threshold": 3}},
     "trimmed_consecutive_larger_than": {"RTemp": [0, 6]}}, "event_dict": {"
    conditional_poschange_event_end": {"RWndSta": {"lower_threshold": 0.1, "
    upper_threshold": 0.9, "pre_window": 6, "change_window": 3, "post_window": 6, "
    end_threshold": 0.9, "max_length_after_condition": 30}}}, "test_specs": {"
    reaction_signal": "RTemp", "window_shift_event_start": 0, "window_shift_event_end":
     0, "feature": "diffEndMax", "kind": "larger", "threshold": -0.1, "second_threshold
    ": 0.0}, "time_start_utc": "now() - 31 days", "time_end_utc": "now()", "
    violation_plausibility_code": "suspicious"}
```

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Output

```
{"test_method": "reaction_single_window_threshold_test", "test_preconditions": {"
    rolling_feature_threshold": {"DTOaTR": {"county": true, "friendship": true, "love":
     true}}}, "time_start_utc": "now() - 31 days", "time_end_utc": "now()", "
    violation_plausibility_code": "suspicious", "test_specs": {"reaction_signal": "
    RTemp", "window_shift_event_start": 0, "window_shift_event_end": 0, "feature": "
    diffEndMax", "kind": "larger", "threshold": -0.1, "second_threshold": 0.0}, "
    event_dict": {"conditional_poschange": {"OpenWindowIndicaitor": {"change": 1}}}}
```

10. Room Temp 6

Input

The test checks whether the room temperature reacts on a VAV supply air flow increase

Expected Output

```
{"test_method": "reaction_single_window_threshold_test", "test_preconditions": {"
    trimmed_consecutive_larger_than": {"RTemp": [0, 6], "FanSu_OpMod": [0.9, 1]}, "
    trimmed_consecutive_smaller_than": {"TSu": [19, 1]}}, "event_dict": {"
    conditional_poschange": {"VavSuAirFl": {"lower_threshold": 5, "upper_threshold":
    30, "pre_window": 12, "change_window": 5, "post_window": 24}}}, "test_specs": {"
    reaction_signal": "RTemp", "window_shift_event_start": 6, "window_shift_event_end":
     24, "feature": "diffEndMax", "kind": "larger", "threshold": -0.1, "
    second_threshold": 0}, "time_start_utc": "now() - 31 days", "time_end_utc": "now()
    ", "violation_plausibility_code": "suspicious"}
```

Output

```
{"test_method": "reaction_single_window_threshold_test", "test_preconditions": {"
    delayed_consecutive_larger_than": {"FanSu_OpMod": [0.9, -6]}}, "time_start_utc": "
    now() - 31 days", "time_end_utc": "now()", "violation_plausibility_code": "
    implausible", "test_specs": {"reaction_signal": "RTemp", "window_shift_event_start
    ": -2, "window_shift_event_end": 6, "feature": "diff", "kind": "lower", "threshold
    ": "tshAirFlSu", "second_threshold": 0}, "event_dict": {"conditional_poschange": {"
    VavSuPos": {"lower_threshold": 10, "upper_threshold": 30, "pre_window": 3, "
    change_window": 5, "post_window": 7}}}}
```

11. Room Temp 7

Input

The test checks whether the room temperature reacts on a radiant ceiling cooling valve opening

Expected Output

```
{"test_method": "reaction_single_window_threshold_test", "test_preconditions": {"
    trimmed_consecutive_larger_than": {"RTemp": [0, 6], "MxCrt_Pu_Cmd": [0.9, 0]}, "
    trimmed_consecutive_smaller_than": {"MxCrt_TFl": [19, 2]}}, "event_dict": {"
    conditional_poschange_event_end_monotonous": {"RcgVlvPos": {"lower_threshold": 1, "
    upper_threshold": 20, "pre_window": 12, "change_window": 5, "post_window": 16, "
    end_threshold": 20, "max_length_after_condition": 60}}}, "test_specs": {"
    reaction_signal": "RTemp", "window_shift_event_start": 0, "window_shift_event_end":
     3, "feature": "diffEndMax", "kind": "larger", "threshold": -0.1, "second_threshold
    ": 0.0, "accepted_ratio_invalid": 0.2}, "time_start_utc": "now() - 31 days", "
    time_end_utc": "now()", "violation_plausibility_code": "suspicious"}
```

Output

```
{"test_method": "reaction_single_window_threshold_test", "test_preconditions": {"
    trimmed_consecutive_larger_than": {"RTemp": [0, 6], "Cho2PipeModCmd": [0.99, −6], "
    MxCrt_Pu_Cmd": [0.9, 0]}, "trimmed_consecutive_smaller_than": {"MxCrt_TFl": [19,
    2]}}, "time_start_utc": "now() − 31 days", "time_end_utc": "now()", "
    violation_plausibility_code": "suspicious", "test_specs": {"reaction_signal": "
    RTemp", "window_shift_event_start": 0, "window_shift_event_end": 3, "feature": "
    diffEndMax", "kind": "larger", "threshold": −0.1, "second_threshold": 0.0}, "
    event_dict": {"conditional_poschange_event_end_monotonous": {"RcgVlvPos": {"
    lower_threshold": 1, "upper_threshold": 20, "pre_window": 12, "change_window": 5, "
    post_window": 16, "end_threshold": 20, "max_length_after_condition": 60}}}}
```

12. Room Air Quality 1

Input

The test checks whether the room air quality measurement does increase without presence detected

Expected Output

