FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Methods and Framework for Data-Driven Building Service Analytics

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Software Engineering & Internet Computing

eingereicht von

David Bese Molnar, BSc
Matrikelnummer 01326891

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof.Dr. Wolfgang Kastner

Wien, 26. April 2020

David Bese Molnar	Wolfgang Kastner

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.ac.at

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Methods and Framework for Data-Driven Building Service Analytics

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Software Engineering & Internet Computing

by

David Bese Molnar, BSc
Registration Number 01326891

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof.Dr. Wolfgang Kastner

Vienna, 26th April, 2020

David Bese Molnar	Wolfgang Kastner

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.ac.at

Erklärung zur Verfassung der Arbeit

David Bese Molnar, BSc
Burggasse 128/16-17, 1070 Wien

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.

Wien, 26. April 2020

David Bese Molnar

Danksagung

Zuerst möchte ich mich bei Herrn Ao.Univ.Prof.Dr. Wolfgang Kastner bedanken, der mir die Möglichkeit gegeben hat, an diesem spannenden Projekt mitwirken zu können und der immer bereit war mit seiner Betreuung und wertvollem Input das Projekt weiterzubewegen.

Ohne Lukas Krammer, Konrad Diwold und Daniel Lechner hätte das Projekt nicht zustande kommen können.

Mein besonderer Dank ergeht an meine Eltern, die mein Studium ermöglicht haben und mich immer unterstützt haben.

Danke Viki!

Acknowledgements

At first, I would like to thank Mr. Ao.Univ.Prof.Dr. Wolfgang Kastner who gave me the opportunity to participate in this exciting project and who was always ready to move the project forward with his support and valuable input.

The project could not have been succeeded without Lukas Krammer, Konrad Diwold and Daniel Lechner.

My special thanks go to my parents who have made my studies possible and have always supported me.

Thank you Viki!

Kurzfassung

Bestandsgebäude werden oft über einen langen Zeitraum unter unentdeckten, suboptimalen Bedingungen betrieben. Optimierungspotenziale bleiben weitgehend unbemerkt, wodurch zusätzliche Betriebskosten, welche aus den erhöhten Energieverlusten resultieren, verursacht werden. Wärmeverluste durch Beschädigungen der Gebäudestruktur, mangelhafte Isolierung sowie Defekte oder ein sinkender Wirkungsgrad von Teilsystemen in der Gebäudehydraulik können sich negativ auf die Energiebilanz des Gebäudes auswirken. Eine schnelle Erkennung und damit rechtzeitige Behebung solcher Probleme kann zu einem reduzierten Energieverbrauch führen, indem eine optimale Energiebilanz im Gebäude erzielt wird. Neubauten verfügen über die modernste Gebäudetechnologien, einschließlich moderner Bautechniken und bewährter Gebäudemanagementsysteme (BMS). Diese Gebäude sind in der Regel mit mehreren invasiven Sensoren ausgestattet, die eine umfassende Überwachung und Optimierung des Gebäudesystems ermöglichen. Im Gegensatz dazu verfügen ältere Gebäude in der Regel nicht über diese Funktionen und daher ist es nicht einfach, die optimalen Betriebsbedingungen zu ermitteln. In dieser Masterarbeit wurde ein Gebäudeanalyse-Framework (Buiding Service Analysis Engine, BSAE) entwickelt, das anhand von nicht-invasiven Sensoren gesammelten Daten der Gebäudehydraulik Systemprobleme und Optimierungspotenziale erkennen kann. Die Erhebung der Anforderungen an das Analyse-Framework erfolgte anhand von Anwendungsfällen. Das Wissen über die Position der Sensoren im Gebäude ist ein unumgänglicher Bestandteil für die Analyse. Aus diesem Grund wurden die notwendigen Schritte zur Erkennung des zugrunde liegenden Sensornetzwerks auf der Grundlage von Hydraulikdaten identifiziert und entwickelt. Die Evaluierung der Konzepte und Methoden erfolgte anhand von Daten eines Bürogebäudes und anhand der identifizierten Anwendungsfälle. Schließlich wurde ein Teil der Ansätze in ein auf Micro Service Architecture (MSA) basierendes Softwaresystem integriert, um ein einfaches Deployment zu ermöglichen und die Wartbarkeit zu verbessern.

Abstract

Existing buildings often operate on undiscovered suboptimal conditions for a long period of time. Optimization potentials remain mainly unnoticed, which causes additional operation costs that arise from the increased energy loss. Heat loss caused by damage in the building structure, poor isolation as well as defects or falling efficiency of subsystems in building hydraulics may have a negative impact on the energy balance of the building. A quick detection and consequently timely fix of such problems can lead to a more optimal and reduced energy consumption by ensuring an optimal energy balance in the building. Recently built buildings provide state of the art building technologies including modern construction techniques and well established Building Management Systems (BMS). These buildings are usually equipped with several invasive sensors which enables a wide range of building system monitoring and optimization. Contrarily, older buildings typically do not have these capabilities and thus, the identification of suboptimal operating conditions for them is not easy to perform. In this master thesis, we developed a so-called Building Service Analysis Engine (BSAE) which is able to identify system flaws and optimization potentials by means of building hydraulics data gathered by non-invasive sensors. The requirements for the analysis framework were defined based on use cases. The knowledge about the spatial arrangement of the sensors in the building is an indispensable part for the analysis. Therefore, we identified and developed the necessary steps to detect the underlying sensor network based on hydraulic data. The evaluation of the concepts and methods was done on data from an occupied office building and against the identified use cases. Finally, we integrated a part of the approaches into a Micro Service Architecture (MSA) based software system to enable easy deployment and to improve maintainability.

Contents

CHAPTER 1

Introduction

1.1 Motivation

Energy usage of buildings makes up about 40% of the total energy consumption worldwide [MF15]. Both residential and commercial building facilities show an increasing demand for energy consumption in the future due to the growing need for comfort and usage time. Studies illustrate, that almost 32% of the energy demand of building facilities is required by heating, ventilation, and air-conditioning systems (HVAC) [SHPV11]. The misconfiguration of components in HVAC systems covers up to 40% of energy waste in building systems [BNGA14]. Buildings often operate on undiscovered suboptimal conditions for a long period of time. These optimization potentials remain mainly unnoticed, which causes additional operation costs and a huge amount of energy waste [STD+18]. For this reason, the European Union aims to decrease energy consumptions for buildings and to improve building energy efficiency [1].

Recently built buildings provide modern construction techniques with state of the art building technologies and well established Building Management Systems (BMS). These building are usually equipped with invasive sensors, which enables a wide range of building system monitoring and optimization services [STD+18]. Contrarily, older buildings typically do not have such technology installed and thus, the identification of suboptimal operating conditions for them is not easy to execute. Retrofitting an existing building with a building energy management system and the associated sensor network technology would require a lot of construction work and investment, and therefore, is often not feasible. Cheap sensor technologies should be designed that are able to gain information about building behaviour with low engineering effort as minimizing construction work and cost. The manual description of equipped sensor topology would require huge engineering effort, which is not feasible on a big scale. Therefore, this thesis aims to identify the

[1] https://ec.europa.eu/energy/en/topics/energy-strategy-and-energy-union

underlying sensor topology based only on measurement data (flow temperature, return flow temperature, flow rate).

Fault detection and diagnosis (FDD) is a process that aims to discover malfunctioning components in building systems and tries to identify optimization potentials by detecting the route cause of faults and flaws. Many methods and concepts have been proposed to detect faults in HVAC systems using available sensor data [KB05]. This thesis aims to identify possible faults in the building's heating system and to analyze and integrate simple fault detection concepts. The identified concepts and methods will be integrated into a building analytics framework that is able to locate optimization potentials on existing buildings.

1.2 Problem Definition

Heat loss caused by damage in the building structure (e.g. broken window), poor isolation as well as defects or falling efficiency of subsystems in building hydraulics may have a negative impact on the energy balance of the building. A quick detection and consequently timely fix of such problems can lead to a more optimal and reduced energy consumption by ensuring an optimal energy balance in the building. Smart buildings already have their own Building Management Systems (BMS) [SZ12] which often provide fault detection features [CM13]. In these buildings, many different types of data are collected, that allows a wide range of analysis. However, buildings have a very long life cycle - and the majority of the buildings is deployed with legacy automation systems. In buildings with no such analysis systems or without a variety of equipped sensors, the analyses in order to identify problems in the building could be very cost-intensive and complicated.

The analysis of existing buildings in a cost effective manner requires solutions with low engineering effort, such as designing non-invasive sensor networks that can be deployed in legacy building systems. However, data gathered from such sensor networks is unstructured and thus, the task is to find structures in the unstructured data and extract information from it. The quality and quantity of this data is also limited, missing or incomplete data has to be completed. In order to enable the analysis (e.g. detect flaws or defects), the spatial arrangement of the distributed sensors has to be known. Since the application has to be able to be applied to any kind of buildings, there is no topological information of the sensor positions available. The manually specification of this topology is not feasible.

1.3 Aim of the Work

The expected output from this thesis is a framework for building analysis. This analysis engine should be able to detect problems in the building, such as heat loss, defects or inefficiencies in building hydraulics by means of data from the building hydraulics gathered by non-invasive sensors. This framework will be designed to be able to be

applied to any type of building. However, the kind of the recognizable problems and the precision of the analysis can not be determined in advance.

The knowledge about the position of the sensors in the building is an indispensable part for the analysis. This underlying problem, the detection of the sensor network topology in the building using hydraulic data from non-invasive sensors, is another focus of the work. The topology discovery algorithm has to be designed and developed in order to localize sensors within the building based on the collected sensor data. If for some reason this automated detection could not work, potential reasons or limitations will be provided.

1.4 Methodological Approach

The initial step of the thesis is the determination of use cases in the problem domain as we specify requirements and necessary conditions of the framework. This is followed by an extensive literature study about the state-of-the-art solutions to similar problems in the problem space and to better understand the components and needs to cover our entire problem domain. This literature study will cover topics of fault detection and diagnosis and sensor topology discovery, and it aims to gain knowledge about possibilities and limitations about the components of such a system. More precisely, we will study possible solution methods for sensor topology discovery based on data form building's hydraulics and analysis methods to detect flaws and faults within a building. In order to successfully identify and design solution methods for our problem domain, we will gain a deeper knowledge about statistical and machine learning methods on a higher level. Moreover, we will identify promising fault detection and diagnostics concepts in order to be able to draw conclusions about certain problems in buildings. This literature study and selection process is followed by the design and implementation of the topology discovery methods to cover all related use cases along with the identification of possible limitations. This step is followed by the design and implementation of the building analysis methods. The evaluation of concepts and methods will be done using real-world building data from a test building. In the final step, we will integrate the so-called Building Service Analysis Engine (BSAE) using appropriate software architecture concepts and make it ready for deployment in a cloud environment.

1.5 Related Work

In this section, we will perform an extensive literature study to analyze and summarize state-of-the-art solution approaches in the field of sensor topology discovery and fault detection and diagnosis.

1.5.1 Sensor Topology Discovery

Most scientific works and studies on sensor topology discovery in buildings try to identify sensor structures using energy data. Topology detection for sensors in the buildings's hydraulic system has been moderately addressed so far.

Konrad Diwold et al. [DSZ15] claim that the knowledge about the exact placement of a certain sensor in a network is indispensable for applications operating on the specific sensor network. In their study, they investigate, which meter data can help to identify the underlying sensor network topology. The proposed methods are based on smart meter activity patterns and voltage measurements. Their approach can be used to assign smart meters to network branches without manual effort. They tested their solution on three distinct low voltage grids located in Upper Austria. They concluded, that given the activity pattern, their approach is able to establish meter groups regarding their grid affiliation.

Xiao Li et al. [LPS13] studied in their work the identifiablity of the grid topology and proposed an efficient algorithm to estimate the Laplacian matrix of the grid. For this purpose, they use only power injection data. In contrast to previously existing solutions on topology estimation which address detecting the changes in the topology, their approach focuses on leveraging on power injections at different buses. They concluded that their solution method is able to estimate the Laplacian matrix with high accuracy by using the sparsity of the grid as a constraint, which was also verified on IEEE test case data.

Romain Fontugne et al. [FOT+13] presented a methodology called Strip, Bind and Search (SBS) to uncover relationships between devices in order to detect misbehaving devices. Their approach consists of three steps: first they *strip* raw sensor data by decomposing them with Empirical Mode Decomposition (EMD) [HSL+98]. In the next step, they *bind* devices that are highly correlated, which is done by aggregating decomposed signals on different frequency levels. This technique enables to detect meaningful inter-device relationships, i.e. assign sensors that behave similarly, thus are located close to each other. In the final step *search*, their method monitors devices over time and detects deviations from the normal system behavior. Authors claim to have detected abnormal energy consumption by comparing heater and cooler, that caused a large energy waste.

1.5.2 Fault Detection and Diagnostics

Detecting faults in building's HVAC (Heating, Ventilation, Air Conditioning) system has been widely investigated in recent years. These studies mostly intent to detect component failures using different approaches. The long term goal of these analyses is to detect undiscovered optimization potentials of building's energy consumption and also to identify more complex errors such as aging of components or to proactively identify system failures.

Gerhard Zucker et al. [ZMH+14] claim that historic data analysis could be the key for quick recognition of building flaws and that the algorithms performing analyses have to focus primarily on automatic identification and classification especially for building states. They proposed an automatic detection of duty cycles in adsorption chillers using the X-Means algorithm introduced by the Auton Lab [PM00], which determines optimal clustering based on the computed values. They could successfully classify different types of chiller behavior. Although, in the study it was shown that it is possible to

automate system state detection for the domain, the study admits the need for manual pre-configuration of the system components before the analyses.

Imran Khan et al. [KCCC13] compared three different data mining techniques for detecting abnormal lighting energy consumption based on sensor measurements from an office building in Rome, Italy. Due to the lack of annotated data for erroneous system behavior and thus, in order to verify the reliability of their approach, they simulated two artificial faults in their data set. In the study, they applied different clustering algorithms: K-Means and DBSCAN. Additionally, they used classification and regression methods to analyze lighting energy and power consumption, and they also tested two different outlier detection methods. They concluded, that the generalized extreme studentized deviate (GESD) [Ros83] outliers's detection with the classification and regression tree algorithm was highly correct, i.e. it could detect the two artificial faults. The study has also shown that the K-Means approach was able to detect abnormal energy consumption and one of the artificial faults.

Stephen Frank et al. [FHJ+16] presented a hybrid, automated FDD approach to detect and diagnose faults non-invasively with a small number of sensors. The authors of the study claim that a combination of two FDD approaches may lead to a high accuracy fault detection engine. They combine energy modeling (model-based approach) and mathematical analytics (data-driven approach). The fault detection engines compares actual sensor measurements with expected performance using a statistical model based on historical measurements. Due to the lack of annotated erroneous sensor data, the authors created a pre-simulated database of modeled faults using the EnergyPlus energy simulation software [CPLW00]. For the fault detection phase, they tested two different regression and five different machine learning algorithms. After, the concepts and methods have been applied to baseline building, they also applied them on a real-world environment. Despite success in decreasing long training periods of pure data-driven approaches, and high accuracy in fault detection, their solution faces challenges like false positive results or performance for multiple buildings.

Balakrishnan Narayanaswamy et al. [BNGA14] designed and presented an algorithm, Model, Cluster and Compare (MCC) that was able to detect anomalies in an unsupervised manner. The MCC algorithm consists of three phases: based on sensor data establish zone-centric black-box models, apply clustering techniques to identify zone categories and finally compare zones to identify erroneous zones. They concluded that MCC can be a competitor to model-based and process history based anomaly detection methods by not requiring physical models or large amounts of historical data while improving both methods.

Mathieu Le Cam et al. [LCZD14] presented the practical use of information extraction from Building Automation Systems (BAS). In a case study, they show that the target variable, i.e. demand for chilled water depends on the supply air temperature and humidity in the air-handling unit. The data mining steps cover five techniques presented in another study [Kam09]. These are anomaly detection, association rule learning, cluster analysis, classification and regression. The case study showed that the designed techniques

can be applied not only to chilled water flow rate but also to any other energy-related indication of the building.

Dan Li et al. [LZHS16] proposed a data-driven FDD approach that can recognize faulty working conditions, identify their types, and additionally, it can also determine the severity of the system flaw. Their method is called Tree-structured Fault Dependence Kernel (TFDK). The algorithm aims to detect seven different faulty working conditions and also to recognize predefined severity levels of the identified faults. Unlike in traditional classification, their algorithm also focuses on the relationship between the different faults by assigning tree-structured labels to the faults. Their approach has been successfully applied to the buildings cooling system, and it is supposed to be also applied to other subsystems of the building.

Samuel R. West et al. [WGWW11] presented in their study a new FDD method based on HVAC operational fault modeling using statistical machine learning techniques. Their approach is able to model HVAC subsystems for normal and faulty behavior and detect deviation from the expected behavior patterns. They have also showcased their algorithm on an occupied commercial office building in Newcastle, Australia. They tested their method in the air-handling unit (AHU) using Hidden Markov Models (HMMs) [GKA19] that can simulate relationships between sensors. In order to detect faults, they trained a model to learn the sensor relationships from historical sensor data for normal operation.

Ying Guo et al. [GWLW13] proposed a model-based Fault Detection and Diagnosis approach using statistical machine learning techniques with data fusion methods. Their algorithm combines a Hidden Markov Model (HMM) based FDD method with data-based methods (i.e. clustering) in order to improve FDD accuracy. Based on different conditions, different models can be generated. Their methods include optimization techniques to avoid models that approaches local minimum. They concluded that their approach works well on air-handling units (AHUs) in the HVAC system.

1.6 Structure of the Work

The thesis is divided into seven chapters.

Chapter 1 covers motivation and essential introduction to the thesis' problem domain and additionally, it presents the methodological steps, how to approach the problem. In addition, it contains an extensive literature study about the problem domain covering topics about theoretical background and potential solution mechanism for sensor topology discovery and fault detection and diagnosis.

Chapter 2 describes the required use cases for the thesis and contains limitations and bounds of the problem space.

Chapter 3 introduces methods and concepts for sensor topology discovery and identifies relevant and essential solution methods to detect connections between distributed sensors based on meter data (flow temperature, return flow temperature and flow rate).

Chapter 4 covers solution approaches to implement a fault detection and diagnosis system.

Chapter 5 evaluates the identified concepts and methods using real-world data from a test building in addition to data preparation.

Chapter 6 contains the aspects and components of the testbed implementation as we integrate a part of the semi-automated fault detection system.

Chapter 7 sums up the results and limitations, and identifies potential future improvements and outlook of the thesis.

CHAPTER 2

Use Cases

The objective of the thesis is to establish a methodology for energy and resource efficient localization of optimization potentials in existing building systems in order to decrease primary energy demand of single building facilities and whole areas. This approach is not designed to replace existing, common building management systems but it should expand analysis possibilities of building facilities through monitoring relevant hydraulic system components independently from the used HVAC systems. The methodology was developed use case driven in order to identify all relevant mechanisms and components. The use case dependency graph considers three different use case categories of the given problem space which however form a coherent unit as depicted in Figure 2.1. The first part *Installation* covers concepts and methods to effectively deploy and maintain a sensor network on building's hydraulics in order to monitor building behaviour in a cost efficient manner regarding installation, maintenance and data provisioning. Even though the second and third component of the dependency graph deal with a similar section of the problem space, they operate on different levels of domain abstraction. While the second block *Operation of building facility* considers a single building facility as objective for optimization localization, the third block tries to draw conclusions about possible enhancements of whole areas regarding primary energy demand optimizations. Although designing the entire use case spectrum of the project is an essential part to understand the components and needs of the system, only selected elements of the use case graph will be processed and implemented in this thesis.

In the following sections, we will take a closer look at each dependency component, however the thesis only focuses on designing and implementing methods for *OBFa*. In addition to the use case dependency graph, each use case will be described with a unique identifier, dependencies, preconditions and postconditions. The identifier is constructed by concatenating the abbreviations of the nested use case components.

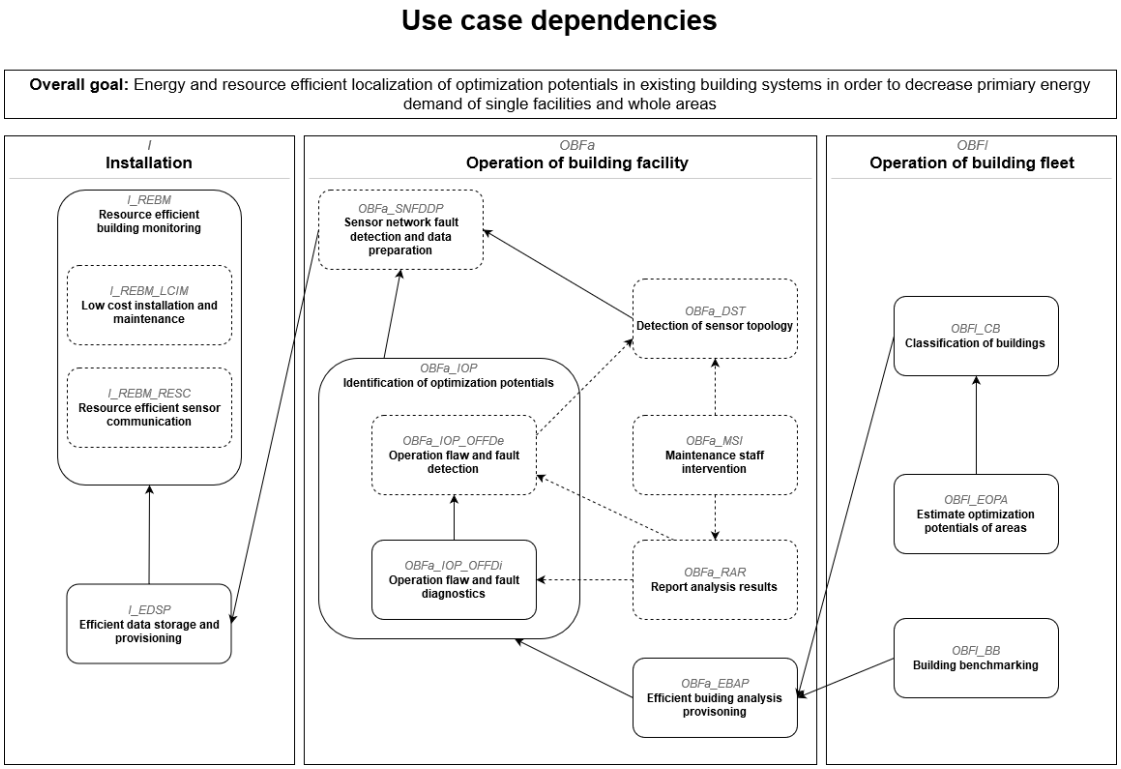


Figure 2.1: Overview of the use cases and their dependencies

2.1 Installation

This use case category covers concepts and methods to effectively deploy and maintain a sensor network on building's hydraulics in order to monitor building behaviour in a cost efficient manner regarding installation, maintenance and data provisioning.

2.1.1 Resource efficient building monitoring

Faults, flaws and inefficiencies in a building's hydraulic system are usually potential causes of energy waste and suboptimal operating conditions. Some modern industrial and commercial buildings are already equipped with numerous sensors that monitor building hydraulics [STD+18]. However, these sensor networks barely provide a high resolution of building structure. The goal of our approach is to develop a resource and cost effective methodology for monitoring building hydraulic components in high resolution and independently from existing HVAC systems as shown in Figure 2.2. Measuring flow temperature and rate is not trivial if we want to do it in a cost efficient manner. To achieve this goal, the building will be equipped with non-invasive sensors, hence there is no intrusion in pipes or expensive construction work needed. The design and implementation of such sensors is not part of this thesis and we assume them to be

existent for the sake of use case completeness. Since these measurement units are clip-on, non-intrusive sensors, they can be removed and reused for more analysis processes in other building facilities. To further minimize the engineering effort, sensors could use energy harvesting techniques and concepts to omit external energy demand for operation as shown in Figure 2.3. Wireless communication technologies will be applied in order to provide effective sensor communication, while a gateway will be used to serve as a single access point for data gathered by the sensors.

