MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„Behaviour Pattern Recognition Based Context-Aware Access Control in Smart Home Environments“

verfasst von / submitted by

Ivo Vidović BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2021 / Vienna, 2021

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 066935
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Medieninformatik UG2002
Betreut von / Supervisor:	Univ.-Prof. Dipl.-Math. Dr. Peter Reichl, Privatdoz.

Acknowledgements

This thesis would not have been possible without the aid of Professor Peter Reichl and my supervisor Nemanja Ignjatov. Their help and encouragement at every step of the work were instrumental in getting this work through the finish line. I would also like to thank my family for always supporting me whenever I needed them, allowing me to focus on my work. Furthermore, my friends and colleagues, who have shown understanding for my lower availability and many absences, deserve an acknowledgement for their patience with me. I thank all of the named people from the bottom of my heart.

Abstract

The everyday schedule and habits of each person result in a sequence of actions that can be observed, which can also be referred to as their behaviour. An in-depth context analysis is performed on a smart home environment to create a context-aware solution for recognizing patterns in the behaviour of the users and the system. These patterns are made available for access control, which makes it possible for the access control mechanism (ACM) to include the knowledge of the usual behaviour found in the system during the access control decision. The central questions of this thesis are how behaviour can be modeled as behaviour patterns in a smart home environment, how the modeled behaviour can be used in access control, and how long the learning time of these behaviour patterns is. The behaviour engine is introduced as a smart home component responsible for determining behaviour and an extensive evaluation is performed against it to confirm the results of this thesis. The evaluations are partly based on CosyHome [25], which is a big dataset created in a case study as part of this work.

Kurzfassung

Jede Person hat alltägliche Angewohnheiten und Terminpläne, die in einer Sequenz von erkennbaren Aktionen resultieren, was auch als deren Verhalten betrachtet werden kann. In einer heimautomatisierten Umgebung wird eine ausführliche Kontextanalyse durchgeführt um Muster in dem Verhalten der Benutzer und des Systems zu erkennen. Diese Muster werden dem Zugriffskontrollmechanismus der Umgebung zur Verfügung gestellt, wodurch dieser in der Lage ist auf das Wissen über das übliche Verhalten im System zuzugreifen. Die zentralen Fragen dieser wissenschaftlichen Arbeit sind wie Verhalten als Verhaltensmuster in einer heimautomatisierten Umgebung modelliert werden kann, wie das modellierte Verhalten in der Zugriffskontrolle eingebunden werden kann und wie lang die Anlernzeit der Verhaltensmuster ist. Die behaviour engine wird als heimautomatisierte Komponente zum Erkennen von Verhaltensmustern eingeführt und eine ausführliche Evaluation wird durchgeführt um die Ergebnisse dieser Arbeit zu verifizieren. Die Evaluationen basieren teilweise auf dem CosyHome Datensatz [25], welcher im Rahmen dieser Arbeit in einer Fallstudie erzeugt worden ist.

Contents

1. Introduction

Every person has habits or a daily schedule that their everyday activities are built around and a huge part of their actions throughout the day take place in their homes. The repeated actions of residents inside their home environment give insight into their daily behaviour and enable specific estimations of what and when an inhabitant will perform an action [10]. Many examples of easily estimable behaviour in a home environment are quickly determined such as the time a resident leaves the home or when the residents prepare their meals.

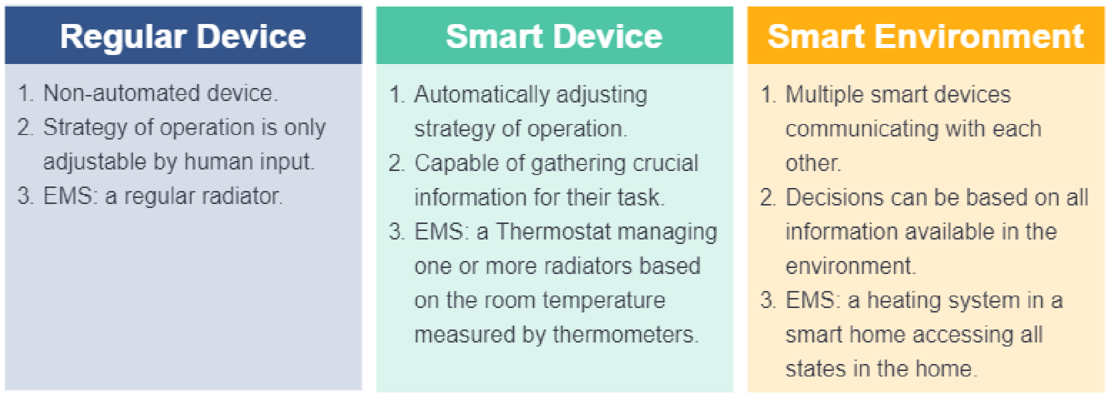
Having access to the expected behaviour offers new possibilities for smart home applications such as a smart coffee machine preparing a coffee for residents to drink before they go to work. Furthermore, application domains in smart home systems can improve their service based on the behaviour found in the system. A smart heating system automatically keeps the usually measured temperature at the home, i.e. does not heat while all residents are absent and only heats up shortly before the first resident returns from school or work. Almost every form of home automation can benefit from the knowledge of how the user will act or what will happen in the system by adjusting their operational strategy to the events that will happen before they happen.

The contextual information this work mainly focuses on is the behaviour of the user and the behaviour commonly seen in the smart home and the application domain this work mainly focuses on is the security of a smart home environment. Security and safety are important factors in home environments since the homeowner needs to be secure and safe in his own place [63] and has to be ensured that the property and privacy are protected. When the security system of a smart house can approximately estimate what the user does at a certain time, it is possible to provide an additional layer of security based on that knowledge.

1.1. Smart Environments

Intelligent systems make very sophisticated and optimized decisions based on the heterogeneous information gathered by various internal sources such as smart sensors in a smart home [85], and external sources such as a server providing traffic information to smart cars [67]. Figure 1.1 displays the differences between a regular device, a smart device, and a smart environment using an energy management system (EMS) as an example due to its significance for home environments [35].

In an Internet of Things (IoT) environment like a smart home, there are various interconnected devices present, which are able to exchange information with each other [101]. These so-called smart devices can get access to a lot of heterogeneous information



Regular Device	Smart Device	Smart Environment
1. Non-automated device. 2. Strategy of operation is only adjustable by human input. 3. EMS: a regular radiator.	1. Automatically adjusting strategy of operation. 2. Capable of gathering crucial information for their task. 3. EMS: a Thermostat managing one or more radiators based on the room temperature measured by thermometers.	1. Multiple smart devices communicating with each other. 2. Decisions can be based on all information available in the environment. 3. EMS: a heating system in a smart home accessing all states in the home.

Figure 1.1.: Smart Environment comparison with EMS as an example

gathered by the other devices in the system and then use that information as context for their own service, which is typical for an IoT environment [73]. Many examples of useful and interesting smart home applications and smart devices either rely on communication with other devices or excel once they get access to more information than they can obtain only by themselves such as an EMS [5] or a smart kitchen [59].

Smart environments provide the optimal ground for context-aware applications by offering a huge amount and variety of information. The context model of a context-aware application set in a smart home environment can be very extensive and offer all the data produced by smart sensors and devices [71] for the application in a meaningful and effective way. Context-aware applications excel the more information is accessible while choosing a strategy for carrying out a task or for offering a service, which allows them to improve the quality of the service [1] or the quality of the result for the task significantly [32].

1.2. Security Concerns

Connecting the devices of a home to a network adds the security concern of enabling external and unwanted access to control the devices of the system [81]. In the attack pattern known as "malicious outsider" [62], a system is remotely accessed without the knowledge of the usual behaviour while the attacker pretends to be a user of the system, which is called impersonation of the user [53].

If a user can successfully be impersonated, a system usually becomes very vulnerable, which is especially a problem when sensitive data is at risk, which is the case in environments such as a smart home [53]. An application capable of detecting suspicious behaviour successfully can preserve the integrity of the system against the malicious outsider attack pattern and notify the user to take proper action to keep the environment secure [81]. This work integrates behaviour pattern recognition into an access control system of a smart home to enhance its security by providing knowledge of the usually observed behaviour to the system.

The aspect in the security of a smart home this work focuses on is authorization, which is the process of deciding whether an access request can be granted based on the configured access rights in the system [11]. This mechanism is necessary to ensure that the incoming control inputs in the system are valid requests, meaning that these requests try to access a resource the user is allowed to access. Authorization is sometimes referred to as access control and in the scope of this work, the terms access control and authorization are treated as synonyms.

1.3. Research Questions and Key Contributions

This master thesis aims to analyze the events in a smart home to determine patterns in the behaviour and enhance access control in a smart home environment based on these context-aware behaviour patterns. Three research questions are formulated to set the scope of the work and to provide well-formulated goals. This section is listing these research questions and is giving a brief overview of the contribution of this work to find answers to these questions.

1. How can the daily routine and the habits of an inhabitant of a smart home be translated into a context-aware behaviour pattern?

Behaviour patterns serving as prediction models for behaviour are the central component of this work and the essential part of the first research question. The goal of this research question is to automatically analyze the actions of a user or any other means of determining behaviour in the underlying smart home environment and to determine regularities. Furthermore, these behaviour patterns need to be usable as estimation models representing the expected behaviour to evaluate new actions of the users and events in the environment.

Behaviour patterns need to be adapted every time new actions or events are observed to properly represent the behaviour after the newly gained information. This is necessary due to the nature of behaviour, which causes it to be only estimable over a period of time [10].

In this work, behaviour patterns are created based on a context model, which has been created as the result of a comprehensive context analysis of a smart home environment. Furthermore, the relationships between the behaviour patterns are introduced as groups, a supporting feature for estimating behaviour more precisely.

2. How can context-aware behaviour pattern recognition be used in a smart home environment for access control?

In the first research question, the behaviour patterns have already been defined as estimation models for behaviour in the smart home environment. The goal of the second research question is to integrate the behaviour estimations into the access control model of the smart home.

Including the behaviour of the users and the system in the process of access control is challenging due to their seemingly incompatible temporal characteristics. Behaviour can only be approximately predicted over time due to its repetitive nature [10] while an access control request has to be processed and enforced in a timely manner at the point of the request [45]. In this thesis, this problem has been overcome by separating the process of creating behaviour patterns and evaluating access control requests completely.

A crucial task for the developers of context-aware applications is to make them understandable and comprehensible for the target user group due to their complex and non-transparent nature [1]. This needs to be especially considered in this thesis due to the additional transparency challenges of IoT environments [69]. The requirement of providing transparency to the users adds a non-technical perspective to the research question.

3. How long is the learning time for a behaviour pattern?

The last research question is formulated to set a goal for the evaluation of the introduced solution design, complementing the other two research questions, which set goals and requirements for the design and implementation. The answer to this research question gives insight into how much data is needed for the behaviour patterns to stabilize and how the learning time is influenced by the introduced features.

The evaluations of this work are split into two groups by which type of data the evaluation is based on. One group is based on artificially generated data in parameterized test scenarios, exploiting the advantages of quickly generated test data to evaluate how the solution of this thesis performs under certain circumstances [38]. The key evaluations of answering this research question are part of the other group, which is based on the real-world dataset CosyHome [25] that has been created in a case study as part of this work.

The requirements for a dataset to be useful in the scope of this thesis are to feature multiple users and devices in at least one smart home, clear relationships indicating which smart homes the devices and users belong to and events over the course of at least eight weeks with timestamps. Different types of households are not equal in regard to how interesting they are for the evaluation. Therefore, the dataset should contain at least one smart home with a dual-income family, which is the most interesting type of household in smart home environments due to their daily structure around work, school, and leisure activities [27]. Optional features that can be used to gain interesting insight are the relationships between the users and the devices in their respective homes and information about the daily routines of the users.

1.4. Structure of this Master Thesis

There are three big topics related to this thesis, namely:

- Smart home
- Context-awareness
- Access control

The master thesis leads with one chapter of theoretical introduction into the fundamentals of each of the three central topics related to this work. Chapter 2 is about the smart home and its characteristics as an IoT environment, introducing the setting of the thesis to discuss the requirements, restrictions, and possibilities in such an environment. The second theoretical chapter, Chapter 3, introduces the terms context and context-awareness and describes the crucial details that need to be considered when designing a context-aware application. Chapter 4 is the third and last theoretical chapter and provides an understanding of the important aspects of access control and describes various forms that have been implemented over the years. Each of these chapters gives a basic introduction into its respective topic and provides a deep enough knowledge to understand the solution design presented in this thesis and some insight into the design choices made in later chapters.

After the background knowledge has been established, the solution design of this work is presented and explained in detail in Chapter 5. Every feature introduced in this thesis is presented in combination with the design choices that lead to its creation, the dependencies and relations to the other features, and how they fit into the big picture. This chapter aims to create a thorough understanding of the concept presented in this thesis and of the decisions that lead to the design. Furthermore, Chapter 5 provides the design details required to answer the first two research questions, which makes the chapter crucial for two of the three research questions.

Chapter 6 describes the behaviour engine, a software component set in a smart home environment based on the solution design introduced in the previous chapter. Every feature introduced in the previous chapter is integrated into a smart home environment and the implementation details are presented and discussed. This chapter has the same relations to the research questions as the previous chapter by being important for the first two research questions.

The already mentioned big dataset is described as part of Chapter 7. This chapter has the main purpose of showing several experiments performed against the behaviour engine to validate the functionality and applicability of the concept in the real world. For making suggestions of how applicable a concept or an application is in a real-world scenario, it is necessary to come as close as possible to a realistic setting during the evaluation, which can be done based on data gathered in the real world. All research questions are related to the evaluation chapter with the last research question even being entirely processed in this chapter.

Chapter 8 presents a conclusion to the thesis by summing up the results of the thesis and how the requirements set by the research questions are fulfilled. Furthermore, this chapter discusses the possible future work resulting from the previously presented findings.

2. Smart Home Fundamentals - An Internet of Things Environment

In 1960 [41], the first types of smart homes were introduced under the name of “wired homes” [41] and have since been referenced to with many different terms, like “adaptive houses” [66], “intelligent buildings” [80], “automated homes” [44], “integrated home systems” [57], or “domotics” [93]. A smart home is usually defined by the inclusion of additionally deployed sensors and actuators as part of an internal network with a middleware [59]. The goal of improving the home’s services to be “smart” by automating them [85] is the significant difference between a smart home and a non-smart home.

The interconnected devices with an additional connection to the Internet put the smart home into the category of IoT applications [101]. The IoT is a paradigm in which the so-called “Things” of a system, which can be any electrical component, are connected to each other, enabling inter-device communication, and connected to the Internet, enabling user-device communication [101].

This chapter serves as an introduction to the topic of smart homes and therefore starts with an introduction to the IoT, which is the enabling technology. After the IoT paradigm is introduced and explained, the supporting technologies are presented, including a short excerpt of their development. Insight into the development of the underlying technologies gives a better understanding of the characteristics and challenges of a smart home environment, which is what is presented in the last part of this chapter.

2.1. Internet Of Things

The IoT, also referred to as the IoT paradigm [73], is the third big development in the field of Internet technologies, with the first two being the World Wide Web (WWW) in 1990, followed by the Mobile Internet in 2000 [101]. Around the year 2000 [101], the term “Internet of Things” has been coined for the first time, describing a network of electrical devices such as sensors connected to computers and the Internet [101]. There have been many definitions for IoT since its beginning years, but due to its broadness, there has never been a unifying definition [73]. Three example definitions of the IoT are:

- “The Internet of Things links the objects of the real world with the virtual world, thus enabling anytime, anyplace connectivity for anything and not only for anyone. It refers to a world where physical objects and beings, as well as virtual data and environments, all interact with each other in the same space and time.” [101]

- “In the context of “Internet of Things” a “thing” could be defined as a real/physical or digital/virtual entity that exists and move in space and time and is capable of being identified. Things are commonly identified either by assigned identification numbers, names and/or location addresses.” [107]
- “Things have identities and virtual personalities operating in smart spaces using intelligent interfaces to connect and communicate within social, environment, and user contexts.” [102]

The multiple definitions of the IoT share common characteristics [73], indicating a general direction of the vision provided by the IoT paradigm. The central quality is the difference to the traditional Internet [73], in which devices are connected to humans and other devices, unlike the Internet, which only interconnects people [6]. Since every object is connected at any time in any place in the IoT like humans are on the Internet, the IoT enables all environments, objects, virtual data, and humans to communicate with each other simultaneously [101]. It is important to note that the goal of the IoT is not to create a separate, global network beside the Internet, but rather to integrate and build components upon the existing Internet [101].

Devices got smaller in size while keeping a meaningful capability of sensing and computing and got interconnected [4]. This advancement enables many different applications and features in many different domains [101], which would not have been possible otherwise, such as networks of vehicles for assisted driving [109], a waste management system depending on the loads in a smart city [74], or a location-based learning platform utilizing the GPS of mobile phones [69]. Furthermore, there are applications, which profit from the IoT like an intelligent kitchen system analyzing the contents of the fridge for healthier meal preparation [59], a healthcare supporting network application [100], or an energy management system using smart sensors to regulate the system for optimal energy consumption [5].

In 2008 [74], the number of people on the earth was overtaken by the number of Things connected to the Internet, and the number of devices connected to the Internet in the year 2020 is around 30 billion [3]. Current estimations for the further development of the number of Things forecasts growth to 75 billion devices connected to the Internet until 2025 [3].

2.2. Underlying Technologies

The technologies required to create the IoT and therefore required to create IoT applications have already been established in the 1990s [101]. The most important technologies, which are needed by the IoT to actually be realizable, are the networking technologies enabling communication between devices, which started in the late 1960s when the first communication between two computers over a network was performed [73]. The development of TCP and IP happened in the early 1980s and the Internet became adopted in the commercial sector in the late 1980s [73], which would later become important building blocks for the IoT [101].

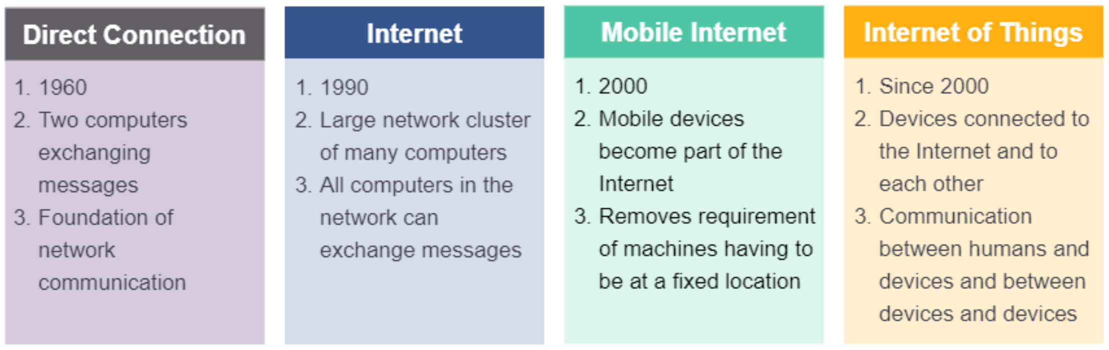

Direct Connection
1. 1960
2. Two computers exchanging messages
3. Foundation of network communication
Internet
1. 1990
2. Large network cluster of many computers
3. All computers in the network can exchange messages
Mobile Internet
1. 2000
2. Mobile devices become part of the Internet
3. Removes requirement of machines having to be at a fixed location
Internet of Things
1. Since 2000
2. Devices connected to the Internet and to each other
3. Communication between humans and devices and between devices and devices

Figure 2.1.: An overview of the development of the IoT and its related technologies

In the 1990s, computers could be built to be small enough, that a tiny sensor or a tiny actuator could be attached to them, which was huge progress for the IoT since these devices could be used to effectively monitor the real world [37]. Furthermore, in 1991 [73] the WWW was introduced, which has its own IoT part, called the Web of Things (WoT) [37]. In WoT, objects are using the WWW as a platform to communicate by being integrated into it, making devices able to communicate with any computer and addressable with any computer browser [37]. After mobile devices got connected to the Internet in the 2000s [101], the IoT was opened up to new application domains, since the sensors and actuators could move freely and were not bound to a location anymore. Furthermore, social networking leads the users to be connected to the Internet constantly via their phones in the Mobile Internet [73], and automation was enabled for many fields with smart devices becoming interconnected and able to exchange information with each other [111]. An overview of the steps leading up to the IoT is displayed in Figure 2.1.