```
{"test_method": "reaction_single_window_threshold_test", "test_preconditions": {"
    trimmed_consecutive_larger_than": {"RAQual": [0, 3]}}, "event_dict": {"
    variable_conditional_poschange": {"RAQual": {"change": 150, "time_period": 12, "
    pre_window": 3, "post_window": 3}}}, "test_specs": {"reaction_signal": "RPscDet", "
    window_shift_event_start": −12, "window_shift_event_end": 0, "feature": "sum", "
    kind": "lower", "threshold": 0.1, "second_threshold": 0.0}, "time_start_utc": "now
    () − 31 days", "time_end_utc": "now()", "violation_plausibility_code": "implausible
    "}
```

Output

```
{"test_method": "reaction_single_signal_threshold_test", "test_preconditions": {"
    trimmed_consecutive_larger_than": {"RAQual": [0, 3]}, "presence_not_detected": {}},
     "time_start_utc": "now() − 31 days", "time_end_utc": "now()", "
    violation_plausibility_code": "implausible", "test_specs": {"reaction_signal": "
    RAQual", "window_shift_event_start": 0, "window_shift_event_end": 6, "feature": "
    diffEndMax", "kind": "larger", "threshold": 0, "second_threshold": 0}, "event_dict
    ": {}}
```

13. Concrete Temp 3

Input

The test checks whether the concrete temperature reacts on a TABS cooling valve opening

Expected Output

```
{"test_method": "reaction_single_window_threshold_test", "test_virtual_timeseries": {"
    DTConTFl": "ConTemp − TFlC"}, "test_preconditions": {"rolling_feature_threshold":
    {"DTConTFl": {"feature": "mean", "window_size": 12, "kind": "larger", "threshold":
    3}}, "trimmed_consecutive_smaller_than": {"HCmd": [0.01, 12]}}, "event_dict": {"
    conditional_poschange_event_end": {"CCmd": {"lower_threshold": 0.1, "
    upper_threshold": 0.9, "pre_window": 12, "change_window": 2, "post_window": 5, "
    end_threshold": 0.9, "max_length_after_condition": 60}}}, "test_specs": {"
    reaction_signal": "ConTemp", "window_shift_event_start": 0, "window_shift_event_end
    ": 0, "feature": "diffEndMax", "kind": "larger", "threshold": −0.05, "
    second_threshold": 0.0, "accepted_ratio_invalid": 0.2}, "time_start_utc": "now() −
    31 days", "time_end_utc": "now()", "violation_plausibility_code": "suspicious"}
```

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Output

```
{"test_method": "reaction_single_window_threshold_test", "test_preconditions": {"
    rolling_feature_threshold": {"DTConTFl": {"feature": "mean", "window_size": 12, "
    kind": "larger", "threshold": 3}}, "trimmed_consecutive_larger_than": {"PrOpModC":
    [0.01, 0]}, "trimmed_consecutive_smaller_than": {"HCmd": [0.01, 12]}}, "
    time_start_utc": "2022−06−01 00:00:00", "time_end_utc": "2022−07−01 00:00:00", "
    violation_plausibility_code": "suspicious", "test_specs": {"reaction_signal": "
    ConTemp", "window_shift_event_start": 2, "window_shift_event_end": 0, "feature": "
    diffEndMax", "kind": "larger", "threshold": −0.05, "second_threshold": 0.0}, "
    event_dict": {"conditional_poschange_event_end": {"CCmd": {"lower_threshold": 0.1,
    "upper_threshold": 0.9, "pre_window": 6, "change_window": 2, "post_window": 12, "
    end_threshold": 0.9, "max_length_after_condition": 36}}}}
```

14. Concrete Temp 4

Input

The test checks whether the concrete temperature reacts on a TABS cooling valve opening

Expected Output

```
{"test_method": "alternative_test", "test_method_sub": "
    reaction_single_window_threshold_test", "test_virtual_timeseries": {"DTConTFl": "
    ConTemp − TFlC"}, "test_preconditions": {"rolling_feature_threshold": {"DTConTFl":
    {"feature": "mean", "window_size": 12, "kind": "larger", "threshold": 3}}, "
    trimmed_consecutive_larger_than": {"PrOpModC": [0.01, 0]}, "
    trimmed_consecutive_smaller_than": {"HCmd": [0.01, 12]}}, "event_dict": {"
    conditional_poschange_event_end": {"CCmd": {"lower_threshold": 0.1, "
    upper_threshold": 0.9, "pre_window": 6, "change_window": 2, "post_window": 12, "
    end_threshold": 0.9, "max_length_after_condition": 36}}}, "test_specs": {"
    reaction_signal": "ConTemp", "window_shift_event_start": 2, "window_shift_event_end
    ": 0, "feature": "diffEndMax", "kind": "larger", "threshold": −0.05, "
    second_threshold": 0.0, "accepted_ratio_invalid": 0.2}, "time_start_utc":
    "2022−06−01 00:00:00", "time_end_utc": "2022−07−01 00:00:00", "
    violation_plausibility_code": "suspicious"}
```

Output

```
{"test_method": "reaction_single_window_threshold_test", "test_preconditions": {"
    rolling_feature_threshold": {"DTConTC": {"feature": "mean", "window_size": 12, "
    kind": "larger", "threshold": 3}}, "trimmed_consecutive_smaller_than": {"CCmd":
    [0.01, 12]}}, "time_start_utc": "now() − 31 days", "time_end_utc": "now()", "
    violation_plausibility_code": "suspicious", "test_specs": {"reaction_signal": "
    ConTemp", "window_shift_event_start": 0, "window_shift_event_end": 0, "feature": "
    diffEndMax", "kind": "larger", "threshold": −0.05, "second_threshold": 0.0}, "
    event_dict": {"conditional_poschange_event_end": {"TCool": {"lower_threshold": 0.1,
     "upper_threshold": 0.9, "pre_window": 12, "change_window": 2, "post_window": 5, "
    end_threshold": 0.9, "max_length_after_condition": 60}}}}
```

15. Room Air Quality 2

Input

The test checks whether the room air quality measurement does decrease after window opening

Expected Output

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