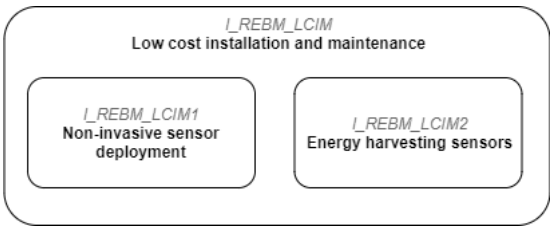


Figure 2.2: Use case: Low cost installation and maintenance

Non-invasive sensor deployment	
ID	I_REBM_LCIM1
Dependencies	-
Preconditions	• Existing or newly designed low-cost, clip-on temperature sensors are available • Existing or newly designed low-cost, clip-on ultrasonic flow sensors are available
Postconditions	The building to be analyzed is equipped with numerous temperature and flow measurement sensors. This is done by minor engineering effort and without intrusion into building's hydraulic system. Since these measurement units are clip-on, non-intrusive sensors, they can be removed and reused for more analysis processes in other building facilities.

Use of energy harvesting sensors	
ID	I_REBM_LCIM2
Dependencies	-
Preconditions	• Existing or newly designed energy harvesting sensors are available
Postconditions	The building facility is equipped with sensors which do not demand external energy source to operate to further lower maintenance expenses.

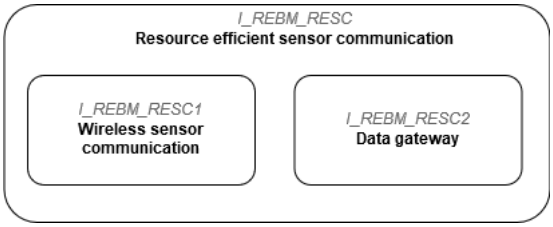


Figure 2.3: Use case: Resource efficient sensor communication

Wireless sensor communication	
ID	I_REBM_RESC1
Dependencies	-
Preconditions	• Existing or newly designed temperature and flow sensors capable of wireless data communication are available
Postconditions	Sensors monitoring building's hydraulic system communicate and transport data without being wired, which decreases installation costs and system error-proneness.

Data gateway	
ID	I_REBM_RESC2
Dependencies	-
Preconditions	-
Postconditions	The data gateway is developed to serve as a single access point for data gathered by newly deployed and existing sensors by integrating legacy protocols.

2.1.2 Efficient data storage and provisioning

An efficient approach to analyze building operation is to process data in a cloud environment, that provides the ease of scalability and accessibility. To provide this, the system will store and serve building data in a cloud environment which the other components of the system have access to. The collection of sensor data is not part of this thesis.

Efficient data storage and provisioning	
ID	I_EDSP
Dependencies	I_REBM
Preconditions	• Accessible cloud environment and resources
Postconditions	Data gathered by sensors of the building is stored in a cloud environment and serves for further analyses.

2.2 Operation of building facility

Our work focuses on this category of the use case spectrum, that considers a single building facility as objective for optimization localization. It contains use case groups which are necessary to analyze building facilities and detect possible flaws.

2.2.1 Sensor network fault detection and data preparation

Analyses could fail on faulty sensor data, hence it is necessary to detect faults of the underlying sensor network and label the data unreliable. To achieve these reliable and cleaned sensor time series, we will consider following steps as shown in Firgure 2.4: Data of all measurements will be resampled to have a unified resolution, while missing data will be interpolated from the nearest data points. Based on anomaly detection engines, anomalous data points will be identified. For shorter intervals or single data points, strategies of replacement will be applied. Faults of the underlying sensor network will be identified (e.g. network connection failure) and the detected intervals within the time series will be labeled. The identified unreliable data will be labeled and potential faults will be derived and recorded. The preprocessed sensor data will be equipped with additional parameters, e.g. weekday/weekend, holiday etc. in order to simplify analyses and to support complex detection methods.

Uniform sensor data resolution	
ID	OBFa_SNFDDP1
Dependencies	I_EDSP
Preconditions	-
Postconditions	All measurement data has unified recording resolution and the same time interval.

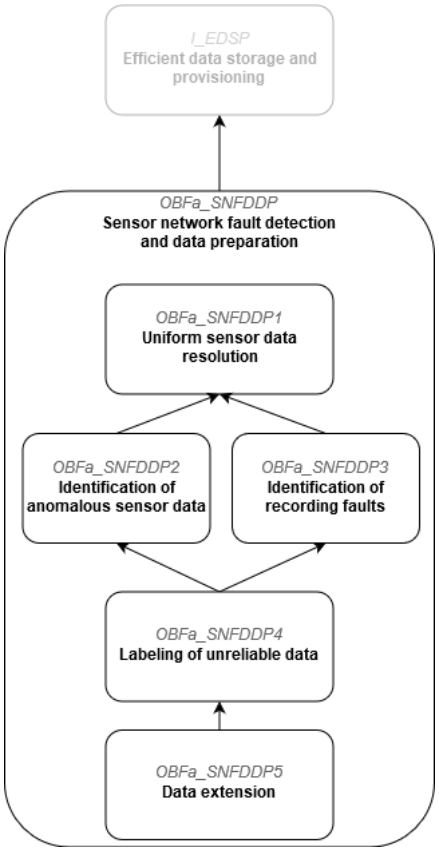


Figure 2.4: Use case: Sensor network fault detection and data preparation

Identification of anomalous sensor data	
ID	OBFa_SNFDDP2
Dependencies	I_EDSP, OBFa_SNFDDP1
Preconditions	-
Postconditions	Anomalous intervals or data points in measurement data are identified.

Identification of recording faults	
ID	OBFa_SNFDDP3
Dependencies	I_EDSP, OBFa_SNFDDP1
Preconditions	-
Postconditions	Intervals or data points with faults from underlying data recording system are identified within the measurement data.

Labeling unreliable data	
ID	OBFa_SNFDDP4
Dependencies	I_EDSP, OBFa_SNFDDP2, OBFa_SNFDDP3
Preconditions	-
Postconditions	Unreliable data points or intervals are labeled in the data set and recorded for further analysis.

Data extension	
ID	OBFa_SNFDDP5
Dependencies	I_EDSP, OBFa_SNFDDP4
Preconditions	-
Postconditions	Measured data is divided into daily splits and additional attributes (e.g. weekday/weekend, holiday etc.) are added to the data records to enables complex analyses.

2.2.2 Detection of sensor topology

The knowledge about the position of the sensors in the building is an indispensable part for the building analysis. Since this approach has to be able to be applied to any kind of building, the position of the sensors within the building is not known apriori. The manual creation of a sensor topology, i.e. the expert records the position of each sensor manually, requires a high amount of extra effort, which we want to avoid. Following steps will be applied to accomplish this need as shown in Figure 2.5: To minimize the engineering effort, measured data in the building hydraulics is recorded in an unstructured manner, i.e. the types of measurement data are not known for the individual time series. The first and initial step of detection of the underlying sensor topology is to identify sensor measurements regarding the type of measured data (i.e. flow temperature, return flow temperature and flow rate). Then, we group different sensors at same location of measurement (i.e. heating circuit). We establish clusters of sensor groups, which are close to each other in order to achieve a segmentation of building, which is followed by the identification of spatial closeness of building parts (i.e. rooms) and the correct sequence of rooms will be derived, ideally.

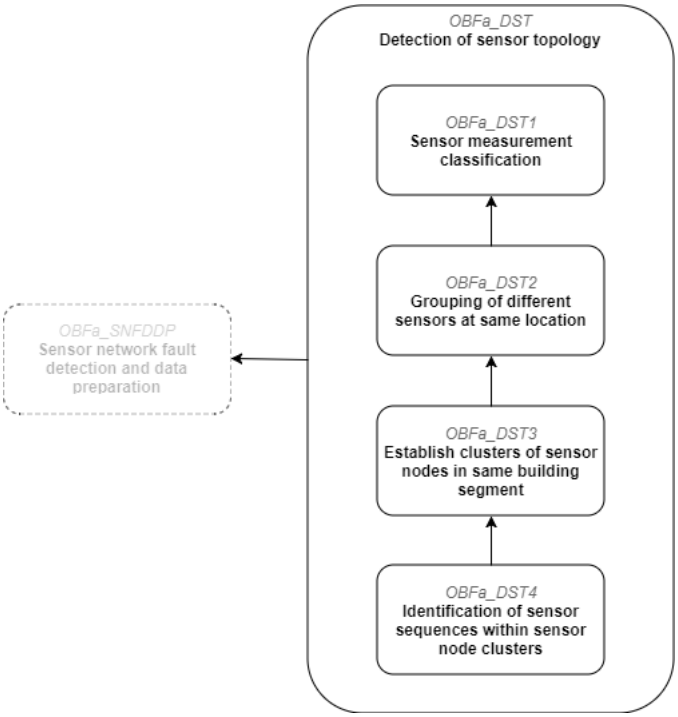


Figure 2.5: Use case: Detection of sensor topology

Sensor measurement classification	
ID	OBFa_DST1
Dependencies	OBFa_SNFDDP
Preconditions	• Number of different sensor types are known • Intervals of measurements with similar temperatures are present • All examined sensors are in use for recorded time interval
Postconditions	Time series are correctly labeled with corresponding measurement type.

Grouping of different sensors at same location	
ID	OBFa_DST2
Dependencies	OBFa_SNFDDP, OBFa_DST1
Preconditions	• All sensor types of a group are present
Postconditions	Established groups of different sensors at same building part (i.e. heating circuit)

Establish clusters of sensor nodes in same building segment	
ID	OBFa_DST3
Dependencies	OBFa_SNFDDP, OBFa_DST2
Preconditions	• Number of building segment clusters are known • Wide range of data is available
Postconditions	Established clusters of sensors in same building segment.

Identification of sensor sequences within sensor node clusters	
ID	OBFa_DST4
Dependencies	OBFa_SNFDDP, OBFa_DST3
Preconditions	-
Postconditions	Correct sequences of rooms within room clusters are identified.

2.2.3 Identification of optimization potentials

The core component of our approach is to identify defects and flaws in a building's hydraulic system. Recognizing such problems can lead to a more optimal and reduced energy consumption. Heat loss in parts of the building, defects in different parts of the hydraulic system, or falling efficiency of subsystems are only a few examples which have to be detected in order to ensure an optimal energy balance in the building. The detection process will be based on thresholds, underlying sensor network topology and on history data as shown in Figure 2.6. The second objective of this mechanism is the identification of the possible cause of a fault. Therefore, a simple rule based diagnostics engine will be implemented in order to identify the cause of the faults as shown in Figure 2.7. The rules will be defined by experts in a suitable definition language and can also be configured during run time. For example, an irregular rise and falling of the heat storage temperature (oscillation of flow temperature) indicates a faulty behavior of this building component.

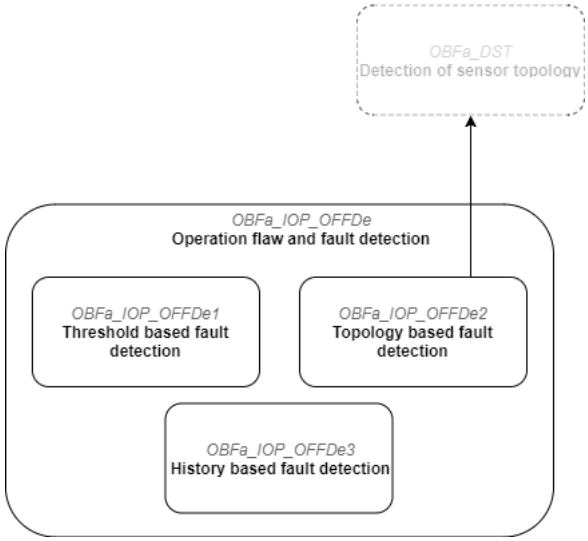


Figure 2.6: Use case: Operation flaw and fault detection

Threshold based fault detection	
ID	OBFa_IOP_OFFDe1
Dependencies	OBFa_SNFDDP
Preconditions	• Predefined thresholds are present • Exactly one of each sensor (flow temperature, return flow temperature, flow rate) is equipped on one hydraulic circuit
Postconditions	Faults, inefficiencies are detected based on thresholds and labeled in measurement data.

Topology based fault detection	
ID	OBFa_IOP_OFFDe2
Dependencies	OBFa_SNFDDP, OBFa_DST
Preconditions	• Exactly one of each sensor (flow temperature, return flow temperature, flow rate) is equipped on one hydraulic circuit
Postconditions	Faults, inefficiencies are detected based on topology (i.e. similarity, closeness) and labeled in measurement data.

History based fault detection	
ID	OBFa_IOP_OFFDe3
Dependencies	OBFa_SNFDDP
Preconditions	• History data of same sensors are available • Exactly one of each sensor (flow temperature, return flow temperature, flow rate) is equipped on one hydraulic circuit
Postconditions	Faults, inefficiencies are detected based on history analyses and labeled in measurement data.

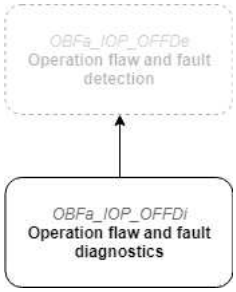


Figure 2.7: Use case: Operation flaw and fault diagnostics

Operation flaw and fault diagnostics	
ID	OBFa_IOP_OFFDi
Dependencies	OBFa_SNFDDP, OBFa_IOP_OFFDe
Preconditions	• Predefined rules are available
Postconditions	Causes of faults and inefficiencies are identified.

2.2.4 Report analysis results

Detecting faults alone is not sufficient to propagate newly gathered knowledge about flaws in the building back to the system. Thus, the Building Service Analysis Engine (BSAE) will provide features to notify building facility managers or experts about the building's misbehavior as shown in Figure 2.8. Analysis reports will regularly be generated automatically to keep facility manager informed about building behavior and detected faults. If diagnostics of a specific fault is possible, the maintenance personal will be notified and so they are able to perform an adequate action in the building system. The implementation of the reporting system is out of the thesis' scope.

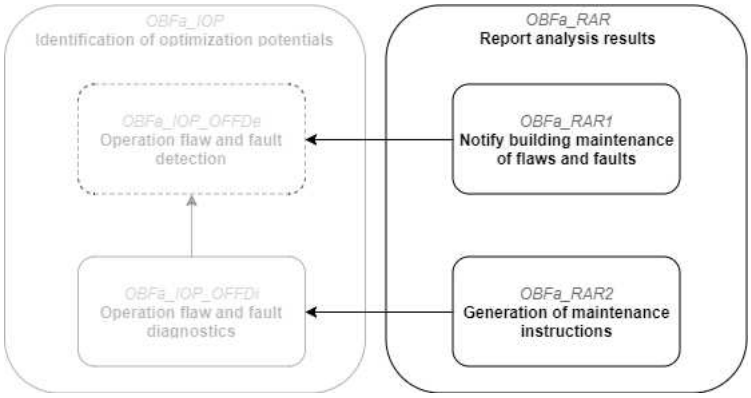


Figure 2.8: Use case: Report analysis results

Notify building maintenance of flaws and faults	
ID	OBFa_RAR1
Dependencies	OBFa_IOP_OFFDe
Preconditions	-
Postconditions	Building's facility manager is notified of building analytics results and is aware of flaws and inefficiencies of building's hydraulics.

Generation of maintenance instructions	
ID	OBFa_RAR2
Dependencies	OBFa_IOP_OFFDi
Preconditions	-
Postconditions	Building's facility manager gets maintenance instructions based on building analysis results and diagnostics process.

2.2.5 Maintenance staff intervention

If flaws and faults were detected and the facility manager is informed about these problems, building experts are able to react and act accordingly. The system defines three potential intervention actions to be carried out by experts: *adaption of building system parameters by experts*, *manual improvement of discovered topology*, *adjustment of analysis components* as shown in Figure 2.9. These actions operate on different application levels. It should be possible to manually improve discovered sensor network topology, and experts should also be allowed to adjust building analysis components. Furthermore, maintenance staff should adapt the building level parameters based on the detected optimization results. The implementation of the parameter configuration system is not part of this thesis.

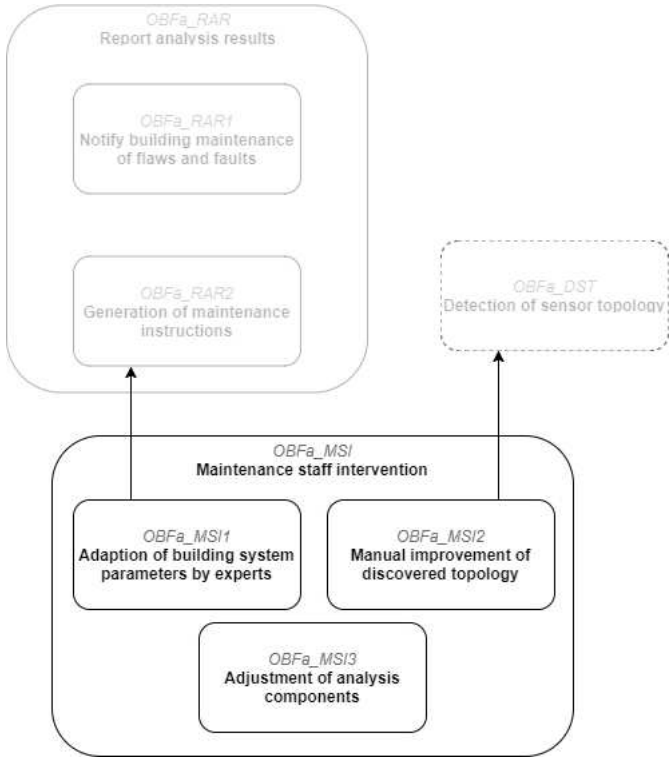


Figure 2.9: Use case: Maintenance staff intervention

Adaption of building system parameters by experts	
ID	OBFa_MSI1
Dependencies	OBFa_RAR
Preconditions	• Expert knowledge is available • Access point to building's management system exists
Postconditions	Building system parameters are optimized, energy demand of building is reduced, building operates on a more optimal level.

Manual improvement of discovered topology	
ID	OBFa_MSI2
Dependencies	OBFa_DST
Preconditions	• Knowledge about building structure
Postconditions	Building sensor topology is improved and has a more fine-graded structure.

Adjustment of analysis components	
ID	OBFa_MSI3
Dependencies	-
Preconditions	-
Postconditions	Types of analysis methods are defined and analysis process is adjusted to accomplish needs of experts.

2.2.6 Efficient building analysis provisioning

To efficiently analyze building operation, data should be processed in a cloud environment that provide the ease of scalability and accessibility. This approach will also be used to serve history analyses to make analyses possible on a higher domain abstraction level, building fleet.

Efficient building analysis provisioning	
ID	OBFa_EBAP
Dependencies	OBFa_IOP
Preconditions	-
Postconditions	Data analysis objects are stored in a cloud environment and served to enable complex building fleet analyses.

2.3 Operation of building fleet

The third and last application level of our building analysis approach is the localization of optimization potentials for whole areas. Building classification and benchmarking, or rather estimation of optimization potentials of areas are some examples that carry high potential to reduce energy demand of building fleets. Following tables show details on use cases for optimization of building fleet operation. The localization of optimization potentials of building fleets is not in the scope of this thesis.

Classification of buildings	
ID	OBFl_CB
Dependencies	OBFa_EBAP
Preconditions	-
Postconditions	Correct classification of buildings within an area.

Estimate optimization potentials of areas	
ID	OBFl_EOPA
Dependencies	OBFl_CB
Preconditions	-
Postconditions	Optimization estimation for city areas are derived based on building classification.

Building benchmarking	
ID	OBFl_BB
Dependencies	OBFa_EBAP
Preconditions	-
Postconditions	Comparison of buildings regarding optimization potentials based on previous building facility analysis.

CHAPTER 3

Non-invasive Sensor Topology Discovery

The knowledge about the spatial arrangement of the sensors in the building is important to locate detected faults and inefficiencies of the building hydraulics. Manual provision of such a plan is time-consuming and difficult to maintain because all deployed sensors with their identification numbers had to be digitized and assigned to the building plans. The building plans are also often not available digitally and the assignment of a large amount of sensors to these plans would require enormous engineering effort.

A sub-goal of this thesis is to detect the relative position of the deployed measuring points with as little effort as possible. In case of successful detection of the positioning of the sensors relative to each other, reference sensors can be used to apply this arrangement to the specific floor plan. Finding out the location of the sensors with little engineering effort is not trivial. We want to develop a methodology that identifies the necessary steps to detect the sensor topology, and we also want to identify any additional conditions and limitations.

The optimal result would be the correct identification of measuring point dependencies only based on measurement data. Some preconditions are already available in the planning phase: the engineering effort cannot exceed an optimal limit and since we are working on the hydraulic system domain, measurements of flow and return flow temperature and flow rate are available. It may also be possible to use operational data for topology detection, but since these data are not always available in the same form, we will primarily limit ourselves to the three sensor types mentioned before.

3.1 Classification of sensor measurements

To minimize the engineering effort, measured data in the building hydraulics is recorded in an unstructured manner, i.e. the types of measurement data are not known apriori for the individual time series. The first and initial step of detection of the underlying sensor topology is to identify the type of measured data (i.e. flow temperature, return flow temperature or flow rate) for each incoming data stream. Machine Learning offers different approaches to group these time series: supervised and unsupervised learning [SA13]. For our problem, we will apply a candidate for each approach.

In the supervised learning approach, a part of the data set is annotated for training of a model and then, based on this trained model, the rest of the data will be correctly labeled. Supervised learning can be divided into two main sub-areas [Kot07]. While *regression* tries to predict a continuous variable, a *classification* problem selects the output variable from a predefined set of objects. As obvious from the definitions, our domain can be categorized as a classification problem, i.e. a measurement type from a predefined set of objects must be assigned to each time series. In order to handle the problem based on this approach, some of the time series have to be annotated with the corresponding sensor types. For the learning phase, we will use the K-Nearest Neighbors (K-NN) algorithm [ZLZ+17]. K-NN is a straight forward classification algorithm which classifies the input data based on the k *closest* training examples. This algorithm is a lazy learner, i.e. it does not build a model in front, thereby no training time is needed. The downside of such an approach is that - in case of a large amount of training data - the algorithm may become computationally infeasible regarding run time and memory usage. The selection of various parameters for a successful classification is crucial. To minimize the manual effort, only the smallest possible group of sensors should be annotated. Furthermore, the selection of k and the distance metric are relevant aspects to consider. A necessary requirement for the selection of k is, that it must be odd in order to be able to make the decision in the majority voting unambiguously. On one hand, if k is chosen too small, relevant information may be lost, which may have a negative impact on the results. On the other hand, if k is chosen too large, the algorithm can become very inefficient, which could end up in bad run times and results. As mentioned above, the selection of the distance metric is also crucial. Since our problem space consists of time series, Euclidean distance calculation seems to be effective. To account temporal shifts of patterns in time series, Dynamic Time Warping could also be used as a distance metric (see details for DTW in Section 3.4.2).

Unsupervised learning is a type of machine learning for problems where no labeled data is available [Lia05]. Unsupervised learning techniques try to recognize structures in the data set by grouping objects into groups (i.e. clusters). Another application area of unsupervised learning is anomaly detection. For our sensor measurement classification problem, we will apply a clustering approach. Clustering methods can be grouped in different ways [MC87]. While in partitional clustering data objects are subdivided into non-overlapping subsets and can be assigned exactly to one cluster, a hierarchical clustering consists of a set of nested clusters as a hierarchical tree. Exclusive clustering assigns data

points to a single cluster, while non-exclusive clustering may map data objects to multiple clusters. For our problem, we take a closer look at K-Means Clustering [KMN+02]. K-Means Clustering is a partitional clustering approach, where a central point (centroid) is assigned to each cluster, followed by an assignment of all data points to the nearest cluster, which is based on feature similarity. The number of clusters must be specified in advance. The selection of centroids generally happens randomly and therefore, the composition of clusters may differ from iteration to iteration. As can be seen from the properties mentioned above, the algorithm must be configured with different parameters. The number of iterations or the termination criterion are just a few of the parameters that must be chosen efficiently to yield appropriate result. As well as for the K-NN classification, the selection of the distance metric plays an important role in K-Means Clustering. Mostly Euclidean distance is applied, which can be fine tuned by DTW, while correlation can also be used as measure of similarity.

The evaluation of the approaches can be done using different metrics [Kot07]. For classification accuracy (acc), precision (prec), recall (rec) and f1-score (f1) will be applied.

$$acc = \frac{TP + TN}{\#samples} \tag{3.1}$$

$$prec = \frac{TP}{TP + FP} \tag{3.2}$$

$$rec = \frac{TP}{TP + FN} \tag{3.3}$$

$$f1 = \frac{2 * (prec * rec)}{prec + rec} \tag{3.4}$$

Where TP: true positives, TN: true negatives, FP: false positives, FN: false negatives. The most suitable evaluation metrics will mostly be decided based on the considered problem. For instance, if problem considers a classification of health data, which decides whether a patient has cancer or not, the focus will probably be on avoidance of false-negative cases (patients classified as healthy but have actually cancer), i.e. minimizing recall. In our case, we need a high, even perfect classification of sensor measurement type because in case of incorrect classification of the sensors, the analyses will result in wrong conclusions. For example, if a flow temperature sensor is classified as flow rate, the analysis engine will yield completely incorrect results. In addition to accuracy, Sum of Squared Errors (SSE) is used to evaluate the clustering approach.