The IoT paradigm is gaining popularity and significance after the year 2000 [73]. A wider interconnection supporting exchange of more detailed information is made possible due to the enabling technologies getting established and more embedded into the world [111]. Another factor increasing the popularity of the IoT is that the supporting technologies are being constantly improved and are becoming more available [56], i.e. sensors becoming cheaper to deploy [74]. Research shows rapid growth in the number of deployed sensors [73] with a trend pointing towards an exponential growth of Things overall connected to the Internet [3]. Today, applications benefit from an environment with many sensors, even if these sensors belong to different stakeholders, like in a smart city where the data produced by the sensors is supplied to other services via a provider [74]. The sensors can also all be part of an enclosed environment like in a smart home, where the devices exchange information with each other [59].

2.3. Smart Home

A smart home is considered to be “smart” due to its interactive nature. This is achieved by replacing regular devices with smart devices, with the key difference being that the latter are connected to the Internet and connected to each other. Furthermore, connecting devices enables remote control by the user and exchange of information with each other, improving the offered service. The number of interconnected devices varies between smart homes and there is a big selection of smart devices, such as a smart TV [50], a smart heating system [5], or smart kitchen devices in general [59]. In fact, most electrical devices are nowadays having a smart counterpart [80] since all electrical devices can be put on the network [81].

In smart environments in general, the user benefits from the offered services being automated [2], which also applies to a smart home environment due to it falling into that category [98]. Waking up with the coffee already being ready, warm water already waiting in the bath after a long day of work, or every device being controllable via a smartphone are common scenarios in smart homes.

A smart home can be considered an upgrade to a regular home without any specific requisites besides having a network or at least having the capability of supporting a network [41] since the technologies required for the upgrade are independent of other external factors. The benefit of not having many prerequisites makes virtually all homes in developed countries capable of being upgraded to smart homes. Communities also benefit from upgrading as many homes as possible due to the advantages of smart homes, i.e. lower emissions [85].

The appeal of upgrading a home to a smart home is the promised automation of the daily activities and tasks, which leads to an easier and more effective day for the inhabitants, who are crucial stakeholders for a smart home system [5]. Dual-income families are a subgroup of homeowners, that are especially interesting in the smart home context [27], because of their additional challenges in organizing their lives [68]. A dual-career marriage introduces a dynamic structure into the household [68], requiring a careful balance between family, work, school, and self-enriching activities [27]. A home capable of adjusting itself to the daily routines of its residents is a promising solution for facilitating the issues of families with these challenges [66].

Early smart devices automated tasks in a home environment by using internal sensors and reacting to certain states in the system, which limited their features to the information they could acquire directly [85]. Before wireless communication was available, the only way of interconnecting the devices inside a home would have been to hardwire the devices into a network [85]. This would be a very expensive and cumbersome solution when the network itself is installed and every time a smart device is added or removed [41]. Therefore, wireless network technologies were an important advancement for smart homes, since they solved the issues stemming from the requirement of connecting devices into a network [85].

2.3.1. Use Cases

The preconditions offered by a smart home environment enable many possible features in different fields [56]. This opens up a big field of improvements to the quality of life for many people and many economic opportunities by providing a big market for new technologies and smart devices. In this section, three example use cases are presented.

Energy Management System

An interesting application domain in regard to smart homes is EMS [5]. Studies show that 41% of residential energy consumption is wasted in US homes [85]. Multiple factors are determined to be the reason for the huge amount of energy being wasted such as older and less efficient devices not being replaced or simply users not using their devices optimally [85]. The latter factor requires an automated solution to fully limit the energy consumption to the users' needs. An example of how an EMS profits from smart technologies and smart environments can be seen in Figure 1.1.

Elderly Care

One field highly benefiting from the remote control possibilities and home automation are applications designed for elderly assistance. Living in a smart home environment enables monitoring multiple aspects of life, which can be used to improve the maintenance of the health of elderly people. [22]

Smart Kitchen

Home automation improves kitchen technologies by i.e. making an oven smart in the sense that it has access to the knowledge of the optimal cooking time and heat for the cooking ingredients. Furthermore, a smart oven can be improved by being interconnected with other electrical kitchen devices, which i.e. enables it to use the content of a smart fridge for recipe suggestions. [59]

2.3.2. Common Challenges

Besides the benefits the IoT paradigm brings to several application domains, there are also clear challenges the IoT and applications relying on the IoT are facing [101]. Many of these common challenges are also relevant for smart home environments.

Security

One problem stemming from the necessity for storing and managing data is the security that needs to be upheld for different kinds of sensitive data, especially if it is related to a person [6]. In a scenario where the medical records of patients need to be handled, a model for regulating access properly for different users in the system is required [100]. IoT applications generally have the same problem any network systems have, which is that the necessary tools and resources to breach the security of any network application are publicly available and widely spread [81].

Smart homes deal with an additional hazard compared to a non-smart home due to the sensitive data being accessible via the Internet, but by being connected to the Internet, it also has possibilities for enhanced security a regular home could not achieve [81]. In a smart home, all states in the home are obtainable regardless of the physical location of the homeowner. Therefore, a smart home removes all fears such as not knowing whether the stove has been turned off or whether the door is locked after leaving the home since the residents can at any time and from any place look up the states of the system. This approach can even be improved by the smart home itself recognizing bad states in the system and raising an alarm to the smart home residents, i.e. if a thermometer captures an unreasonably high temperature, the user might be warned of a fire.

Computational Power

To process data, it is required to spend certain resources such as computational power, and the more data needs to be processed the more resources are required to process the data. Executing a task based on the data generated by a big number of devices in an IoT system can become unfeasible due to the cost increasing with the computation times [108], which can lead to the task becoming too expensive to execute leading to a lower Quality of Experience (QoE) [18]. Other tasks in IoT environments are time-critical and need to be executed based on immediately available information [16] such as a smart car having to adjust its maneuver to a hazard [106]. In a smart home environment, the focus lies on the QoE of the user, which includes dealing with big amounts of data rather than time-critical operations. There are no hazards that have to be dealt with in a period of microseconds to prevent injuries or damage in smart home applications. In a sensor network, a lot of data is constantly produced [6], which introduces the necessity for a filter to extract the knowledge from the raw data [108].

A way of optimizing the usage of the computational power of weak computational nodes is possible in an IoT network by using a task scheduler [31]. Distributing tasks over an IoT network should enhance the QoE for the user of an application [18] since most tasks can be finished faster and more reliably due to multiple nodes being able to offer a service to the user [31]. A task scheduler in a non-distributed environment only needs to operate with an optimal strategy for its own resources to execute tasks effectively. In a distributed environment, it is additionally necessary to take meta information into account about which nodes are accessible, which are prone to outages, and which nodes have a reliable connection [31]. A smart home usually has only a limited number of rather weak computational nodes [72], which is why it is necessary to distribute the tasks and components efficiently.

Data Management

A common example of issues shared in many IoT applications stems from the high number of approximately 30 billion devices that are part of the IoT [3], which produce data that has to be stored and processed [73]. The storage and management of the amount of heterogeneous and unstructured data the IoT deals with require more sophisticated approaches than traditional relational databases can handle [43]. The usual strategy to mitigate this problem is to distribute the data between many nodes with a crucial fault tolerance strategy to ensure the preservation of the data despite its distribution [43].

Another problem stemming from the processed data being heterogeneous in nature is that the different applications in an IoT environment need to communicate with each other. Since many applications are interested in different data due to their application domain [2] and due to the components sharing information they gather themselves, it becomes complicated to norm the data so that all applications and components can access the data they need [37]. Furthermore, there are no standards or protocols after which these applications are operating, leading to challenges in the development of an IoT application [37], which can mostly be addressed by synchronizing the different components of a resulting environment and by providing clear interfaces.

Obscure Technology

The lack of public awareness regarding the underlying technology, its potential, and issues, is a more general and less technical problem for IoT applications [69]. Most of what brings the benefits in an IoT environment is perceived by the user to be running in the background, which makes it difficult to understand [69]. This leads to the public opinion being formed by superficial information about the concepts of the IoT paradigm presented by commonly consumed mainstream media, which displays ideas of a world with connected devices ranging from a dangerous dystopia to a euphoric utopia [69]. Another related issue is the lack of commonly known IoT-related reference business models since this is what would attract more investors for IoT applications and systems [56].

The important consideration for smart home technologies related to this issue is that the applications and systems need to be understandable by the users, who cannot be assumed to have knowledge about IoT or any technological knowledge at all [27]. For this reason, it is necessary for designers and developers to create applications that are accessible for the inhabitants of the smart home while dealing with this environment. For example, an EMS for a smart home is simpler to understand for a common user if it measures the consumed energy in how much money the user has to pay instead of displaying a value on a scale like kilowatt per hour [85].

3. Context Fundamentals

The phrase “to put something into context” is commonly understood as providing additional information, while the phrase “taking something out of context” is commonly understood as the removal of information. The first phrase is usually seen as desirable, as more information tends to deliver a clearer picture, which enhances decision-making, while the second phrase is usually seen as undesirable because removing information leads to worse decisions. Therefore, being context-aware means being able to access additional information and in the scope of a software application, this characteristic implies the possibility of the application to adapt to the accessed information during their operations [8]. The goal of making a computer application or system context-aware is to improve the human-computer interaction without adding the requirement for additional and complex user inputs [4].

While the concepts of context and context-awareness seem clear to most people at least from a linguistic perspective, it is a challenging task for most people to properly define these terms [1]. The intuitive grasping of these concepts comes from the human characteristic of being very capable of conveying and understanding ideas due to their implicit knowledge of the real world, which is not inherently available in communication including computers [1]. The inclusion of this implicit knowledge has already made many computer applications and systems context-aware in many different fields, including intelligent environments, pervasive computing, and ubiquitous computing, which includes smart homes and other IoT environments [4].

Before designing a context model for a context-aware application, it is necessary to analyze the underlying context domain to understand what context is available and create a context model exploiting as much of the context as possible. Many context-aware systems and applications have already been designed in several different application domains [73], providing insight into the challenges and demands coming with the development process of a system or application with an emphasis on context-awareness in a given environment [4]. Besides the challenges and demands also lots of design methods are presented in the literature, which is why it is advisable to look into already existing ways to approach the creation of a context-aware application.

To create a context-aware application, it is necessary to have a fundamental understanding of what context-awareness is, which implies a necessity for a clear and concise definition of what context is. Therefore, the first half of this chapter is about the terms context and context-awareness and starts with providing definitions for these terms. Working based on clear definitions provides a solid foundation for the scientific aspect of this work and reveals possible restrictions from a technical perspective. After introducing the two central terms, some examples are presented in various context domains and a very important meta-model for context models is introduced, which is called the life cycle of context. The second half of this chapter discusses various methods for creating a context-aware design.

3.1. Background

The term context is commonly used in the English language and finding an example for contextual information for a topic or an application is fairly easy. Humans are generally very capable of recognizing related information or drawing links between information to recognize a pattern due to their implicit knowledge of various topics in the world, which is also why the communication between humans does not require the explicit expression of all contextual information [1]. This inherent ability of humans is only possible because both communicating parties are able to access the mentioned implicit knowledge. In communication between two computers or between a computer and a human, it is not possible to implicitly access information outside of what is explicitly exchanged [1].

Definition Challenge

Early definitions of context limited the possible contextual information to specific context types, like the identifiers of objects and users and their locations, the time, the environment and there were even some more specific context types like the emotional state of a user or the focus of attention [1]. These early iterations of definitions for context seem more like a resulting context model and not like a definition since they are predefined lists of all possible contexts, which made them very little useful in practice when used as a definition [1]. Some of the iterations introduced huge differences, which occurred due to the dual origin of the definitions [30]. One view on the term comes from a technical perspective and the other from a social perspective, which leads to the researchers not being able to find common ground for a satisfying definition of context [4].

On top of the dual origin, the iterations of the definitions also show big differences because even the various fields within the technical and social views are using context in their own way, which leads to the definitions being drawn by multiple fundamentally incompatible theories [4]. For example, some theories center around activities, assigning every context to a certain activity, others are more centered on how humans perceive the world, which makes context the common knowledge about the entity it is associated with [4]. In these two examples, the difference is introduced by the chosen focus of each context definition. The first example puts a technical center into its model, where all context refers to an activity, while the other theory has a social view of life and puts the human understanding of the world into the center of the theory.

Context

In the end, all of the previously created definitions failed to satisfy due to being too specific to fully describe every possible situation relevant to a user or to an application [1]. The most agreed-upon definition of context is “context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves" [1], which is also the definition for context used in the scope of this work. Figure 3.1 illustrates this definition by displaying that context refers to the information relevant to describing or characterizing an entity.

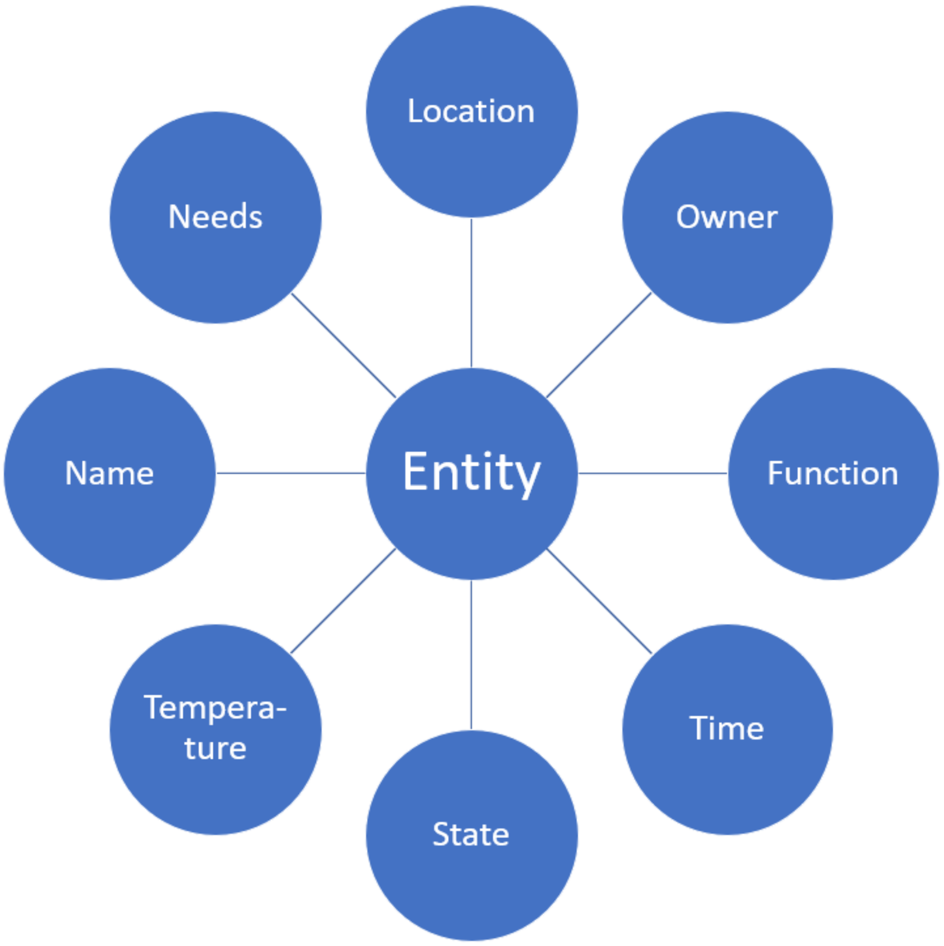


Figure 3.1.: Example of the context of an entity

The presented definition has the main advantage of being capable of including any context in any environment, allowing any available information to be considered as context. This broadness and comprehensiveness are also its biggest disadvantage because using this definition for context carries the necessity to design a specific context model to set boundaries for what is ultimately used as context. This is necessary due to the definition itself not providing any restrictions and no limitations for what can be included as context. Therefore, a context model can be considered the subset of a context domain containing only the most relevant context.

Context-Awareness

Context-awareness faces the same definition challenges that context faces, but unlike with context, there is no commonly agreed-upon definition for context-awareness [4]. The same author giving the definition of context also gave a definition for context-awareness: "a system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task" [1], and the other views and definitions of context-awareness share the same core concepts of this definition. All projects designing a context-aware application recognize the crucial capability of adapting to the context found in their environment [8].

Context always depends on the underlying environment, since the tasks, the goals, the problems, and many other factors differ even between similar environments [2]. Therefore, it is necessary for a context model designer to tailor a context model to the domain while considering all of the given possibilities and constraints coming with the environment to optimize the context-aware nature of the design. The process of modelling is crucial when creating context-aware applications and systems. If the exact environment and requirements are not properly investigated, it is possible that meaningless information gets included, which leads to an ineffective process, or meaningful information is not determined, which lowers the usability of the context model for the context-aware application.

Due to the presented definition for context, the designer of a context model can work without any restrictions, but the challenge for the designers is not to find all possible context for the context domain, but instead to find all meaningful context for the context domain. An international corporation tracking the location of their devices might include which country the device is in for the location-based information of a device as context, but a corporation working in only one country does not need to consider the country due to the information being redundant. This simple example shows that small differences can change the relevance of context significantly.

Finding all meaningful context for a context model designed for a computer application or system is a challenging task. Besides basic design models, it is also helpful to create a classification for types of context, especially due to the broad nature of the definition. A way of categorizing context from an operational perspective is based on the way the data value was acquired [73]. A context type is classified as primary context if the data values are acquired directly by sensors and no further computations, and as secondary context, if primary context is used to compute the data values to provide additional context [73].

3.1.1. Context Domain

In this subsection, a few mostly IoT-related context domains are presented and serve as general examples of showing how the environment places specific requirements on a context-aware design [105], dictating what the context model needs to achieve and what context needs to be included in the model. The smart home is not included in this section, because it is discussed in more detail in Chapter 5.1.2 with an in-depth context analysis.

IoT Middleware

A middleware for an IoT environment deals with the dynamic, heterogeneous, and big-scale nature of the infrastructure [71] and would consider quality of service for search times in form of precision and recall [21] as a relevant contextual factor. A sensor network produces big amounts of data [74], which leads to challenges [108] such as the storage of the data needing to be distributed over several machines, making the current capacity of all nodes responsible for data storage useful context when determining data storage management strategies.

A middleware put in place of a sensor network is required to provide accessibility to the data stored in the distributed nodes and to support the processing and dissemination of the context [47]. Therefore, reliability or any other common networking attribute can be considered context in a context-aware middleware in an IoT environment.

Smart City

There are many possible applications in a smart city that use location-based context. Some examples include an interactive tourist guide or a monetary incitement for people to distribute them more evenly across the city based on their locations in the city [69]. Some projects include the location-based context together with the consumer behaviour such as smart fridges keeping track of their content to notify the local grocery store of which products will more likely sell [74]. Smart garbage cans are capable of informing the city about the state of their capacity to improve the waste management infrastructure [64]. Context-awareness is very interesting while designing applications for smart cities because of the heterogeneous nature of the available information that can be used in the services and the huge range of information the various services can offer to each other [114].

Vehicular Networks

Related to the smart city and in some cases, part of a smart city is the topic of a vehicular network [67], where cars exchange information with each other and with the system to provide a safer and more efficient traffic network [109]. In this scenario, the real-time locations of the cars on the road are used to choose the optimal route for the car to get to the goal [67]. Furthermore, the computational tasks can efficiently be divided between more powerful computational nodes in a centralized network and the less powerful computational nodes in smart cars [110]. Including context reduces congestion without the existing physical infrastructure having to be extended [67].