```
{"test_method": "reaction_single_window_threshold_test", "test_virtual_timeseries": {"
    RAQual_corr": "RAQual - RAQual.quantile(0.05) + 400"}, "test_preconditions": {"
    trimmed_consecutive_larger_than": {"RAQual": [0, 3]}}, "event_dict": {"
    variable_conditional_poschange": {"RWndSta": {"change": 0.9, "time_period": 2, "
    pre_window": 6, "post_window": 6}}}, "test_specs": {"reaction_signal": "RAQual_corr
    ", "window_shift_event_start": 0, "window_shift_event_end": 6, "feature": "
    diffEndMax", "kind": "larger", "threshold": -100, "reaction_start_min": 800}, "
    time_start_utc": "now() - 31 days", "time_end_utc": "now()", "
    violation_plausibility_code": "implausible"}
```

Output

```
{"test_method": "reaction_single_window_threshold_test", "test_preconditions": {"
    trimmed_consecutive_larger_than": {"RAQual": [0, 3]}}, "time_start_utc": "now() -
    31 days", "time_end_utc": "now()", "violation_plausibility_code": "implausible", "
    test_specs": {"reaction_signal": "RAQual", "window_shift_event_start": -3, "
    window_shift_event_end": 0, "feature": "diffEndMax", "kind": "lower", "threshold":
    0.1, "second_threshold": 0.0}, "event_dict": {"conditional_poschange_event_end": {"
    RWndSta": {"lower_threshold": 0.1, "upper_threshold": 0.9, "pre_window": 3, "
    change_window": 3, "post_window": 3, "end_threshold": 0.1, "
    max_length_after_condition": 30}}}}
```

16. Room Temp 8

Input

The test checks whether the room temperature reacts on a radiator heating valve opening

Expected Output

```
{"test_method": "reaction_single_window_threshold_test", "test_preconditions": {"
    trimmed_consecutive_larger_than": {"RTemp": [5, 12], "MxCrt_Pu_Cmd": [0.9, 12], "
    MxCrt_TFl": [30, 12]}}, "event_dict": {"conditional_poschange_event_end_monotonous
    ": {"RadVlvPos": {"lower_threshold": 10, "upper_threshold": 30, "pre_window": 12, "
    change_window": 5, "post_window": 12, "end_threshold": 30, "
    max_length_after_condition": 60}}}, "test_specs": {"reaction_signal": "RTemp", "
    window_shift_event_start": 0, "window_shift_event_end": 6, "feature": "diffEndMin",
     "kind": "lower", "threshold": 0.1, "second_threshold": 0.0, "
    accepted_ratio_invalid": 0.2}, "time_start_utc": "now() - 31 days", "time_end_utc":
     "now()", "violation_plausibility_code": "suspicious"}
```

Output

```
{}
```

17. Air Flow 1

Input

The test checks whether the opening of a VAV supply air damper causes an air flow rise

Expected Output

```
{"test_method": "reaction_single_window_threshold_test", "test_virtual_variables": {"
    tshAirFlSu": "max(VavSuAirFl.quantile(0.95) * 0.05, 2)"}, "test_preconditions": {"
    delayed_consecutive_larger_than": {"FanSu_OpMod": [0.9, -6]}}, "event_dict": {"
    conditional_poschange": {"VavSuPos": {"lower_threshold": 10, "upper_threshold": 30,
     "pre_window": 3, "change_window": 5, "post_window": 7}}}, "test_specs": {"
    reaction_signal": "VavSuAirFl", "window_shift_event_start": -2, "
    window_shift_event_end": 6, "feature": "diff", "kind": "lower", "threshold": "
    tshAirFlSu", "second_threshold": 0}, "time_start_utc": "now() − 31 days", "
    time_end_utc": "now()", "violation_plausibility_code": "implausible"}
```

Output

```
{"test_method": "reaction_single_window_threshold_test", "test_preconditions": {"
    trimmed_consecutive_larger_than": {"FlMtr": [0, 6]}, "conditional_poschange": {"
    VavSuAirFl": {"lower_threshold": 5, "upper_threshold": 30, "pre_window": 12, "
    change_window": 5, "post_window": 24}}}, "time_start_utc": "now() − 31 days", "
    time_end_utc": "now()", "violation_plausibility_code": "implausible", "test_specs":
     {"reaction_signal": "FlMtr", "window_shift_event_start": 6, "
    window_shift_event_end": 24, "feature": "diffEndMax", "kind": "larger", "threshold
    ": 0, "second_threshold": 0}, "event_dict": {}}
```

18. Air Flow 2

Input

The test checks whether a air flow is measured, when the damper is open but ther should no airflow

Expected Output

```
{"test_method": "alternative_test", "test_method_sub": "
    single_signal_threshold_test_moving_window", "test_virtual_variables": {"
    tshAirFlSuLo": "max(VavSuAirFl.quantile(0.95) ∗ 0.1, 10)", "tshAirFlSuHi": "max(
    VavSuAirFl.quantile(0.95) ∗ 0.2, 20)"}, "test_preconditions": {"weekday_selection":
     {"week_day_list": [0, 1, 2, 3, 4, 5, 6]}, "trimmed_consecutive_larger_than": {"
    VavSuPos": [5, 6]}, "trimmed_consecutive_smaller_than": {"FanSu_OpMod": [0.9, 6]}},
     "test_specs": {"kind": "larger", "threshold": "tshAirFlSuLo", "second_threshold":
    "tshAirFlSuHi", "stat_feature": "mean", "window_length": 12, "min_periods": 10}, "
    time_start_utc": "now() − 30 days", "time_end_utc": "now()", "
    violation_plausibility_code": "implausible", "violation_agg_func_runs": "
    mean_time_weighted"}
```

Output