In order to accomplish reasonable outputs, some domain-specific assumptions have to be made and some additional constraints must be defined. In our case, the flow and return flow temperatures barely differ, i.e. they take values from ranges near to each other. Therefore, in the classification process, we will consider only data points, where flow rate

takes positive values. This ensures that the flow temperature for all data points is greater than the return temperature (in the case of heating). This constraint ensures that the clusters are distinct so there will be no overlaps which could lead to misclassification for edge cases. Another important question to investigate and answer is how large intervals from the time series are required for classification. Feeding the algorithm with long time intervals, the algorithm may become computationally infeasible (e.g. DTW calculation) which results in a useless approach.

As we can see from the details mentioned above, heating components have to be in use (switched on) to provide relevant data for training purposes, since unused heating circuits may be grouped incorrectly. In such a case, flow and return flow temperatures absorb the room temperature and therefore, their behaviour may not reflect the behaviour of components located near to them. In order to get rid of this problem, a so-called *footprint* phase will be applied in the building. In this initial phase, all heating circuits are heated to the same temperature. After a short period of time all flow temperatures take values from a relatively similar value range. Therefore, possible classification errors could be avoided by applying this methodology. Details for *Footprint Method* an be found in Section 5.1.3.

3.2 Establishment of sensor groups at same location

While designing the fault detection system, it became clear that another precondition for the system has to be defined. The analysis of the building behaviour requires the presence of measuring points, i.e. on an examined hydraulic circuit all three sensor types (flow temperature, return flow temperature and flow rate) must be present exactly once. These sensors should be grouped to measurement points as depicted in Figure 3.1. For example, to calculate the energy output of a hydraulic circuit, it is necessary to examine the data points where the circuit is heated, and to include flow and return flow temperatures for the calculation.

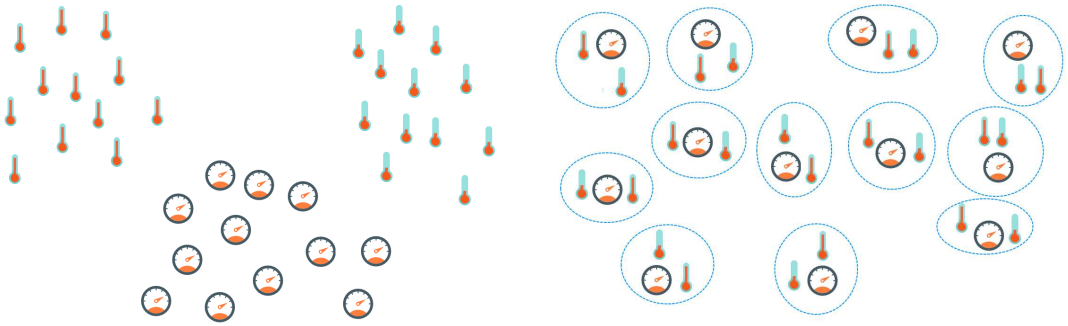


Figure 3.1: Idea of grouping sensors at same measurement point

Even if this requirement is met, the data is recorded in an unstructured way, i.e. we do

not know which time series belong to which measuring points. In order to be able to form sensor groups consisting of a flow, return flow temperature sensor and a flow rate sensor, we use correlation analysis of all sensors. We assume that sensors at a measuring point have a strong influence on the behaviour of the others and that there is a strong relationship between the states of these sensors. If, for example, a hydraulic circuit is heated, the flow rate increases and at the same time the flow temperature increases as well. After a short delay in time, the return flow temperature on this hydraulic heating circuit will also increase. If it is no longer heated, then there is no more flow rate and the temperature measurements begin to adapt to the room temperature promptly.

In order to verify the assumption that there is a strong relationship between the behavior of the sensors at a measuring point, we will develop a naive algorithm that performs the mapping of each sensor type based on correlation calculation. As a measure of the degree of linear correlation between two time series, we use the correlation coefficient (also called Pearson correlation). This can only be used for interval scaled characteristics, but it is the case for sensor measurements. The Pearson correlation coefficient is calculated as follows:

$$\rho = \frac{\mathrm{cov}(X, Y)}{\sigma_x \sigma_y}, \tag{3.5}$$

where cov is the covariance and σ is the standard deviation. The coefficient can take values between -1 and 1, where 1 is positive correlation, -1 is negative correlation, and 0 represents no correlation. With a negative correlation (value close to -1), the values between two time series will be the same but linearly reversed. A good example is when the number of products produced increases for the same demand, the number of products negatively correlates to the price of the product i.e. the number is increasing while the price is decreasing. In our domain, positive correlations are relevant. If the flow temperature is higher, then the return flow temperature on the same hydraulic circuit is also increased.

Our method uses a smaller time interval (i.e. one week) for this algorithm. When selecting this interval, it is important to note that the heating system must be in operation within the investigated data points. When the heating system is switched off, the measured temperature values adapt to the room temperature. This could end in an information loss about the behaviour of the heating system. This criterion can be reached either manually or automatically. If we use time series for the calculation where the building was heated for sure, then the condition is reached immediately. But if we want to further reduce the engineering effort, filtering data points for the calculation, where flow rate is present, i.e. the heating system is in operation can be done. A possible error in this procedure occurs if the heating system is also used for cooling during the summer months. Cooling of buildings is often realized by cold water circulation in the heating system. This means that the flow rate measurement provides positive values as the system is in operation, but in this case there is a negative correlation between the sensor types at a

measuring point. In other words, when cooling is in progress, the flow rates increase, but at the same time the temperatures decreases. For these reasons, it is necessary to select the time series intervals in such a way that the hydraulic system is either in heating or cooling phase, exclusively.

The algorithm 3.1 works as follows: the correlation matrix between two arbitrary selected sensor types is calculated for all sensors. The largest value (correlation) in the matrix is selected and so the first group of sensors is established. Iteratively, the other correlation coefficients are processed as in the step before. If the coefficient describing the dependency between sensors are already assigned to a group, then this case is ignored and the next largest coefficient is selected. These steps will be repeated until all sensor pairs have been formed. As the next step, the sensors of the third sensor type are correlated with the measured values of one of the already assigned sensor types and the resulting matrix is processed iteratively as described before. As a result, we expect correctly assembled sensor groups containing the three sensor types at one measuring point.

Algorithm 3.1: Sensor Grouping Algorithm

Data: time series of predefined length
Result: sensor group assignments
`// Create flow temperature (TFI) - flow rate (PrVlm) assignments`
tfi_prvlm_correlation_matrix ← [];
for *tfi in tfi_data* **do**
  for *prvlm in prvlm_data* **do**
    calculate pearson correlation of tfi and *prvlm*;
    append result to *tfi_prvlm_correlation_matrix*;
  end
end
sort *tfi_prvlm_correlation_matrix* by correlation value;
tfi_prvlm_assignments ← {};
for *value in tfi_prvlm_correlation_matrix* **do**
  if *tfi_index not in assignments AND prvlm_index not assigned* **then**
    $tfi\_prvlm\_assignments[tfi\_index]$ ← *prvlm_index*;
  end
end
`// Create flow temperature (TFI) - return flow temperature (TRt) assignments`
tfi_trt_correlation_matrix ← [];
for *tfi in tfi_data* **do**
  for *trt in trt_data* **do**
    calculate pearson correlation of tfi and *trt*;
    append result to *tfi_trt_correlation_matrix*;
  end
end
sort *tfi_trt_correlation_matrix* by correlation value;
tfi_trt_assignments ← {};
for *value in tfi_trt_correlation_matrix* **do**
  if *tfi_index not in assignments AND trt_index not assigned* **then**
    $tfi\_trt\_assignments[tfi\_index]$ ← *trt_index*;
  end
end
combine *tfi_prvlm_assignments* and *tfi_trt_assignments*;

3.3 Establishment of clusters of sensor nodes in same building segment

While examining the sensor sequences, we noticed that the analysis of the direct connections of all sensors in the building is inefficient and inaccurate. The applied concepts and methods for the identification of the arrangement of sensors (see Section 3.4) do not provide adequate results at the building level, i.e. these methods are suitable to sequence rooms within a circle correctly, but are unsuitable to sequence all rooms and sensors in one step. The reason for this is that most buildings are heated by several hydraulic circuits, which operate independently from each other. For example, if there are two heating circuits in the heating system, they will be heated at the same time if necessary, and therefore heating components will change their state similarly at the same time, but this does not say anything about the physical proximity of the components. Therefore, we will form clusters of sensors that may be located close to each other, i.e. in the same building segment, before examining the concrete arrangement of heating components. Based on the correlation analysis of rooms, conclusions can be drawn about the relationships between rooms. The assumption is that rooms are closer to each other because they are in similar states (e.g. outside temperature), have similar functionality, and have similar usage patterns. Figure 3.2 shows the idea of sensor group clustering on the building level.

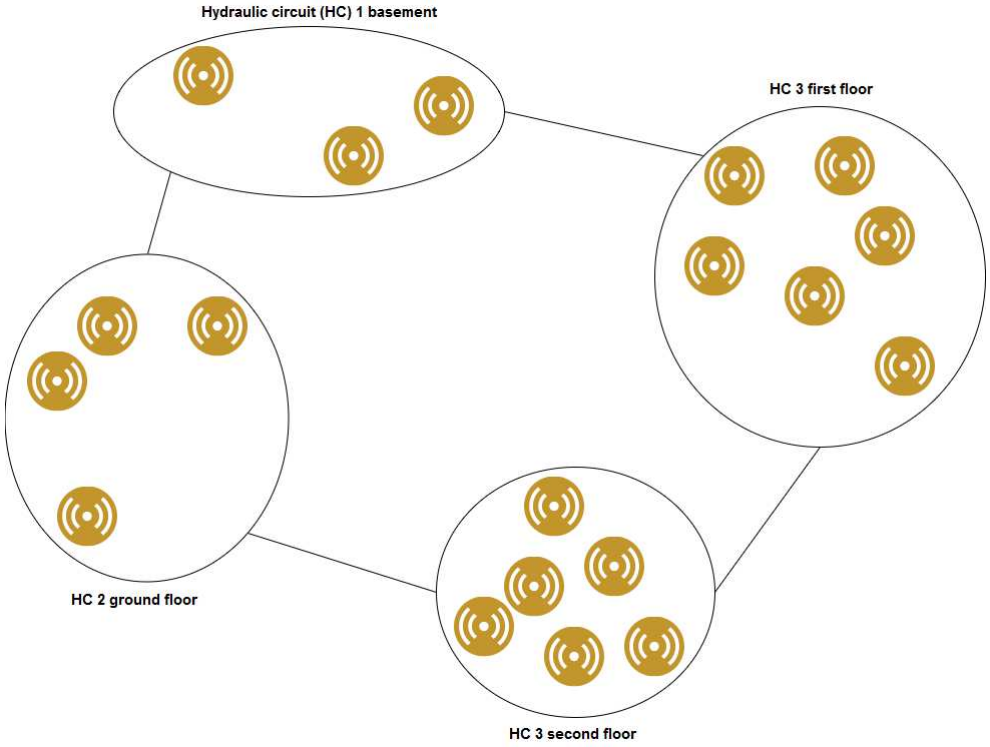


Figure 3.2: Idea of room clusters of different building parts

To investigate the correlation of longer time series, it is not sufficient to use the raw data as input [FOT+13]. Longer time series may contain periodic components that behave as non-stationary trends. In such a case, no reasonable statement can be made about the relationship of devices. These trends must be removed if they exist.

Empirical Mode Decomposition (EMD) offers a practical approach to this problem. EMD tries to decompose the signal into different frequency ranges [HSL+98]. These are also called Intrinsic Mode Functions (IMF) and each of these functions describes the signal on a different frequency range. EMD is an iterative process and is performed as follows: [1]

Algorithm 3.2: Empiric Mode Decomposition Algorithm

```
Data: time series as TS
Result: decomposed time series
while TS has IMFs do
    signal ← TS;
    while signal is not an IMF do
        upper_bound ← connect(local_maxima);
        lower_bound ← connect(local_minima);
        signal ← signal − mean(upper_bound, lower_bound);
    end
    remove signal from TS
end
```

We will apply a methodology proposed by Fontugne et al. [FOT+13], as the derived IMFs are aggregated into three frequency groups and the signal is examined in these three aggregated frequency ranges. The correlation analysis of the sensor measurements is only performed on these three frequency ranges. The assumption is that by analyzing the signal at different frequencies we can derive more adequate inter-device relationships, and by removing the trends we can determine a better metric for spatial proximity for the building sensors. This metric will not be used for direct spatial relationships, but will be used as a metric for cluster building. Based on this distance calculation, we form groups of sensors that hopefully are close to each other. Clustering of sensors to building parts is done using an appropriate clustering algorithm. Unlike for sensor type clustering in Section 3.1 where we know the expected number of clusters, in this case we cannot define the number of clusters prior to the grouping. Thus, we cannot apply K-Means Clustering. DBSCAN is an unsupervised clustering algorithm that relies on density-based notion of clusters [EKSX96]. The algorithm can detect arbitrary shapes of clusters and does not require to define the number of clusters in advance while it accepts precomputed distance metrices for distance calculations. The algorithm requires two parameters to be set: the *eps* parameter is the maximum distance between two entity to be seen as neighbors of each other, while *minPts* is the minimum number of samples that can build a cluster. It groups observations with many neighbors (high-density spaces) to one group, while

[1] https://pyhht.readthedocs.io/en/latest/tutorials/hilbert_view_nonlinearity.html

observation in a low-density spaces will be marked as outliers. We will use Scikit's [2] DBSCAN implementation [PVG+12]. Ideally, we will form sensor groups of rooms that are near to each other, i.e. they are located on the same hydraulic circuit.

3.4 Identification of sensor sequences within sensor node clusters

The final step of our topology discovery process is to identify connections between heating system components within clusters of sensor nodes. We will achieve this by the means of hierarchical clustering. To overcome the problem of shifts of heating events in time, we will apply Dynamic Time Warping.

3.4.1 Hierarchical Clustering

Hierarchical clustering is a method to build hierarchies of clusters by merging or dividing clusters to achieve a sequence of clusters and elements [Pop14]. We will apply an approach proposed by Saute et al. [STD+18]. The idea of this approach is to establish connections based on distance measurements of time series and in our case to create a map of components of the hydraulic system within the building. There are two types of hierarchical clustering algorithms: agglomerative and divisive. While agglomerative clustering starts with a cluster for each object and tries to merge clusters near to each other, divisive hierarchical clustering algorithm follows a top-down approach by starting with a single cluster of the entire problem space. It splits up clusters in order to achieve an optimal clustering by means of the defined distance metric [Pop14]. Ideally, data series with a low distance will be clustered in close clusters, while higher level clusters will form the clusters for rooms and floors. The natural way to represent the clusters is a dendrogram which is a visual representation of the clusters and could reflect the topology of the underlying system as depicted in Figure 3.3.

[2] https://scikit-learn.org/stable/index.html

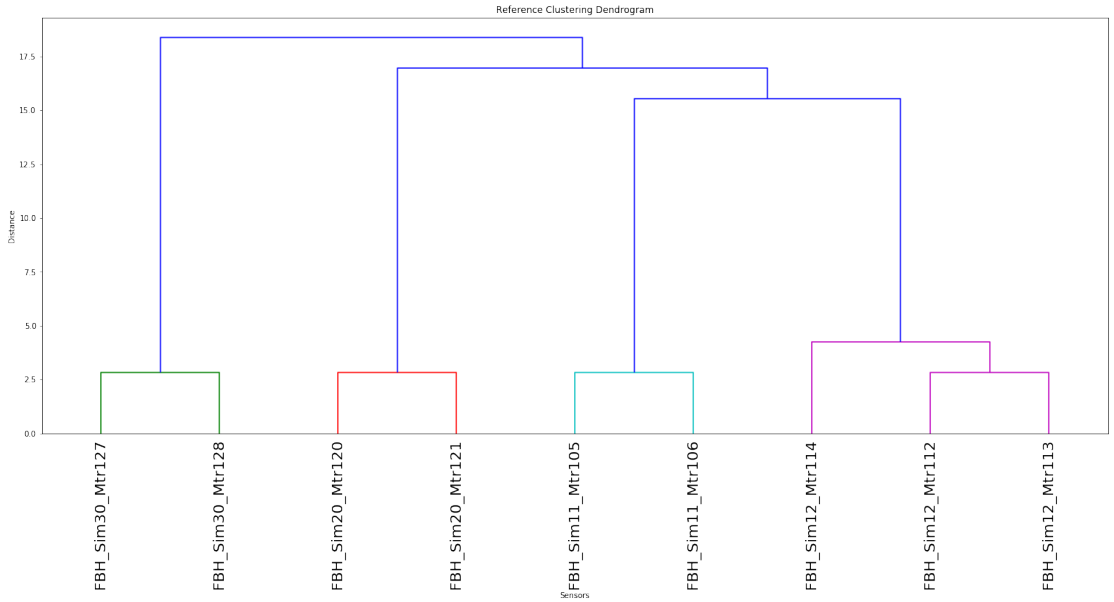


Figure 3.3: Example of a dendrogram

We will use SciPy's hierarchical clustering algorithm [3]. This Python implementation is an agglomerative hierarchical clustering, i.e. it is based on a bottom-up approach to create the hierarchy levels. There are three main parameters to configure the clustering process. *Linkage method* is used to compute the distance between two clusters. The option *single* implements a minimum spanning tree based algorithm, while *complete*, *average*, *weighted* and *ward* use a nearest-neighbors chain algorithm. For distance measurement between data series, the function allows us to use different metrics: Euclidean distance, Manhattan distance, correlation between time series, or even user defined distance metrics. Another important factor of hierarchical clustering is the selection of an appropriate time series length for examination. Processing too long or too short time series intervals in the algorithm may have a negative impact on the outcome of the process. Due to the high number of parameter combinations and in order to evaluate a great number of configuration sets we will brute force a predefined set of configurations against a reference clustering of the time series. This means, that we will create a reference clustering containing the optimal solution for the given problem instance and evaluate the output of the clustering algorithm against this reference clustering. This allows us to measure the performance of different configurations (linkage method, distance metric and times series interval).

[3] https://docs.scipy.org/doc/scipy/reference/cluster.hierarchy.html

3.4.2 Dynamic Time Warping

While applying hierarchical clustering on our real-world problem domain, we encountered several problems. In order to detect sequences of hydraulic components, they have to be heated sequentially. That means, that the hydraulic setup of the building plays an important role in the success of the topology discovery process. We expect older buildings to have line-like heating systems established, where rooms and hydraulic components are heated sequentially in the heating phase. More recent buildings often have a star-like topology for their heating system where each room is individually connected to the management system, therefore using hierarchical clustering will fail due to independent management of the temperatures in the rooms [STD+18]. This means, that we cannot derive meaningful information about spatial arrangement of rooms considering time shifts in temperature measurement changes. To overcome this issue, we will use the Footprint Method, see Section 5.1.3.

Hierarchical clustering can fail due to the slight time gradient of heating curves. We expect hydraulic components to be found near each other (i.e. assigned close clusters) even if the measurements do not correlate directly due time shifts (e.g. the heat reaches components that are further from the beginning later). The system must take this time shifts into account to create a more optimized topology layout. To boost up the clustering of sensors based on their pairwise correlation, Dynamic Time Warping (DTW) will be applied to find matching shapes among different time series that are shifted in time as introduced by Sauter et al. [STD+18]. DTW tries to find the optimal alignment of two time series. The sequences are mapped in a non-linear manner to match to each other fulfilling some rules, like the first and last index of the first time series have to be the first and last index of the second time series [dtw07]. DTW is well established in speech recognition, shape matching etc. but it has been also applied to other types of time series [BC94]. We will use DTW as distance metric for the hierarchical clustering to achieve more accurate results. This method works fine on data gathered from the Footprint Method, see Section 5.1.3, where the heaters are turned on in a sequential order. However, this technique only works on heaters within one hydraulic circuit.

CHAPTER 4

Building Service Analysis Engine

Heat loss in building segments, defects in different parts of the hydraulic system or a falling efficiency of a subsystem are some examples which have been identified as potential cause of energy waste in buildings. A quick detection and consequently timely fix of such problems can lead to a more optimal and reduced energy consumption. We identified following flaws and defects in building's hydraulics: Flow temperature oscillations, abnormal operation time of hydraulic circles, abnormal heat emission of hydraulic circles, extreme deviation from desired temperature.

4.1 Fault Detection and Diagnostics

The objective of an FDD system is an early detection of system and component failures and the diagnostics of the detected flaws. Figure 4.1 represents the generic application of an FDD system.

The first and initial step of FDD is to monitor the system in order to be able to detect problems in the system. If a flaw is detected, the diagnostics engine is responsible for determining the cause of this misbehavior. In the generic application, the following step is to evaluate the detected fault regarding the severity of the problem, which is followed by operational decisions. Another feature of such an FDD is prognostics which intends to help for maintenance decision making. Predictive maintenance identifies potential system and component failures based on history data. It tries to predict the time before failure of a component.

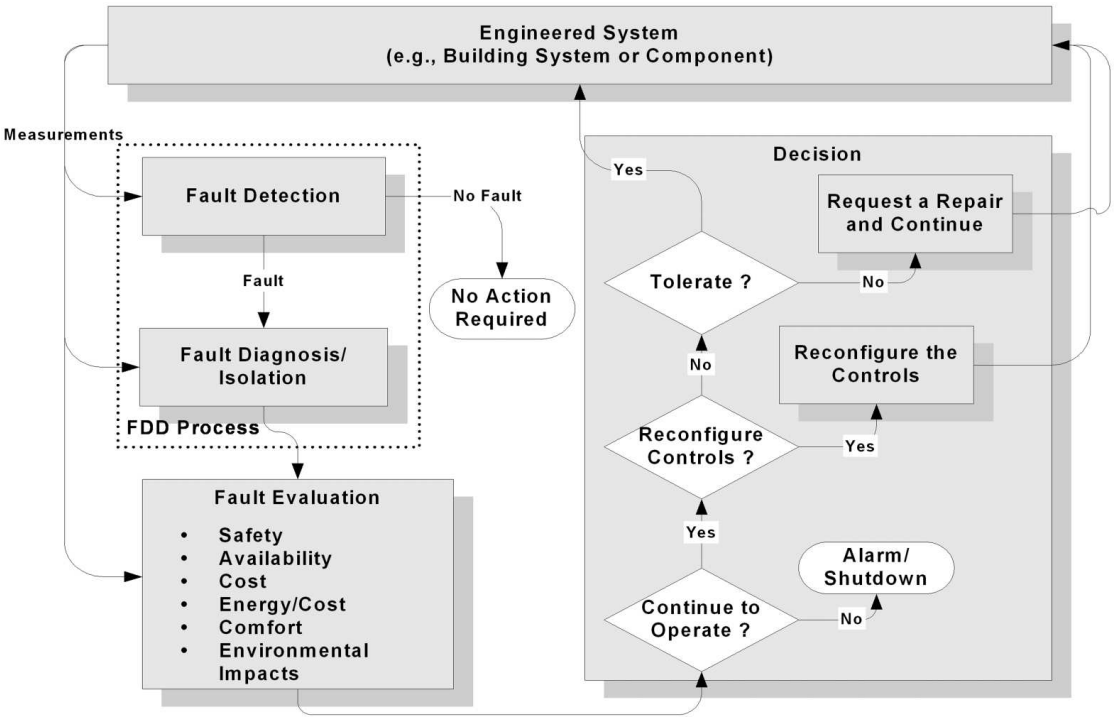


Figure 4.1: Generic application of FDD [KB05]

There are different ways to group FDD methods and concepts:

- *Quantitative Model-Based, Qualitative Model-Based, Process History Based [KB05]:* Model-based approaches use prior knowledge to build a model that can be used as basis for identifying discrepancies between expected and measured values. In contrast, data-driven approaches derive models from history data, so no prior knowledge of the system is needed. Quantitative and qualitative models use knowledge of the underlying system's physics. The results from these analyses will be used to draw consequences about certain problems in the system.
- *Rule-based, Model-based, Data-driven [FHJ+16]:* Rule-based FDD compares the measurements with a predefined set of rules. Violating any rules triggers an alarm. This method is time consuming and requires a significant amount of expert knowledge. The definition of the rule set is not trivial and can be hardy applied to any kind of buildings. Model-based FDDs require a model for deriving information about flaws of the system. This model is used as basis for detection of discrepancies between predictions and measured system conditions. This concept requires knowledge of the examined system; therefore, it cannot be applied to any kind of buildings. In addition, the results of such an approach depends heavily on the quality of the model. However, using this approach we can achieve a high accuracy in failure classification. Data-driven methods do not require any models

and only minor knowledge of the underlying system. They mathematically relate measured inputs to measured outputs. Significant discrepancies between inputs and outputs or between sensors measurements indicate a flaw of a system component. The quality of such a method depends on the quality and quantity of the data and cannot detect flaws that are already present in the data set.