Task Scheduler

A task scheduler's goal is to optimize the execution of tasks by scheduling them effectively [77] and one crucial contextual information is the length of a task. A task scheduler in a distributed environment has multiple computational nodes and needs to choose the processing unit that executes the task [31], which is why the current load and capacity of each processing unit is essential context when deciding which unit gets assigned a task. If the distributed environment of the task scheduler furthermore operates with mobile processing units, the task scheduler also has to consider the reliability of each unit while deciding which unit is assigned a task [31]. This scenario shows how the context model needs to be adapted to a specific environment, even if the task is the same. Therefore, using an existent model without investigating the underlying environment thoroughly leads to suboptimal solutions.

3.1.2. Life Cycle of Context

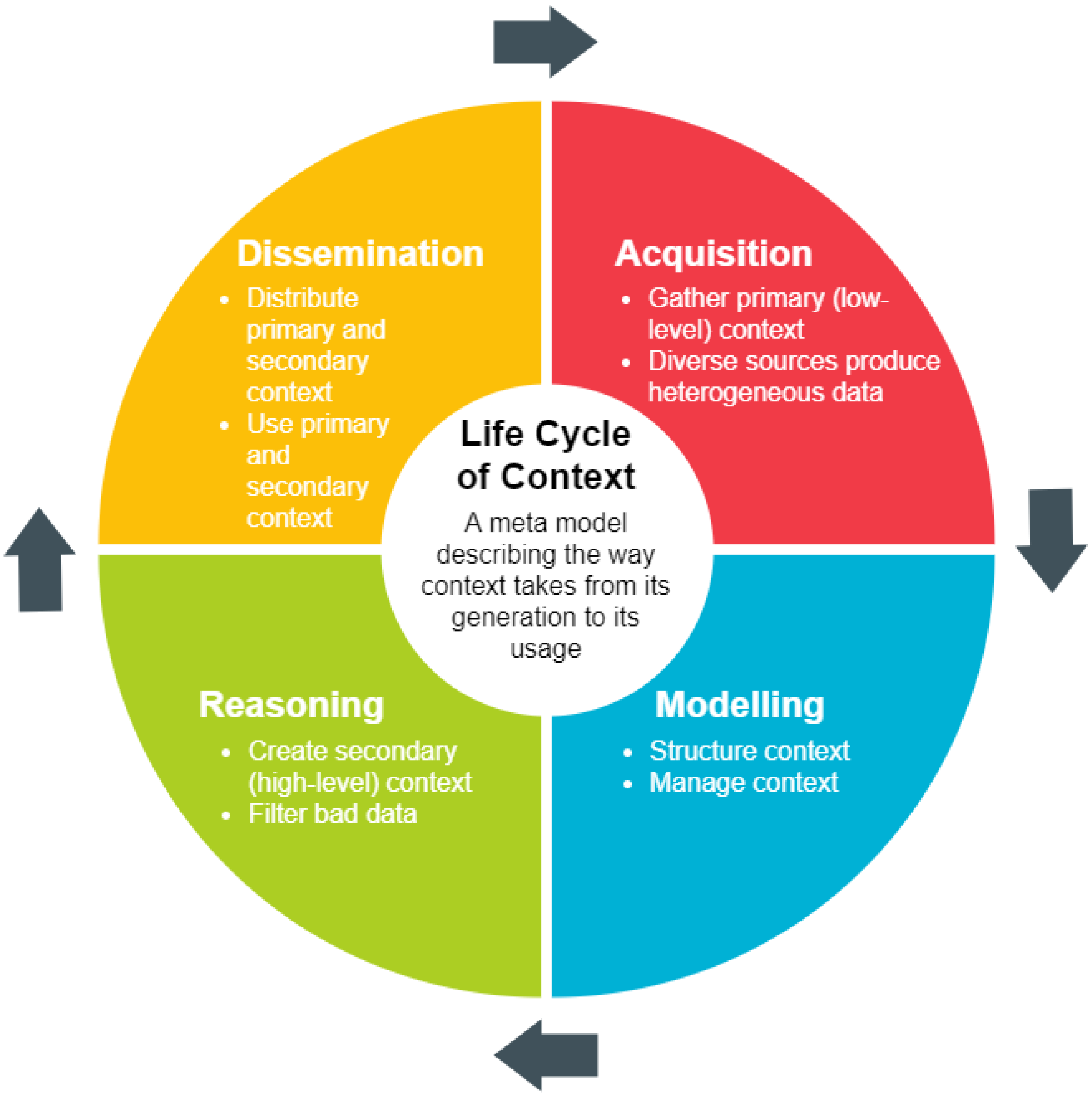

Dissemination
• Distribute primary and secondary context
• Use primary and secondary context
Acquisition
• Gather primary (low-level) context
• Diverse sources produce heterogeneous data
Life Cycle of Context
A meta model describing the way context takes from its generation to its usage
Reasoning
• Create secondary (high-level) context
• Filter bad data
Modelling
• Structure context
• Manage context

Figure 3.2.: The four phases in the life cycle of context

Having an understanding of context is a crucial prerequisite for fully utilizing contextual information in a context-aware application. Next, it is also important to consider the different steps included in the process of making contextual information available. A good model for understanding the phases the information goes through, which almost all context-aware applications follow, is the life cycle of context [73]. Due to the simplicity and the intuitive nature of the life cycle of context, context-aware applications having a context management system following this model often do not cite it [73].

Acquisition Phase

The goal of the first phase is to gather unprocessed data with the important questions of this phase dealing with how, what, and where the data can be acquired [73]. In a distributed environment such as smart homes [105], the information used in an application is gathered by all devices and applications in the environment [73]. There are mainly five topics related to the method of acquiring data, which need to be considered in this phase [73].

The first topic is about whether the information in the environment gets pulled or pushed and results in the first design decision for the context management system [73]. The second and third topics represent whether the context to get is measuring a continuous state or requested with distinguishable timestamps and what the underlying structure of communication looks like [73]. The last two design decisions are based on the types of context sources such as digital or physical, and the type of gathered context, i.e. whether the context is sensed or manually put in or derived [73]. The context created in the acquisition phase is classified as primary context since it is gathered directly and there is no further computation performed on the context in this phase.

Modelling Phase

At the end of the acquisition phase, the raw sensor data is collected and has now to be translated into a context model to represent the low-level context in the system [73], which happens in the context modelling phase. Depending on the way the context model is organized, this phase might also be responsible for organizing and including further context generated after the first phase.

This phase in the life cycle of context is very dependent on the underlying environment since the resulting low-level context needs to be meaningful to the application in the environment. Several methods can be used to design a context model for a context-aware application [73] and many times these methods do not exclude each other [4]. In Chapter 3.2, some example design methods are presented.

Reasoning Phase

The reasoning phase has the responsibility to analyze the low-level context present at the end of the second phase and generate high-level context or conclusions about the contextual information [73]. The context created in this phase needs to provide precise knowledge to the application efficiently and due to the fact that it is generated by computational operations instead of being sensed directly, it is classified as secondary context [73]. Furthermore, the reasoning phase has the responsibility to filter out bad primary context [73], which can be present in any environment due to any form of malfunction [108]. After filtering bad context, the remaining context needs to be grouped and processed [73].

There are many suitable mathematical models, such as k-means clustering [99], or logical models, such as decision trees [76], for processing various types of low-level context. Choosing a fitting method for the reasoning phase needs to be based on an examination of the possibilities in the low-level context and based on the goals of the reasoning. A good example for the distinction of the possibilities enabled by the low-level context is whether reference inputs can be provided for a supervised method [42]. Alternatively, the reasoning phase needs to discover patterns without any external aid, which would classify the reasoning method as unsupervised [9]. It is also possible that multiple methods are included in the reasoning phase if multiple, different kinds of high-level contexts need to be created.

Dissemination Phase

In the final phase, the high-level context that was analyzed in the reasoning phase and if necessary also the low-level context present in the modelling phase need to be made available for the context-aware system [73]. The main question for this phase is how the contextual information is transmitted to the consuming application and is mainly dealing with the infrastructure of the system [73]. Depending on the environment, context gets queried from the context managing component or the application needs to subscribe to receive published updates about the context [33].

After the dissemination phase, the context is deployed in the system and further context can be gathered with or without the aid of the already established context, which means that the cycle restarts from the data acquisition phase. Whether new context needs to be integrated into already existing context or all context exists separately in the system is a question answered by the design of the context model [73]. Depending on the second and third phases, new context can lead to existing context requiring reevaluation [73].

3.2. Design Methods

The used definition for context allows any information, that can be related to the currently examined entity, to be considered as context. Since no restrictions are imposed on a context-aware design through this definition due to nothing being excluded from qualifying as context, every possible information can potentially be considered context. The all-inclusiveness of the definition potentially leads to ineffective designs of context models due to the designers not being able to properly determine all meaningful context or not being able to disqualify unimportant context [32].

The importance of how dependent the context is on the underlying environment or task has to be considered thoroughly at this point. Related literature provides a wide range of lessons learned for context-awareness in many different fields such as IoT [73], industry [83], government [32], and pervasive computing [4]. Simply orienting the own context-aware model by an existing model can easily lead to sub-optimal usage of context-awareness or even problems in the design itself.

A promising approach to creating a context-aware design is to use specific design methods created to aid designers in determining all meaningful contexts and in structuring the context-aware application. These methods are distinguishable by the environment they are created for and by included modelling techniques [73]. The six most popular techniques for designing context models are based on key-value maps, tags, relationships, objects, logic, and ontologies [73].

There is a big discrepancy in how detailed the guidance is given by a design method due to the context being highly dependant on the underlying environment [2]. Generally, the more guidance a design method offers for creating a context model, the more requirements and restrictions it imposes on the created model or the environment. On the other hand, design methods that are kept as generic as possible are environment-neutral and put no to minimal restrictions on the design, but the more generic a method is the less in-depth guidance it offers in the design process.

3.2.1. In-Depth Support Design Methods

The more detailed context-aware design methods are appealing for their vast support in the design process and are making sure that the design makes the most out of the context found in a specific environment. These methods are usually influenced by the experience gained during the creation of other context-aware designs in the respective field, which turns lessons learned from other projects into a useful tool while working on a context model in a related setting.

Businesses commonly work closely with the government in many different fields, which opens the field of business to government applications, typically set in a large-scale environment with multiple actors and stakeholders [32]. Many different fields with widely varying requirements fall into this group, but these requirements have some intersections due to their similar environment. Based on many reference context models, a systematic approach is provided [32], featuring a step-by-step guide to easily discover the meaningful context in an environment navigating many different actors and legislation considerations.

When a field does not have many reference solutions, it might be necessary to compromise on the similarity between the projects to be able to use one of the more specific context model design methods. A lesser related project might still provide a design method that is useful, but a developer needs to be careful to not overlook the differences between the respective requirements and environments. The industrial field does not have many reference projects with context-aware designs, but picking a specific method in this field might help to discover context types not being suggested in other models. A context model for industrial applications could utilize the temperature of the operating machines as context, as suggested in a design method created for this field [83].

3.2.2. Generic Support Design Methods

The more generic context-aware design methods are based on the goal of being applicable to all environments, which is why these methods usually provide basic guidelines that can help every project regardless of the environment it is set in. In the scope of context-awareness, which is heavily dependent on the underlying environment, looking into design methods created to be environment-neutral gives insight into the most common design decisions. Furthermore, generic methods can provide systematic approaches to easily determine meaningful context, even though no specific context is suggested.

One generic method of designing a context-aware application has already been discussed in Chapter 3.1.2 as the life cycle of context, which can be found in all projects including context-awareness, even in those not citing it, showing its intuitiveness and its general applicability [73]. Another generic example, which is very similar to the life cycle of context in being detectable in most context-aware projects, is a layered architecture introducing the most common components of context-aware systems and applications divided into multiple layers [8]. The first three layers introduced in the architecture are comparable to the first three stages of the life cycle of context, with the sensor layer being the equivalent to the context acquisition phase, the raw data retrieval layer being the equivalent to the context modelling phase and the preprocessing layer being the equivalent to the context reasoning phase. The top two layers are the storage/management layer and application layer, which are together similar to the last phase of the life cycle of context, the context dissemination phase.

An easy and systematic approach to designing a context-aware application without being built for any specific environment is CAMeOnto [2]. This ontology is based on the principles of 5Ws, which are the interrogatives who, where, when, why, and what [2]. The who refers to the information about the user, i.e. the identity of the user or the preferences of the user, and encourages thinking about who exactly is supposed to use the application. The when refers to temporal contexts such as the time of day an action is being performed or intervals between actions and is used to determine how time plays a role in the task of the application. The country, the city, the exact GPS location or the distance to something are all examples for location-based context, expressed by the where. Activities and devices fall both into the category of why in this ontology, which encourages asking why the user is able to get a service with the answer indicating that an activity or device is involved. The service provided to the user is the what, which is used to determine what the user trying to get from the application.

4. Access Control Fundamentals

Security is one of the most important concerns in an information system and security breaches due to unauthorized access need to be prevented in particular in the growing field of distributed environments [14]. The goal of access control is to prevent these security breaches and this is achieved by restricting inappropriate actions in the system, which can happen due to operations by programs on behalf of users or due to requests by users [88]. The mechanisms of an access control system limit the activities and operations a user and in extension, a program on behalf of a user is allowed to perform in the system [88].

An access control mechanism (ACM) can be defined as: "the logical component that serves to receive the access request from the subject, to decide, and to enforce the access decision." [45] Using this definition puts the component responsible for access control in charge of evaluating access requests and in charge of enforcing the decision made in the evaluation process. Enforcing access control means to decide whether a subject, the requesting entity, is allowed to access an object, the requested entity [89]. An example of a traditional form of access control can be imagined as traveling to a country with the border officials enforcing whether a person is allowed to enter the country or not.

The process of enforcing access control is called authorization, which ensures that illegal access to information is prevented [11]. Authorization is related to the similar term authentication, which is the process of verifying the identity of a subject in a system [20], and therefore, authorization is enabled by authentication [88]. It is important to consider that the terms access control and authentication are not referring to the same process and that authorization usually assumes that the subject is already authenticated [88]. The access rights of the users in a computing system are stored in an authorization database and are maintained by a security administrator [88]. Furthermore, access rights are organized following an access control model, which has a long history full of many different types of models with each generation introducing a more sophisticated approach than the previous one.

The hierarchy of an access control model is formed by policies and rules [55]. Policies consist of rules and can therefore be seen as the set of rules in place to decide whether access should be given or not. Each object is assigned exactly one policy and each policy can have any number of rules. During the decision process of the enforcement of a policy, each rule is examined and the decision is made based on whether at least one or all of the rules are fulfilled, depending on the access control model.

The purpose of this chapter is to give a general overview of the different forms of access control models, commonly seen in enterprise environments, government, or information technology. The goal of this overview is to create an understanding of how access control was developed, giving an overall insight and understanding of how access control based on attributes is working. This chapter does not compare the different access control models and is not meant to be seen as an evaluation of which approach of designing an access control model performs the best.

4.1. Identity-Based Access Control

In access control approaches based on identities, the access control is only utilizing the identities of subjects and objects and are therefore called Identity-Based Access Control (IBAC) models. These models have assigned identities to every object and every subject in the system and keep access control lists (ACLs) of these identifiers for which subject may access which object [113]. ACLs hold the associations between identities and are used during the enforcement of the access control decisions [45], where access is permitted if the association between subject and object is given [113].

Translated into the example of border officials enforcing whether a person is allowed to enter the country, the officials would either have a list of all people that are allowed to enter the country and look whether this person is on that list, or there would be lists for each person that specify which country this person is allowed to enter. Both options display the biggest weakness of IBAC approaches, which is the very low scalability due to the lack of natural support for large-scale systems [45]. Besides the issue of potentially billions of entries on an ACL having to be processed during an access request, another issue is the high maintenance overhead introduced in updates whenever changes to the access rights need to be applied [45]. A necessity for maintenance arises in the example of border enforcement in situations such as when a country would ban the entry of citizens of another country because of a conflict, which would lead to countless entries in many lists requiring an update.

In computing systems or networks, the number of access rules would possibly require more computing power to resolve big ACLs. This is less of a problem nowadays since most computers would likely be powerful enough that no issues become noticeable. IBAC approaches are attractive because they are easily and quickly implemented and are suitable for systems with a low or at least a constant number of users or if the access rights are rather static [113]. Issues in IBAC models arise in systems with large and dynamic subject and object numbers, since the high maintenance efforts become a burden and error-prone, especially when a human updates the access lists, leading to possible security breaches [113]. One unattractive aspect of IBAC is the low support for the creation of policies since this approach uses exclusively identifiers, which does not accurately represent how i.e. a company handles the access rights in its informational system [113].

IBAC finds application in secure communication over a network [55] since the mutually exchanged information at the beginning of the communication gives each party an identity on which the access control is performed. Another application domain for IBAC is operating systems [113], where the file access rights are commonly set as user lists as a matrix containing the information of which user has what access rights for an object.

4.2. Lattice-Based Access Control

The early counterpart to access control models based on identity is called Lattice-Based Access Control (LBAC) [87]. The name-giving lattice in these approaches comes from the featured security labels used in the access control process, which are combined into a grid [113], in which they are divided into a totally ordered set of security levels and a set of categories [87].

In LBAC, each category can be combined with any security level, forming distinct security classes, which are assigned to each object, called their security label [87], and to each subject, called their security clearance [86]. To gain access to an object, the security clearance of a subject must be of the correct category and must dominate the security label of the object [90]. A security level dominates another security level if it is equal or higher to it in the predefined, totally ordered set of security levels.

When there are four security classes in the order “top-secret”, “secret”, “confidential” and “unclassified”, a subject with the security level of “secret” in a certain category is allowed to access objects of all levels but “top-secret” [90]. Even when a subject has the security level of “top-secret” in one category, the access to objects of other categories is not necessarily granted and depends on the security level the subject is assigned in the category of the object.

Information being translated after successful access control is called the flow of information [87] and a secure information flow guarantees that unauthorized flow of information is prevented [29]. The goal of LBAC is to ensure that information flows only one way, which is into the dominant direction, and never the other, which is the dominated direction [90]. This characteristic makes LBAC not policy-neutral since the fundamental structure of LBAC models enforces the flow of information to only go in the dominant direction [86].

While objects can be destroyed and created dynamically in the system, the security classes remain static and must be predefined [87]. All security levels are assigned by the security administrator or security officer instead of the resource owner like in other access control models and usually cannot be changed dynamically, which is a characteristic that is called tranquility [86]. Not being able to modify the security classes in the lattice and not being able to update the security labels and security clearances makes LBAC very static in nature.

In the example of border control enforcement, no information flow is present so that a lattice can be formed properly. Furthermore, the only possibility for creating categories is to create a separate category for each country. This makes the environment undesirable for LBAC because a lot of categories cause management overhead [113]. Furthermore, changing the lattice, which is necessary each time a new country is founded or becomes part of another country, and changing the security labels and clearances, which becomes necessary each time the travel rights for someone change, is a cumbersome activity in the static LBAC.

The access control systems developed with a lattice-based approach are mostly set in government environments and the military sector [113]. LBAC is popular and widespread in both of these environment types [87]. The popularity in these specific sectors comes from the concern about the flow of information being present in the environment. Therefore, most LBAC models are driven by the requirements of governmental or defense sectors [87]. The scenarios commonly found in these environments tend to create a structure of security classes that can be described as a lattice with a one-directional flow of information. LBAC is a fitting model in these cases because it is specifically designed to have policies, which enable this flow [86].

4.3. Role-Based Access Control

The first attempts of creating an access control model unveiled the challenges of enforcing access control and maintaining access rules, which are mainly performance issues in case many rules need to be processed and maintenance issues due to low support from the access control model. A promising approach, called Role-Based Access Control (RBAC), was introduced as an alternative [86] and the first access control models using RBAC emerged in the 1990s [90], which is 25 years after the early LBAC and IBAC approaches [48].