```
{"test_method": "alternative_inv_test", "test_preconditions": {"weekday_selection": {"
    week_day_list": [0, 1, 2, 3, 4, 5, 6]}, "trimmed_consecutive_larger_than": {"
    VavSuPos": [5, 6]}}, "time_start_utc": "now() − 30 days", "time_end_utc": "now()",
    "violation_plausibility_code": "implausible", "test_method_sub": "
    single_signal_threshold_test_moving_window", "test_virtual_variables": {"
    tshAirFlSuLo": "max(VavSuAirFl.quantile(0.95) ∗ 0.1, 10)", "tshAirFlSuHi": "max(
    VavSuAirFl.quantile(0.95) ∗ 0.2, 20)"}, "event_dict": null, "test_specs": {"kind":
    "larger", "threshold": "tshAirFlSuHi", "second_threshold": "tshAirFlSuLo", "
    stat_feature": "mean", "window_length": 12, "min_periods": 10}, "
    violation_agg_func_runs": "mean_time_weighted"}
```

19. Room Presence 1

Input

The test checks whether there is very high room presence over a long period of time

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Expected Output

```
{"test_method": "single_signal_threshold_test_moving_window", "test_preconditions": {"
    weekday_selection": {"week_day_list": [0, 1, 2, 3, 4, 5, 6]}}, "test_specs": {"kind
    ": "larger", "threshold": 0.9, "stat_feature": "mean", "window_length": 4032, "
    min_periods": 3024}, "time_start_utc": "now() - 31 days", "time_end_utc": "now()",
    "violation_plausibility_code": "implausible", "violation_agg_func_runs": "
    mean_time_weighted"}
```

Output

```
{"test_method": "single_signal_threshold_test_moving_window", "test_preconditions": {"
    weekday_selection": {"week_day_list": [0, 1, 2, 3, 4, 5, 6]}, "cond_larger_than":
    {"RoomOccupancy": 0.8}}, "time_start_utc": "now() - 31 days", "time_end_utc": "now
    ()", "violation_plausibility_code": "implausible", "test_method_sub": null, "
    test_virtual_variables": null, "event_dict": null, "test_specs": {"kind": "larger",
     "threshold": 0.8, "second_threshold": null, "stat_feature": "mean", "window_length
    ": 17520, "min_periods": 12960}, "violation_agg_func_runs": "mean_time_weighted"}
```

20. Air Flow 3

Input

The test checks whether a significant supply VAV flow is measured although the damper is commanded closed

Expected Output

```
{"test_method": "single_signal_threshold_test_moving_window", "test_virtual_variables":
     {"tshAirFlSuLo": "max(VavSuAirFl.quantile(0.95) * 0.1, 10)", "tshAirFlSuHi": "max(
    VavSuAirFl.quantile(0.95) * 0.2, 20)"}, "test_preconditions": {"weekday_selection":
     {"week_day_list": [0, 1, 2, 3, 4, 5, 6]}, "trimmed_consecutive_smaller_than": {"
    VavSuPos": [0.1, 6]}}, "test_specs": {"kind": "larger", "threshold": "tshAirFlSuLo
    ", "second_threshold": "tshAirFlSuHi", "stat_feature": "mean", "window_length": 12,
     "min_periods": 10}, "time_start_utc": "now() - 31 days", "time_end_utc": "now()",
    "violation_plausibility_code": "implausible", "violation_agg_func_runs": "
    mean_time_weighted"}
```

Output

```
{"test_method": "alternative_inv_test", "test_preconditions": {"weekday_selection": {"
    week_day_list": [0, 1, 2, 3, 4, 5, 6]}, "trimmed_consecutive_smaller_than": {"
    VavSuPos": [0.1, 6]}}, "time_start_utc": "now() - 30 days", "time_end_utc": "now()
    ", "violation_plausibility_code": "implausible", "test_method_sub": "
    single_signal_threshold_test_moving_window", "test_virtual_variables": {"
    tshAirFlSuLo": "max(VavSuAirFl.quantile(0.95) * 0.1, 10)"}, "event_dict": null, "
    test_specs": {"kind": "larger", "threshold": "tshAirFlSuLo", "second_threshold":
    null, "stat_feature": "mean", "window_length": 12, "min_periods": 10}, "
    violation_agg_func_runs": "mean_time_weighted"}
```

21. Room Air Quality 3

Input

The test checks whether the estimated room air quality measurement is high when unoccupied

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Expected Output

```
{"test_method": "single_signal_threshold_test_moving_window", "test_preconditions": {"
    weekday_selection": {"week_day_list": [0, 1, 2, 3, 4, 5, 6]}, "
    consecutive_smaller_than": {"RPscDet": [0.1, 288]}}, "test_specs": {"kind": "larger
    ", "threshold": 600, "second_threshold": 900, "stat_feature": "percentile", "
    feature_params": {"percentile": 5}, "window_length": 288}, "time_start_utc": "now()
     - 31 days", "time_end_utc": "now()", "violation_plausibility_code": "implausible",
     "violation_agg_func_runs": "mean_time_weighted"}
```

Output