There are further ways to classify FDD methods, but we will limit ourselves to the presented ones. After analyzing the problem statement and the requirements, we decided to combine rule-based and data-driven approaches to detect flaws in the system and diagnose their cause.

4.2 Fault Detection for BSAE

Fault detection aims to find defects and flaws in the examined data set based of analysis results, thresholds, sensor topology and history analyses. Due to restricted sensor data (flow temperature, return flow temperature and flow rate) availability and the lack of expert knowledge, we implemented naive algorithms to detect flaws in the building's hydraulic system.

4.2.1 Oscillation Detection

Oscillations with high frequency and amplitude of temperature signals may indicate erroneous behavior of hydraulic system components. A possible reason for such a wrong behavior is the simultaneous request of hot and cold water. In order to design and implement an oscillation detector that is capable of detecting oscillations with different shapes (frequency and amplitude), we will implement a naive, difference-based algorithm.

The algorithm calculates the first derivative of the signal. Thereafter, a rolling window analysis is performed on this detected signal. Based on the configuration of the algorithm, data points will be filtered, where a window contains the corresponding number of data points with the configured amplitude. The determined data points will be annotated in the original data set and passed on to the visualization component. The visual representation of the selected set forms an entry point for the facility manager to locate the abnormal component behavior.

Algorithm 4.1: Oscillation Detection Algorithm

Data: data, frequency and amplitude
Result: data set marked with oscillation flag

```
1  add column for oscillation;
2  detect data time resolution;
3  while not at end of this data set do
4  |  calculate derivative of data;
5  |  count sign changes;
6  |  move rolling window;
7  end
8  for i ← 0 to length of oscillations do
9  |  if oscillations > threshold then
10 |  |  mark data as faulty;
11 |  end
12 end
```

4.2.2 Target Temperature Analysis

The target temperature for a hydraulic component is the value set by the hydraulic system that the water temperature should ideally approach. A deviation from the target temperature may also mean an inefficiency or component failure. The deviation of the measured sensor data from these target temperatures shows whether a component behaves correctly or not. As a measure of deviation of the measured values from the estimated values, we will apply Mean Squared Error (MSE). MSE measures the average of the squares of the errors between two signals [WB09]. Then, outliers of the MSE values will be detected and annotated as erroneous signals. These results are visualized, allowing the facility manager to locate and analyze faulty hydraulic circuits.

4.2.3 Heat Emission Analysis

The calculation of the amount of emitted heat in a hydraulic circuit is not trivial and is hardly possible with little engineering effort. Since our approach intends to use only moderate engineering effort, we will assume the surface of the hydraulic circuits to be of the same size. The algorithm calculates the temperature differences between flow temperature and return flow temperature of a hydraulic circuit weighted by the flow rate in the corresponding measuring unit for a certain period of time. Only data points with flow rate are considered, i.e. heating is in operation. These determined energy units for the data points will be summed up and form the basis for the energy release analysis. These results are analyzed for individual sensors over several days and tested against each other for hydraulic circuits as well. To find the outliers, we examine the deviation from the standard deviation.

Algorithm 4.2: Heat Emission Algorithm

Data: time series of predefined length
Result: heat emission of each component

```
for components in data do
    heat_emission ← 0;
    previous ← null;
    for row in component do
        if row.prvlm > 0 AND previous is NOT null then
            heat_emission ← heat_emission + (previous − row.trt) ∗ row.prvlm;
        end
        previous ← row.trt
    end
end
```

4.2.4 Operation Time Analysis

The analysis of the operation time of hydraulic components is suitable for identification of inefficient system components. An abnormal deviation from the average operating time indicates a possible building failure, such as a broken window in the room or even poor building isolation. Operation time is simply calculated by summing up the data points where flow rate takes a positive value. The analysis results in a set of sensor-measured daily operation hours pairs. This method can be applied to the entire building at once over a certain amount of days, e.g. one week. We will analyze the distribution of daily operation hours in the rooms and the distribution of daily operation hours for specific rooms. This step is followed by the outlier detection by the means of the deviation from the standard deviation.

4.2.5 Anomaly Detection

In order to make adequate analyses possible, we have to eliminate erroneous sensor measurements. These incorrect values could result in misconfiguration of the underlying sensor topology as well as in inaccurate analysis results. There are two types of defects that have to detected: defects of single sensors, i.e. broken sensors or missing sensor connectivity which causes missing values in the data sets. We have to detect these anomalies in the time series automatically. For general anomaly detection, we use Luminol [1]. This lightweight Python library enables anomaly detection for time series as well as other functionalities like correlation analysis. It provides with various algorithms to perform the analysis: *Bitmap Detector*, *Default Detector*, *Derivative Detector* and *Exp-avg Detector*. The algorithm scores each data point to create the anomaly score, which is followed by the annotation of anomalous intervals based on predefined thresholds.

[1] https://github.com/linkedin/luminol

4.3 Fault Diagnostics for BSAE

A simple rule based diagnostics engine will be implemented in order to identify the cause of the faults. The rules should be defined by building experts and they should be configurable during run time of the application. For example, an irregular rise and falling of the heat storage temperature (oscillation) indicates a faulty behavior of this building component. This component of the system is only provided as a proof-of-concept due to the lack of expert knowledge and support, but it makes further development possible.

CHAPTER 5

Experimental Application of Methods

In this section, we will apply the identified concepts and methods on real-world data. By evaluating the methods, we will examine the applicability of the theoretical concepts and identify possible limitations of the concepts on real-world objects.

5.1 Experimental data

5.1.1 Data description

The data used for the application of methods and their evaluation come from an office building. The building is equipped with numerous invasive sensors that measure room and water temperatures, flow in pipes, electricity consumption, etc. Operational data is also recorded at the same time. Our problem instance requires that the solution uses only minimal engineering effort and therefore only limited data is available for topology discovery and building analysis. We only use flow and return flow temperature sensors, flow rates and room temperature sensors. These sensors can be implemented as non-invasive sensors, which is why they are in the focus of our analyses.

As shown in Figure 5.2, four office rooms are available on the upper floor of the test building, each of which is equipped with several heating circuits. Each room is equipped with heating and cooling systems (two concrete core activation, *BKA 1* and *BKA 2*; and floor heating, *FBH*) as it can be seen in Figure 5.3. We will only analyse the floor heating system. In the first, third and fourth room are two floor heating circuits, in the second there are three heating circuits. There is also an observation room, which is only partially used, and a corridor connecting the rooms. All rooms are used as offices, so we can derive relevant analyses regarding usage patterns. In addition to the upper floor,

there is a measuring point that measures characteristics of the ground floor, along with temperature sensors from different room on that floor.

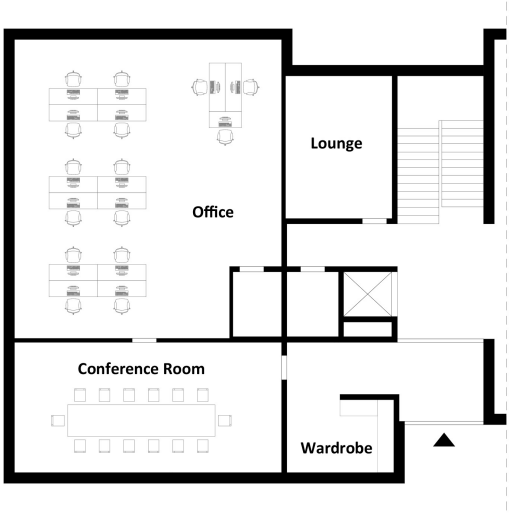


Figure 5.1: Floor plan of the ground floor

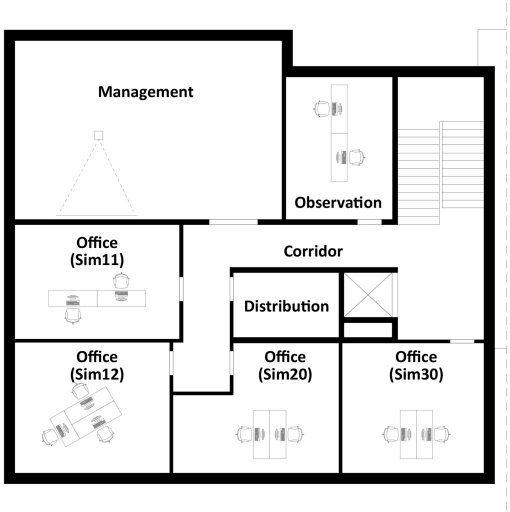


Figure 5.2: Floor plan of the first floor

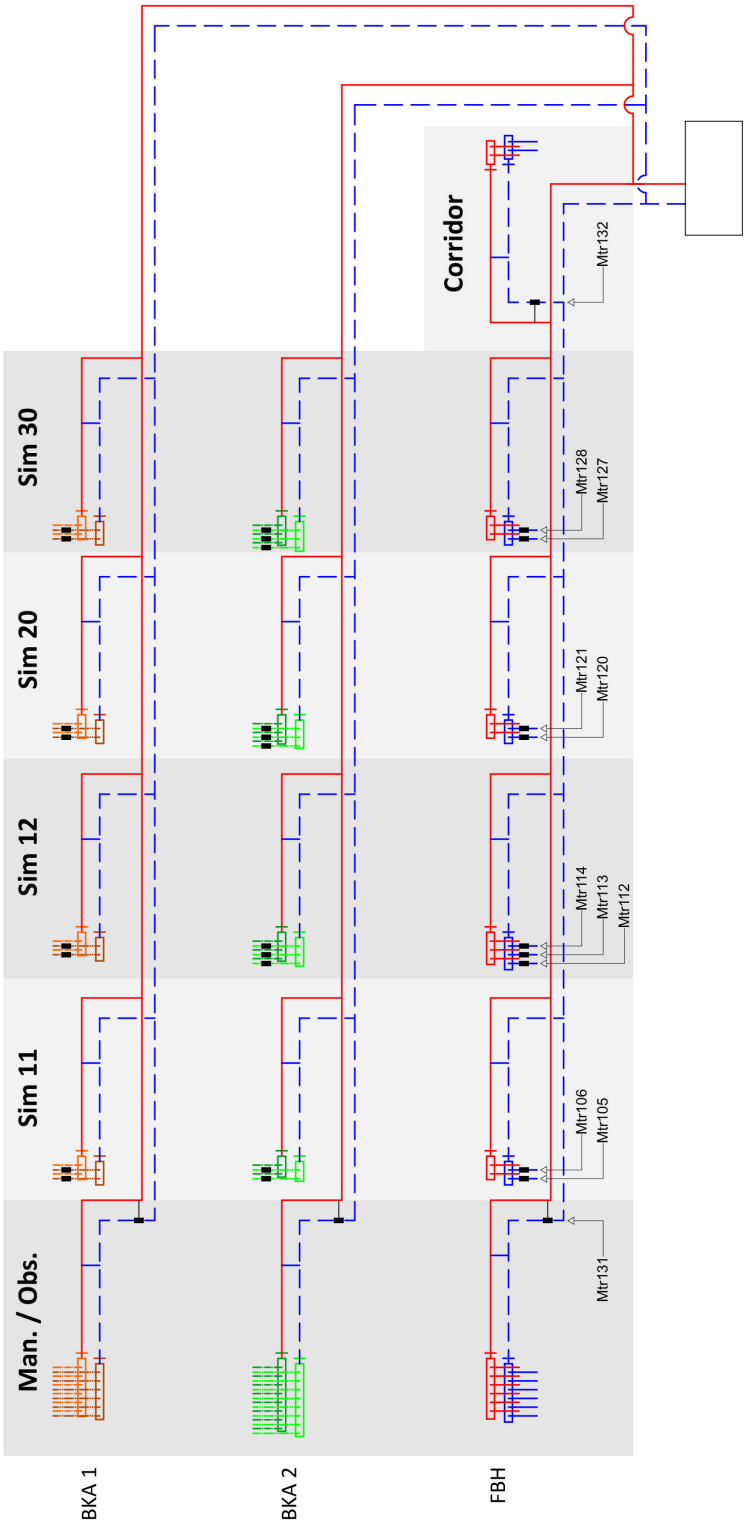


Figure 5.3: Scheme of the heating system

Each measuring point has three sensors, flow temperature, return flow temperature and flow rate. The data is collected in the test building via a gateway using the BACnet [1] communication protocol. The gateway is also responsible for sending the collected data to the cloud. The collected data is stored in the cloud and made available for further analysis.

Even if the concepts and methods are well applicable in theory, it can lead to problems when applying the concepts to real data which must either be eliminated or identified as an additional precondition. The following errors have occurred during data recording:

1. **No decimal places:** The data from the test building comes without decimal places due to the specification and hardware architecture of the sensors. (Later the accuracy of the measured values was improved to one decimal place.) This problem prevents a detailed analysis of the sensor behavior. Heating circuits close to each other, e.g. in a room, show only small differences during a heating phase. The absence of decimal places therefore causes the loss of relevant information. Figure 5.4 shows the flow temperature signal for the sensor *Mtr121* in one of the office rooms *Sim20*. As we can see, the signal does not record decimal places, and therefore, relevant information will be ignored.

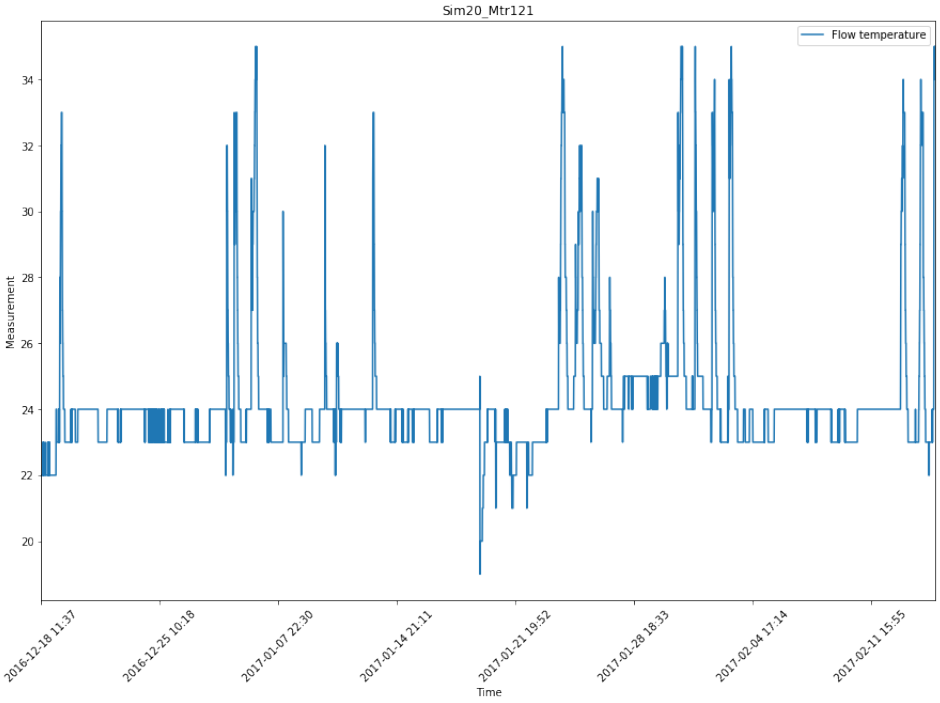


Figure 5.4: Example for a signal with missing decimal places

[1] http://www.bacnet.org/

2. **Turned off heating components:** This problem occurs when the sensors on the heating circuit are working properly, i.e. they are constantly sending correct data, but the component (e.g. radiator) is switched off. In this case, the water in the radiator absorbs the room temperature and shows only moderate changes in the measured data and only affected by the temperature of the room. Therefore, it is not possible to derive information about inter-device relationships and the measuring point cannot be located in the building. This problem cannot be solved using software methods and is therefore identified as an additional constraint for the system, i.e. for correct topology discovery and building analyses, all investigated heating components must be in operation. Figure 5.5 demonstrates the lack of temperature changes for a turned off heating component *Mtr102* in the office room *Sim11*. Switched off heating circuits lead to misclassifications and false analyses.

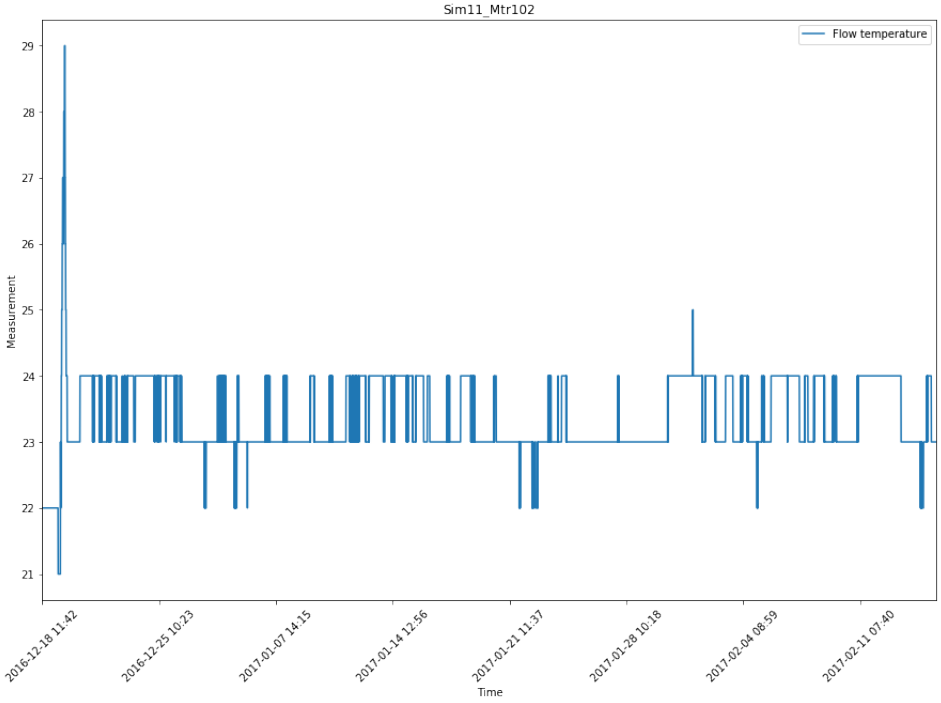


Figure 5.5: Flow temperature measurement for a turned off heating component

3. **Building Hydraulics Structure:** The heating system in the test building has a star-shaped structure in which a central distribution room distributes the heat on request. This means that there is a room on the upper floor from which the rooms are directly heated or cooled, i.e. the hot water for room A never flows through room B but is routed directly to the respective room. This behaviour of the heating system is not common in older buildings, where the hot water sequentially reaches the rooms and the room relationships can be recognised. In addition, the temperatures in the offices are controlled independently. These properties of the

test building prevent the precise analysis of room relationships, which is the basis for the topology discovery of the sensors. In order to be able to test our methods with semi-real data, we will simulate the behavior of older buildings in the test building. In this simulation, the rooms are heated one after the other and the hot water reaches the rooms sequentially. This simulation is called Footprint Method in our project. (Details about the Footprint Method can be found in Section 5.1.3.)

5.1.2 Data preparation

In addition to hardware limitations (see in above sections), other errors and problems can occur during data recording that need to be fixed by the software. Collected sensor data is stored in the cloud and can be accessed by our Building Service Analysis Engine (BSAE). This data must be cleaned and prepared before analyzing it. In the following sections, we will summarize and explain the identified preparation steps.

1. **Uniform sensor data resolution:** To reduce the amount of data transported to the cloud, the data is not recorded in a fixed resolution, but the system listens for changes in the sensor values, i.e. if the measured value changes, a data point is recorded. This procedure implies unstructured data recordings regarding time. However, a uniform data resolution is required for data analysis. For this reason, the data must be resampled, which is easy to implement with Pandas [2].

2. **Interpolate missing values:** The missing data points can be filled in different ways. The simplest solution is to fill the values with the last measured value during resampling. Another possibility is to interpolate the missing values, where again several strategies can be applied: linear, nearest, quadratic, etc. We will apply several interpolation strategies and use the best one for the analyses.

3. **Identification of network failures:** A common error at data collection in the buildings is the failure of the network. If there is no Internet connection available, or an internal error occurs in the data collection network, no data is sent to the cloud, resulting in *gaps* in the time series. These errors should not affect on topology discovery and building analyses and must be detected and fixed by the software. If the entire network fails, no more data will be sent. If the network goes online again, new data will be sent to the cloud. The data gaps caused by this failure are filled by resampling and interpolation. However, these sensor values do not reflect the correct system behavior and must therefore be identified and marked accordingly. If the data from all sensors remain the same after several points in time, we can assume that it is a network failure. Pandas provides features that can help detect this no-change behavior. With a rolling window analysis, sensor values can be detected that do not change within a specified time interval. Since our resampling manager resamples the data points to a ten minute resolution, we can limit this

[2] https://pandas.pydata.org/

rolling window to two data points, i.e. if no sensor changes in ten minutes (it is very unlikely that any of the sensor values will not change in ten minutes), we can identify a network failure. The detected time intervals must be marked and are therefore treated as irrelevant data in the analyses. This information about the network failure is also disclosed to the facility manager. Figure 5.6 contains flow temperature signals for different heating components over six weeks. As we can see, after two weeks all temperature sensors stopped sending measurement data which is represented by the constant measurement values.

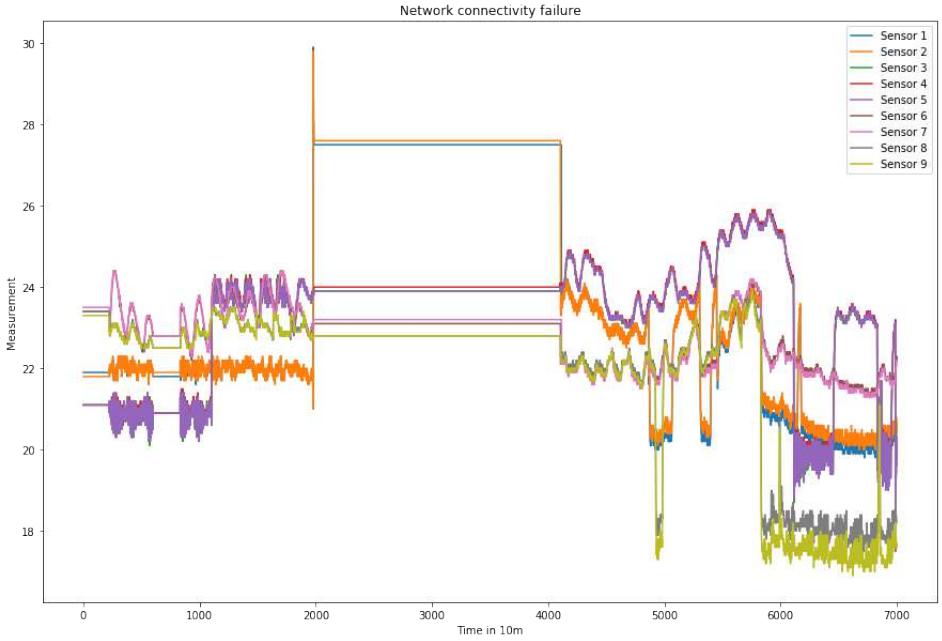


Figure 5.6: Signals without value changes imply a network connectivity failure

4. **Identification of sensor failures:** The failure of a sensor can be described in the same way as a network failure. If a sensor does not send any data, but the remaining sensors provide correct data, the sensor may be broken. Detecting the failure of a single sensor is not as trivial as detecting a network failure. Because of the communication architecture described above, there are several reasons for the lack of incoming data from a sensor. Since the data recording system only records the changes in the measured values, it is possible that the measured value has not changed at the measuring station. Since the measured values are only recorded with one (or no) decimal place, minor changes in the measured values may be ignored by the data recording system. In this case, no data will be sent to the cloud. Another possibility for the failure of the measured value transmission is that the heating circuit is switched off and there are hardly any changes in the data. This case is excluded by setting a precondition: each heating circuit must be in operation. The last possibility is that the sensor is broken and no longer able to write data to

the gateway, which is the case we want to detect. For the identification of such cases, we have to do a rolling window analysis analogous to the failure of the entire network. The size of the time interval in which no changes in measured values are permitted is determined manually and is empirically evaluated. If a sensor does not transmit a change within one hour, we assume that the sensor is broken. However, this only applies if the other sensors transmit data normally parallel to this time period, i.e. it is not a network failure. These time periods are marked accordingly unreliable as in the case of a network failure. Figure 5.7 shows the failure of the flow temperature sensor *Mtr105* in the office room *Sim11*. We can observe the lack of incoming senor measurements for a longer time period with the constant sensor values.