In RBAC, the name-giving roles serve as a semantic relationship between subjects and objects [113], representing the access rights of subjects and the permissions on objects [70], and are maintained and assigned by security administrators according to the access rights in the system [90]. The assigned roles of a subject are examined by the ACM and compared against the list of roles in the access policy of an object and access is granted if a match is found. This means that in RBAC the decision for an access control request is predetermined implicitly by the security officials of a system [45].

RBAC models for border officials would work like passports, effectively giving each country a respective list of countries their citizens are allowed to enter and border officials only need to look at the passport to check whether the passport is of a country whose citizens are allowed to enter. The enforcement of the access control is more effective than in IBAC since the time to check each role is significantly smaller than the necessary time to iterate through all ids in the system.

IBAC and LBAC are both special cases of RBAC, which means that their ACMs can be expressed using the ACMs offered by RBAC [90], which allows a combination of easier to maintain ACLs and lattice-based structures with secure information flow at the same time [86]. In comparison to IBAC, roles can be considered predefined sets of access control rules, enabling central management of ACLs, lowering the necessary maintenance significantly [113]. Compared to LBAC, roles can be considered as unrelated security classes [86], which are assigned to each subject and object, just as the security clearances and security labels are. The difference between RBAC and LBAC is that RBAC is a policy-neutral access control approach [70] since there is no dependency between the roles. To mimic this defining characteristic of LBAC in RBAC, it is necessary to provide an extra mechanism to preserve the flow of information [86].

The main benefits of RBAC are its good scalability [113] and easy maintainability [86], while it is still simple to implement [89]. Additionally, the significant factors for the spread of RBAC were its compatibility with existing access control models [45] and its adaptability to the typical enterprise environment [113]. The biggest weakness in an RBAC model is that its nature lacks support for exceptions [45] since every exception requires a new role, which in the worst-case scenario provides one role for each subject in the system. In that case, the efforts of creating an RBAC model are ultimately reverted, turning the approach into an IBAC model.

RBAC is especially popular in enterprise access control management since it mirrors the access control rights commonly seen in the corporate environment [113]. The possible central management for RBAC systems makes it easy, effective, and less error-prone when the changing needs of an organization require the access policies to be updated [86].

4.4. Attribute-Based Access Control

In earlier approaches, access control has been restricted to specific identifiers, security labels, and roles and each of these approaches still see success in different fields. Each of the earlier approaches of access control also imposed limitations on resulting access control models, like RBAC requiring a new role for every exception [113], IBAC scaling poorly [45] or LBAC being very restrictive due to its static nature [87]. The two problems these limitations share are that they are fundamentally integrated into the approach itself and that they impose restrictions on the writing of the rules and policies [45]. The design of an access control model should limit the possible access control policies as little as possible [45]. A newer approach tries to mitigate these issues by allowing any security-relevant characteristic of objects, subjects, and the environment in access rules [113]. These security-relevant characteristics are called attributes and therefore this approach is called Attribute-Based Access Control (ABAC) [45].

ABAC is defined as “an access control method where subject requests to perform operations on objects are granted or denied based on assigned attributes of the subject, assigned attributes of the object, environment conditions, and a set of policies that are specified in terms of those attributes and conditions.” [45] This definition displays the difference between ABAC and other access control approaches due to the fact that it does not prescribe a certain core for the access control [48]. Instead, it allows any attribute featured in the system to be the base in access rules, making the writing of rules and policies more complex compared to the earlier approaches [48]. Another difference introduced by this definition of ABAC is the interesting possibility to allow for ACMs to build rules upon attributes of objects and the environment, unlike previous approaches that exclusively enabled rule types related to the subject, like identities, roles, or security classes [45].

Possible subject attributes are any information that can be related to the user in the system such as the identity, the role, or the job title [113]. The object’s attribute usually depends on the type of object, i.e. files in a file storage system could use the title, date, and author of the files as object attributes, while machines in a factory might use their states or their temperature as meaningful attributes [113]. Environment attributes are used to describe operational or situational states of the system unrelated to subject and object, like the current date, network security, or room temperature, depending on which makes sense for the environment the access control model is created for [45]. Environmental attributes are especially interesting since environmental information is widely unused in previous approaches to designing access control models [113].

A benefit of ABAC is furthermore, that it does not exclude the possibility to integrate policies and rules mimicking the rules and policies of RBAC [45] and in extension also of IBAC and LBAC. More generally formulated, RBAC is a special case of ABAC [45], and IBAC and LBAC are special cases of RBAC [86]. The identities used in the ACLs of IBAC can be linked to the attribute “identity” and the roles can be assigned to subjects like in RBAC with rules written for the attribute “role”, which compares the role of the subject with the requested role of the access rule [45]. The flow of information can also be mimicked using attributes by making the ACM recognize that the access rights of one security class include the access rights of all security classes it dominates, preserving the hierarchy of security classes found in LBAC [90].

Designing an access control model based on attributes for border officials would make it possible to introduce additional attributes, while still keeping the benefits of the other approaches. RBAC already gave a good solution that would allow the border officials to use predefined ACLs of countries as roles and assign those roles to their respective citizens. The issues arise from the exceptions such as a criminal not being allowed to leave a country or at least not allowed to enter a specific country. In this case, a new subject attribute could be defined called “criminal status”, which marks the subjects and is used in a policy that allows each citizen to enter all countries according to the access rights given by their citizenship with the additional restriction that the subject is not allowed to have a criminal status. “Crisis status” can be used as an object attribute, which rejects access rights when the country has to refuse further travel due to a civil war or a disease. Environmental attributes could be derived from neighbouring countries as “threat level” since travel might need to be restricted when there is a conflict close to the country since the concern arises that the conflict could cross the border.

ABAC can become cumbersome when the access control is shared between organizations since several other attributes need to be shared between these organizations as well [45]. An ABAC model furthermore takes longer to implement and it requires more understanding on the user side to properly use its features [45].

4.5. XACML Architecture

The Extensible Access Control Markup Language (XACML) [45] provides a framework compatible with ABAC. It introduces an architecture to structure the access control components by dividing the responsibilities of an access control model [36].

The four crucial components used in the architecture of the introduced structure in XACML are the Policy Administration Point (PAP), the Policy Decision Point (PDP), the Policy Enforcement Point (PEP), and the Policy Information Point (PIP) [36]. Figure 4.1 shows the four components of the XACML architecture and the relationship between them is displayed by linking the components that interact with each other.

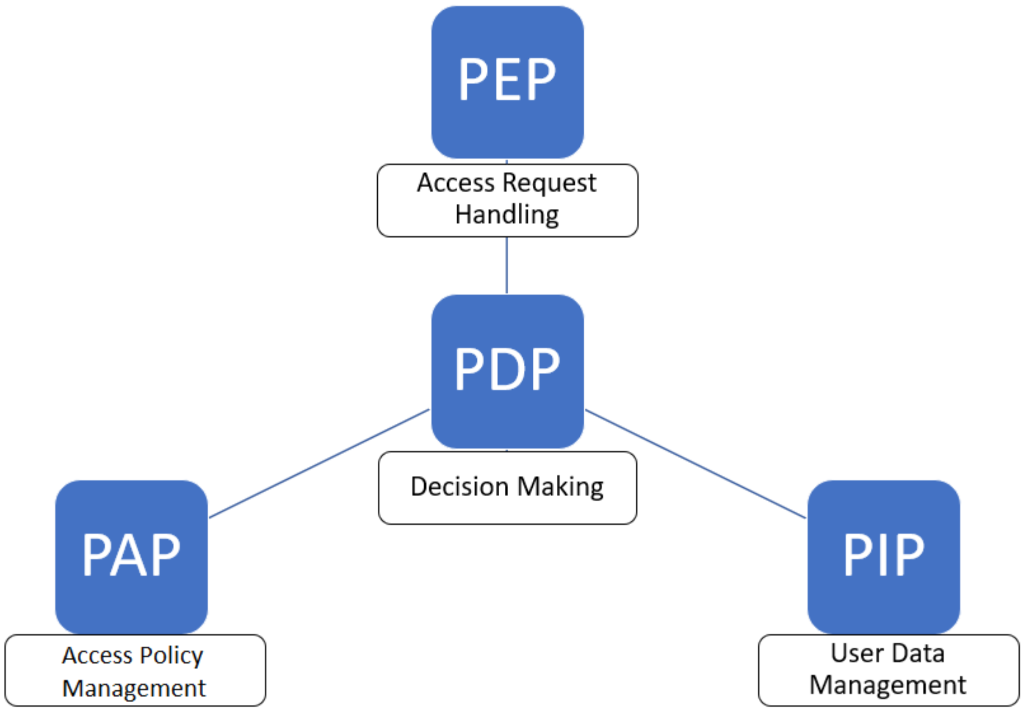


Figure 4.1.: XACML components and which components they interact with

Policy Administration Point

The PAP is defined as "the system entity that creates a policy or policy set" [36]. As the definition suggests, this component is responsible for managing the access control policies. During an access control request, this component supplies the policies that are evaluated.

Policy Information Point

The PAP is specifically responsible for the management of the access control policies, but besides the policies, there are still other data that need to be managed in an access control system. For each other type of information that has to be stored, the PIP is responsible, which is defined as "the system entity that acts as a source of attribute values" [36]. During an access control request, the PIP is responsible for the retrieval of data.

Policy Decision Point

The PIP and PAP components cover the data management and access policy management, but they are not responsible for evaluating access requests or using the data in other ways in the process of access request evaluations. The component responsible for making use of the data and policies is the PDP, which is defined as “the system entity that evaluates applicable policy and renders an authorization decision” [36].

An evaluation of an incoming request is based on the access control policy associated with the object, which is accessed through the PAP, and the related information stored in a database, which is accessed through the PIP [45]. The PDP is the central component in the XACML scheme and is the only component linked with each other component, while all other components are only linked with the PDP [36].

Policy Enforcement Point

The PEP is the final component in the XACML architecture and it is the component responsible for handling access requests. PEP is defined as “the system entity that performs access control, by enforcing authorization decisions” [36].

Whenever a request comes in, the PEP forwards the request to the PDP, which responds to the PEP with an authorization decision result. Based on the result the PEP then has to either block or grant access to the requested object, enforcing the decision made by the PDP.

5. Solution Design

Chapter 1.3 presents three research questions, which serve as the goal of this thesis and are used to define the design goals of this chapter. Note that only the first two research questions set goals for the design of this work, while the third question sets goals for the evaluation. Therefore, this chapter discusses the first two questions and the third research question is processed in Chapter 7.

Determining Behaviour

The aim of this work is to determine patterns in the behaviour found in a smart home environment. These patterns are utilized as indicators for what actions and states in the system are to be anticipated and are therefore utilized to determine whether observed behaviour matches the expected behaviour. The smart home environment this work is set in is called COSYLab and its most important parts in relation to this thesis are presented in Chapter 5.1.

The first research question is formulated to set the goal of creating context-aware behaviour patterns within the observable behaviour of smart home residents. This question asks for a translation of the inhabitants' interactions with the home into prediction models, what behaviour in the system can be observed influencing the residents' actions, and other observable factors in the smart home. The requirements to satisfy this research question are:

- Thorough context analysis of the underlying smart home environment
- Behaviour patterns capable of evaluating whether actions or events are expected
- Investigation and inclusion of further factors influencing the behaviour

First, the context in the smart home needs to be examined to create a context model exploiting the context present in the environment. Based on the available context, patterns in the behaviour need to be determined and created so that they can be used to evaluate whether further events match the behaviour. Lastly, the context model needs to be examined for further possibilities for predicting behaviour more accurately.

Access Control

With the additional knowledge about the expected behaviour, an access control system can improve the safety and security by detecting deviating behaviour in the system and react to it, i.e. alarm the user and the fire department in case of a dangerously high room temperature. The malicious outsider attack pattern, which is based on an outsider without the knowledge about the usual behaviour [62], can be minimized if the ACM is capable of detecting deviating behaviour, even if the user is successfully impersonated by the adversary [53].

The goal of the second research question is to integrate the knowledge of expected behaviour into the existing ABAC model of COSYLab. This research question is satisfied when these requirements are fulfilled:

- Defining attributes for including behaviour estimations into the ABAC model of COSYLab
- Behaviour is processible at runtime during access request evaluation
- Aid the smart home residents in using the behaviour related features

The goals of this research question make the patterns' capability of determining whether the observed behaviour is matching the expected behaviour a crucial requirement in answering the first research question. Furthermore, the evaluation of access requests and the evaluation of events in the smart home is the bridge between the two research questions.

Analyzing behaviour at runtime is unfeasible because potentially too much data has to be processed [19]. Therefore, it is necessary for the knowledge to be prepared by extracting the information out of the data before an access request is processed [108]. This is one crucial requirement set implicitly by the evaluation process being included in access control by the research question.

Additional Considerations

One crucial design consideration originates in the residents in a smart home, who cannot be expected to have much technical knowledge to understand how the system works [27], making transparency an important design goal. In an access control model, only designated people, typically one or more security administrators, have permission to manage access policies [86].

A company can hire a person with sufficient knowledge about IoT and security to manage the access rules and other configurations effectively, but in a smart home environment, the residents of the smart home themselves have to carry out these tasks. Therefore, the management of the access rules and the management of the configurations in the system have to be made as understandable as possible by supporting the user in crucial decisions.

Chapter Preview

The context-aware approach to recognize patterns in the behaviour found in a smart home environment and including these patterns in an ACM requires an enormous design with many considerations and design choices. This chapter serves as an overview of the solution design presented in this thesis and is split up into sections, starting with the lowest-level feature and finishing with the highest-level features.

Section 5.1 is responsible for every context-related design decision and information featuring an in-depth context analysis performed on the underlying smart home environment and the design decisions made for each phase of the life cycle of context. Smart home events are introduced and described in Section 5.2, which are provided by COSYLab and serve as the transmission medium for the low-level context.

In Section 5.3, behaviour patterns are introduced, the high-level context type that serves as the core of the concept and as the central component for answering the first research question. Groups represent semantic relationships in the system and are presented as another high-level context type in Section 5.4.

Section 5.5 deals with the evaluation of smart home events and is the important connection between the first and the second research question. All design decisions related to the integration of the solution design into COSYLab are discussed in Section 5.6, which furthermore provides a central part of the answer to the second research question.

The last section of this chapter is not intended to add any information but to serve as an overview of the crucial design decisions that have been presented throughout this chapter. It introduces several tables to present as much information as possible in a comprehensive way.

5.1. Context Information Management

In Chapter 3, a definition for context is presented and discussed in detail for the scope of this work. The crucial characteristic of the used definition is that any information can qualify as context, as illustrated in Figure 3.1. This puts the designer of a context-aware application into the position, that a context model needs to determine the subset of relevant information in the set of possible information.

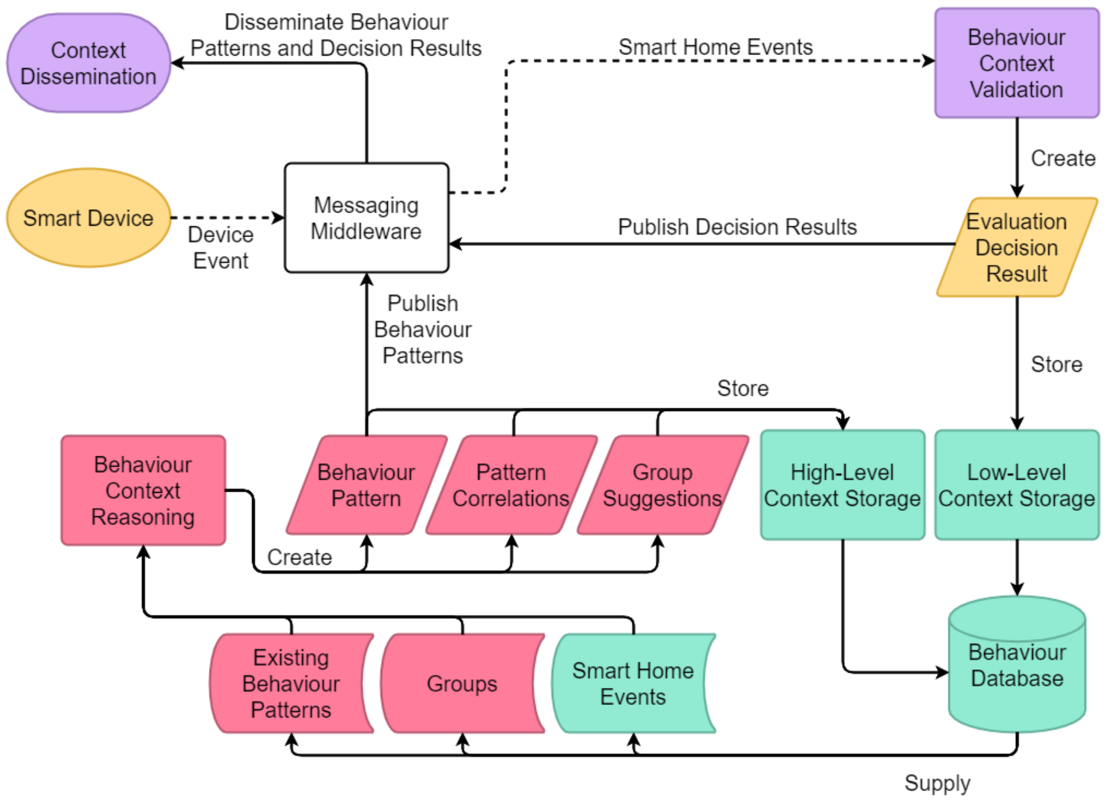
This work aims to provide a behaviour analysis for access control and is set in the smart home application COSYLab, which is why it has to adapt to the present environment and the context it offers. Identifying the useful context of an application or a system is a complex task with many possible starting points, which is why a systematic approach is recommended to determine context effectively [32]. Many different design methods have already been established with different granularity and restrictions regarding the environment they are set in [4].

COSYLab

This work is part of COSYLab, an application in a smart home environment that focuses on making the smart devices inside the home remotely accessible to its inhabitants. The central goal of applications like COSYLab is to provide the offered service to a legitimate user requesting the service at the time the service is requested and at the correct location with the correct device [47]. A crucial part of this goal is to ensure the legitimacy of the requests, which introduces a necessity for access control [81].

The COSYLab environment features an authentication process with a username and a password for logging in a user [20] and the session token resulting from the authentication [54] is used to determine the subject in access requests. Since authentication is provided by the environment, it is not examined closer in this work, which allows this thesis to be focusing on authorization.

The access control in COSYLab is based on attributes, which makes it an ABAC model. The environment features a messaging middleware based on message queues for the communication between components, which is used by components to publish new information to make it available to other components. The smart devices in the smart home regularly send updates about their states or about measured states in the system and can be remotely accessed, which means that smart sensors are part of the environment as well as devices that residents commonly interact with.



Context Dissemination
Disseminate Behaviour Patterns and Decision Results
Smart Home Events
Behaviour Context Validation
Create
Smart Device
Device Event
Messaging Middleware
Publish Decision Results
Evaluation Decision Result
Publish Behaviour Patterns
Store
Store
Behaviour Context Reasoning
Behaviour Pattern
Pattern Correlations
Group Suggestions
High-Level Context Storage
Low-Level Context Storage
Create
Behaviour Database
Existing Behaviour Patterns
Groups
Smart Home Events
Supply

Figure 5.1.: Top-level overview of the life cycle of context.

Design Methods

In this section, two intuitive and environment-neutral approaches are used to present the context-related part of this work. The life cycle of context model [73] presented in Chapter 3.1.2, is used to describe the design decisions made in regards to context in a well-structured way. The design method for the context model is CAMeOnto [2], which is a design method used to facilitate the search for the necessary contextual information and is described in Chapter 3.2.2. With these two models, the underlying smart home environment is examined and an in-depth context analysis is performed.

Figure 5.1 displays a top-level view of what is presented in this section in a flow chart. The parts involved in the context acquisition phase are coloured yellow, the context modelling is coloured green, the context reasoning is displayed as red, and the context dissemination is purple. Context starts its journey when it is created by smart devices and once the event is evaluated, the low-level context is stored in the behaviour database. Note that the previously established context is used to evaluate new context, which is why "Behaviour Context Validation" is coloured purple. The context reasoning is based on existing low-level and high-level context and its results are stored in the behaviour database as well. The evaluation results of new events and newly created behaviour patterns are disseminated to the smart home.