```
{"test_method": "single_signal_threshold_test_moving_window", "test_preconditions": {"
    weekday_selection": {"week_day_list": [0, 1, 2, 3, 4, 5, 6]}, "
    cond_nan_or_larger_than": {"occupancy_status": 0}}, "time_start_utc": "now() - 31
    days", "time_end_utc": "now()", "violation_plausibility_code": "implausible", "
    test_method_sub": null, "test_virtual_variables": null, "event_dict": null, "
    test_specs": {"kind": "larger", "threshold": 200, "second_threshold": null, "
    stat_feature": "percentile", "window_length": 4032, "min_periods": 2016}, "
    violation_agg_func_runs": "mean_time_weighted"}
```

22. Air Flow 4

Input

The test checks whether a air flow is measured, when the damper is open but ther should no airflow

Expected Output

```
{"test_method": "alternative_inv_test", "test_method_sub": "
    single_signal_threshold_test_moving_window", "property_inv": {"Datapoint": "
    VavSuPos", "ValueMax": 100}, "test_virtual_variables": {"tshAirFlSuLo": "max(
    VavSuAirFl.quantile(0.95) * 0.1, 10)", "tshAirFlSuHi": "max(VavSuAirFl.quantile
    (0.95) * 0.2, 20)"}, "test_preconditions": {"weekday_selection": {"week_day_list":
    [0, 1, 2, 3, 4, 5, 6]}, "trimmed_consecutive_larger_than": {"VavSuPos": [5, 6]}, "
    trimmed_consecutive_smaller_than": {"FanSu_OpMod": [0.9, 6]}}, "test_specs": {"kind
    ": "larger", "threshold": "tshAirFlSuLo", "second_threshold": "tshAirFlSuHi", "
    stat_feature": "mean", "window_length": 12, "min_periods": 10}, "time_start_utc": "
    now() - 30 days", "time_end_utc": "now()", "violation_plausibility_code": "
    implausible", "violation_agg_func_runs": "mean_time_weighted"}
```

Output

```
{"test_method": "single_signal_threshold_test_moving_window", "test_preconditions": {"
    weekday_selection": {"week_day_list": [0, 1, 2, 3, 4, 5, 6]}, "
    trimmed_consecutive_larger_than": {"VavSuPos": [5, 6], "FanSu_OpMod": [0.9, 6]}}, "
    time_start_utc": "now() - 30 days", "time_end_utc": "now()", "
    violation_plausibility_code": "implausible", "test_method_sub": null, "
    test_virtual_variables": {"tshAirFlSuLo": "max(VavSuAirFl.quantile(0.95) * 0.1, 10)
    ", "tshAirFlSuHi": "max(VavSuAirFl.quantile(0.95) * 0.2, 20)"}, "event_dict": null,
     "test_specs": {"kind": "lower", "threshold": "tshAirFlSuHi", "second_threshold": "
    tshAirFlSuLo", "stat_feature": "mean", "window_length": 12, "min_periods": 10}, "
    violation_agg_func_runs": "mean_time_weighted"}
```

23. Air Flow 5

Input

The test checks whether a significant supply VAV flow is measured whenever the damper is commanded open

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Expected Output

```
{"test_method": "single_signal_threshold_test_moving_window", "test_virtual_variables":
     {"tshAirFlSuLo": "max(VavSuAirFl.quantile(0.95) * 0.1, 10)", "tshAirFlSuHi": "max(
    VavSuAirFl.quantile(0.95) * 0.2, 20)"}, "test_preconditions": {"weekday_selection":
     {"week_day_list": [0, 1, 2, 3, 4, 5, 6]}, "trimmed_consecutive_larger_than": {"
    VavSuPos": [5, 6], "FanSu_OpMod": [0.9, 6]}}, "test_specs": {"kind": "lower", "
    threshold": "tshAirFlSuHi", "second_threshold": "tshAirFlSuLo", "stat_feature": "
    mean", "window_length": 12, "min_periods": 10}, "time_start_utc": "now() - 31 days
    ", "time_end_utc": "now()", "violation_plausibility_code": "implausible", "
    violation_agg_func_runs": "mean_time_weighted"}
```

Output

```
{"test_method": "alternative_test", "test_preconditions": {"weekday_selection": {"
    week_day_list": [0, 1, 2, 3, 4, 5, 6]}, "trimmed_consecutive_larger_than": {"
    VavSuPos": [5, 6], "FanSu_OpMod": [0.9, 6]}}, "time_start_utc": "now() - 30 days",
    "time_end_utc": "now()", "violation_plausibility_code": "implausible", "
    test_method_sub": "single_signal_threshold_test_moving_window", "
    test_virtual_variables": {"tshAirFlSuLo": "max(VavSuAirFl.quantile(0.95) * 0.1, 10)
    ", "tshAirFlSuHi": "max(VavSuAirFl.quantile(0.95) * 0.2, 20)"}, "event_dict": null,
     "test_specs": {"kind": "larger", "threshold": "tshAirFlSuHi", "second_threshold":
    "tshAirFlSuLo", "stat_feature": "mean", "window_length": 12, "min_periods": 10}, "
    violation_agg_func_runs": "mean_time_weighted"}
```

24. Air Flow 6

Input

The test checks whether a significant supply VAV flow is measured although the damper is commanded closed

Expected Output