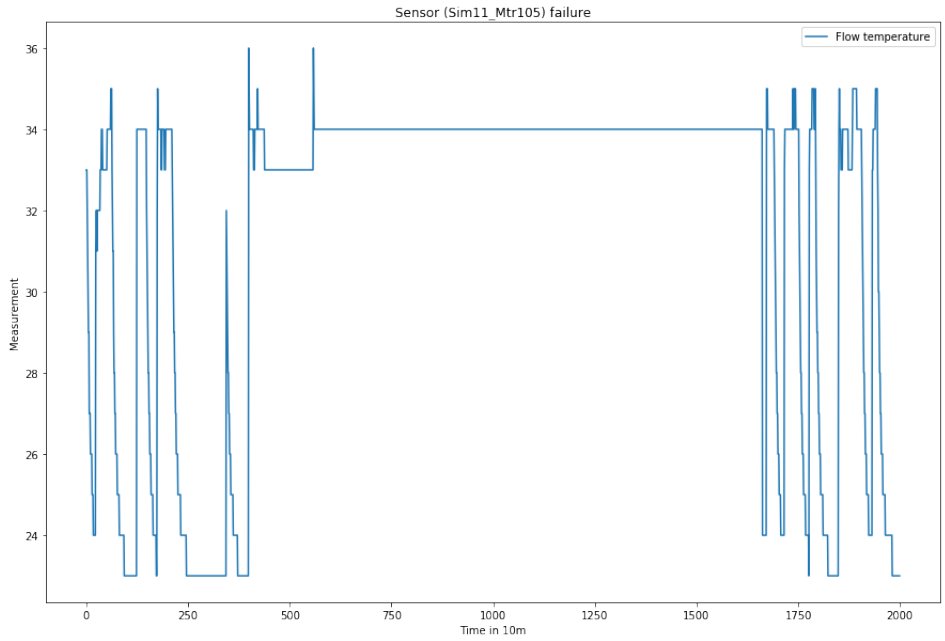


Figure 5.7: Single signal without value changes implies sensor failure

5. **Detection of anomalous sensor values:** In addition to the failure of the entire network or the failure of a single sensor, incorrect values may possibly come from the sensors. In this case, measured values will arrive in the cloud and will be stored. This erroneous data must be detected and removed. We will use anomaly detection engines to detect the anomalous data in the time series. The Python library Luminol[3] is a time series analysis tool that can detect anomalous data points. The library offers several anomaly detection methods, such as *Bitmap-Detection*, *Derivate-Detector* or *Exp-Avg-Detector*. For our use case, the *Derivate-Detector* works best. This detector is meant to be used when abrupt changes of value are of

[3]https://github.com/linkedin/luminol, accessed 22.01.2019

main interest. We use it with the default smoothing factor 0.2. As you can see in Figure 5.8, abnormal data points (in this case zeros) are detected as anomalous marked with colored lines in the graph. These data points can then be removed and interpolated.

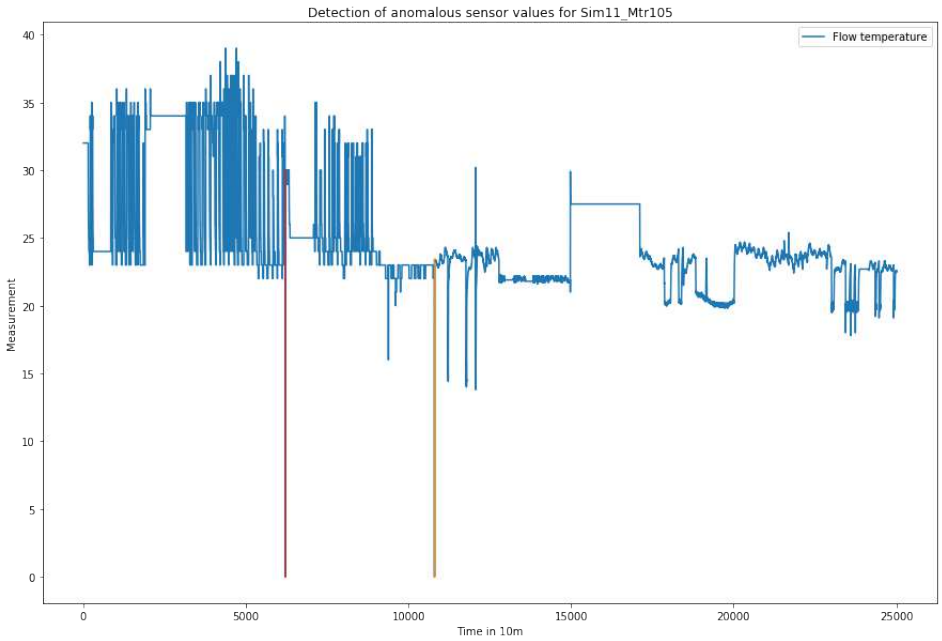


Figure 5.8: Anomalous sensor values for a flow temperature signal are colored

5.1.3 Footprint method

The Footprint Method is an initialization phase in the BSAE. After the installation of all non-invasive sensors and the communication network in the building, the building will be cooled down. Then all heating circuits and radiators are turned on and the building is heated. This special data are used exclusively for topology discovery, they are not suitable for building analyses purposes. This short phase takes little time and engineering effort, but it is still very important for many topology detection steps. On one hand, this data can be used for the classification of the sensor measurements while on the other hand, they form the basis for the detection of the room sequences in building parts. In this initialization phase, all temperatures are heated to the same set point, which simplifies the classification of sensor values and makes more accurate analyses possible. Since all heating components are turned on, the rooms in a building part are heated one after the other, thus enabling the detection of room relationships.

Figure 5.9 shows the flow temperature changes for a hydraulic circuit in the training phase which enables the detection of the sequence of sensors on this hydraulic circuit which corresponds to physical arrangement of these sensors.

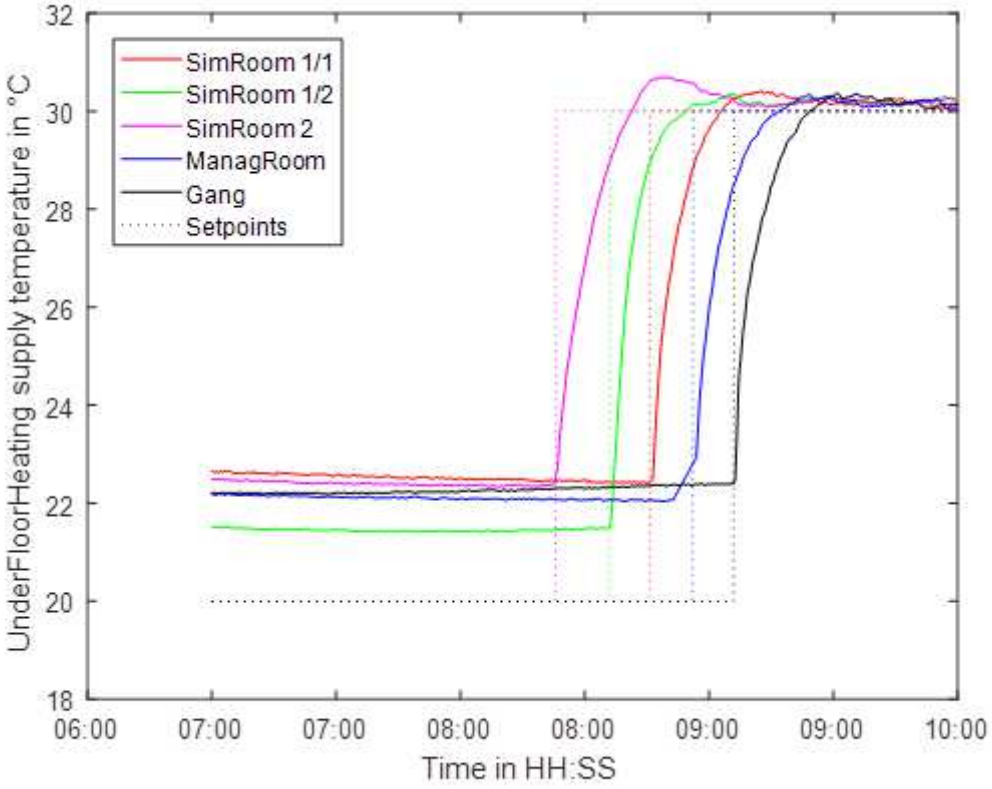


Figure 5.9: Initialization phase for flow temperature measurements

5.2 Evaluation of Methods

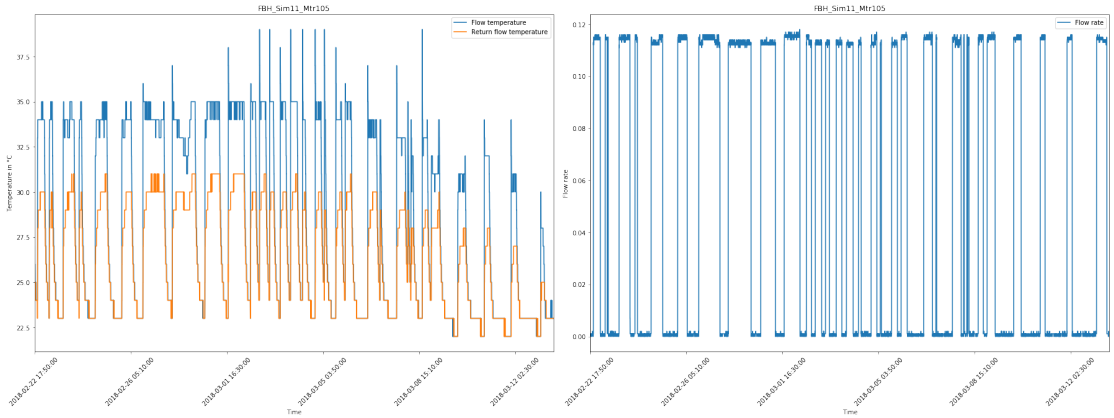
In this section, we will apply concepts and methods on the real-world test data set. The evaluation of the methods is based on different metrics and against the use cases.

5.2.1 Topology Discovery

Classification of sensor measurements

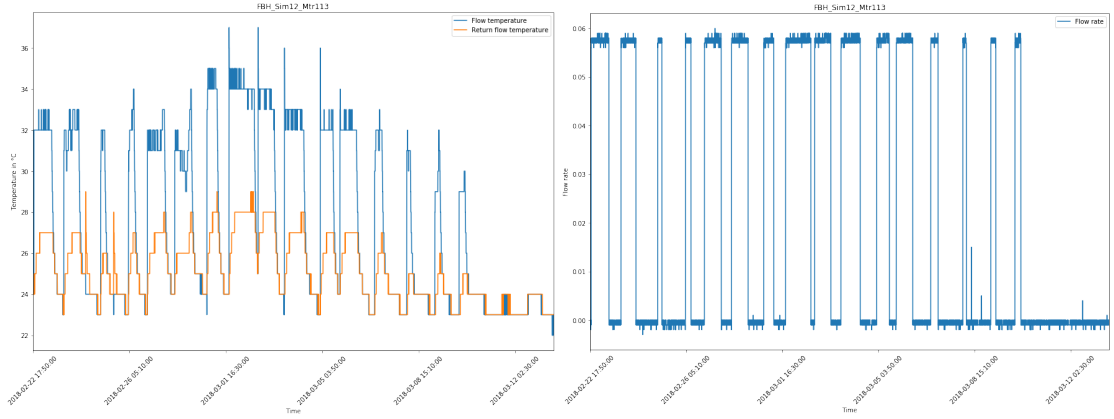
As described in the sections above, two approaches for the classification of sensor measurements are applied and evaluated. The K-Nearest Neighbour Algorithm and K-Means Clustering are two concepts used for the classification. Figure 5.10 shows the sensor recordings for two selected measurement points. On the left hand side, the flow and return flow temperatures are shown, while on the right hand side, the flow rate is plotted. The values are recorded over twenty days in winter time. During the heating phase of a building, the return flow temperatures are always lower or equal to the flow temperatures. As can be seen in Figure 5.10 in the bottom figures, the heating circuit

was no longer heated during the last couple of days of the recording (as there was no flow rate, see bottom right figure). In this case, the water temperatures are only influenced by the room temperature and absorb the temperature of the room.



(a) Flow and return flow temperature for *Mtr105*

(b) Flow rate temperature for *Mtr105*



(c) Flow and return flow temperature for *Mtr113*

(d) Flow rate temperature for *Mtr113*

Figure 5.10: Signals of two measurement stations

While applying the classification methods, we identified an error: the analysed data contained heating and cooling phases at the considered time interval. To illustrate this, Figure 5.11 depicts sensor data from March to May. At the beginning of the data set, the building was heated, which can be seen from the fact that the return flow temperatures are below the flow temperatures. In the middle range, the flow and return flow temperatures are alternately higher and lower. This occurs when heating and cooling are used at the same time. In April, at night and in the morning hours, the building was heated, while during the day, when the outside temperatures were already higher, the hydraulic system was used to cool the offices. Beginning from end of April the offices were only cooled by the hydraulic system. These changes in the behaviour of the flow and return flow temperatures usually have a negative impact on the classification of the sensor

measurements. For this reason, it is important that only heating or cooling phase data is used when applying these methods. This is an additional engineering effort, but can be handled by the Footprint Method (see Section 5.1.3).

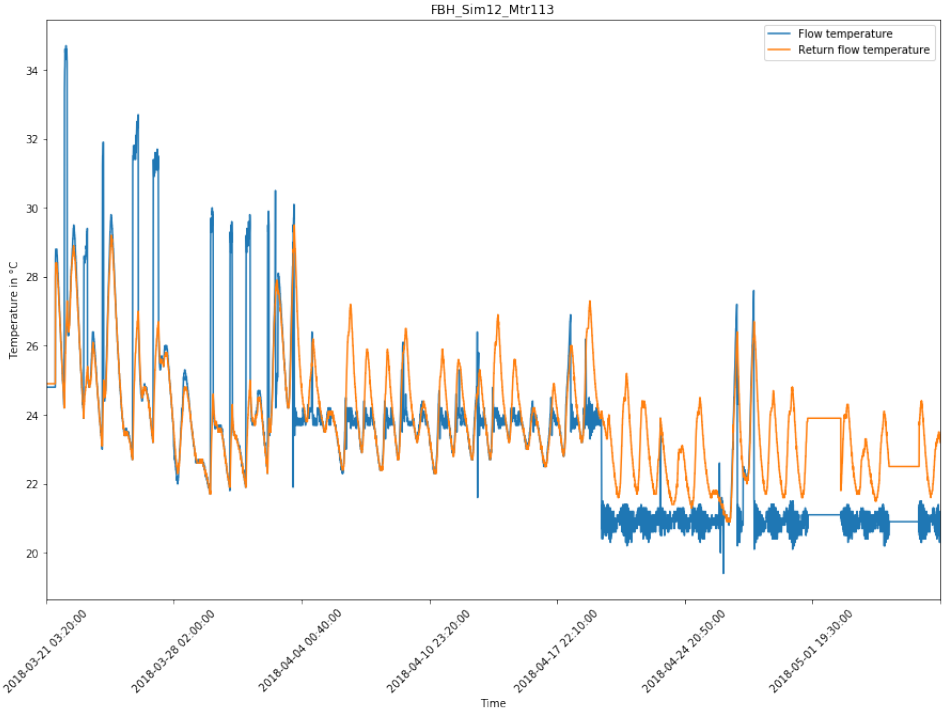


Figure 5.11: Heating and cooling in the same time series

For the K-Nearest Neighbour algorithm, we will annotate the smallest possible subset of sensors with the correct measurement type (flow temperature, return flow temperature or flow rate). The selection of sensors was done randomly and was tested in multiple iterations. We used Euclidean distance with Dynamic Time Warping as the measure of distance between two time series. Table 5.1 and Table 5.2 show the results of the training with several iterations. In the first experiment, 10% of the sensors were annotated (i.e. labeled manually with the correct sensor type) and the after applying the algorithm we achieved an accuracy of 83%, i.e. 83% of the sensors were correctly classified. In the second experiment, the number of annotated sensors was drastically and unrealistically increased to 50%, i.e. the engineers should annotate half of the sensors with the correct sensor type, which is not compatible with the use case about little engineering effort. In this case, we achieved an average accuracy of 87%, which is an improvement to the previous experiment. The results could be seen as successful, but for our use case there is a need for an almost perfect assignment of the sensor types to the actual time series. If a sensor is misclassified, the results of the analysis will be inadequate.

Class	Avg. Precision	Avg. Recall	Avg. F-Score
Flow rate	1.00	1.00	1.00
Flow Temperature	0.92	0.50	0.62
Return Flow Temperature	0.65	0.93	0.77

Table 5.1: Labeled Training Dataset 10% - Average Accuracy: 83%

Class	Avg. Precision	Avg. Recall	Avg. F-Score
Flow rate	1.00	1.00	1.00
Flow Temperature	0.95	0.70	0.80
Return Flow Temperature	0.77	0.95	0.85

Table 5.2: Labeled Training Dataset 50% - Average Accuracy: 87%

As depicted in the tables, the flow rate sensors were always classified correctly. The reason for this is that these sensors take completely different values than the temperature sensors (flow rate < 1). In contrast to the flow rate sensors, the flow and return flow temperature sensors are often wrongly classified. Figure 5.12 shows the possible reason for the incorrect classifications. The figure shows the time series of two return flow temperature sensors and one flow temperature sensor. Intuitively, one would say that the blue signal is more similar to the green one than to the orange one. The Euclidean distance metric used with DTW also provides these results. However, this means that the return flow temperature sensor in the middle is identified as flow temperature sensor. This error can be avoided using the *Footprint Method*, where all heating components are heated to the same temperature.

For our domain, K-Means Clustering can be more efficient and successful than K-NN. As described in the sections above, no annotated data is required for clustering. For the first experiment, we took a time interval of 12 days and achieved an accuracy of 92% with the default configuration of the algorithm. Figure 5.13a shows the cluster centroids for the three sensor types. The reason for the misclassifications can be found in the overlaps of the cluster centroids. These occur when the heating component is not used and in these time periods the signals of flow and return flow temperatures will be very similar.

To work around this problem, we have implemented an algorithm that filters the points in time from the records where there is no flow rate. This results in time series where all heating components are in operation. With the help of this methodology, we achieved an accuracy of 100%, i.e. all sensors are assigned to the correct sensor type. Figure 5.13b shows the cluster centroids for the second experiment. We can observe, that now, all clustering centroids are distinct, and there are no overlappings between them. This ensures that there are no misclassifications of the sensors. We could show that even without manual effort, we can classify sensor measurements, by selecting an appropriate time interval and eliminating the data points where the heating is not in use, i.e. there is a positive flow rate.

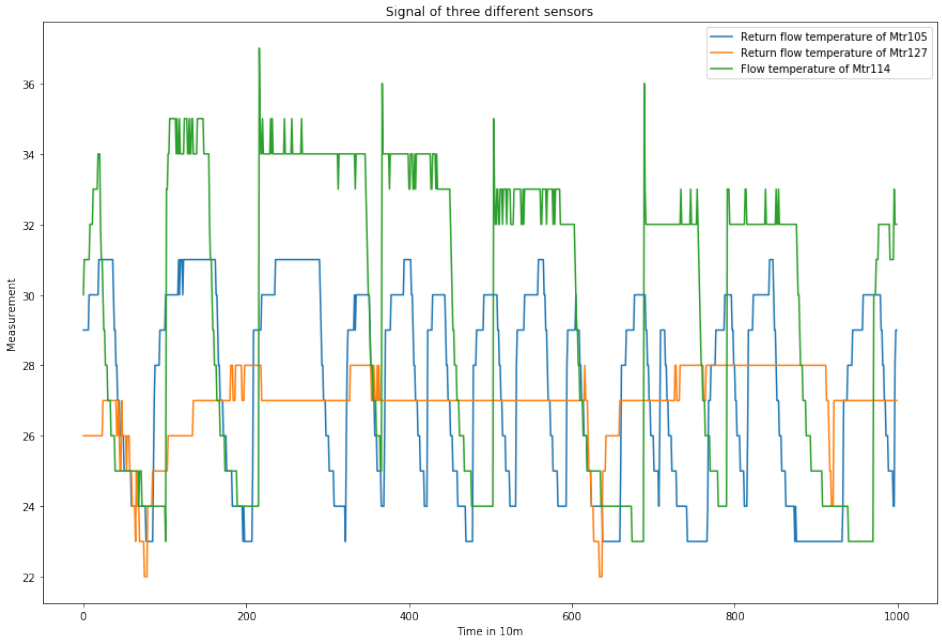
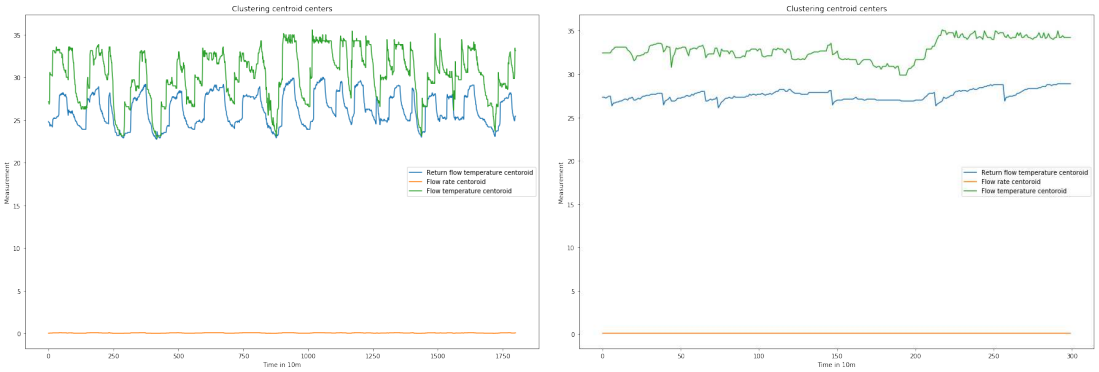


Figure 5.12: Possible cause of misclassification of sensors



(a) Centroids without flow rate restriction (b) Centroids with flow rate restriction

Figure 5.13: Clustering centroids of the different sensor types

Establishment of sensor groups at same location

In order to establish sensor groups of flow temperature, return flow temperature and flow rate sensors, we will implement the algorithm described in Section 3.2. This is important to enable building analysis potentials, since some analysis methods require the existence of different sensor types in the same building segment. As described above, our approach expects every hydraulic component to be equipped with exactly one of each sensor type (flow temperature, return flow temperature and flow rate sensor). First, the

algorithm calculates sensor assignments between flow temperature and flow rate sensors based on there correlation values, i.e. the correlation matrix of flow temperature and flow rate measurement time series will be calculated using the Pearson correlation matrix. The resulting assignments are shown in Table 5.3. As we can observe, the assignments were correctly identified except for three relationships. All three wrong assignments are heating circuits in the same room, because they behave similarly. This can be avoided, be using the Footprint Method as described in Section 5.1.3.

Flow rate	Flow temperature	Success
Mtr105	Mtr105	True
Mtr106	Mtr106	True
Mtr112	Mtr114	False
Mtr113	Mtr112	False
Mtr114	Mtr113	False
Mtr120	Mtr120	True
Mtr121	Mtr121	True
Mtr127	Mtr127	True
Mtr128	Mtr128	True

Table 5.3: Flow rate - Flow temperature assignments

Table 5.4 shows the assignments of return flow and flow temperature sensors. We have achieved a completely correct assignment of the sensors. Thus, the measuring points consisting of flow and return flow temperatures and flow rate sensors were established and can be used for the analyses. As mentioned before, this approach assumes that each one of the sensor types is installed on each heating component exactly once. Additionally, we applied the algorithm to assign flow temperature measurements to room temperature sensors. Table 5.5 shows the assignments of room temperatures to flow temperatures. We have shown an efficient way of establishing sensor groups without manual effort, based only on available sensor measurements.