5.1.1. Acquisition Phase

The first phase in the life cycle of context deals with raw data directly gathered from its sources in the environment [73]. The crucial information this phase discovers is about how context can be extracted from the environment and about technical details such as the characteristics of the underlying messaging infrastructure.

The data produced in this phase is classified as primary context and the resulting design decisions of this phase need to provide extractable context [73]. A short summary of the most important points in this section can be found in Table 5.1.

Responsibility

The context is extracted from smart home events, which are pushed by smart devices in the COSYLab environment. This means that the components acquiring the context are designed to listen to new events for gathering contextual information [33]. The alternative to listening to events would be repeatedly requesting new context, which would be necessary if the environment was set up to support pulling information [73].

Frequency

An important distinction in the phase of gathering context is whether the context is obtained instantaneously or periodically with the main difference between periodic and instantaneous events being whether a certain point of time can be assigned to the event or not [73]. In a smart home, an instantaneous event represents an input such as a request to open a door, while a periodic event represents a measurement of a continuous state in

Where does the responsibility for acquisition lie?	The context is gathered from smart home events, which are pushed by the devices and sensors in the environment.
Is the context generated periodically or instantly?	The events of sensors are pushed regularly while the events representing user interactions are instant. Therefore, both types of context generation exist.
How does the underlying communication infrastructure look like?	COSYLab features a messaging middleware, which is responsible for the distribution of events.
What kind of sensor types are present in the environment?	Sensor types are distinguished by the way of data acquisition and the three types are measuring directly, querying other services, and publishing the result or a combination of the two. All three are present in the environment.
Is the context manually provided, sensed, or derived?	Smart sensors provide sensed data and the interactions between the residents and the smart home are manual input. Derived context is not interesting in the scope of this work.

Table 5.1.: Context Acquisition: Crucial Insights

the system such as a sensor repeatedly sending updates about the room temperature. Therefore, both types of frequency need to be considered by the design while obtaining context.

Source

The third design decision is based on the present communication infrastructure, which is used to obtain the context [73]. In COSYLab, a messaging middleware is used for the communication between the various components in the system and the events with context are exchanged over messages in the middleware.

One important responsibility of a middleware in a smart home is to provide an interface between context-gathering devices and other components in the environment, which do not necessarily work with the same data types [105]. Effectively, this means that the middleware is hiding the more fine-grained sensor details and making the context more accessible for other components [47]. This allows for a resulting design decision of simply adapting to the present messaging infrastructure to get the smart home events for the context acquisition.

Sensor Types

Data sources for contextual information can be distinguished by whether the resulting data is measured directly, like a thermometer does, or polled from other sources and published as data, like a calendar, or a combination of the two, like a device for gathering information about the weather [73]. All of the mentioned types of data sources are present in the underlying environment, which is why the design needs to be able to deal with all types of smart devices and sensors.

Data Types

The last design question in the acquisition phase is what kind of data is extractable in the environment and distinguishes between the types sensed, derived, and manually put in [73]. Sensed data is a common scenario with smart sensors and smart devices and manually put in data is present due to user interactions and configurations in the smart home.

The last form of data type is less interesting in the scope of this work since deriving data refers to external services being included or data being derived in the form of i.e. calculating the distance between two received locations by performing a mathematical operation. This means, that only two of the three data types are considered in the design.

In conclusion, the design decisions of the acquisition phase result in a smart home component listening to periodic and instantaneous events over a middleware communication infrastructure. Furthermore, the data sources can gather the data directly themselves or through polling external information or be a combination of the two and the types of data are raw sensor data or user inputs.

5.1.2. Modelling Phase

At the end of the first phase the raw sensor data is obtained and the next step is to represent the context as meaningful low-level attributes in the system. In this work, the design method to determine which contextual information is read out of the raw sensor data is based on CAMeOnto [2]. The benefit of using CAMeOnto is its environment neutrality, imposing very few restrictions on resulting context models.

CAMeOnto neither suggests nor imposes an order in which its interrogatives need to be used [2]. Therefore, they will be ordered to make this section easier to read and more understandable by starting with boundary context, then discussing the internal context, and at the end introducing external context. At the end of this section, there is a short summary in Table 5.2, presenting the crucial information gained by each interrogative of CAMeOnto [2].

What

The first question, what, asks for what exactly needs to be provided to the user and steers the designers to define design goals for the context model and to determine the boundary context [2]. This work is set in a smart home environment and for the first research question, the behaviour found in the environment needs to be represented by estimation models in the form of behaviour patterns. The second research question requires the context model to provide access control based on these behaviour patterns, which means that the service supported by the context model is an access control service.

Access control usually assumes that the virtual user sending a request is also the valid person behind the virtual user [20], which makes the system vulnerable to a malicious outsider attack [62]. Using behaviour as the basis for access control can help to minimize the threat of that attack pattern by comparing the current behaviour of a user against the usual behaviour of the user and if they mismatch, the system can take further actions to prevent harm.

The security of the smart home system can be further improved by comparing states found in the system against the usual states present in the system. To achieve this, the sensed states are monitored and compared to their respective expected reference value. If a significant difference is found between an expected state and a current state, the smart home can use its advanced options for security and warn the inhabitants or notify the respective public authorities [81], i.e. the fire department in case of extremely high temperatures.

Who

The second question in the CAMeOnto ontology encourages questions about the target users in the system, giving insight into internal context related to the users [2], i.e. how much technical knowledge can be expected of a user. In the case of a smart home application, the target user group cannot be narrowed down to a specific subset of people, because home appliances are in an environment used by everybody. Another insight gained by this interrogative is what information is necessary to distinguish the users. In the case of COSYLab, a simple identifier is sufficient.

Besides the direct information about a user, it is also useful to consider possible relationships between the users. This enables further meaningful context in smart homes since people living together also have a relationship with each other, which can influence their behaviour. The relationships between users are further discussed in Chapter 5.4.

Why

In CAMeOnto, the third question refers to the devices and the activities in the system by asking about the ways of the service being provided to the user. The question has to be asked like “Why is this service provided to the user” and the answer can be “The service is provided to the user because an activity is performed” or “The service is provided due to a device”.

The devices in the environment refer to the smart devices in the smart home, while the activities refer to the measurements taken by sensors and the interactions the users have with the devices. A more detailed look into the activities provides related external context such as descriptions of activities, context related to the activities, if and how the activities are related, or what information provided by other activities is influencing an activity [2]. In COSYLab, activities are called events and are divided by what triggered their creation. Passively created activities are called periodic events due to the represented sensor measurements being taken periodically while the activities created in response to user inputs are called control events due to the user making an input to control a device.

Periodic events represent measurements created without any user input by smart sensors and are defined by which sensor took the measurement, the value of the measurement, and the time the measurement was taken. Control events are created in response to user input, which represents the users interacting with the devices in the smart home and can be described by the user triggering the activity, the device involved in the activity, and the time the activity is processed. Events are described in more detail in Chapter 5.2.

Besides the activities, CAMeOnto furthermore encourages to identify additional external context related to the devices in the context model [2]. Similar to the information needed to identify a user, the information needed to identify a device is a simple identifier. Furthermore, there can also be semantic relationships between devices, which can be used to more accurately predict behaviour in the system. Relationships between devices are discussed in more detail in Chapter 5.4.

When

The behaviour of the residents in the smart home can be influenced by various temporal factors and in the scope of this work the temporal context is the central and most important context for determining the behaviour in the smart home. Periodic and instantaneous events in the system are expected to vary heavily at different times, even in shorter time intervals.

Periodic events represent regularly captured states in the system by a smart sensor and different states are expected to be found depending on the time of day, i.e. a temperature sensor will measure lower temperatures at night than during the day. Instantaneous events, which are also called control events, represent the interactions a smart home resident has with the devices in his home, which also depends heavily on the time of day since i.e. meals will usually be prepared before breakfast, lunch, and dinner. A working user is usually absent from home during working hours and will not have any interactions with the smart devices during working hours.

Considering bigger time intervals, behaviour also changes between the days in a week. A sensor will possibly find different states in the home on weekends compared to weekdays and even between the weekends and between the weekdays a lot of differences can be encountered due to various factors such as a flexible duty roster of a smart home inhabitant. An example for changing behaviour in the states of the system is the thermometer, which recognizes the heating of the rooms when a resident is present and since the presence of the inhabitants can differ between the respective daily schedules, the thermometer will measure different behaviour in different days. For control events, the same situations can be used to describe an expected change in behaviour. A smart home resident working or going to school is not at home interacting with devices during working or school hours, but during the same times of the day on weekends or free days, interactions are to be expected.

Another insight that can be derived from investigating the temporal requirements of the context model offers a possibility to group the behaviour. Control events are grouped into clusters, which are used in the reasoning phase to create the control event patterns, and the periodic events are divided into time slots, which are used in the periodic event pattern as reference points.

Where

Locations are commonly used in many context-aware applications and in the scope of a smart home, it can be very beneficial to include where the actions are taking place. In the scope of this work, the location of the smart home resident is assumed to always be in the appropriate location because COSYLab does not provide sensors to track the locations of the device and resident.

It is furthermore possible to include other location-based context types influencing the behaviour in the smart home. One example is the number of present users in one location, which changes for example how often a user is interacting with a coffee machine or which devices for entertainment are interacted with. This feature based on location-based context is only discussed as future work in the scope of this thesis due to the lack of support in the underlying environment.

5.1.3. Reasoning Phase

Similar to how other applications dealing with big amounts of data are facing performance challenges [108], using low-level context becomes unfeasible quickly due to the big amount of data produced in an environment with many sensors [73]. Due to runtime requirements set implicitly by the second research question of this thesis, the low-level context described in Chapter 5.1.2 is not sufficient to use behaviour properly.

In the scope of this work, it would be necessary to iterate through all previous events for each access request without a loss of QoE, since the user is waiting for the request to be processed. Therefore, computational time needs to be saved during an access request by introducing high-level context, also referred to as secondary context [73]. Multiple types of high-level context are introduced in the scope of this work and a short description of each secondary context type including a reasoning method is provided in this section. A more detailed description of every high-level context type can be found in their respective sections. This section serves as an overview of these context types and furthermore, short summaries of what is provided in this section can be found in Table 5.3.

What?	The service that shall be provided by the context model is behaviour pattern recognition in the smart home environment COSYLab, which sets the boundary context [2]. The ultimate goal of these patterns is to be used during access control, which would work especially well against malicious outsider attacks [62].
Who?	The users of home appliances are expected to have every possible background, which includes the possibility of them not having any technical knowledge [27]. Simple identifiers are enough to identify users in the system. Relationships between users are supported in the context model due to the smart home residents having a relationship with other inhabitants.
Why?	Activities in the smart home are divided into periodic events, representing sensors publishing measurements without user input, and control events, representing user interactions with the environment. Devices in the smart home are distinguished by simple identifiers and semantic relationships between devices are supported by the context model.
When?	The temporal aspect of the context model is the crucial context used to distinguish behaviour. In smaller time spans, the day is divided into time slots for periodic events, while control events get grouped into clusters to determine the interaction times. In larger time spans, the days in the week are expected to have different behaviour, which is why context is divided by the days of the week.
Where?	Context-based on location is often used in context-aware applications, but in this work, there is no location-based context due to a lack of support in the underlying environment.

Table 5.2.: Context Modelling: Summary of low-level Context

Behaviour Pattern

The analysis performed on the low-level context needs to estimate the behaviour in the smart home environment and store the results as prediction models capable of being evaluated against quickly. Therefore, behaviour patterns are introduced as a high-level context type in the form of persistent data models, which allows for the evaluation of newly incoming events to be performed against a behaviour pattern directly retrieved from the database. The behaviour patterns are described in more detail in Chapter 5.3.

The method of calculating the behaviour patterns needs to be decided, which includes the decision of how many methods need to be included [73]. There are different types of methods that can be included and the method used needs to be chosen based on what low-level context is available and on what the resulting high-level context should be. The smart home events are the present low-level context and the behaviour patterns are the target high-level context, which makes supervised learning [42] and unsupervised learning [9] seem like promising approaches.

For a supervised method of examining behaviour of users, supervision needs to be provided [42], which would have to be in the form of either the users themselves having to provide estimations of their usual behaviour or in the form of predefined examples by the developers, but both alternatives are rather poor options. Users are not trained for such a complex task and the reasoning would put an unreasonable effort on the users to work properly, while predefined examples provided by the developers before deployment are too error-prone since the daily habits vary heavily between users, which is why no global example fitting all patterns can be provided.

While including supervision leads to suboptimal choices, unsupervised learning is a more promising approach to analyzing the behaviour of the user. Without supervision, the application has to determine the patterns in the behaviour solely based on the events raised in the system, making the reasoning fully automated and independent from any further input [9]. More details about patterns can be found in Chapter 5.3 and a visual representation for a better understanding can be found in Chapter 6.10.

Pattern Correlation

The second research question requires behaviour patterns to be included in access control. Therefore, the policy administrator needs to choose specific patterns for the evaluation process, which is a role usually filled by the homeowner in a smart home environment. For this reason, the user needs to be provided with information on which the decision which pattern to choose can be based.

Besides a visual representation of behaviour patterns displaying an understandable way of what a pattern allows and disallows, an additional indication of when a pattern should be replaced needs to be provided. This is necessary due to the nature of behaviour being approximated over time [10] and the possibility of the behaviour adjusting to external factors. Therefore, an additional high-level context type is created from behaviour patterns by comparing them with each other to express their degree of similarity.

The design decisions for pattern correlations need to be based on how to convey the information as easily as possible to the user. A proper indicator of how similar two patterns are displays the similarity in percent because this is a universally understood metric. This indicator is called the correlation between two patterns and fuzzy logic [112] promises to provide proper reasoning methods for determining a degree of similarity. The previously used unsupervised learning methods are suited for creating user profiles or to find patterns in unlabeled data [73] but are less recommended in this use case. Correlation is discussed in Chapter 5.6.3, where more details about the integration of the features are provided.

Group Suggestion

The context model supports relationships between users and devices, which are introduced as device groups and user groups respectively. Furthermore, these relationships are integrated into the access control process as well. Details about the feature introducing groups are discussed in Chapter 5.4, with more details about what can be grouped and how groups are evaluated in the access control model is discussed in Chapter 5.5 and Chapter 5.6.

One important thing about the groups is that they are completely managed by the users, which can be challenging for some users. To facilitate group management, suggestions for groupings are created as a high-level context type and are provided as two separate values indicating to what degree a relationship is determined by the system. The two values represent how many common interaction counterparts a considered subject or object has and how many counterparts would be possible. This means, that the two values indicate how many devices two users commonly use or how many common users two devices have. This metric, which is created with fuzzy logic [112], is used because it is more intuitive than a percentage value.

5.1.4. Dissemination Phase

In the fourth phase of the life cycle of context, the resulting high-level context of the third phase and the resulting low-level context of the second phase needs to be made available to the system [73]. COSYLab uses a messaging middleware for the communication between the components, which is also used to bring the context, that needs to be distributed, to the components consuming it.

In the scope of this concept, it is mainly important to disseminate the behaviour patterns, since they are the main feature introduced and since the behaviour patterns are also created to make the gained insight easily accessible. The other high-level context types also provide useful information for the user and for the other components, but disseminating the low-level information does not give as many benefits, since the raw context is a huge amount of data that cannot be processed in real-time [108].

The higher-level context types presented in this work are distributed by being published and other components can subscribe to message queues for the information [33]. The evaluation results used in access control are furthermore made available in queries to

Behaviour Pattern	Behaviour is evaluated in runtime for access control requests, which is why behaviour patterns are required to be evaluated quickly to preserve the QoE for the users. Therefore, the behaviour patterns are created with unsupervised learning methods [9] and stored persistently so that they only need to be retrieved and evaluated against for access control.
Pattern Correlation	Correlation serves as an indicator of how similar two patterns are, so that policy administrators are aided in their decision on which behaviour pattern to set. The correlation is displayed in percent and uses fuzzy logic [112] in the analysis process.
Group Suggestion	Groups are a feature introduced to support the access control and are completely managed by users. To aid the users in the management process, group suggestions are created with a fuzzy logic method indicating a degree of relation [112].

Table 5.3.: Summary of the high-level context types with reasoning method

make the provided features properly usable to fulfill the requirements of behaviour being estimable at runtime given by the second research question.

5.2. Events

Smart home events are used for the communication between components in the environment and a middleware architecture is used for the exchange of messages. In the scope of this work, smart home events hold the raw data gathered in the first phase of the life cycle of context [73], and a messaging middleware is responsible for the context acquisition [47].

Two types of events generated in the environment are interesting since these are the events needed to create estimations about the behaviour of the users and the system. The two types of events are called “control events” and “periodic events” and are semantically divided by whether a device or a user has triggered them. The first research question of this work requires the behaviour in the smart home to be translated into behaviour patterns. In the smart home environment, user behaviour is represented by control events, and system behaviour is represented by periodic events. Each event encompasses a set of low-level contexts, which is also referred to as the metadata of the event. Which context types an event has depends on the type of the event and a general overview is given in Table 5.4.

Event Type	Context Type		
	User	Device	Temporal
Control Event	yes	yes	yes
Periodic Event	no	yes	yes

Table 5.4.: Events and their metadata, a set of low-level context

5.2.1. Control Events

The smartphone became ubiquitous in the daily life of most people [13] and being able to control the smart devices in the home remotely with it is a common benefit of smart homes, facilitating the daily activities of users [80]. Control events are classified by their characteristic of being created in response to user interactions. Whenever a smart home resident uses his smartphone to remotely interact with a device in the home, i.e. signaling the coffee machine to make a cup of coffee, a control event is raised.

The crucial context types in a control event are the initiating user, the device on which the event is performed upon and the time of the execution itself. These three low-level context types serve as the metadata of control events.

5.2.2. Periodic Events

In an IoT environment, the devices, also called Things, communicate with each other [101] and since smart homes are IoT environments, devices found in a smart home send various information such as their status, in the form of an event to the system [80]. All events created by devices in the environment are referred to as “periodic events” in the scope of this work. Periodic events are regularly created by smart devices and are either updates on their status or measurements of smart sensors.

Periodic events have the same context as control events, except that periodic events do not include user information. This leaves periodic events with the identifier of the device it was created by and the weekday as metadata. Besides their metadata, periodic events also include a timestamp and a numerical value, if the event is a measurement by a smart sensor.

5.3. Behaviour Pattern

Behaviour patterns represent the translated behaviour as a high-level context type in the smart home environment and they are used to represent this behaviour in the access control process. This makes the behaviour pattern a central component in the answers to both design-related research questions. The first research question is provided with the requested way of translating behaviour into context-aware behaviour patterns while the second research question is provided an estimation model usable during the access control request.

The two important processes behaviour patterns are involved in are the analysis, the process creating them, and access control, the process using them. Details about the analysis process are provided in this section, while the details about the decision-making process are discussed in Chapter 5.5. Furthermore, Chapter 5.6 includes information about the integration of the behaviour patterns into ABAC.

Analysis Schedule

Since behaviour is defined by repeatability [10], big changes affecting the behaviour are observable due to the repeated diversion from existing patterns and need to be analyzed and integrated as soon as possible into behaviour patterns. Therefore, behaviour patterns are updated regularly, resulting in a new version of every pattern being released periodically after a set time interval. The only restriction the nature of the behaviour patterns puts on the analysis schedule is that behaviour patterns need to be ready before the time span in which new events for a pattern can appear.

Event Grouping

A separate pattern type is introduced for both event types since different types of behaviour are represented by each event type. Due to their differences, the analysis and the evaluation process need to be different for the pattern types. Dividing events into control events and periodic events is not sufficient to distinguish behaviour meaningfully, which is why events need to be grouped in the analysis process.