```
{"test_method": "alternative_inv_test", "test_method_sub": "
    single_signal_threshold_test_moving_window", "property_inv": {"Datapoint": "
    VavSuPos", "ValueMax": 100}, "test_virtual_variables": {"tshAirFlSuLo": "max(
    VavSuAirFl.quantile(0.95) * 0.1, 10)", "tshAirFlSuHi": "max(VavSuAirFl.quantile
    (0.95) * 0.2, 20)"}, "test_preconditions": {"weekday_selection": {"week_day_list":
    [0, 1, 2, 3, 4, 5, 6]}, "trimmed_consecutive_smaller_than": {"VavSuPos": [0.1,
    6]}}, "test_specs": {"kind": "larger", "threshold": "tshAirFlSuLo", "
    second_threshold": "tshAirFlSuHi", "stat_feature": "mean", "window_length": 12, "
    min_periods": 10}, "time_start_utc": "now() - 30 days", "time_end_utc": "now()", "
    violation_plausibility_code": "implausible", "violation_agg_func_runs": "
    mean_time_weighted"}
```

Output

```
{"test_method": "single_signal_threshold_test_moving_window", "test_preconditions": {"
    VavSuPos": {"equal_to": 0}}, "time_start_utc": "now() - 30 days", "time_end_utc": "
    now()", "violation_plausibility_code": "implausible", "test_method_sub": null, "
    test_virtual_variables": {"tshAirFlSuLo": "max(VavSuAirFl.quantile(0.95) * 0.1, 10)
    "}, "event_dict": null, "test_specs": {"kind": "larger", "threshold": "tshAirFlSuLo
    ", "second_threshold": null, "stat_feature": "mean", "window_length": 12, "
    min_periods": 10}, "violation_agg_func_runs": "mean_time_weighted"}
```

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

25. Air Flow 7

Input

The test checks whether a significant supply VAV flow is measured whenever the damper is commanded open

Expected Output

```
{"test_method": "alternative_inv_test", "test_method_sub": "
    single_signal_threshold_test_moving_window", "property_inv": {"Datapoint": "
    VavSuPos", "ValueMax": 100}, "test_virtual_variables": {"tshAirFlSuLo": "max(
    VavSuAirFl.quantile(0.95) * 0.1, 10)", "tshAirFlSuHi": "max(VavSuAirFl.quantile
    (0.95) * 0.2, 20)"}, "test_preconditions": {"weekday_selection": {"week_day_list":
    [0, 1, 2, 3, 4, 5, 6]}, "trimmed_consecutive_larger_than": {"VavSuPos": [5, 6], "
    FanSu_OpMod": [0.9, 6]}}, "test_specs": {"kind": "lower", "threshold": "
    tshAirFlSuHi", "second_threshold": "tshAirFlSuLo", "stat_feature": "mean", "
    window_length": 12, "min_periods": 10}, "time_start_utc": "now() − 30 days", "
    time_end_utc": "now()", "violation_plausibility_code": "implausible", "
    violation_agg_func_runs": "mean_time_weighted"}
```

Output

```
{"test_method": "single_signal_threshold_test_moving_window", "test_preconditions": {"
    weekday_selection": {"week_day_list": [0, 1, 2, 3, 4, 5, 6]}, "
    trimmed_consecutive_larger_than": {"VavSuPos": [5, 6], "FanSu_OpMod": [0.9, 6]}}, "
    time_start_utc": "now() − 30 days", "time_end_utc": "now()", "
    violation_plausibility_code": "implausible", "test_method_sub": null, "
    test_virtual_variables": {"tshAirFlSuLo": "max(VavSuAirFl.quantile(0.95) * 0.1, 10)
    ", "tshAirFlSuHi": "max(VavSuAirFl.quantile(0.95) * 0.2, 20)"}, "event_dict": null,
     "test_specs": {"kind": "lower", "threshold": "tshAirFlSuHi", "second_threshold": "
    tshAirFlSuLo", "stat_feature": "mean", "window_length": 12, "min_periods": 10}, "
    violation_agg_func_runs": "mean_time_weighted"}
```

26. Room Air Quality 4

Input

The test checks whether the estimated room air quality offset is high

Expected Output

```
{"test_method": "single_signal_threshold_test_moving_window", "test_preconditions": {"
    weekday_selection": {"week_day_list": [0, 1, 2, 3, 4, 5, 6]}, "
    cond_nan_or_larger_than": {"RAQual": 0}}, "test_specs": {"kind": "larger", "
    threshold": 200, "second_threshold": 500, "stat_feature": "percentile", "
    feature_params": {"percentile": 5}, "window_length": 4032, "min_periods": 2016, "
    comparison_offset": 400}, "time_start_utc": "now() − 31 days", "time_end_utc": "now
    ()", "violation_plausibility_code": "implausible", "violation_agg_func_runs": "
    mean_time_weighted"}
```

Output

```
{"test_method": "single_signal_threshold_test_moving_window", "test_preconditions": {"
    weekday_selection": {"week_day_list": [0, 1, 2, 3, 4, 5, 6]}}, "time_start_utc": "
    now() − 31 days", "time_end_utc": "now()", "violation_plausibility_code": "
    implausible", "test_method_sub": null, "test_virtual_variables": null, "event_dict
    ": null, "test_specs": {"kind": "larger", "threshold": 500, "second_threshold":
    null, "stat_feature": "percentile", "window_length": 1440, "min_periods": null}, "
    violation_agg_func_runs": "mean_time_weighted"}
```

27. Air Flow 8

Input

The test checks whether a significant supply VAV flow is measured whenever the damper is commanded open

Expected Output