Return flow temperature	Flow temperature	Success
Mtr105	Mtr105	True
Mtr106	Mtr106	True
Mtr112	Mtr112	True
Mtr113	Mtr113	True
Mtr114	Mtr114	True
Mtr120	Mtr120	True
Mtr121	Mtr121	True
Mtr127	Mtr127	True
Mtr128	Mtr128	True

Table 5.4: Return flow temperature - Flow temperature assignments

Room temperature	Flow temperature	Success
Sim11	Sim11_Mtr105	True
Sim12	Sim12_Mtr112	True
Sim20	Sim20_Mtr120	True
Sim30	Sim30_Mtr127	True

Table 5.5: Room temperature - Flow temperature assignments

Establishment of clusters of sensor nodes in same building segment

In order to establish clusters of sensor nodes that are near to each other, it is necessary to have sensor data from different parts of the building. Our test building only provides room temperature data from two different floors of the building: three rooms are located in the ground floor (conference room, lounge and wardrobe, see Figure 5.3) while another four temperature sensors are located in the first floor offices. Figure 5.14 shows the room temperature measurements over three weeks in these rooms. Room with id from 0 to 2 are located downstairs, while rooms with id from 3 to 6 are located in the first floor.

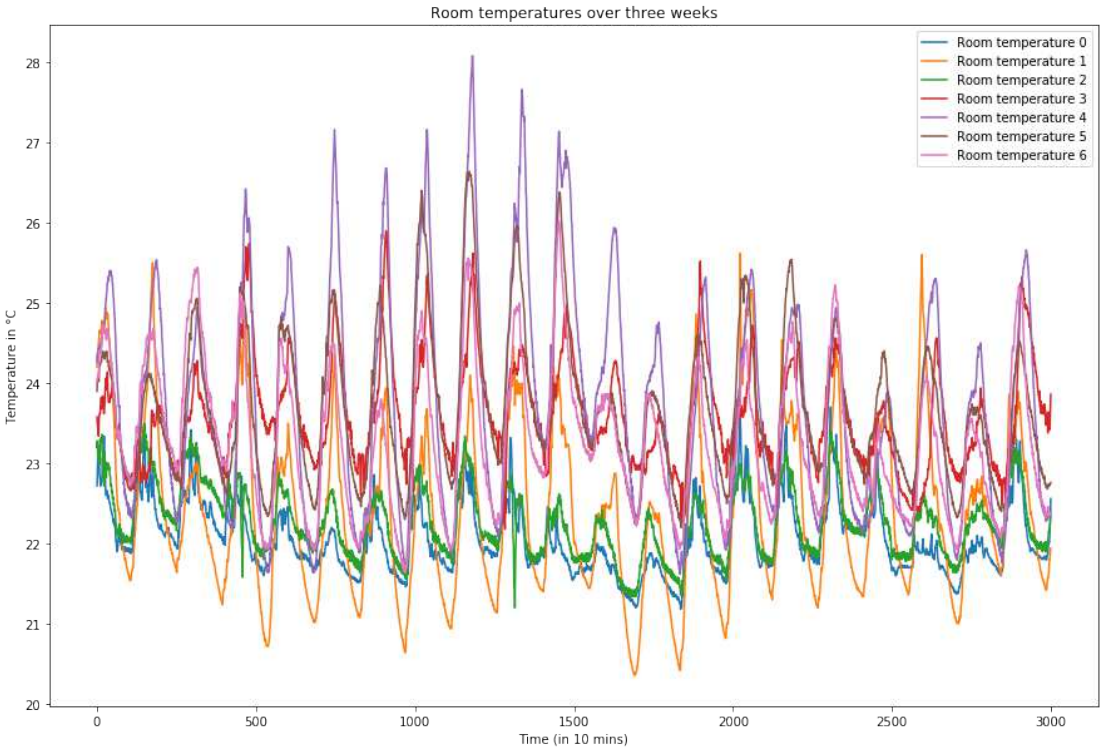
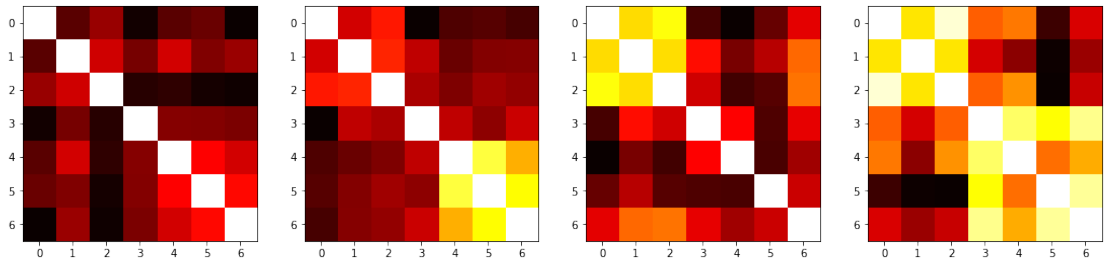


Figure 5.14: Room temperatures for selected time period

As described above, we will decompose the sensor measurements using EMD into IMFs. This is followed by creating clusters of the IMFs. We will create four clusters of the IMFs:

high frequency, middle frequency 1, middle frequency 2 and low frequency. Each IMF for each sensor will be assigned to exactly one cluster. After that, we aggregate signals from each clusters by taking the mean of each signal within a cluster. These signals form the basis for our correlation analysis. Figure 5.15 shows the correlation matrices of each aggregated signal set. As we can observe, we cannot identify any meaningful correlation on the high frequency signals. While low frequency signals show a high correlation between the two floors. The first three rooms are grouped into one unit, while the other four units form an obvious cluster as well.

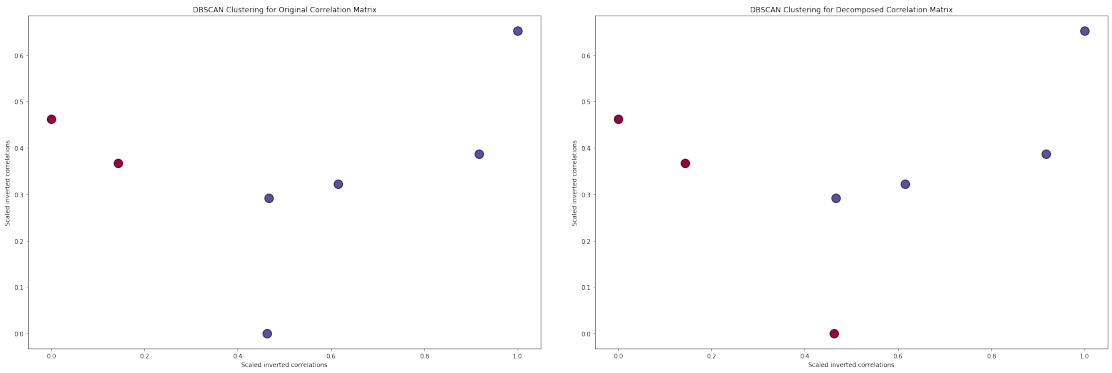


(a) High frequency (b) Middle frequency 1 (c) Middle frequency 2 (d) Low frequency

Figure 5.15: Heat maps of correlation matrices on different frequencies

As described above, we apply DBSCAN clustering on the correlation matrix of the process. In order to use the correlation matrix as distance metric for the algorithm, we have to modify it, since higher correlation means smaller distance. To achieve this, we multiply the correlation matrix by -1 and scale the values to be in the range of 0 and 1. To successfully configure the DBSCAN algorithm, we have to define the *eps* parameter which is the maximum distance between two entity to be seen as neighbors of each other [4], we set it to 0.3 and additionally, we set the minimum number of samples that can build a cluster to 2. We run the clustering algorithm for both the original correlation matrix generated by the raw data and also for the best promising EMD result. Figure 5.16 shows the results of the experiments. For the original correlation matrix (see on the left-hand side of the figure), the algorithm groups almost everything correctly except one room, i.e. the first and second time series have been grouped together (colored red), but the third time series was assigned to the other cluster (colored blue). The experiment that uses the best EMD correlation matrix for the clustering, outperforms the other one by clustering each room to the correct floor, i.e. the first three dots are colored red together while the other four time series were colored blue (see right-hand side of the figure).

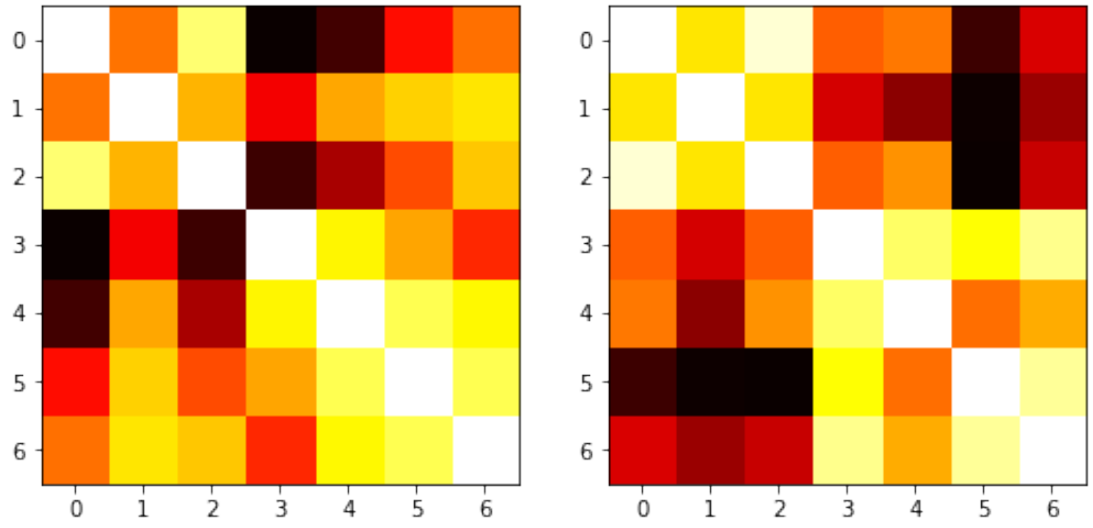
[4] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.dbscan.html



(a) Clustering before EMD (b) Clustering after EMD

Figure 5.16: Clustering results of flow temperature signals with DBSCAN

We can observe, that correlating signals on different frequency levels may lead to an improvement of spatial localization of sensors. Figure 5.17 illustrates the comparison of the original (left-hand side) and the best decomposed correlation matrices (right-hand side). As we can see, on the EMD correlation matrix (right-hand side) the two floors are distinguished on a clear way, i.e. the first three rooms have high correlation and so do the other four rooms. However, due to the lack of available sensor data from different parts of the buildings, we cannot prove the accuracy of the algorithm for any kind of sensors.



(a) Original correlation matrix (b) Best EMD correlation matrix

Figure 5.17: Comparison of the original and the best decomposed correlation matrices

Identification of sensor sequences within sensor node clusters

- **Hierarchical Clustering:** As described in Section 3.4.1, we will establish hierarchical relationships between sensors based on flow temperature using hierarchical clustering methods. To evaluate the success of the clustering, we created a reference hierarchical clustering of the sensors based on prior knowledge of the sensor location relationships. Figure 5.18 shows the reference (correct) clustering for rooms on a hydraulic circuit. This dendrogram represents our desired hierarchical clustering. In the next step, we evaluated multiple clustering configuration sets regarding linkage method, distance metric, sensor type and time interval against the reference clustering by measuring the correctly identified sensor location relationships. These experiments could not identify sensor relationships correctly, thus brute forcing the configuration of the algorithm was not successful.

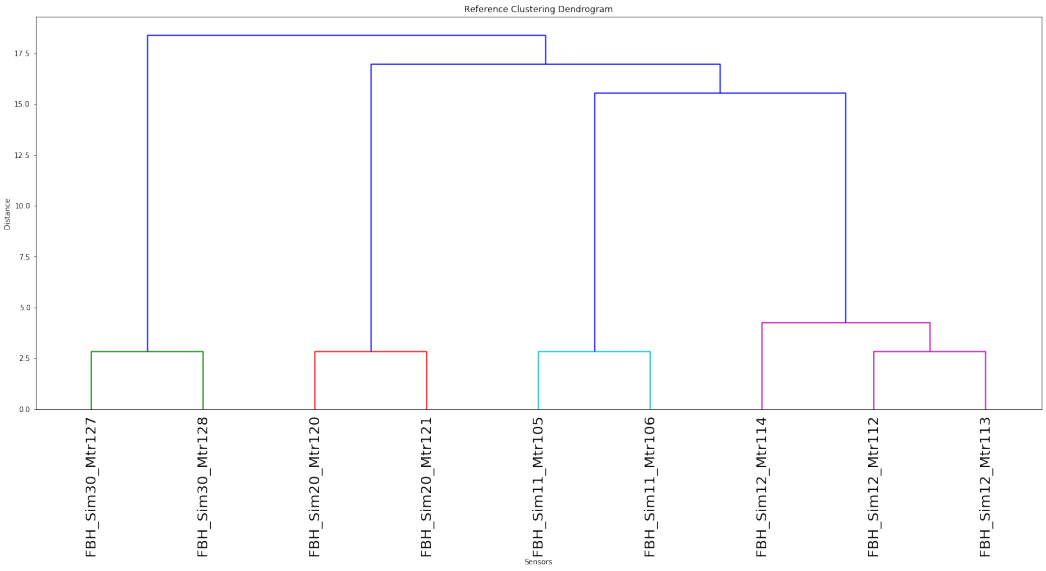


Figure 5.18: Reference hierarchical clustering

Figure 5.19 depicts the experimental hierarchical clustering for the same hydraulic circuit as for the reference clustering. We can observe, that in most cases the algorithm could identify relationships within one room, but failed to identify relationships between rooms correctly. The reason for this may be the star-like layout of hydraulic components in the test building. To take time shifts of heating events in account, we will improve the distance calculation using DTW as described in Section 3.4.2.

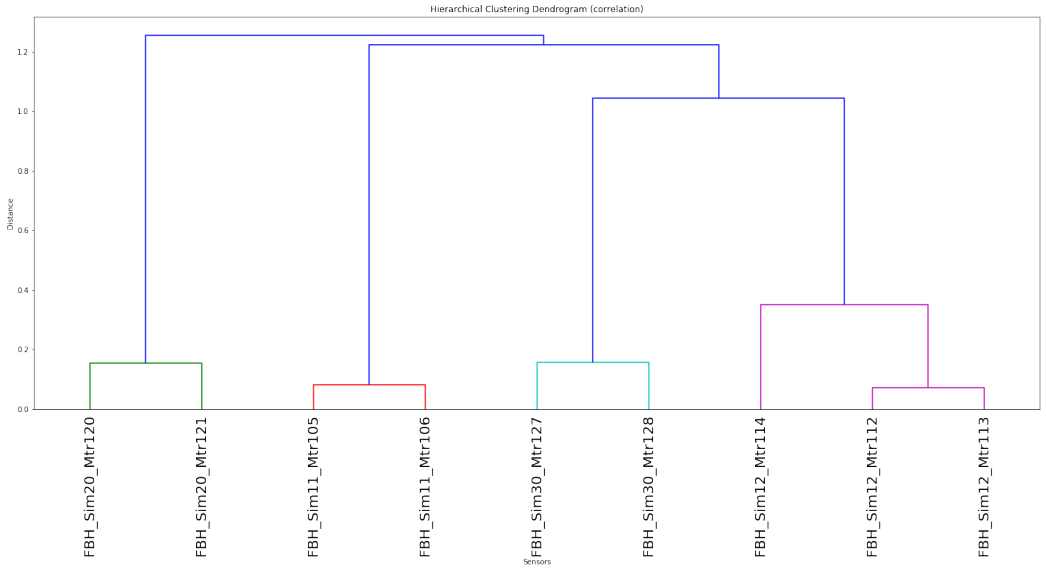
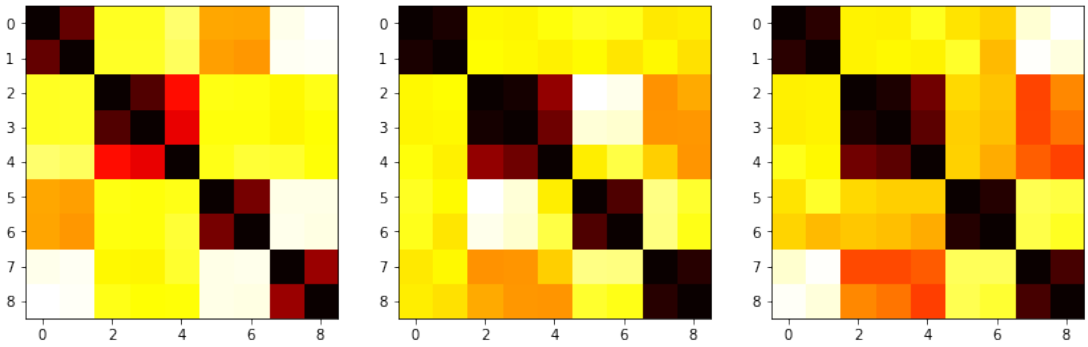


Figure 5.19: Correlation based hierarchical clustering

- **Dynamic Time Warping:** We applied the proposed solution to our data set, by calculating the distance between two time series using DTW. This was done by a Python implementation of this method, fastdtw [5]. The algorithm can be configured with distance metric and radius. Radius is the key parameter that defines the number of data points to both sides from the current data point that should be evaluated for finding the optimal matching. Setting the radius too high increases the runtime of the algorithm extremely. We used 5 as radius to balance between potentially better solution and runtime. This technique results in more adequate distance values, however, the connections between rooms are still not discovered correctly. Figure 5.20 shows the results of the clustering with our different approaches. Since we expect distances to be smaller between close rooms, on the heat map, we expect sensors near to each other to be darker (due to the smaller distance). For the sake of comparability, we multiplied the correlation values by -1 to be able to compare these results with the distance measurements. This was necessary, because we expect close rooms and sensors to have a higher correlation. As we can see, along the diagonal there are dark fields where the distance is smaller. Already for Euclidean hierarchical clustering, we can observe sensors in the same room to be near to each other. The second experiment shows better distance measurements especially for sensors 2, 3 and 4. The last heat map shows our experiment of comparing Euclidean distances fine tuned with DTW. We can observe further improvements in distance calculation but still weak relationships between rooms. To overcome this issue, we have to use the *Footprint Method* as described in Section 5.1.3 to initialize the building for the topology discovery

[5] https://pypi.org/project/fastdtw/

process.



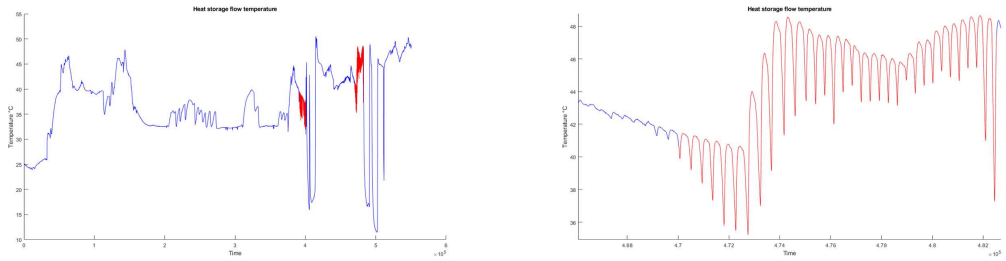
(a) Euclidean distance (b) Pearson Correlation (c) Euclidean with DTW

Figure 5.20: Heat map of sensor relationships using different distance metrics

5.2.2 Building Service Analysis Engine

Oscillation Detection

The oscillation detection algorithm uses a derivative approach, where we run a sliding window on the dataset and count the sign changes with a predefined amplitude and frequency. After applying the algorithm on the experimental data, we observed that the configuration of the algorithm is crucial for delivering meaningful results. To detect the desired kind of oscillations, we needed to configure the algorithm parameters precisely. After some test runs, we managed to annotate the desired parts of the signal regarding their amplitude and frequency. The algorithm is capable of performing on different data resolutions without adjusting the parameters. The algorithm takes three parameters: the size of the rolling window in seconds (windowSizeInSeconds), the minimum difference to consider an oscillation (minDiff) and the minimum count of oscillations within a time window to consider it erroneous (minOscillationsInWindow). As a result, we identified the parameters to be 3600 for *windowSizeInSeconds*, 0.06 for *minDiff* and 15 for *minOscillationsInWindow* for our test case. Figure 5.21a shows the annotated signal of a heat storage and marks the erroneous parts of the graph. Figure 5.21b shows the same annotation in higher resolution. The idea behind this approach is to identify high frequency oscillation for hydraulic components, for example for a heat storage. This high frequency oscillations may exist due to the simultaneous request for hot and cold water that is a potential flaw in the hydraulic system causing suboptimal energy management of the system.



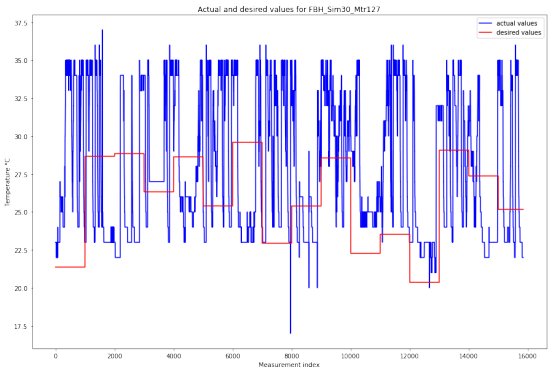
(a) Low resolution (b) High resolution

Figure 5.21: Identified high frequency periods of a sensor

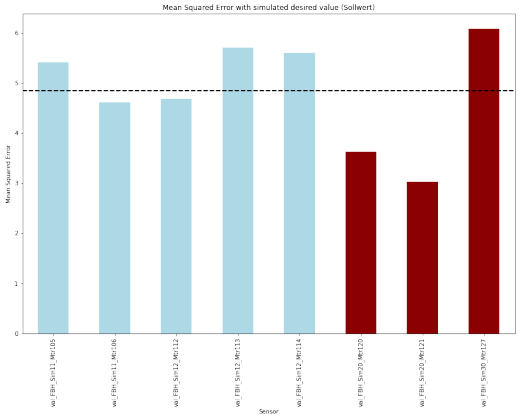
Target Temperature Analysis

The deviation analysis of the current temperature from the target temperature is an essential approach to verify the correctness of the hydraulic system operation. The idea is to measure the deviation of measured flow temperature values from the values set by the hydraulic system to fulfill. Since our test BMS does not provide target values for flow temperature, we will simulate these values to enable the evaluation of our approach. Figure 5.22a shows measured sensors values and the corresponding, simulated target temperatures for a selected sensor *FBH_Sim30_Mtr127* over a month. On the X-axis, we see the time while the Y-axis holds the temperature values in Celsius. The blue line represents the exactly measured values of this particular sensor, while the red line shows our simulated target flow temperature values. We simulated the target temperature within a specified temperature range and with variable length. This may not represent the real-world behavior of the target temperature set points, but for sake of our analysis it is sufficient.

Then, we calculate the Mean Squared Error (MSE) between these two time series. Applying the algorithm on the experimental data results in a set of sensor-MSE value pair. MSE measures the average of the squares of the errors which can be used to identify deviations from the sufficient behaviour. Then, outliers of the MSE values will be detected and annotated as erroneous signals. Figure 5.22b shows the detected outliers within the building and depicts these sensors as faulty. The X-axis contains boxes for each sensor while the Y-axis represents the MSE value. The higher the MSE value is, the bigger is the error, i.e. the deviation of actual sensor measurements differs heavily from the target temperature values. As we can observe, the algorithm marks three sensors as faulty, by measuring the deviation from the standard MSE deviation. The actual sensor values for these hydraulic components differ significantly from the normal standard deviation, which may refer to a fault in the building's hydraulic system.