During the analysis process, the events' metadata is used for grouping, and for each of the resulting sets of metadata, a distinct behaviour pattern is created. All related events belong to exactly one pattern and have exactly matching metadata, which is also referred to as their metadata being considered as equal. Furthermore, low-level context is not shared between behaviour patterns.

The metadata used to group events is designed to include the crucial contextual factors, which cause significant and observable differences in the behaviour if one of these factors is exchanged. The reason for this design decision is that each set of metadata is treated separately in the pattern creation process. Therefore, newly created events are assigned to a specific pattern based on all of the factors given by its metadata and evaluated against that pattern. The pattern creation process is based on all previously created events available at the time of the analysis.

Access Control Considerations

Waiting times for responses of systems need to be minimized and operations performed on every request in a network application should especially be as efficient as possible [21]. Since behaviour patterns are used during access control as requested by the second research question, they need to be quickly accessible and processible at runtime, which is why they are stored persistently as prediction models rather than the unfeasible approach of always reanalyzing all available low-level context [108]. Due to this design decision, the reasoning and dissemination phase are completely independent of each other [73].

5.3.1. Control Event Pattern

The daily routine of a person is a very complex schedule involving a lot of different locations, devices, and activities. Control event patterns are responsible for capturing the behaviour determined by interactions a smart home resident has with the home environment. Tracking the behaviour of the users is based on control events since each control event represents one interaction the users initiated with the system. Putting all control event patterns of a specific user together can be considered the user profile of that user. Table 5.5 presents the crucial details of control event patterns.

Smart home environments have various available devices with distinct semantic usage [80]. Therefore, a unifying analysis approach is needed that is applicable independently of device type and results in a control event behaviour pattern equally serving as prediction models for all device types.

The unifying approach of analyzing user interactions is based on the interaction times, resulting in a device type independent user profile for each resident in the smart home. A fitting mathematical model needs to be selected to achieve the goal of the analysis process as part of the reasoning phase [73]. Providing references is suboptimal as described in Chapter 5.1.2 and therefore, the behaviour patterns are created with unlabeled and unstructured data, which provides a ground for unsupervised learning [73].

K-means clustering [99], an unsupervised learning method [9], is used to create clusters within all control event groups based on the timestamps of the control events. The algorithm is set up to create as few clusters as possible and if predefined acceptance criteria aren't met, the algorithm is repeated trying to create a proper clustering with one additional cluster.

The acceptance criteria for a cluster require it to stay under a threshold regarding the number of events that are too far away from the center. The resulting usage clusters are also referred to as control event peaks, control event pattern peaks, or usage peaks. A visualization of a control event pattern can be found in Chapter 6.10.2 in Figure 6.6.

5.3.2. Periodic Event Pattern

Periodic event patterns represent usually found states in the environment that can be measured by a numerical value, i.e. measured by thermometers, humidity sensors, light sensors, etc. Note that there are more types of periodically sent events in the smart home environment such as updates about the current state of the device, but only periodic events containing numerical values are analyzed due to their comparability. Table 5.6 presents the crucial details of periodic event patterns.

A smart home can have any number of smart sensors deployed, which do not necessarily measure the same state or in the same unit. Therefore, it is necessary to apply a unifying analysis approach independent of measuring unit or device type, which produces an equally valid prediction model for any kind of underlying measurement type.

Reasoning Method	The reasoning method used to create control event patterns is K-means clustering [99].
Semantic Representation	One control event pattern represents the interaction one user has with one device on a specific day of the week.
Insight	Control event patterns provide the expected interaction time spans in the environment.
Components	One control event pattern consists of usage clusters, which feature two values: their usage center, a timestamp within a day, and a standard deviation.
Accuracy	The usage times of the different devices are not going to be predictable to the minute. This is why the standard deviation is important for a usage cluster.

Table 5.5.: Control Event Pattern: Crucial Details

Measurements by smart sensors are continuously taken throughout the day, which is why clustering algorithms do not give any meaningful insight into periodic events. Instead, periodic events are divided into fixed time slots throughout the day, effectively creating an interpolation. Each time slot has the same size and together the time slots cover exactly a whole day, which introduces a linear distribution model for the time slots.

Besides the timestamp used to determine the time slot an event belongs to, periodic events furthermore encompass a numeric value from the measurement. The numerical value is used to calculate an average value and a standard deviation for every time slot. Using standard deviation and average value assures that periodic event patterns work equally for each smart sensor regardless of the amplitude and variance of the data. The reference values in the time slot and the time slot itself in a periodic event pattern is also referred to as periodic event pattern value. A visualization of an example periodic event pattern can be found in Chapter 6.10.1 in Figure 6.5.

5.4. Semantic Groupings

Humans have certain relationships with each other, which potentially impact their behaviour, i.e. being family or work colleagues. For a context-aware application aiming to analyze behaviour of users in the system, the ability to include relationships between users is highly beneficial. Furthermore, devices can also be grouped based on their similarities and relationships, i.e. their physical location or their function.

Reasoning Method	The reasoning method used to create periodic event patterns is a calculation of the average value and its standard deviation for each time slot.
Semantic Representation	Periodic event patterns represent what one sensor usually measures during a specific weekday.
Insight	Periodic event patterns give insight into what states are usually observable throughout the day in the smart home environment.
Components	Periodic event patterns encompass a set of reference values for measurements with standard deviations, which are divided into several time slots.
Accuracy	By providing several values, the periodic event pattern allows newly created periodic events to be compared against a localized value instead of a global value, allowing for higher accuracy.

Table 5.6.: Periodic Event Pattern: Crucial Details

So far the context model has covered low-level information, encompassing only data directly accessible in the events, and one form of high-level context, behaviour patterns for user interactions and smart sensors. Control event patterns represent the relationship between devices and users, which is not the only relationship found in the underlying environment. In this section, groups are introduced as relationships between users and between devices respectively as another form of high-level context.

Purpose of Grouping

Until now, behaviour is modeled by assigning one behaviour pattern to a specific situation. This means that new events are assigned to a behaviour pattern based on their metadata. User groups furthermore enable the consideration of every behaviour pattern matching the metadata after the requesting subject is replaced with a different subject from the group. Device groups work the same way but replace the device context in the metadata.

Note that including multiple patterns is only intended for control events. This feature does not work for periodic events, because they include measurements with numerical values, and due to the heterogeneous nature of the data of smart sensors in the smart home [85], the resulting pattern cannot be used interchangeably in a meaningful way.

Management Considerations

It is possible to determine relationships between entities by analyzing common behaviour, but the knowledge of similar interactions within the system does not imply what kind of relationship two users or two devices have with each other. This could cause errors such as children getting grouped with their parents, a scenario in which children might get access to devices they are not intended to have. Therefore, it is a necessary design choice to put the users in charge of the groups. The users are aware of what relationships exist in the environment and can create meaningful groups based on that information, which makes the system less error-prone.

Both group types are totally controlled by the users in the system, including creation, maintenance, etc. Therefore, the feature must be made understandable and usable for the average smart home resident, since the users in a smart home environment can have any possible background [27].

Group Suggestions

To help users, who do not necessarily have a deep understanding of smart homes or IoT, it is necessary to present aid in a commonly understandable form. One high-level context created in the reasoning phase [73] to provide this help is group suggestions, indicating how many devices are commonly used between two users or how many common users two devices have. Even though behaviour has so far only been defined by behaviour patterns, which are created using unsupervised learning [9], groups are adding a supervised element to the observed behaviour and the ACM. How groups are involved in the ACM is described in more detail in Chapter 5.6.

As mentioned in Chapter 5.1.3, a proper way of displaying context needs to be chosen, which means a readable and understandable way for users needs to be chosen. While a percentage is understood by users, it would be confusing to display how much the behaviour of two users or two devices have in common as a percentage value, which is why the method of common users or common device usages is chosen.

5.4.1. User Groups

Humans have many different relationships with each other and forming groups is a natural characteristic of people in society [7]. Some people are related, others are friends or are living together. These associations between users are represented by user groups and are integrated into the behaviour of one user by including the behaviour of associated users.

The shared goals, interests, activities, and routines of groups of people like families lead to similarities in behaviour such as the routines and tasks built around the family commonly having dinner together. There are several different scenarios in which behaviour of other people in a home can be used in estimations for the behaviour of a person.

Absent Resident

Many example scenarios in which user groups are useful include the absence of a resident or at least the temporary inability of a person to carry out a task. When a person is on vacation, housework usually done by this person needs to be done by another person, which means in many cases that already tracked routines can be used as behaviour patterns for the substituting person. Furthermore, behaviour patterns of infrequently substituting residents take much longer to form due to the data being collected much slower.

A dishwasher that is always used after dinner by the same person will, even if that person is absent, likely be used after dinner. In this case, the tracked behaviour of the substituting person likely does not recognize the transferred task as expected behaviour without including the already established behaviour pattern of the absent person.

Fixed Time Activities

Tasks that are highly inflexible in regards to when they can or have to be performed benefit from the inclusion of user groups. The preparation of breakfast in the morning of a family has to be done in time for the children to eat before going to school. One adult person potentially prepares the breakfast more often than the others, even though all adult residents in the smart home can carry out the task.

In this case, the behaviour pattern of the person performing the task more frequently becomes accurate faster than the behaviour patterns of the other users. By using the behaviour pattern that becomes reliable first also for other users in the group nullifies the additional learning time of all other patterns once the first pattern captures the behaviour.

Shifting Chore Schedule

Chores are often divided by weekdays with a planned schedule, in which every weekday a different user can be assigned to carry out the task. In a situation, where another resident takes over a task, a new pattern needs to form for the resident the task is transferred to. Then, the process of a pattern forming needs to be gone through again, because the behaviour is determined as unusual for some time after the handover. This additional learning time is completely avoidable and unnecessary, since the behaviour itself did not change, but was transferred to a different resident.

5.4.2. Device Groups

Devices can be related to each other in meaningful ways such as using their physical locations or semantic similarities to draw connections between them. Using device groups gives generally the same benefits as using user groups by making behaviour patterns more flexible in many real-life scenarios, shortening the learning times of behaviour patterns.

Reasoning Method	There is no reasoning method necessary for the creation of groups since they are not created in an analysis process. Group suggestions are created by using fuzzy logic [112].
Semantic Representation	Groups represent the relationships between users and the relationships between devices.
Insight	Groups enable the insight gained by other behaviour patterns to be shared between them.
Components	A group consists of all behaviour patterns associated with the entities in the group.
Management Responsibility	The users of the system are responsible for the management of the groups, which includes their creation.

Table 5.7.: Groups: Crucial Details

Location

Building groups of devices based on locations needs consideration of granularity and abstraction level [49]. In a smart home, high-level abstractions such as the room where the device is located serve better as the basis for a device group than low-level abstractions such as the physical distance between two devices.

When a resident is in a location or zone a device group is located in and interacts with a device, then interactions with other devices of the same group are likely, i.e. a resident using multiple kitchen devices while cooking. The assumption for the pattern sharing to be meaningful is that if a behaviour pattern indicates interactions between a device and a user, then it furthermore indicates user and device to be close during these usage times as well. Therefore, it is likely that further interactions with other devices of the group take place.

Use Case Similarities

Devices with similar use cases enable multiple scenarios in which the information about their similarities improves the reliability of behaviour recognition. Creating groups based on semantic similarities might be less intuitive since related scenarios lean towards a device being used as an alternative for another device in the group instead of devices being used together like in scenarios where groups are based on physical location.

One example scenario featuring semantically similar devices is a resident using multiple devices in a daily leisure time span. Some devices can have a fixed time schedule such as the resident watching a TV series every day from 7 pm to 8 pm, while the other devices are used more flexibly during the remaining leisure time. In this scenario, a resident having a fixed leisure time span observable between 7 pm and midnight might decide to watch a movie after his favourite series finished.

Deviating from usual behaviour, i.e. watching a movie after the favourite series at 8 pm, is considered as deviating behaviour, even though the inhabitant always performs recreational activities during that time span. Furthermore, behaviour patterns usually form over several weeks after enough data is available and this process is delayed for devices that are not always used. Treating the behaviour patterns as completely isolated causes the information of the residents' free time on that day to be approximately between 7 pm and midnight to be lost. Putting the devices into a group is implicitly providing this knowledge.

5.5. Behaviour Evaluation Decision

Behaviour has so far been discussed with a focus on the first research question about translating behaviour into context-aware behaviour patterns and different features supporting this translation have been introduced. The second research question deals with including the translated behaviour into an ACM and this section deals with a central part of the supported ACM, namely the evaluation of smart home events. Note that the details of how exactly the evaluation of events is integrated into the ACM are described in Chapter 5.6.

Whenever an event is raised in the smart home environment, it is necessary to assure that the event is valid by being raised by a legitimate source and acceptable by displaying a desired state in the system. An ACM needs to enforce an interaction in response to it being requested by a resident, while a malicious outsider [62], impersonating a resident [53] and requesting an interaction with a device in the home needs to be blocked. For this reason, the ACM includes an evaluation process to determine whether an event in the system is valid or not [88].

Decision Result

The output of the decision-making process is called the decision result and consists of a classification, which is introduced in this section, and a probability, which is a numerical value in percent, indicating to what degree the event is matching the usual behaviour. The main purpose of the decision result is to indicate whether the system needs to react to an event, which can be a suspicious request from a seemingly valid user account or a suspicious state captured by a smart sensor. In the scope of this work, the decision-making process is also referred to as the evaluation process.

False Negatives

It is important to consider that patterns are statistical approximations of the real behaviour of users and must be used as prediction models and not as forced schedules. Humans are always able to make unpredictable choices based on the available context, which cannot always be predicted by a context-aware system [12]. A human who is sick might show differences in his interactions with smart devices such as using the water boiler in unusual frequency. The user needs to be able to perform any or no actions at all at any time in the smart home even if the resulting behaviour mismatches the predicted behaviour [12]. This introduces a requirement for additional mechanics providing control over the system for the user [27].

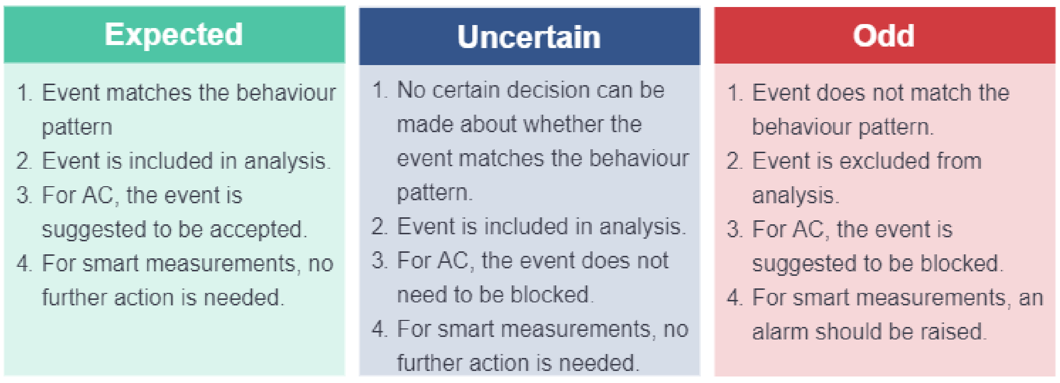
Assuming a person usually cooks between four and five in the afternoon, but after being rescheduled at work, the person starts coming home two hours later that day from work, which also shifts the cooking schedule. Ideally, behaviour patterns adapt after a few analysis cycles, reflecting the new usage times for kitchen devices, but if all usages of these kitchen devices get rejected and excluded from the analysis by the system, the new usage cluster can never develop.

It is desirable for events to be classified falsely as rarely as possible, since residents of the smart home would start to feel less safe and even controlled by the system instead of in control of their home [27]. While behaviour patterns need to detect deviating behaviour, an oversensitive evaluation creating many false negatives would downgrade the QoE, since users would wrongly be blocked by the system too often. Identifying and blocking every malicious event is obviously the goal, but a bad precision, which is a low rate of true positives compared to all positives [28], would lead to constant false alarms by the system.

Groups are one feature introduced to minimize the false negatives by using behaviour patterns interchangeably when appropriate, as described in Chapter 5.4. Another feature needed for the user to not be controlled or locked out of the smart home system [27] is external validations, which allow the user to override the decisions when a false negative is found and are introduced in this section.

5.5.1. Classification

The goal of the decision result is to represent the information of whether an event matches a pattern or not and the classification reflects this by providing different categories for events. An overview of the possible classification categories can be seen in Figure 5.2.



Expected	Uncertain	Odd
1. Event matches the behaviour pattern 2. Event is included in analysis. 3. For AC, the event is suggested to be accepted. 4. For smart measurements, no further action is needed.	1. No certain decision can be made about whether the event matches the behaviour pattern. 2. Event is included in analysis. 3. For AC, the event does not need to be blocked. 4. For smart measurements, no further action is needed.	1. Event does not match the behaviour pattern. 2. Event is excluded from analysis. 3. For AC, the event is suggested to be blocked. 4. For smart measurements, an alarm should be raised.

Figure 5.2.: The three possible classifications for a decision.

The classification is a property of the decision designed a scale ranging from rhythmic, in this scope referred to as expected, to random, in this scope referred to as odd [49]. The purpose of the renaming of the values in the scale is to adjust for better intuitiveness for an access control system.

It is important to exclude odd events from the process of analyzing the existing events to create behaviour patterns because if they were included a malicious outsider would only have to repeatedly trigger the same events until these events change the behaviour pattern enough so that the attacker can interact with the system like a legitimate user.

5.5.2. External Validation

It is always possible that a valid event gets falsely classified as odd, which happens in the expected scenario of a user deciding or having to deviate from the usual behaviour [27]. Therefore, the user is given the possibility to override evaluation decisions. It is important for applications to not lock out the user [27], which is especially true in smart home environments, since it deals with home technology and sensitive data [6], and in context-aware applications, which are complicated to understand for a user [12]. Users might have trouble understanding what exactly leads to a control request being blocked and therefore, users are given the right to override evaluation results, putting them in charge of the evaluation process.

Another important aspect of external validations is that they change the classification of an event into “expected”, including the event in the analysis process. Without external validation, behaviour patterns would be rigid and would disallow big changes due to them discarding all deviating events, which are necessary for patterns to adapt. This would make the behaviour patterns useless after the first time the life of the users goes through a big change.

5.5.3. Processing an Event

The process of evaluating an event is visualized in Figure 5.3 and the three steps are further described in this section. Before the first step, the evaluation starts with a decision based on whether a corresponding behaviour pattern for the processed event exists.

The corresponding pattern is chosen based on the metadata of the event. If no corresponding pattern exists, all other steps are immediately skipped resulting in a classification of “uncertain”, which includes the event in the analysis, enabling a corresponding pattern to be created.

External Validation

The first step in the evaluation process is to check whether an event recently evaluated against the resolved pattern was externally validated, indicating that the evaluation resulted in a false negative. Further prevention of false negatives can be achieved by simply marking a behaviour pattern as externally validated for a reasonable period of

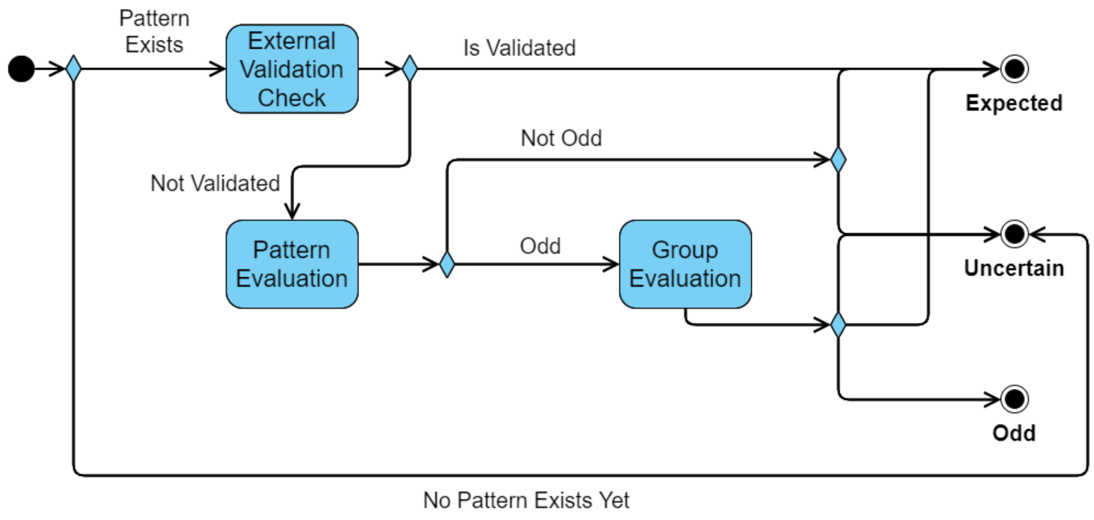


Figure 5.3.: General Evaluation Process of a smart home event

time. During the time a pattern is marked, all events are automatically treated as externally validated, which results in a classification of "expected".