```
{"test_method": "alternative_test", "test_method_sub": "
    single_signal_threshold_test_moving_window", "test_virtual_variables": {"
    tshAirFlSuLo": "max(VavSuAirFl.quantile(0.95) * 0.1, 10)", "tshAirFlSuHi": "max(
    VavSuAirFl.quantile(0.95) * 0.2, 20)"}, "test_preconditions": {"weekday_selection":
     {"week_day_list": [0, 1, 2, 3, 4, 5, 6]}, "trimmed_consecutive_larger_than": {"
    VavSuPos": [5, 6], "FanSu_OpMod": [0.9, 6]}}, "test_specs": {"kind": "lower", "
    threshold": "tshAirFlSuHi", "second_threshold": "tshAirFlSuLo", "stat_feature": "
    mean", "window_length": 12, "min_periods": 10}, "time_start_utc": "now() - 30 days
    ", "time_end_utc": "now()", "violation_plausibility_code": "implausible", "
    violation_agg_func_runs": "mean_time_weighted"}
```

Output

```
{"test_method": "single_signal_threshold_test_moving_window", "test_preconditions": {"
    weekday_selection": {"week_day_list": [0, 1, 2, 3, 4, 5, 6]}, "
    trimmed_consecutive_larger_than": {"VavSuPos": [5, 6], "FanSu_OpMod": [0.9, 6]}}, "
    time_start_utc": "now() - 30 days", "time_end_utc": "now()", "
    violation_plausibility_code": "implausible", "test_method_sub": null, "
    test_virtual_variables": {"tshAirFlSuLo": "max(VavSuAirFl.quantile(0.95) * 0.1, 10)
    ", "tshAirFlSuHi": "max(VavSuAirFl.quantile(0.95) * 0.2, 20)"}, "event_dict": null,
     "test_specs": {"kind": "lower", "threshold": "tshAirFlSuHi", "second_threshold": "
    tshAirFlSuLo", "stat_feature": "mean", "window_length": 12, "min_periods": 10}, "
    violation_agg_func_runs": "mean_time_weighted"}
```

28. Room Presence 2

Input

The test checks whether there is no room presence over a long period of time

Expected Output

```
{"test_method": "single_signal_threshold_test_moving_window", "test_preconditions": {"
    weekday_selection": {"week_day_list": [0, 1, 2, 3, 4, 5, 6]}}, "test_specs": {"kind
    ": "lower", "threshold": 1e-06, "stat_feature": "mean", "window_length": 4032, "
    min_periods": 3024}, "time_start_utc": "now() - 31 days", "time_end_utc": "now()",
    "violation_plausibility_code": "implausible", "violation_agg_func_runs": "
    mean_time_weighted"}
```

Output

```
{"test_method": "single_signal_threshold_test_moving_window", "test_preconditions": {"
    weekday_selection": {"week_day_list": [0, 1, 2, 3, 4, 5, 6]}, "cond_smaller_than":
    {"room_presence": 0}}, "time_start_utc": "now() - 31 days", "time_end_utc": "now()
    ", "violation_plausibility_code": "implausible", "test_method_sub": null, "
    test_virtual_variables": null, "event_dict": null, "test_specs": {"kind": "smaller
    ", "threshold": 0, "second_threshold": null, "stat_feature": "mean", "window_length
    ": 14400, "min_periods": 10080}, "violation_agg_func_runs": "mean_time_weighted"}
```

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

29. Room Air Quality 5

Input

The test checks whether the estimated room air quality offset is low

Expected Output

```
{"test_method": "single_signal_threshold_test_moving_window", "test_preconditions": {"
    weekday_selection": {"week_day_list": [0, 1, 2, 3, 4, 5, 6]}, "
    cond_nan_or_larger_than": {"RAQual": 0}}, "test_specs": {"kind": "lower", "
    threshold": -200, "second_threshold": -500, "stat_feature": "percentile", "
    feature_params": {"percentile": 5}, "window_length": 4032, "min_periods": 2016, "
    comparison_offset": 400}, "time_start_utc": "now() - 31 days", "time_end_utc": "now
    ()", "violation_plausibility_code": "implausible", "violation_agg_func_runs": "
    mean_time_weighted"}
```

Output

```
{"test_method": "single_signal_threshold_test_moving_window", "test_preconditions": {"
    weekday_selection": {"week_day_list": [0, 1, 2, 3, 4, 5, 6]}, "cond_smaller_than":
    {"RAQualOffset": 50}}, "time_start_utc": "now() - 31 days", "time_end_utc": "now()
    ", "violation_plausibility_code": "plausible", "test_method_sub": null, "
    test_virtual_variables": null, "event_dict": null, "test_specs": {"kind": "smaller
    ", "threshold": 50, "second_threshold": 25, "stat_feature": "mean", "window_length
    ": 1440, "min_periods": 7}, "violation_agg_func_runs": "mean_time_weighted"}
```

30. Air Flow 9

Input

The test checks whether a significant supply VAV flow is measured although the damper is commanded closed

Expected Output

```
{"test_method": "alternative_test", "test_method_sub": "
    single_signal_threshold_test_moving_window", "test_virtual_variables": {"
    tshAirFlSuLo": "max(VavSuAirFl.quantile(0.95) * 0.1, 10)", "tshAirFlSuHi": "max(
    VavSuAirFl.quantile(0.95) * 0.2, 20)"}, "test_preconditions": {"weekday_selection":
     {"week_day_list": [0, 1, 2, 3, 4, 5, 6]}, "trimmed_consecutive_smaller_than": {"
    VavSuPos": [0.1, 6]}}, "test_specs": {"kind": "larger", "threshold": "tshAirFlSuLo
    ", "second_threshold": "tshAirFlSuHi", "stat_feature": "mean", "window_length": 12,
     "min_periods": 10}, "time_start_utc": "now() - 30 days", "time_end_utc": "now()",
    "violation_plausibility_code": "implausible", "violation_agg_func_runs": "
    mean_time_weighted"}
```

Output