(a) Target and actual temperature measurements



(b) Outlier detection for target temperatures

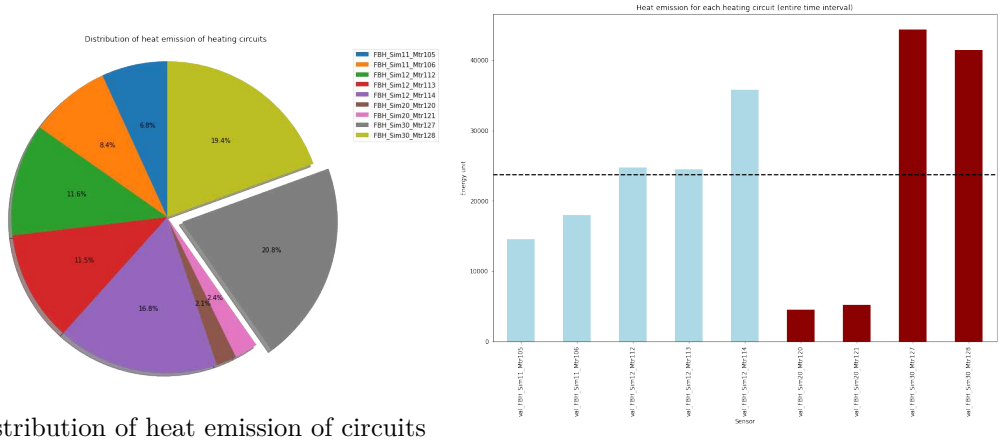
Figure 5.22: Target temperature analysis and outlier detection

Heat Emission Analysis

Heat Emission Analysis is a naive approach for measuring the emitted heat by the hydraulic components. This is done by computing the temperature difference of hydraulic components weighted by the flow rate, i.e. for each data point we subtract the return flow temperature from the flow temperature and multiple it with the rate. These values will be summed up and it builds the amount emitted heat for a specific time interval of a heating component. Due to structure of the algorithm and the available sensor types, the unit of this heat emission is simply an energy unit. Since we do not know the size of the pipes, we cannot measure the concrete energy value. However, for our fault detection purposes, knowing the exact energy value is not relevant. We only want to measure the distribution of the heat emission among hydraulic components and to find extreme values (outliers) in the calculations. Figure 5.23a shows the distribution of the heat emission of a heating circuit over two months in a pie chart. We can observe, that the component *FBH_Sim30_Mtr127* uses 20.8% of the entire energy while *FBH_Sim30_Mtr128* 19.4%. These hydraulic components are located in the same room *Sim30* which means, that the energy consumption of this room is almost the half of the entire building regarding the emitted heat. This unbalanced behavior of heat emission pattern may entail a flaw in that specific room.

To automatically analyze these results, we look for outliers by examining the deviation from the standard deviation. Figure 5.23b shows the sensors that are out of the average heat emission measurement range and marks them as faulty for enabling further analyses of these sensors. This is done by visualizing the measurements with a box plot. The black, horizontal, dashed line represents the mean of the heat emission values. An extreme deviation from the standard deviation could imply an erroneous behavior of the specific heating elements. While we already discussed the outlier of the room *Sim11*, we also found a room with extremely small energy usage over the two months. *Sim20* uses

significantly small amount of energy for heating purposes, which may be due to unused building unit or it may entail broken heating components, which should be analyzed by building experts.



(a) Distribution of heat emission of circuits

(b) Outlier detection

Figure 5.23: Heat emission analysis results

Operation Time Analysis

There are multiple ways to analyze the operation hours of the building's hydraulics. As a short recap, we consider hydraulic components to be in operation if the flow rate sensor on this heating component records a positive value, i.e. the water is moving within this hydraulic circle. To measure this in an effective way, we split the dataset into days and count the data points with a positive flow rate, which adds up to the operation time of the component on a specific day. To be able to derive relevant information from this analyses regarding possible energy loss and inefficiencies, we can either focus on single measurement points or on the distribution of operation time for the entire building. Firstly, we will consider a single measurement point, i.e. hydraulic component with the three different sensors (flow temperature, return flow temperature and flow rate), for our analysis. Figure 5.24 shows the daily operation hours of the heating component *FBH_Sim11_Mtr105* over a month. We can observe moderate or even zero operation hours on the weekends, which makes sense due to the nature of the test building (office). The black horizontal, dashed line represents the mean of the calculated operation time values. Bars colored with dark red are automatically identified outliers within the values using standard deviation measurements. This measured system component was used too much in contrast to the other days in the examined time interval. The facility manager can investigate the specific room or hydraulic component for fault detection or compare with other rooms that show also significant outliers for these days.

Another possibility to find hydraulic components with a suboptimal usage pattern is to investigate the daily distribution of operation hours for the entire building. For this, we

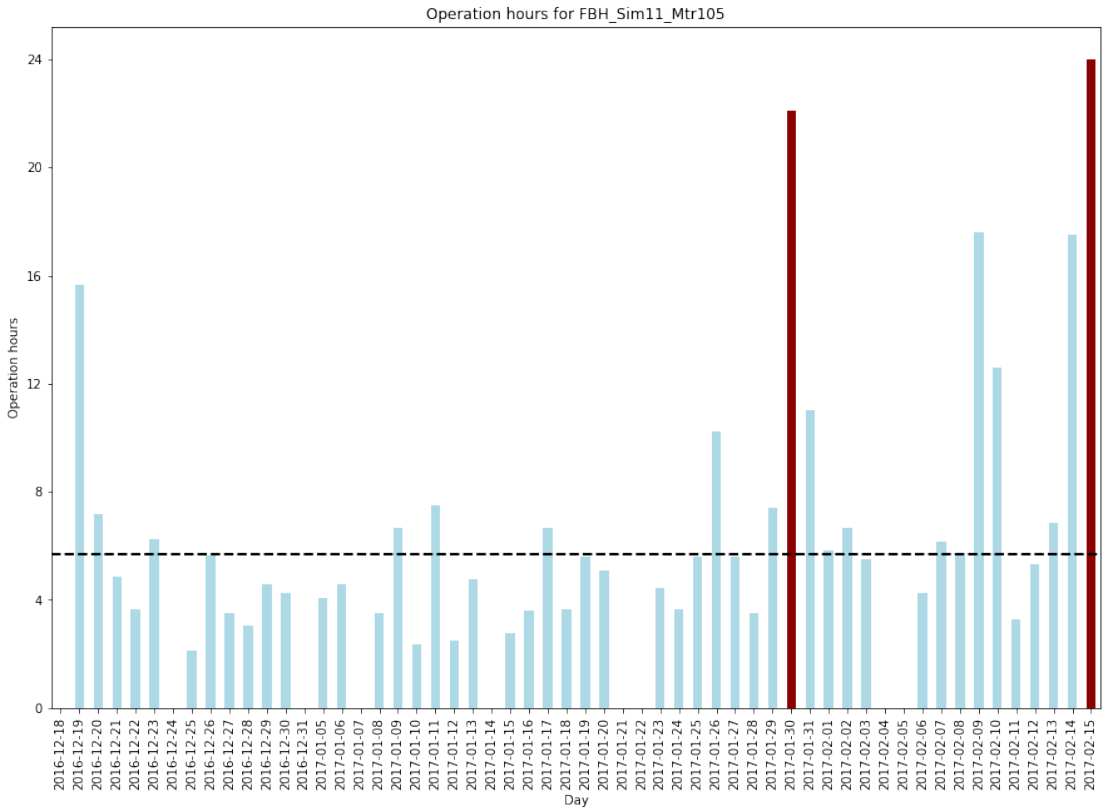


Figure 5.24: Outlier detection for a specific sensor over time

generate box plots to visualize the distribution of component usage for selected dates. Figure 5.25 shows this distribution for six consecutive weeks. On the X-axis, we can see the days of the week, while the Y-axis shows the operation time in hours. The blue boxes contain the majority of measured operation hours, and the separate dots represent outliers, that are significantly far away from the mean of the measured operation time values. Our visualization algorithm is able to identify and extract these dots and show them to the experts. This is an important mechanism, to identify hydraulic components whose usage pattern differs from the others. Furthermore, the algorithm is able to differentiate between positive and negative outliers, i.e. the dot is below the blue box (majority) or above. Our optimization (decrease) use case implies to focus only on outliers (dots) above the blue boxes since we want to detect component with a high operation time to reduce it later. We identified the dots to be of the same hydraulic circuits, *FBH_Sim30_Mtr127* and *FBH_Sim30_Mtr128*, which means, that this room is heated significantly more than other rooms, that may entail a flaw in this room.

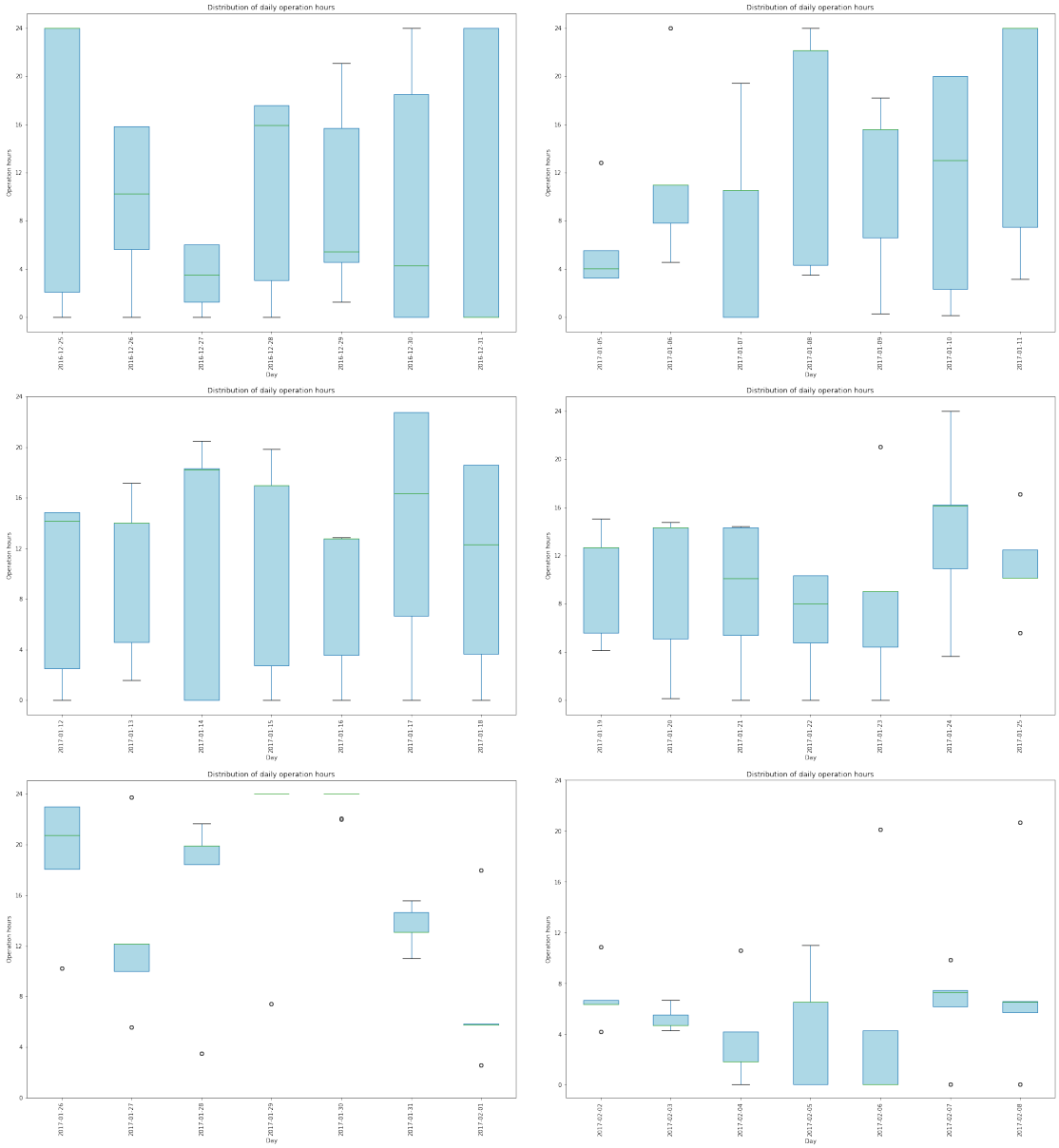


Figure 5.25: Distribution of daily operation hours over consecutive weeks

CHAPTER 6

Integration and Deployment

In this chapter, we will integrate some of the designed and implemented parts of sensor topology discovery along with the analysis methods. We will create small service instances for each task in order to provide easy deployment, testing capabilities and maintenance. We will go through the overall architecture of the BSAE and identify the necessary software components. This will be followed by the description of the components and of the frameworks used for the software integration and deployment.

6.1 Architecture

The architecture of the application will follow the principles of the Micro Service Architecture (MSA). To keep various components of the system decoupled, the tasks of the software will be divided into different responsibilities [AAE16]. The services work independently from each other and the failure of one service has no effect on the functionality of the other services. For example, the failure of the service responsible for report generation has no negative impact on the functionality of the service which is responsible for topology discovery. Each service has its own data management and is deployed in an isolated runtime environment. The communication between services or services and the user interface takes place via REST interfaces or via a message broker. This architecture ensures exchangeability of the individual components and the configurability of the entire system [NMMA16].

The loose coupling of the services in such an architecture (coupling only via messaging interfaces) allows a free choice of infrastructure components for the individual services [FL]. Thus, the most suitable programming language, database or programming style can be selected for each service. For example, if Java is the best choice for report generation, you can do it, while Python is probably more suitable for data analysis. Since the system is primarily responsible for data analysis, Python is used as the primary programming language in most services.

MSA enables easy deployment via containerization, which in an essential factor to reduce time to the market. Although, it comes with a communication overhead and increased resource needs, it increases the software's ability to scale horizontally and the ability for failure resilience [New15].

Figure 6.1 show the overview of the BSAE architecture. Two main components of the system are out of the thesis' scope: the building and the sensor data persistence. We will assume that sensor data is collected in the building, it is transferred to the cloud for further transformation, persistence and it is provided via interfaces accessible for BSAE. The *Topology Discovery Engine* is designed and implemented, but it will not be integrated into our testbed. In following sections, we will describe the software components in detail.

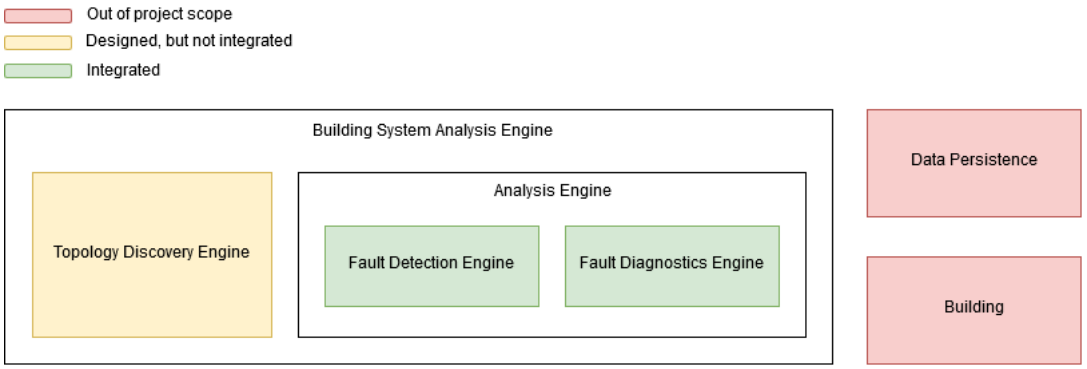


Figure 6.1: Overview of BSAE architecture

6.2 Software Components

Figure 6.2 shows the software components identified as inevitable parts of the BSAE. The system consists of eight services that are able to work independently, the persistence layer for some services and the message broker to enable asynchronous communication between service instances. The API gateway ensures a single point of access for the exposed functionalities for the experts. Software components marked with red are out of the project's scope, while components colored with yellow are designed and implemented but not integrated into our testbed. *Data Persistence* marked with red is the database hosted in the cloud that receives, manages and hosts sensor measurement data provided by the building sensor network. As mentioned in above sections, data gathering from the building's sensor network is out of the project's scope, we simple assume that the data is collected, transferred to the hosted database, stored in an appropriate manner and provided via interfaces accessible from the outside. The *API Gateway* is designed to handle requests to the BSAE but not implemented in our testbed. The *Topology Discovery Engine* is an essential part of the system. It is designed and implemented, however it is not integrated into our testbed implementation. The reason for that is the complexity of the topology discovery process and the identified restrictions of that

process. In the following sections, we will describe the integrated parts of the system in detail.

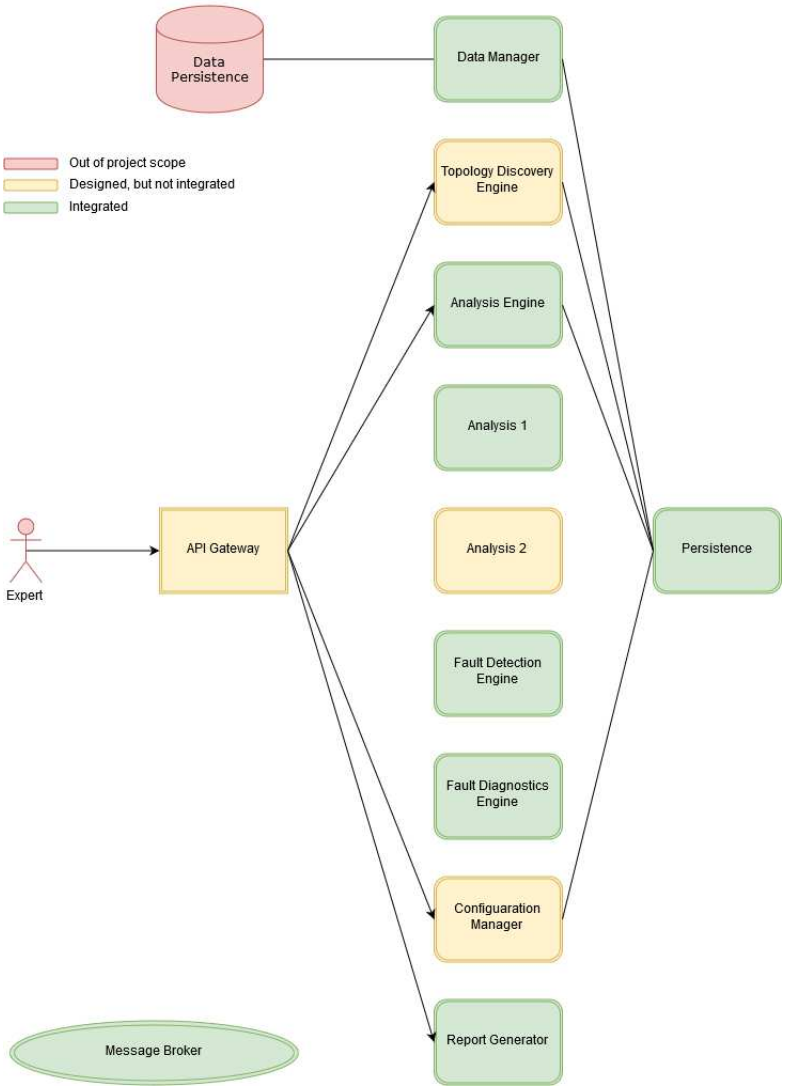


Figure 6.2: Software components of BSAE

Data Manager

The *Data Manager* is responsible for regularly retrieving data from the central sensor data collection and to provide an endpoint to access this data. It can be configured to access the cloud database directly or to load sample data from a CSV file. This enables easy testing and quick evaluation. Furthermore, it is able to schedule data fetching at regular intervals and predefined data amount. Additionally, this module is responsible

for creating unified time series, which means it is responsible to detect anomalous and missing values, and also network connectivity errors. It has to resample and prepare the dataset for further analysis. Figure 6.3 shows the class diagram of this module.

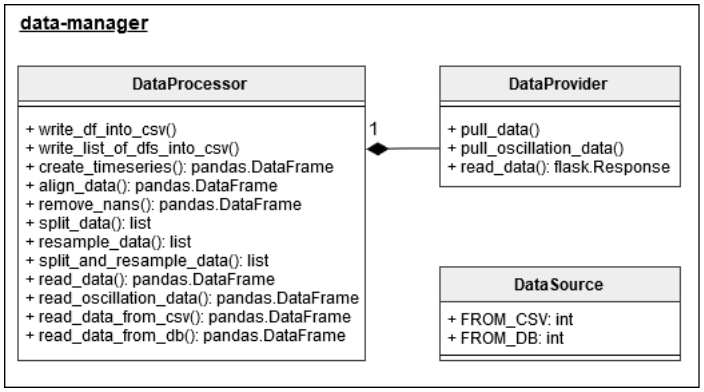


Figure 6.3: Service components of Data Manager

Analysis Engine

The *Analysis Engine* has a central data distribution role. It fetches data from the *Data Manager*, persists a new analysis object in its database and distributes data to be analyzed across the analysis components that are subscribed, i.e. are enabled to process sensor data. Its main responsibility is to orchestrate the analysis process and to keep track on history analyses. After fetching sensor data from the *Data Manager*, this module produces analysis messages on a predefined message queue. This messages will be consumed by subscribed analysis components that return analyzed objects containing the analysis results of the specific component. This messages will be redirected to the *Fault Detection Engine* that decides whether there is a fault in the analysis or not. This information is stored in the module's persistence layer. It will be forwarded to the *Fault Diagnostics Engine* which identifies the cause of the fault and returns it to this module asynchronously. After each step, this engine persists the new state of the analyses while also providing an endpoint to access history and ongoing analyses. Figure 6.4 shows the software components of this module.

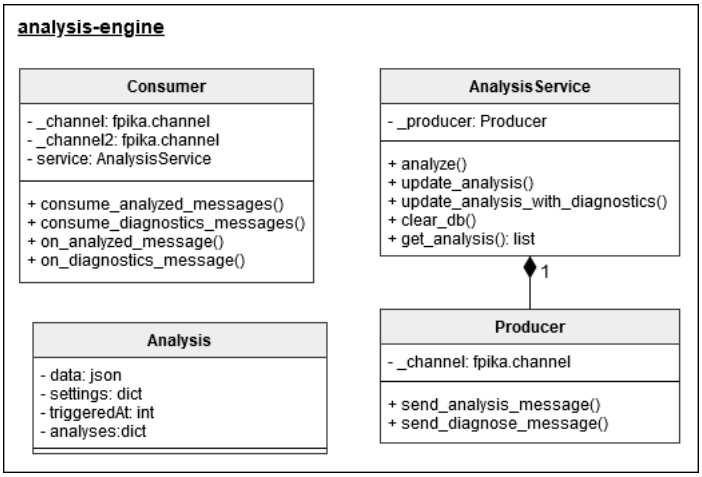


Figure 6.4: Service components of Analysis Engine

Analysis Component

Individual *Analysis Components* are responsible for the analysis of the dataset from a specific aspect. The idea is to encapsulate analysis functionalities into independent services that analyze the dataset fragment for a specific target. These components are designed to provide the ease of maintainability and replaceability. They subscribe to a predefined message queue in order to listen to incoming analysis requests. This uniform interface enables the design and development of different analysis modules without the need to modify other parts of the system or even without downtime of the BSAE. This approach of splitting the analysis process into smaller parts comes with some overhead in implementation and resource requirements, but to create a new *Analysis Component*, one should only implement the messaging interface and provide the functionality with a free choice of technology stack and implementation details. In our testbed, we will implement the *Oscillation Analyzer* as an instance of the *Analysis Component* that subscribes to the analysis message queue, listens to incoming messages, and after processing the data it produces the appropriate analysis object that will be consumed by the *Analysis Engine*. Figure 6.5 shows the class diagram of this particular *Analysis Component*.

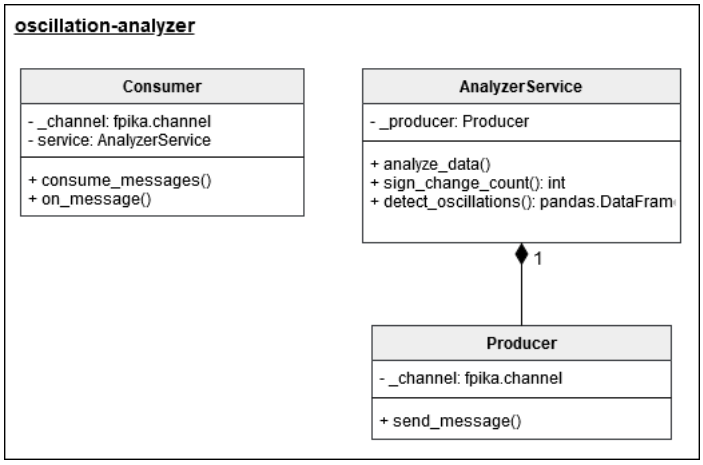


Figure 6.5: Service components of Oscillation Analyzer

Fault Detection Engine

The *Fault Detection Engine* is triggered by the *Analysis Engine* and is responsible to identify faults of analyzed data based on thresholds, history analyses and the discovered sensor topology. Depending on the type of the analysis, it identifies abnormal data points or time intervals. In order to provide this identification functionality, expert knowledge is required to define the thresholds, and fault characteristics. In our testbed, we will restrict our fault detection capabilities to predefined thresholds, that are bound to the specific types of analysis. Due to the lack of expert knowledge, we will identify threshold based on empirical experiments. These thresholds can be modified easily, and due to the software architecture it can be changed without downtime or unnecessary modifications on other parts of the software.