A user can likely interact multiple times with a device in a short time span, which would cause all resulting control events to be classified as "odd". In this case, all of these events would need to be externally validated, which is unnecessary if the user validated the event related to the first performed interaction. For periodic events, it is undesirable to send multiple suspicious state warnings, once the resident confirmed that the unexpected state is valid.

Pattern Evaluation

The second step in the evaluation process is the pattern evaluation, where the processed event is compared against the behaviour pattern to check to what degree the event matches the resolved behaviour pattern. Both event types are evaluated against different behaviour pattern types, which leads to them furthermore having to be evaluated differently. The evaluation of a control event is based on usage clusters in the corresponding control event behaviour pattern and the evaluation of a periodic event is based on reference values in the time slots of the periodic event pattern.

Control events include a timestamp in their low-level context, which allows reading the time of day the interaction is taking place. This time of day value is used to determine the closest usage cluster of the control event pattern and the event is compared against the resulting cluster. The evaluation against a usage cluster is quickly processed due to clusters being defined by only two numbers, their center, and their standard deviation.

A timestamp is also included in periodic events, which is used to extract the time of day the measurement is taken. Periodic event patterns are divided into fixed time slots and the time of day of a periodic event determines to which time slot it gets compared against. Each of the time slots has a reference value, representing the expected state, and a standard deviation, representing a confidence interval. The evaluation against a periodic event behaviour pattern only requires a comparison between the reference value in the respective time slot to the value of the measurement.

For both types of events, a distance is compared to a standard deviation. If the distance is smaller than the standard deviation times one, the event is classified as “expected”, if the distance is bigger than that but smaller than the standard deviation times two, the event is classified as “uncertain” and if the distance is bigger than that, the event is classified as “odd”.

Group Evaluation

Groups are only used in the evaluation of control events as described in Chapter 5.4. Therefore, this step is skipped for the evaluation of periodic events. If a control event is classified as odd at the end of the pattern evaluation phase, the event might still have been falsely declined. Especially during the initial learning time of a behaviour pattern, it might produce many false negatives. Introducing the feature of putting users and devices into their own respective groups aims to reduce the learning times for behaviour patterns.

Groups help reduce the number of false negatives detected in the evaluation of events by including more than one behaviour pattern in the evaluation process. This might change the decision result into accepting the event, improving the feeling of safety due to the system not second-guessing the user [27], which improves the QoE [18].

Group evaluations work by using all behaviour patterns of a group interchangeably. This means, that an event gets evaluated once against each pattern in the group. Then, the best result is chosen as the result for this step, which is simultaneously the final result of the evaluation process.

5.6. Smart Home Integration

So far, this chapter has mostly dealt with the first research question, discussing in detail how behaviour is translated into a context-aware behaviour pattern. The second research question requires the behaviour pattern and other introduced features such as groups to be integrated into the ABAC model [45] of COSYLab. This section deals with the design details related to the second research question and the integration of the design into the environment on a conceptual level. The realization details of the environment integration are not part of this section and are discussed in Chapter 6.

The monitored behaviour in the smart home environment is represented by the behaviour patterns, which are described in Chapter 5.3, and subsequent events in the smart home are evaluated against these patterns as described in Chapter 5.5. The integration of behaviour patterns in the underlying environment’s access control is realized through attributes [45]. An access policy in the ABAC model requires an attribute to be comparable to a reference value. Therefore, an access rule needs to include a proper way of comparing behaviour indicated by a smart home event to a specific behaviour pattern. Simple examples for an attribute set in an access rule work with subject attributes, i.e. requiring the age of the requesting user to be equal to or greater than eighteen.

In this section, two attributes for the direct inclusion of behaviour patterns into ABAC are introduced. Furthermore, one additional attribute is presented, which is related to behaviour patterns but cannot be the sole base of a policy, because of a lack of user involvement. Then, one aid to users for choosing which behaviour patterns to set in their policies is presented in the form of pattern correlations. Furthermore, this section describes the messaging interfaces of the design.

5.6.1. Access Control Attributes

In ABAC, attributes are used in an access request by comparing the value determined while processing the request with a target value set in the access rule [113]. For the second research question, user behaviour needs to be evaluated against its respective behaviour pattern and the evaluation result needs to be made available to ABAC. User behaviour is estimated by requests made by humans to interact with a device in the home, also referred to as control events as described in Chapter 5.2.1. When a control event does not match the usual behaviour of the subject, the system needs to be informed about the discrepancy to take further action.

The nature of the decision results enables two different rule types, namely rules based on numerical comparisons and rules based on string comparisons. For the scope of this work, one of the two options needs to be chosen. String comparisons utilize the classification of the decision result, while the numerical comparison utilizes the probability estimation of the decision result. The numerical option is chosen due to it being more flexible from the user's perspective compared to the alternative, which would restrict the user to a set of predefined values.

For creating a rule including a behaviour attribute, the residents need to choose a specific pattern, which is then used to evaluate all subsequent smart home events with matching metadata. This design decision is a necessary measure to prevent access policies from changing due to the regular releases of patterns as described in Chapter 5.3.

The attributes share the prefix "Context.Behaviour.Pattern.Device" due to the classification given by the ABAC model of the smart home and are distinguishable by their suffix. Two attributes that can be used without any further prerequisites, namely:

- Singular Pattern Evaluation with suffix "Single.Control-Based"
- Group Pattern Evaluation with suffix "Group.Control-Based"

The first attribute covers the integration of a single control event pattern into ABAC. A rule with this attribute requires a specific pattern to be chosen, which is used to evaluate an access request. As a comparable value for the rule, a target confidence value needs to be set for ABAC to compare the result of the evaluation process as described in Chapter 5.5 against. Semantically, a rule of this type represents use cases such as "Behaviour of John Doe with TV on WEDNESDAY equals <target pattern> greater than 30%".

The second attribute is responsible for the integration of groups of control event patterns. This attribute utilizes the groups discussed in Chapter 5.4 for the evaluation process, allowing multiple patterns to be included in one access control request evaluation. An access rule written using this feature includes the same information as the first attribute with one addition, which is an identifier for the group. This attribute semantically translates several concepts to access rules such as "Grouped Behaviour in Adults of John Doe with TV on WEDNESDAY equals <target pattern> greater than 50%" or "Grouped Behaviour in Kitchen Devices for John Doe with Stove THURSDAY equals <target pattern> greater than 30%".

5.6.2. Additional Behaviour Evaluation

System Behaviour alone is not interesting for access control due to it not including a subject but it is still possible to improve the security of smart homes based on periodic events. User behaviour can be influenced by various states in the system, which is why periodic event patterns are still potentially beneficial as building blocks for more complex access policies [81]. An example of an event raising an alarm in the system is an unnaturally high temperature in a room in the home since this could indicate a fire.

The third attribute introduced in the scope of this work serves to evaluate system behaviour by exposing the evaluation of measurements of smart sensors against periodic event patterns introduced in Chapter 5.3.2. An access policy cannot consist of only one rule of this type, because no subject is attempting to access an object in the context of a smart sensor taking a measurement. Instead, rules of this type are meant to be used as building blocks for complex access policies, allowing the inclusion of information about states in the system during the evaluation of other access rules. The third attribute has the name:

- Singular Pattern Evaluation with suffix "Single.Monitoring-Based"

Periodic event patterns allow the access control model to evaluate whether states measured by sensors in the smart home are within an expected range of values, which is determined by a confidence interval around an expected average value. In access rules, this attribute is used by writing rules like "Device Behaviour Pattern for WEDNESDAY equals <target pattern> greater than 50%" and as target pattern, the access rule writer chooses one of the applicable patterns. Note that the device does not have to be specified for the access rules in the underlying access control model, since they are implicitly associated.

5.6.3. Behaviour Pattern Correlations

Smart home residents are responsible for defining and maintaining the access policies in their homes. Once a behaviour pattern is set to be used in an access control policy, every new event raised in the smart home environment with matching metadata is evaluated against that pattern, even after new versions of the pattern including more data are released.

Behaviour patterns are released regularly to adapt them to changes of the estimated behaviour [10], as discussed in Chapter 5.3. Therefore, policies setting a specific behaviour pattern need to consider an indication of when a rule needs to be updated in its design.

Replacement Recommendation

After the initial creation of access control policies, the residents decide when an access policy containing an outdated behaviour pattern is replaced. It is necessary to support users in making this decision due to the unintuitiveness of this task [12]. The time span after which patterns are recommended to be replaced cannot be estimated generally, because it differs between patterns due to several factors such as the frequency of usage of the related device. A better strategy to suggest an update of the policies to the user is to calculate statistical differences between newly created patterns and the currently used patterns, creating an indicator for a replacement becoming necessary.

A new pattern is published regularly for every set of metadata including all events that have been raised in the smart home so far, which means that the new pattern has more data than the previous behaviour pattern with the same metadata. After a few analysis cycles, the newer pattern might not resemble the old patterns anymore, but how much two behaviour patterns are similar needs to be evaluated. Since this is a metric that will be shown to the user, the design emphasis is put on choosing a metric commonly understood, which is why the similarity is presented in percent. In the scope of this work, correlations are used only as an aid for policy administration, which is why pattern correlations is a feature considering only patterns with matching metadata.

Calculation

Correlation is calculated between patterns with matching metadata as part of the context reasoning phase [73]. There are two types of behaviour patterns and both are treated differently when the correlation is calculated between two of their patterns. The most important factor of how similar patterns are is how much they cover each other, which means to what degree they will accept the same events.

Periodic event patterns are divided into time slots with each time slot having expected values and confidence intervals for the numerical values in measurements for smart sensors. The correlation calculates an average over the distance between the expected values in relation to the confidence interval for each time slot of a pattern compared to the respective time slot in the correlating pattern.

Control event patterns consist of usage clusters, which have a time of day center and a standard deviation, and whether an event is matching is determined by the distance to the closest cluster. Similar to periodic event patterns, control event patterns are similar if their usage clusters are covered in the other pattern. Each cluster in both pattern types is compared to the closest previous and the closest subsequent cluster in the other pattern, calculating an average in percent of how much each cluster is covered in the other pattern gives the similarity between the two control event pattern.

5.6.4. Configuration Options for Residents

Configuration interfaces are responsible for adapting the operations to external needs and for providing information, that is not obtainable otherwise. The interfaces are provided separately because grouping them by their responsibility facilitates the development and enhances the comprehensiveness of the design. There are a total of four configuration interfaces for the design, which are built upon the existing messaging infrastructure:

- Pattern Management
- Group Management
- Policy Management
- External Validation

It is necessary to allow for external pattern management since a new pattern is released regularly. This interface is used to delete obsolete patterns and enables tagging patterns with additional information. The group management interface allows all CRUD [104] operations to be performed against groups. All available information about patterns and groups can furthermore be queried. Policy Management is required for configuring complex policies and external validations require an interface as its name suggests.

5.6.5. Output Dissemination

The output of the design needs to be made available to the components that require or are interested in the produced information. There two types of outputs generated by the design are:

- Regularly Published Pattern
- Evaluation Decision

The information of patterns being created is designed to be published asynchronously following the publish-subscribe pattern [33] to make the information available to all components in the system. Decision results are also published for the same reason, but they are also made available in a synchronous exchange for being obtainable during access control.

5.7. Summary

In this chapter, the solution design and the decisions behind the design have been discussed in great detail with each part being presented with a preceding requirement stemming from one or more research questions of this work. The amount of information makes crucial parts difficult to find, which is why this summary features a list of questions

What type of software architecture is used for the acquisition and management of context?	The solution design introduces an explicit context model and interacts via a messaging middleware software with other components in the smart home environment.
Do the interaction options feature active or passive context-awareness?	The users of the smart home have to specifically set the patterns that are being evaluated against, therefore the interaction options are classified as active.
How can the context-aware system feature be described?	The system feature offered by the design can be described as a passively executed service since behaviour is measured passively and once a pattern is set for evaluation, access requests are evaluated against that pattern without further input.

Table 5.8.: Context-aware design: fundamental information

How does the context get acquired?	The context is acquired via a middleware infrastructure provided by the COSYLab environment.

Table 5.9.: Context-aware design: architecture

and answers, which are written to serve as a comprehensive overview of the central design information of the solution design.

The questions and answers are presented in Table 5.8, giving an overview of the most fundamental parts of the solution design, Table 5.9, which discusses details about the architecture in the underlying environment, Table 5.10, presenting a technical overview over the context introduced in this work and in Table 5.11, which uses the life cycle of context [73] as a reference model to present details about the context.

What context attribute properties are provided?	Overall, there are three attributes introduced in the scope of this work. Two of the attributes can be used directly in ABAC, namely the singular control event pattern-based attribute and the grouped control event pattern-based attribute. The last one can only be included in more complex access rules due to the singular periodic event pattern-based attribute not including a user. During the access rule evaluation, all three attributes produce a numerical value and an additional categorization based on the numerical value, with a classification of “expected” indicating that the evaluated event matches the behaviour pattern and a classification of “odd” indicating a mismatch between the usual behaviour and the currently evaluated event. In case no clear result can be given, the event is classified as “uncertain”.
Are the acquisition types classified as primary or secondary?	The used acquisition type is primary since the events are directly taken from the environment without including any additional computational operations. Note that this does not exclude the usage of secondary context, which is generated using the acquired context.
How can the context be categorized from an operational standpoint?	The behaviour patterns are categorized as profiled context due to their nature of being updated periodically after a set time interval.
What entities can be observed in the context model?	The context model features users, devices, and their respective groups as observable entities.
Is the context type provided external or internal?	Behaviour is generally classified as internal context [2].

Table 5.10.: Context-aware design: context overview

What are the central design decisions of the acquisition phase [73]?	The responsibility of the acquisition is based on the acquiring component, which pushes the information to the other components. There are two types of events being consumed in this stage of the life cycle with one of them being gathered periodically and the other one being gathered instantly. The events are gathered via a messaging middleware infrastructure and the sources include physical, logical, and virtual sensor types. The acquisition process includes sensing and manual provision by users.
On what is the context model based?	The context model is ontology-based and is modeled based on CAMeOnto [2].
What reasoning methods are used?	There are three types of high-level context created in the reasoning phase. The first one is the behaviour patterns, which are created using unsupervised learning [9]. The other high-level context types are group suggestions and pattern correlations and both are created using fuzzy logic algorithms [112].
Is the distribution based on queries or on subscriptions?	The decision results, pattern updates, and other higher-level context types are pushed into the messaging queues, enabling communication through subscriptions [33]. Furthermore, there is also the possibility of specifically querying pattern evaluations for the access control.

Table 5.11.: Context-aware design: life cycle of context

6. Implementation

The concept introduced in this work aims to provide the knowledge of behaviour to a smart home environment and focuses on supporting the ACM of the smart home with this information. As part of this work, a software component is created to implement the described solution design and is integrated into COSYLab, an existing smart home application. The resulting smart home component is called behaviour engine, runs parallel to the usual smart home installation in the local network, and is responsible for analyzing the behaviour in the home and disseminating the knowledge that is produced.

An overview of the components interacting with the behaviour engine within the smart home it is set in can be seen in Figure 6.1. In the image, the architecture of the smart home application is shown with the smart home devices being coloured yellow, open-source components being displayed as green, and the software components related to the behaviour engine being blue.

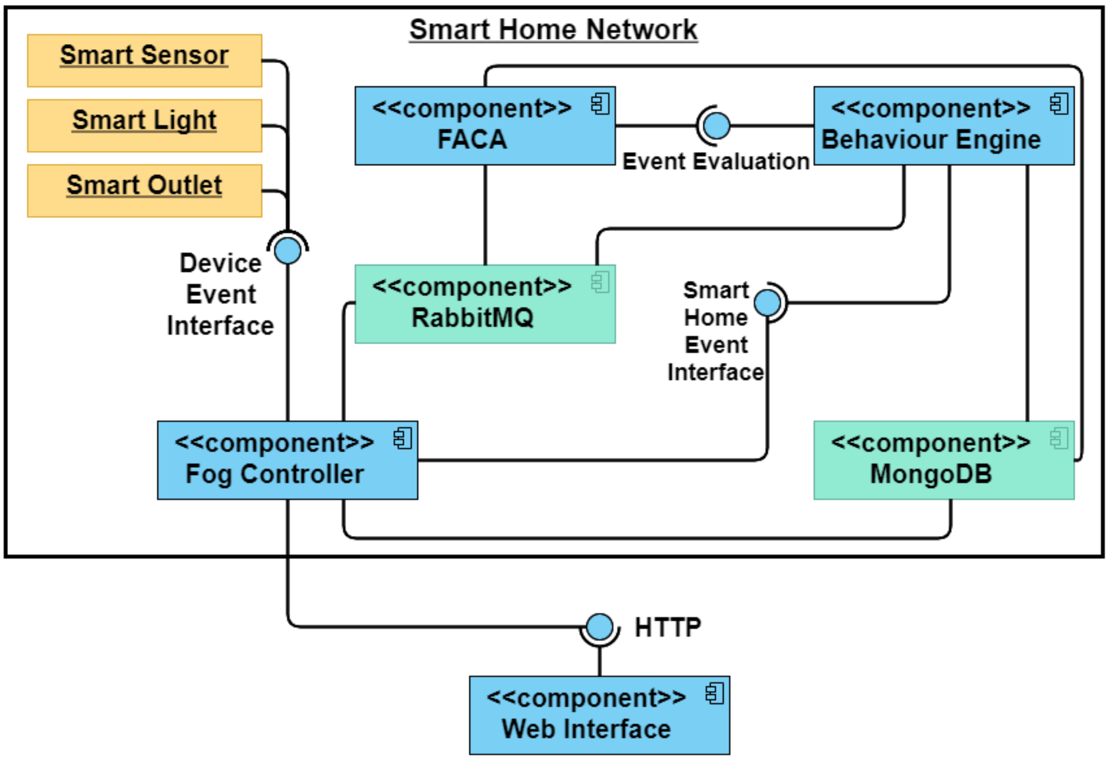


Figure 6.1.: The underlying smart home environment

The internal architecture of the behaviour engine needs to follow the described solution design presented in Chapter 5 and be implemented to support all the described features. Figure 6.2 displays a component view of the behaviour engine, showing its four components. Three of the components are internal, which means that they cannot be interacted with directly, but instead interacted with indirectly through the messaging component. Therefore, the messaging component exposes the functionality of the analysis component, the decision component, and the data management component to the environment. The three internal components do not interact directly with each other but have access to a common database.