```
{"test_method": "alternative_inv_test", "test_preconditions": {"weekday_selection": {"
    week_day_list": [0, 1, 2, 3, 4, 5, 6]}, "trimmed_consecutive_smaller_than": {"
    VavSuPos": [0.1, 6]}}, "time_start_utc": "now() - 30 days", "time_end_utc": "now()
    ", "violation_plausibility_code": "implausible", "test_method_sub": "
    single_signal_threshold_test_moving_window", "test_virtual_variables": {"
    tshAirFlSuLo": "max(VavSuAirFl.quantile(0.05) * 0.1, 10)", "tshAirFlSuHi": "max(
    VavSuAirFl.quantile(0.05) * 0.2, 20)"}, "event_dict": null, "test_specs": {"kind":
    "smaller", "threshold": "tshAirFlSuLo", "second_threshold": "tshAirFlSuHi", "
    stat_feature": "mean", "window_length": 12, "min_periods": 10}, "
    violation_agg_func_runs": "mean_time_weighted"}
```

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

31. Temp Deviation 1

Input

The test checks whether the average concrete and room temperatures over long durations (>= 1 day) deviate

Expected Output

```
{"test_method": "similarity_difference_threshold_test_moving_window", "
    test_preconditions": {"weekday_selection": {"week_day_list": [0, 1, 2, 3, 4, 5,
    6]}, "delayed_consecutive_smaller_than": {"HCmd": [0.1, 288], "CCmd": [0.1, 288]}},
     "test_specs": {"kind": "larger_abs", "threshold": 2.0, "second_threshold": 4.0, "
    stat_feature": "mean", "window_length": 576, "min_periods": 576}, "time_start_utc":
     "now() − 31 days", "time_end_utc": "now()", "violation_plausibility_code": "
    implausible", "violation_agg_func_runs": "mean_time_weighted"}
```

Output

```
{"test_method": "deviation_threshold_test_moving_window", "test_preconditions": {"
    rolling_feature_threshold": {"Concrete_Temperature": {"feature": "mean", "
    window_size": 1440, "kind": "deviation"}, "Room_Temperature": {"feature": "mean", "
    window_size": 1440, "kind": "deviation"}}}, "time_start_utc": "now() − 1 day", "
    time_end_utc": "now()", "violation_plausibility_code": "implausible", "test_specs":
     {"kind": "absolute", "threshold": 1.0, "second_threshold": null, "stat_feature": "
    mean", "window_length": 1440, "min_periods": 24}, "violation_agg_func_runs": "
    mean_time_weighted"}
```

32. Temp Cooling 1

Input

The test checks whether the average concrete temperature is lower than the room temperatures over long durations (>= 1 day) when cooling

Expected Output

```
{"test_method": "similarity_difference_threshold_test_moving_window", "
    test_preconditions": {"rolling_feature_threshold": {"CCmd": {"feature": "mean", "
    window_size": 288, "kind": "larger", "threshold": 0.2}}, "
    delayed_consecutive_smaller_than": {"HCmd": [0.1, 288]}}, "test_specs": {"kind": "
    larger", "threshold": −1.0, "second_threshold": −0.5, "stat_feature": "mean", "
    window_length": 288, "min_periods": 264}, "time_start_utc": "now() − 31 days", "
    time_end_utc": "now()", "violation_plausibility_code": "implausible", "
    violation_agg_func_runs": "mean_time_weighted"}
```

Output

```
{"test_method": "similarity_difference_threshold_test_moving_window", "
    test_preconditions": {"rolling_feature_threshold": {"CCmd": {"feature": "mean", "
    window_size": 288, "kind": "larger", "threshold": 0.2}}, "
    delayed_consecutive_larger_than": {"HCmd": [0.1, 288]}}, "time_start_utc": "now() −
     31 days", "time_end_utc": "now()", "violation_plausibility_code": "implausible", "
    test_specs": {"kind": "lower", "threshold": 1.0, "second_threshold": 0.5, "
    stat_feature": "mean", "window_length": 288, "min_periods": 264}, "
    violation_agg_func_runs": "mean_time_weighted"}
```

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

33. Temp Heating 1

Input

The test checks whether the average concrete temperature is greater than the room temperatures over long durations (>= 1 day) when heating

Expected Output

```
{"test_method": "similarity_difference_threshold_test_moving_window", "
    test_preconditions": {"rolling_feature_threshold": {"HCmd": {"feature": "mean", "
    window_size": 288, "kind": "larger", "threshold": 0.2}}, "
    delayed_consecutive_smaller_than": {"CCmd": [0.1, 288]}}, "test_specs": {"kind": "
    lower", "threshold": 1.0, "second_threshold": 0.5, "stat_feature": "mean", "
    window_length": 288, "min_periods": 264}, "time_start_utc": "now() - 31 days", "
    time_end_utc": "now()", "violation_plausibility_code": "implausible", "
    violation_agg_func_runs": "mean_time_weighted"}
```

Output

```
{"test_method": "similarity_difference_threshold_test_moving_window", "
    test_preconditions": {"weekday_selection": {"week_day_list": [0, 1, 2, 3, 4, 5,
    6]}, "delayed_consecutive_larger_than": {"HCmd": [0.1, 288], "CCmd": [0.1, 288]}},
    "time_start_utc": "now() - 7 days", "time_end_utc": "now()", "
    violation_plausibility_code": "implausible", "test_specs": {"kind": "smaller_abs",
    "threshold": 0.0, "second_threshold": -0.5, "stat_feature": "mean", "window_length
    ": 1728, "min_periods": 1728}, "violation_agg_func_runs": "mean_time_weighted"}
```