Fault Diagnostics Engine

The *Fault Diagnostics Engine* is simple, rule-based diagnostics service. It is only a proof-of-concept to enable the evaluation of an entire FDD system. Simple rules are defined in an *if this, then that* approach, i.e. we define pairs of faults and causes. This module is triggered by the analysis engine with the analysis object after fault detection. This message is published on a predefined message queue that will be processed and completed with the cause of the fault if possible. Due to the lack of expert knowledge, we define arbitrary rules based on our experiments. Figure 6.6 depicts the components of this engine.

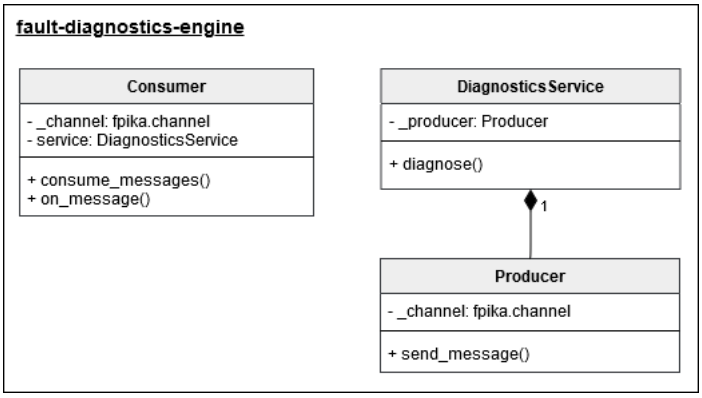


Figure 6.6: Service components of Fault Diagnostics Engine

Report Generator

The *Report Generator* is responsible for creating a human-readable representation of building system analyses. It provides an interface that is accessible for building experts in order to generate a visual representation of an analysis. It creates a PDF file containing meta-information of the analysis along with the FDD process results with graphs. Figure 6.7 shows the classes and methods of the generator module.

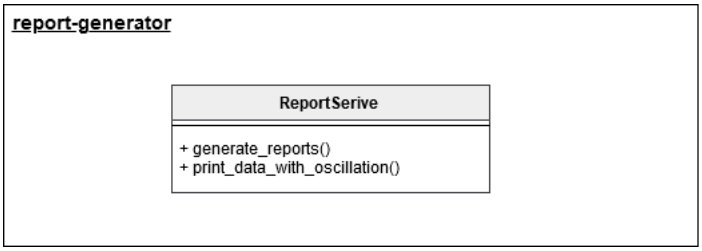


Figure 6.7: Service components of Report Generator

6.3 Communication

As suggested by Fowler et al. [FL], the services should communicate in synchronous or asynchronous manner. Depending on the type of the use case, we will use REST interfaces for synchronous and a message broker for asynchronous communication. Flask [1] will be used to create a simple web application structure. To define the REST API endpoints in a decoupled, standardized manner, we will implement the OpenAPI specification [2] using a Python library Connexion [3]. This approach enables us to define REST API endpoints in a standardized format using the YAML file format, as shown in the Appendix, and it

[1] https://www.palletsprojects.com/p/flask/
[2] https://swagger.io/specification/
[3] https://github.com/zalando/connexion

allows decoupling the API specification from business logic. Furthermore, the framework supports API versioning, which is essential for a MS architecture, because changes to a service always require a new version of the respective service. For asynchronous communication, we will use a message broker, RabbitMQ [4], as shown in the Appendix. RabbitMQ is one of the most popular open source message brokers that provides a high throughput and low latency asynchronous messaging tool which is easy to deploy and scale. We aim to use this type of messages everywhere it is possible. Due to the nature of the Micro Service Architecture, the communication is very complex but fault tolerant. Figure 6.8 shows the sequence diagram of the building analysis process.

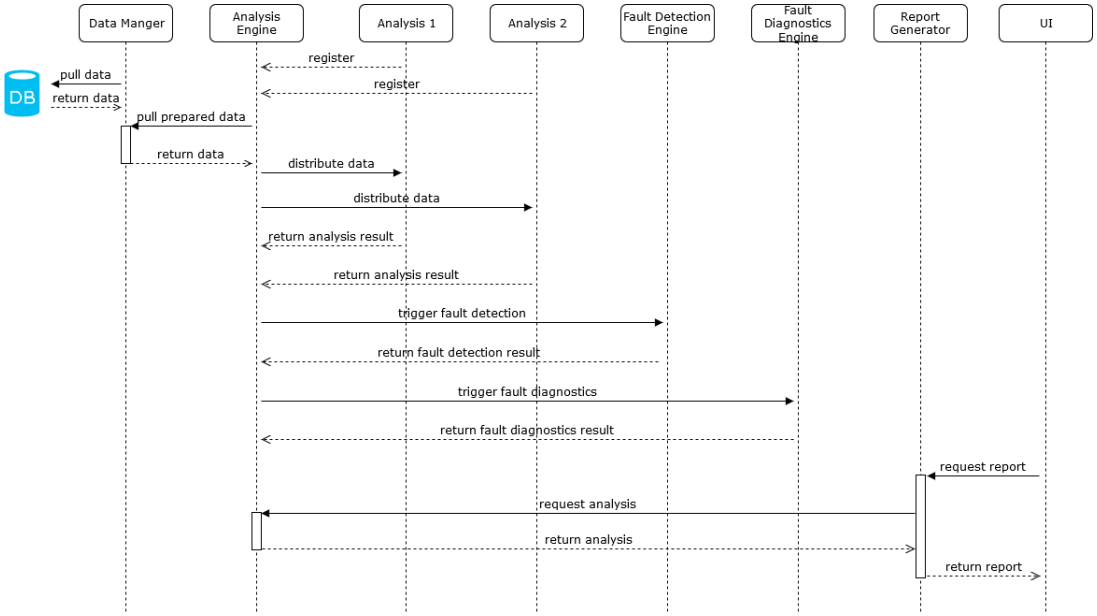


Figure 6.8: Process flow of building analysis

6.4 Deployment

Due to the nature of our problem domain, i.e. analysis of numerous buildings or even building fleets, BSAE should be ready to be deployed in a cloud environment. These environments provide on-demand computer power and resources, enabling flexibility, fault tolerance, while also reducing engineering effort in infrastructure maintenance. Software designed and developed with Micro Service Architecture is naturally suitable to be deployed in the cloud. The components of the system will run in Docker [5] containers and they will be orchestrated with Docker Compose [6]. Docker performs operating-system-level virtualization, which enables to deliver software packages in an isolated environment.

[4] https://www.rabbitmq.com/
[5] https://www.docker.com/
[6] https://docs.docker.com/compose/

The containerization of software components makes the deployment less complicated and enables the use of several cloud services to deal with problems of a cloud deployment.

CHAPTER 7

Conclusion

7.1 Summary

The aim of this thesis was to identify optimization potentials of building's energy consumption based on hydraulic meter data with minimal manual effort. The initial step of the thesis was the determination of use cases in the problem domain as we specified requirements and necessary conditions of the system. We identified three use case groups covering different domain abstraction levels: *Installation*, *Operation of building facility* and *Operation of building fleet*. Although, we defined and described use cases for Installation and Operation of building fleet for sake of completeness, this work focused on detecting building system faults and optimization potentials on building level. This was followed by an extensive literature study about the state-of-the-art solutions to similar problems in the problem space. Recently built building with advanced Building Management Systems (BMS) collect a large amount of sensor data and usually provide fault detection features. In contrast, older building normally have only a few sensors to control their hydraulic system and thus, the analysis such buildings is limited and they often operate on unnoticed, suboptimal configuration. Therefore, the goal of this thesis was to identify flaws in these buildings using non-invasive sensor networks on the hydraulic system. The knowledge about the spatial arrangement of the equipped sensor is inevitable for the analysis. Hence, we identified necessary steps to detect the topology of the sensor network as little manual effort as possible. This resulted in four necessary steps to identify the topology: *classification of sensor measurements*, *establishment of sensor groups at same location*, *establishment of clusters of sensor nodes in same building segment* and *identification of sensor sequences within sensor node clusters*. In the first step, we compared K-NN classification and K-Means clustering in order to identify sensor types for sensor measurements. With some preprocessig and by applying K-Means clustering, we could correctly identify the measurement types for the time series. This was followed by creating sensor groups of each sensor type (flow temperature, return

flow temperature and flow rate), for which we designed and successfully implemented a correlation based algorithm. Detecting sensor sequences on building level failed due to independently operating hydraulic circuits. Therefore, as the next step, we established clusters of sensor nodes using Empirical Mode Decomposition (EDM) and DBSCAN clustering. Results show that we can establish more meaningful inter-device relationships when we look at different frequency levels of signal rather than correlating raw data directly. In order to detect sensor sequences within sensor node clusters, we made use of hierarchical clustering and Dynamic Time Warping (DTW). Although, we could identify connections between sensors in the same room, our approach failed to identify connections between rooms, which resulted in introducing an initial heating phase for the topology discovery, the *Footprint Method*. In this method, we heat up hydraulic components in a sequence within a hydraulic circle, and so we are able to detect room connections correctly. In conclusion, we identified possible steps to detect sensor topology with as little engineering effort as necessary considering limitations of sensor types and sensor placement. Knowing the spatial arrangement of equipped non-invasive sensors, we were able to identify analysis approaches that can help to reduce the energy consumption of the building. We designed and developed following building analysis methods: *oscillation detection*, *target temperature analysis*, *heat emission analysis* and *operation time analysis*. Due to the lack of annotated faulty data, we constructed naive algorithms, that can analyze the data from different aspects and can identify outliers of the analysis. With the oscillation detector, we were able to detect abnormal signal behavior on specific hydraulic components that can be cased by erroneous component operation. Remaining analyses allowed the detection of hydraulic components with significant deviation from other components and room by analysing sensor data and detecting outliers. Our approaches of topology discovery and building analysis were tested on measurement data from an occupied office building. In order to integrate identified analysis approaches into a real-world application, we developed a Micro Service Architecture (MSA) based software. After identifying necessary services, we integrated data management including preprocessing and one analysis component into the software. We decided to use MSA since these services work independently from each other and the failure of one service has no effect on the functionality of the other services. This ensures exchangeability of the individual components and the configurability of the entire system even without downtime while also providing the ease of scalability.

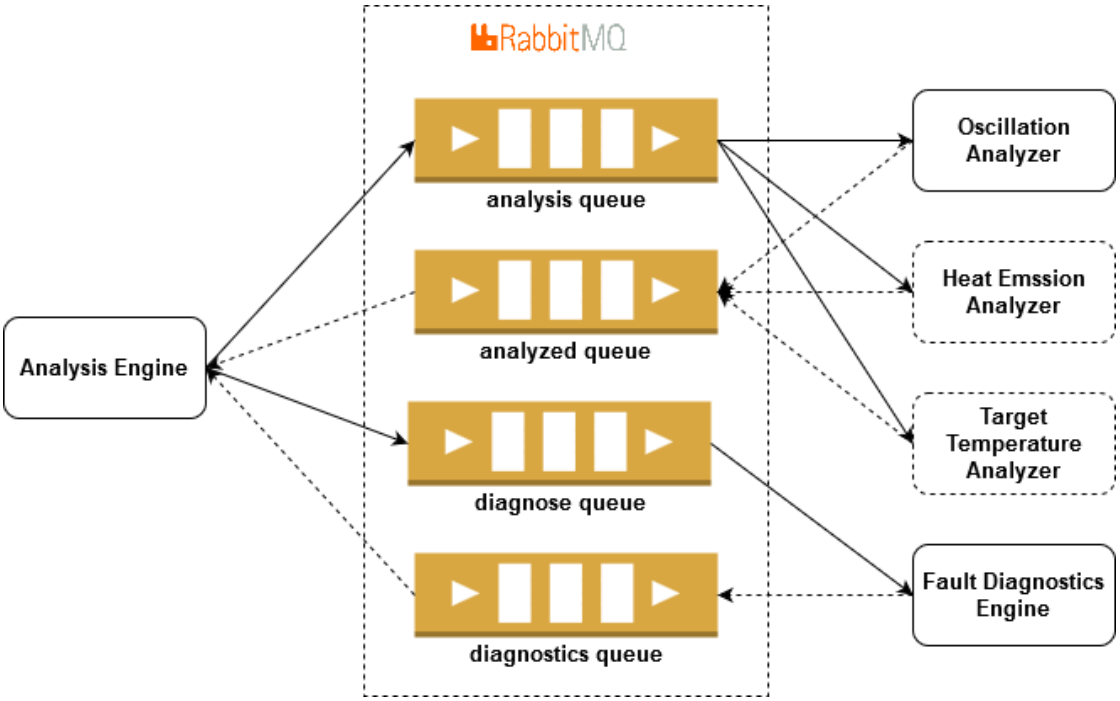
7.2 Future Work

There is wide spectrum of improvement possibilities on this work. As mentioned above, we could only identify senor typologies with certain restrictions, that require manual effort. In order to further reduce costs and engineering effort, future work should focus on reducing or even removing these limitations by improving proposed approaches or by modifications of the identified discovery process. With an increased involvement of building expert knowledge, more accurate and specific building analysis algorithms should be designed and implemented. Additionally, annotated faulty sensor data would

increase analysis possibilities. Future work should also focus on identifying optimization possibilities on building fleet, i.e. analysis on groups of buildings, since this domain level has not been studied in this thesis.

Appendix

Asynchronous communication



Synchronous communication

data_manager.yaml

```
1  swagger: "2.0"
2
3  info:
4    title: Data Manager API
5    version: "1.0"
```

```yaml

basePath: /1.0

schemes:
  - http
consumes:
  - application/json
produces:
  - application/json

paths:
  /data/pull:
    post:
      operationId: api.data.pull_data
      summary: Trigger Data Manager to pull data
      responses:
        '204':
          description: Successfully triggered Data Manager to
            ↪ pull data
  /data/pull/oscillation:
    post:
      operationId: api.data.pull_oscillation_data
      summary: Trigger Data Manager to pull oscillation data
      responses:
        '204':
          description: Successfully triggered Data Manager to
            ↪ pull oscillation data
  /data:
    get:
      operationId: api.data.read
      summary: Get sensor data for a specified time interval
      parameters:
        - in: query
          name: start_date
          type: string
          description: Timestamp to define starting point in
            ↪ time
          default: ''
        - in: query
          name: end_date
          type: string
          description: Timestamp to define ending point in time
          default: ''
```

```
        - in: query
          name: sensor
          type: string
          description: Name of sensor
          default: ''
      responses:
        '200':
          description: Successfully read sensor data from
           ↪ database
```

analysis_engine.yaml

```
swagger: "2.0"

info:
  title: Analysis Engine API
  version: "1.0"

basePath: /1.0

schemes:
  - http
consumes:
  - application/json
produces:
  - application/json

paths:
  /ping:
    get:
      operationId: api.analysis.ping
      summary: Check endpoint accessibiltiy
      responses:
        '204':
          description: Service is online
  /cleardb:
    get:
      operationId: api.analysis.clear_db
      summary: Clear analyses from mongodb
      responses:
        '204':
          description: Task started
```

```
31    /analyze:
32      post:
33        operationId: api.analysis.analyze
34        summary: Trigger Analysis Engine to analyze data
35        parameters:
36          - name: options
37            in: body
38            required: true
39            schema:
40              $ref: '#/definitions/Options'
41        responses:
42          '204':
43            description: Successfully triggered Analysis Engine
                ↪ to analyze data
44    /analyses:
45      get:
46        operationId: api.analysis.get_analyses
47        summary: Retrives all analyses in the system
48        responses:
49          '200':
50            description: Successfully retrived all analyses from
                ↪ the system
51
52  definitions:
53    Options:
54      type: object
55      properties:
56        start_date:
57          type: string
58        end_date:
59          type: string
60        components:
61          type: array
62          items:
63            type: string
64        sensors:
65          type: array
66          items:
67            type: string
```

oscillation_analyzer.yaml

```yaml
swagger: "2.0"

info:
  title: Oscillation Analyzer API
  version: "1.0"

basePath: /1.0

schemes:
  - http
consumes:
  - application/json
produces:
  - application/json

paths:
  /ping:
    get:
      operationId: api.analysis.ping
      summary: Check endpoint accessibiltiy
      responses:
        '200':
          description: Service is online
```

fault_diagnostics_engine.yaml

```yaml
swagger: "2.0"

info:
  title: Fault Diagnostics Engine API
  version: "1.0"

basePath: /1.0

schemes:
  - http
consumes:
  - application/json
produces:
  - application/json
```

```
15
16 paths:
17   /ping:
18     get:
19       operationId: api.diagnostics.ping
20       summary: Check endpoint accessibiltiy
21       responses:
22         '200':
23           description: Service is online
```

report_generator.yaml

```
1  swagger: "2.0"
2
3  info:
4    title: Report Generator API
5    version: "1.0"
6
7  basePath: /1.0
8
9  schemes:
10   - http
11 consumes:
12   - application/json
13 produces:
14   - application/json
15
16 paths:
17   /ping:
18     get:
19       operationId: api.report.ping
20       summary: Check endpoint accessibiltiy
21       responses:
22         '204':
23           description: Service is online
24   /generate:
25     post:
26       operationId: api.report.generate_reports
27       summary: Generates analysis reports
28       responses:
29         '204':
30           description: Service is online
```

List of Figures

List of Algorithms

Bibliography

[AAE16] N. Alshuqayran, N. Ali, and R. Evans. A systematic mapping study in microservice architecture. In *2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)*, pages 44–51, Nov 2016.

[BC94] Donald J. Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, AAAIWS'94, page 359–370. AAAI Press, 1994.

[BNGA14] Bharathan Balaji, Balakrishnan Narayanaswamy, Rajesh Gupta, and Yuvraj Agarwal. Data driven investigation of faults in HVAC systems with model, cluster and compare (mcc). BuildSys 2014 - Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings, 11 2014.

[CM13] Sterling R. Blanes L.M. Howley M. Costa, A. and Keane M.M. A swot framework to investigate the integration between building management systems and fault detection and diagnosis tools. Proceedings of the 5th International Conference of Applied Energy. ICAE 2013., 2013.

[CPLW00] Drury Crawley, Curtis Pedersen, Linda Lawrie, and Frederick Winkelmann. Energyplus: Energy simulation program. *Ashrae Journal*, 42:49–56, 04 2000.

[DSZ15] Konrad Diwold, Matthias Stifter, and Paul Zehetbauer. Meter communication and measurement based topology identification for low voltage networks. *EDST*, 2015.

[dtw07] *Dynamic Time Warping*, pages 69–84. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[EKSX96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

[FHJ+16] Stephen Frank, M. Heaney, Xin Jin, Joseph Robertson, Howard Cheung, Ryan Elmore, and Gregor Henze. Hybrid model-based and data-driven fault detection and diagnostics for commercial buildings. National Renewable Energy Lab. (NREL), Golden, CO (United States), 08 2016.

[FL] Martin Fowler and James Lewis. Microservices a definition of this new architectural term. `https://martinfowler.com/articles/microservices.html`. Last accessed: 2020-02-03.

[FOT+13] R. Fontugne, J. Ortiz, N. Tremblay, P. Borgnat, P. Flandrin, K. Fukuda, D. Culler, and H. Esaki. Strip, bind, and search: A method for identifying abnormal energy consumption in buildings. In *2013 ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 129–140, April 2013.

[GKA19] Jasleen Grewal, Martin Krzywinski, and Naomi Altman. Markov models — hidden markov models. *Nature Methods*, 16:795–796, 09 2019.

[GWLW13] Ying Guo, Josh Wall, Jiaming Li, and Sam West. Intelligent model based fault detection and diagnosis for HVAC system using statistical machine learning methods. ASHRAE Winter Conference, 01 2013.

[HSL+98] Norden Huang, Z Shen, S.R. Long, M.L.C. Wu, H.H. Shih, Quanan Zheng, N.C. Yen, Chi-Chao Tung, and H.H. Liu. The empirical mode decomposition and the hilbert spectrum for nonlinear and non-stationary time series analysis. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 454:903–995, 03 1998.

[Kam09] Chandrika Kamath. *Scientific Data Mining - A Practical Perspective.* SIAM, 01 2009.

[KB05] Srinivas Katipamula and Michael Brambley. Methods for fault detection, diagnostics and prognostics for building systems - a review part I. *HVAC and R Research*, 11, 04 2005.

[KCCC13] Imran Khan, Alfonso Capozzoli, Stefano Corgnati, and Tania Cerquitelli. Fault detection analysis of building energy consumption using data mining techniques. *Energy Procedia*, 42:557–566, 12 2013.

[KMN+02] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. An efficient k-means clustering algorithm: analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):881–892, July 2002.

[Kot07] Sotiris Kotsiantis. Supervised machine learning: A review of classification techniques. *Informatica (Ljubljana)*, 31, 10 2007.

[LCZD14] Mathieu Le Cam, R. Zmeureanu, and Ahmed Daoud. Application of data mining techniques for energy modeling of HVAC sub-systems. 05 2014.

[Lia05] T. Liao. Clustering time series data — a survey. *Pattern Recognition*, 38:1857–1874, 11 2005.

[LPS13] X. Li, H. V. Poor, and A. Scaglione. Blind topology identification for power systems. In *2013 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pages 91–96, Oct 2013.

[LZHS16] Dan Li, Yuxun Zhou, Guoqiang Hu, and Costas Spanos. Fault detection and diagnosis for building cooling system with a tree-structured learning method. *Energy and Buildings*, 127, 06 2016.

[MC87] Glenn W. Milligan and Martha C. Cooper. Methodology review: Clustering methods. *Applied Psychological Measurement*, 11(4):329–354, 1987.

[MF15] G. Mantovani and L. Ferrarini. Temperature control of a commercial building with model predictive control techniques. *IEEE Transactions on Industrial Electronics*, 62(4):2651–2660, April 2015.

[New15] Sam Newman. *Building Microservices*. O'Reilly Media, Inc., 1st edition, 2015.

[NMMA16] Irakli Nadareishvili, Ronnie Mitra, Matt McLarty, and Mike Amundsen. *Microservice Architecture: Aligning Principles, Practices, and Culture*. O'Reilly Media, Inc., 1st edition, 2016.

[PM00] Dan Pelleg and Andrew W. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, page 727–734, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[Pop14] Shraddha K. Popat. Review and comparative study of clustering techniques. volume 5 (1) of *International Journal of Computer Science and Information Technologies*, pages 805–812, 2014.

[PVG+12] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, Edouard Duchesnay, and Gilles Louppe. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 01 2012.

[Ros83] Bernard Rosner. Percentage points for a generalized ESD many-outlier procedure. *Technometrics*, 25(2):165–172, 1983.

[SA13] R. Sathya and Annamma Abraham. Comparison of supervised and unsupervised learning algorithms for pattern classification. *International Journal of Advanced Research in Artificial Intelligence*, 2, 02 2013.

[SHPV11] Anika Schumann, Jer Hayes, Pascal Pompey, and Olivier Verscheure. Adaptable fault identification for smart buildings. In *Proceedings of the 7th AAAI Conference on Artificial Intelligence and Smarter Living: The Conquest of Complexity*, AAAIWS'11-07, page 44–47. AAAI Press, 2011.

[STD+18] T. Sauter, A. Treytl, K. Diwold, D. Molnar, D. Lechner, L. Krammer, B. Derler, C. Seidl, and F. Wenig. Getting fit for the future: Optimizing energy usage in existing buildings by adding non-invasive sensor networks. In *2018 IEEE 27th International Symposium on Industrial Electronics (ISIE)*, pages 963–968, June 2018.

[SZ12] Stefan Soucek and Gerhard Zucker. Current developments and challenges in building automation. *Elektrotechnik und Informationstechnik*, Volume 129:278–285, 06 2012.

[WB09] Z. Wang and A. C. Bovik. Mean squared error: Love it or leave it? a new look at signal fidelity measures. *IEEE Signal Processing Magazine*, 26(1):98–117, Jan 2009.

[WGWW11] Sam West, Ying Guo, Rosalind Wang, and Joshua Wall. Automated fault detection and diagnosis of HVAC subsystems using statistical machine learning. 12th International Conference of the International Building Performance Simulation Association, 01 2011.

[ZLZ+17] Shichao Zhang, Xuelong Li, Ming Zong, Xiaofeng Zhu, and Debo Cheng. Learning k for knn classification. *ACM Trans. Intell. Syst. Technol.*, 8(3), January 2017.

[ZMH+14] Gerhard Zucker, Jasmine Malinao, Usman Habib, Thomas Leber, Anita Preisler, and Florian Judex. Improving energy efficiency of buildings using data mining technologies. IEEE International Symposium on Industrial Electronics, 06 2014.