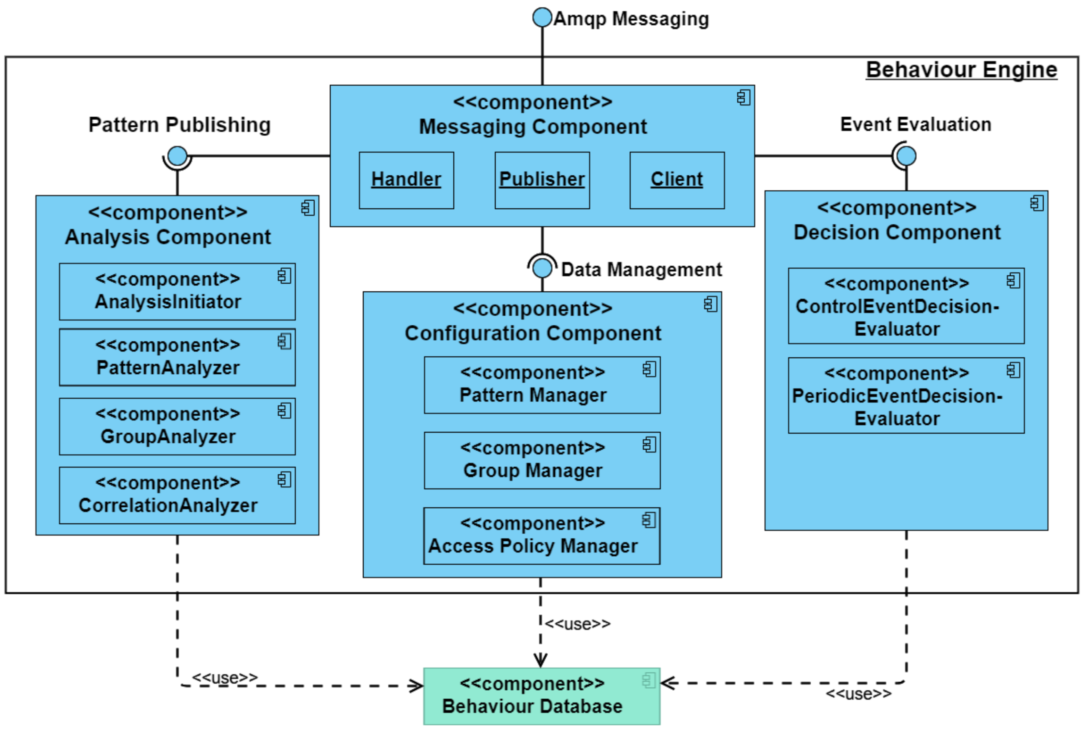


Figure 6.2.: Component View of Behaviour Engine

This chapter aims to provide an overview of the most important details of the implementation to understand how the behaviour engine works and where it fits into the bigger picture in the smart home environment. First, the related smart home components are introduced to provide insight into the underlying environment and the related smart home components. Then, the semantic integration of the behaviour engine into the ACM is presented, which includes the attributes for the ABAC model and a description of the exchanged messages. The behaviour engine features four components and in this chapter, a section is dedicated to each of these components. The last two sections describe the high test coverage present in this project and the graphical visualizations of the behaviour patterns. Note that the web interface is not part of this thesis and therefore no frontend pictures are included in this thesis with the pattern visualizations being the only exception to this rule.

6.1. Underlying Smart Home Infrastructure

The smart home environment features several software components to provide the promised home automation of a smart home. To support the software application, the smart home offers an infrastructure that is responsible for providing services that are shared between all smart home application components. The offered services include a messaging middleware responsible for the entire communication within the smart home and a database responsible for the data storage and retrieval. This section serves to provide technical details about the infrastructure of the smart home environment, which includes the shared services. Furthermore, the smart devices and sensors in the home are presented in this section.

6.1.1. Smart Home Devices

All types of smart devices are supported by the environment, including smart sensors, and these devices generate the data for smart home events. Whenever a user interacts with a smart device, a control event is created, and whenever a smart sensor publishes a measurement, a periodic event is created. After an event is raised by a device, it is wrapped into a smart home event in the Fog Controller, which is a smart home component related to the behaviour engine and is described later in Chapter 6.2.2.

A smart device is any type of electrical device in a smart home, that is connected to the Internet. These smart devices are remotely accessed by the smart home resident, usually via their smartphone but any device with a web browser can initiate the interaction. A smart sensor is deployed in the smart home and takes regular measurements of a specific type and sends updates about the state it measures to the environment. Events based on smart measurements are generated without any user input and are therefore processed automatically in the environment. Whenever an event is received by the behaviour engine, it evaluates whether the event matches the usual behaviour in the system. Details about how the behaviour engine evaluates smart home events are presented in Chapter 6.7 and details about how the evaluation of events is integrated into the ACM are presented in Chapter 6.5.

6.1.2. Open-Source Components

All communication from and to the behaviour engine is going through an AMQP messaging middleware [34], which is represented by an open-source RabbitMQ service in the smart home environment. The behaviour engine accesses the functionality of the messaging infrastructure with the java Spring packages for RabbitMQ [96] and AMQP [94].

The smart home environment uses a NoSQL MongoDB database service [15] for the storage of data and the behaviour engine performs its database operations with the MongoDB package of java Spring [95]. The database scheme of the behaviour engine is described in Chapter 6.3.

6.2. Related Smart Home Components

The two smart home components having a relationship with the behaviour engine are the smart home event coordinating component, called Fog Controller, and the access control component, called Fog Access Control Agent (FACA). These are the only other smart home components that are interacting directly with the behaviour engine. Both of these software components share their maven parent project with the behaviour engine, which enables all versions of related third-party software to be kept equal across the environment.

In this section, an overview of the responsibilities of the two related smart home components is given. A general description of these applications provides sufficient knowledge to understand what other components besides the behaviour engine are important in the scope of this thesis, how they interact with the behaviour engine, and what the behaviour engine needs to provide for them.

6.2.1. Fog Access Control Agent

The FACA is responsible for handling access control requests and is structured internally following the XACML framework [36]. The access control model of FACA is based on attributes, making it an ABAC model. When an access request is received by its PEP, the request is forwarded to the PDP, which retrieves the access policy from the PAP and processes each attribute on the list of rules in that policy individually. The attributes in the policy provide a reference value and an operator for the PDP to compare the actual information against. If the information supplied by the PIP matches the constraint given by the access rule, the rule is fulfilled and the next rule is processed and an access policy is only met if all access rules are met.

In the FACA, context attributes are not fetched by the PIP but are instead retrieved by messaging the respective component responsible for the context type. This means, that the behaviour engine is responsible for providing the behaviour attribute values that are compared in the PDP, which is why the FACA is using the evaluation interface of the behaviour engine whenever an access policy including a behaviour related access rule has to be evaluated. Details about the communication between the FACA and the beahviour engine are described in Chapter 6.5.

6.2.2. Fog Controller

The Fog Controller is a smart home component with the responsibility of being the distributor of requests and events within the smart home. This means, that whenever an event is raised in the smart home or whenever a message is sent to the smart home, it is initially processed by the Fog Controller, which handles the request by forwarding the message to other smart home components or by raising or wrapping the messages into a new event.

The Fog Controller has the role of creating access control requests and send them to the FACA whenever a device event requires access control. Once access is granted, the Fog Controller wraps the device in a smart home event and publishes it, which is then consumed by the behaviour engine. Furthermore, the Fog Controller is responsible for forwarding configuration messages from the user to the behaviour engine.

The communication involving configuration requests for the behaviour engine is never received directly from the user's web application. Instead, all inputs a user sends to the behaviour engine go through a RESTful HTTP interface [79], which is received by the Fog Controller and translated into a request for the behaviour engine. Details about what can be configured in the behaviour engine are presented in Chapter 6.8.

6.3. Database

Every context type used and created in the analysis process is stored persistently, which means that every feature described in Chapter 5 has its own database entries. The behaviour engine stores all of its data types separately from other components in the smart home so that it can work independently by not having to rely upon or interact with the storage strategies of the environment. This section serves to give an overview of the database scheme of the behaviour engine.

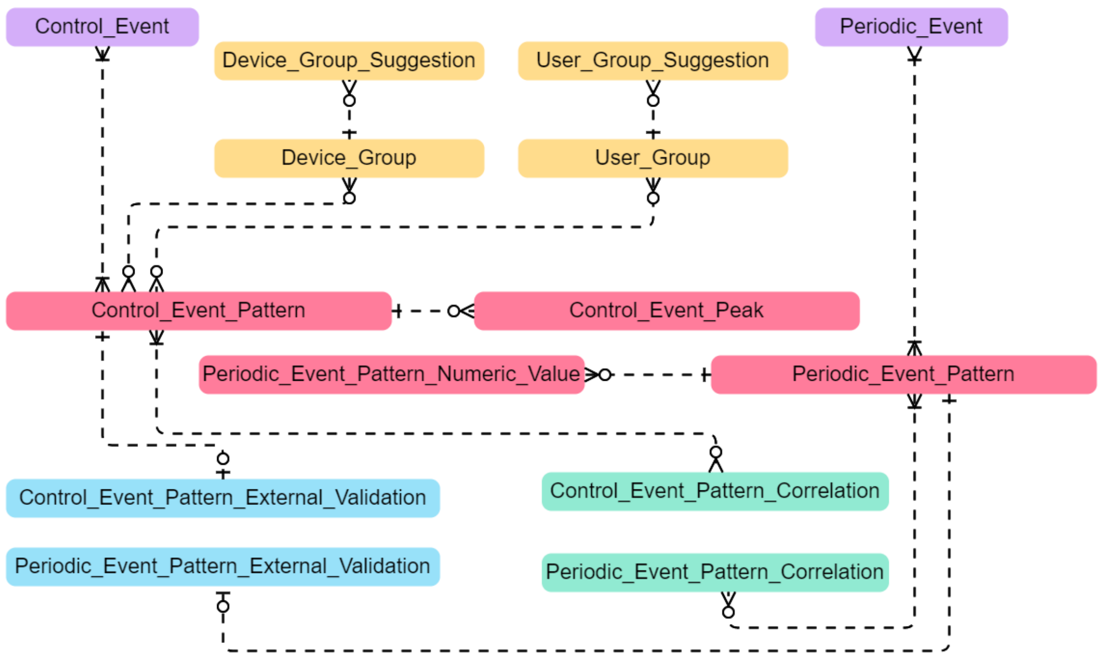

Control_Event
Device_Group_Suggestion
User_Group_Suggestion
Periodic_Event
Device_Group
User_Group
Control_Event_Pattern
Control_Event_Peak
Periodic_Event_Pattern_Numeric_Value
Periodic_Event_Pattern
Control_Event_Pattern_External_Validation
Control_Event_Pattern_Correlation
Periodic_Event_Pattern_External_Validation
Periodic_Event_Pattern_Correlation

Figure 6.3.: Database Scheme

The behaviour patterns are the central component of the design as can be seen in Figure 6.3, the database scheme of the behaviour engine. The behaviour patterns are coloured red and are connected to each other group of data types. Both pattern types have their respective data type for evaluating further events, namely the control event peak and the periodic event pattern numeric value.

The smart home events are coloured purple and are modeled after the events that are created in the environment. These events encompass a set of low-level contexts as described in Chapter 5.2, which is used to determine the respective pattern they belong to during the evaluation and during the analysis cycle. The reason for the patterns and the events to have a relationship of one or more on each side is that no pattern exists without smart home events and all smart home events are assigned to one pattern of which a new version is released every analysis cycle.

The yellow-coloured entities are related to groups, which are introduced and described in Chapter 5.4. Both types of groups are strictly separated and get their respective group suggestions. Groups and patterns have zero or more relationships with each other due to neither of them having to be associated with the other, but there is no limit to how many patterns are part of a group and there is no limit to how many groups a pattern can belong to. Groups are a feature that is exclusive for control event patterns, which is why they do not have a relationship with periodic event patterns.

Pattern correlations are displayed as green and external validations are coloured in blue. There can always be only one external validation for a pattern because either one exists already and therefore no new one is created or one is expired and gets removed. Pattern correlations exist between each pattern with equal metadata, which is why one pattern can have any number of pattern correlations, but a pattern correlation is always related to exactly two patterns.

6.4. Messaging

All communication involving the behaviour engine is performed via the AMQP messaging middleware with JSON messages. The messaging component is divided into handlers, which are responsible for receiving and processing messages from other smart home components, and publishers, which are responsible for publishing messages to the middleware. Furthermore, there is one Message Client, which allows the behaviour engine to send synchronous messages to other components in the smart home.

There are three types of messages that are received or consumed by the behaviour engine. The first type is the *AttributeValueEvaluationRequest* as described in Chapter 6.5.2, the second type is the smart home events as described in Chapter 6.7, and the third type is configuration requests as described in Chapter 6.8. In this section, all of the message clients, publishers, and handlers are presented. Note that not all messages available on the messaging interface are included as JSON examples due to the high number of messages that would have to be included and some messages are presented in other sections of this chapter.

6.4.1. Client

The behaviour engine features one messaging client, which has the purpose of sending messages to other components in the smart home environment. The behaviour engine only sends one message as part of its initial startup, which is a message containing the three behaviour attributes to register them in the FACA. Furthermore, this is the only time when the behaviour engine initiates communication with the environment if the asynchronous publishing of messages is not counted.

6.4.2. Publisher

The messages published by the behaviour engine are made available to any component that needs or is interested in the published information and are not targeted to a specific component. Therefore, the publish-subscribe pattern is used for the message publishers [33] and the messages are sent asynchronously [23]. There are two message publishers, which are divided by what information is being published, namely:

- Decision Message Publisher
- Pattern Message Publisher

The Decision Message Publisher is one part of the “Pattern Publishing” interface and makes the decision results of the evaluation of smart home events available to other components in the environment besides the FACA. This allows components that are not involved in the access control process to be informed about whether the smart home events match the behaviour. The message published is an *AttributeValueChangeNotification*, which can be seen as part of the JSON message in Listing 6.4, but in the published message there is only one notification instead of the list presented in the example of Chapter 6.5.2.

Pattern generation is described in Chapter 6.6.1. Whenever a new pattern is generated, the behaviour engine publishes a *ControlEventPatternUpdateMessage* or a *PeriodicEvent-PatternUpdateMessage*, depending on the pattern type. The messages of this component serve to inform the environment of the release of a new pattern and an example of the content in one of the messages can be seen in Listing 6.16. In the messages of the Pattern Message Publisher, which is the second half of the “Pattern Publishing” interface, only the pattern part of the example is present, and the differences between pattern types are the same as described in Chapter 6.8.1.

6.4.3. Handler

The behaviour engine features seven message handlers and a total of 22 messaging queues it binds on the AMQP interface. The message handlers are:

- Device Group
- Event Evaluation
- Event
- External Validation
- Pattern
- Policies
- User Group

All handlers are exchanging their messages synchronously [103], except for the Event Message Handler, which only consumes the smart home events sent by the environment. Examples of the events this handler consumes can be seen in Listing 6.10 and Listing 6.12. This handler triggers an evaluation of the event as described in Chapter 6.7.2 and hands the result over to the Decision Message Publisher, which publishes the result as described in the previous section. This component is not to be confused with the Event Evaluation Message Handler, which is responsible for access control and interacts with the FACA as described in Chapter 6.5.2.

```
{
    "messageType": "cosy.behaviour.engine.pattern.delete
        .response",
    "success": true,
    "errorMessage": "No Error Occurred"
}
```

Listing 6.1: JSON message: Standard Behaviour Response Message

The message handlers that communicate synchronously respond with a *Behaviour-ResponseMessage* on most operations since signaling that the request is successfully performed is sufficient information in most cases. An example of this standard response message can be seen in Listing 6.1. Every response message includes the three fields of the standard response, which is a message type matching the requested operation, an indicator of whether the operation was successful, and an error message if the operation was not successful.

The Device Group and the User Group Messaging handler offer exactly the same messaging interface for their respective group type. They cover interfaces for all of the CRUD operations [104] as described in Chapter 6.8.2 and all create, update, and delete queues only give the standard response. The two read messages include either a list of groups for which groups a user can manage or a specific group requested in a read operation and its corresponding group suggestions. Listing 6.17 displays the resulting message of a read request for a device group.

The Pattern Message Handler features a messaging interface that covers all operations described in Chapter 6.8.1 and the three possible read operations give the result messages as described in that chapter. Which read operation is triggered is determined by which queue received the request and the input is either a publishing key or the list of metadata depending on the pattern types of the requested related patterns. The update message allows the user only to update a text field of the behaviour patterns as described in Chapter 6.8.1 and the delete pattern operation allows the deletion of one pattern and both of these operations only respond with a standard response.

The Policies Message Handler and the External Validation Message Handler both only return the standard response for all of their operations. The *ExternalValidationWrapper* only has one field for identifying the event to validate. The two queues in the Policies Message Handler work as described in Chapter 6.8.3 and allow the user to associate a list of either device or user groups with a control event behaviour pattern.

6.5. Access Control Integration

Access control integration is one of the central parts of this work due to the second research question. An access control request is sent by the Fog Controller to the FACA whenever a resident in the smart home requests control over an object. The access control requests interesting for this thesis are triggering the evaluation of an access control policy including a rule with a behaviour attribute.

This section mainly deals with the details of the communication between the Fog Controller, the FACA, and the behaviour engine during the processing of an access control request. Furthermore, this section presents which of the behaviour attributes require an additional configuration and how they are configured.

6.5.1. Behaviour Access Control Attributes

As described in Chapter 5.6.1, the full names of the three behaviour access control attributes provided by the behaviour engine are:

- Context.Behaviour.Pattern.Device.Single.Control-Based
- Context.Behaviour.Pattern.Device.Single.Monitoring-Based
- Context.Behaviour.Pattern.Device.Group.Control-Based

The two attributes containing the infix “Single” are evaluating a single control event pattern, with the suffix “Control-Based”, and a single periodic event pattern, with the suffix “Monitoring-Based”. Both of these attributes can be used in access policies without further configuration since the FACA can determine that it needs to relay the retrieval of information to the behaviour engine and the behaviour engine can evaluate this attribute without any further input. For the evaluation process, the attributes are required to add one additional parameter in their attribute name for a specific pattern to be resolvable. The parameter is added similarly to how it would be added to an URL and the parameter value is the unique publishing key of the pattern to evaluate.

The remaining attribute represents behaviour pattern evaluation while including a group, and therefore, this attribute requires one parameter for the publishing key of the pattern like the other attributes do, and furthermore it requires a group identifier in the attribute name. This identifier is used to determine the group the behaviour engine should use during the access control evaluation, which furthermore allows the behaviour engine to evaluate all patterns a user wants to be included in the decision-making process. This is the only behaviour attribute that requires two parameters in its attribute name and it is the only attribute that requires an additional configuration on the behaviour engine.

```
{
    "publishingKey": "4a87e571-adc4-4f5e-9c7f-a0cb1d4241f2",
    "groups": [
        "gabT6cR2tsphw56SMPMyF0GHtj0fF386JycewdI8vuUm9jHIEqqq7a
            Pl0YqSHTkx",
        "DFJc5EoioDwJIdHkqp7j1zTw9DtgtXxnvkjiRn4eWLwPfeX48JiUHq
            cxn6E6pdfy"
    ]
}
```

Listing 6.2: JSON message: configure group evaluation for pattern request

The general event evaluation and the access control attribute evaluation are separate processes due to a limitation in the underlying environment. At the point of the general evaluation, the group that should be part of the evaluation can not be determined just by examining the control event, which is why this information needs to be made available to the behaviour engine. Listing 6.2 shows one example request to make the behaviour engine aware of the relationship between a group and a behaviour pattern. Note that groups are strictly divided into device and user groups, which is why the request can contain only groups of one type and the type of group being configured is determined by which messaging queue is used.

6.5.2. Access Control Request Processing

When the FACA evaluates a behaviour related access rule, it sends an *AttributeValueEvaluationRequest* to the behaviour engine, which evaluates the request and responds with a *ContextAttributeValueList*. The behaviour engine responds with a list of values because the request of the FACA can contain multiple attribute values for a bulk request, in which case the behaviour engine would evaluate all of the requested attributes and send all resulting values in a single response. Once the FACA receives the response from the behaviour engine, it evaluates all access rules in the policy and responds to the access control request of the Fog Controller with the decision.

If the access is granted, a control event is published and the request of the user is processed. If the request is denied, the requested action of the user is blocked and no smart home event is created. The smart home event is used by the behaviour engine in its regular evaluation process as described in Chapter 6.7.2 and a decision result is published to an AMQP messaging queue, where it is made available for all components that are interested in the result. The processing of a control event in response to a request of a smart home resident is displayed in Figure 6.4.

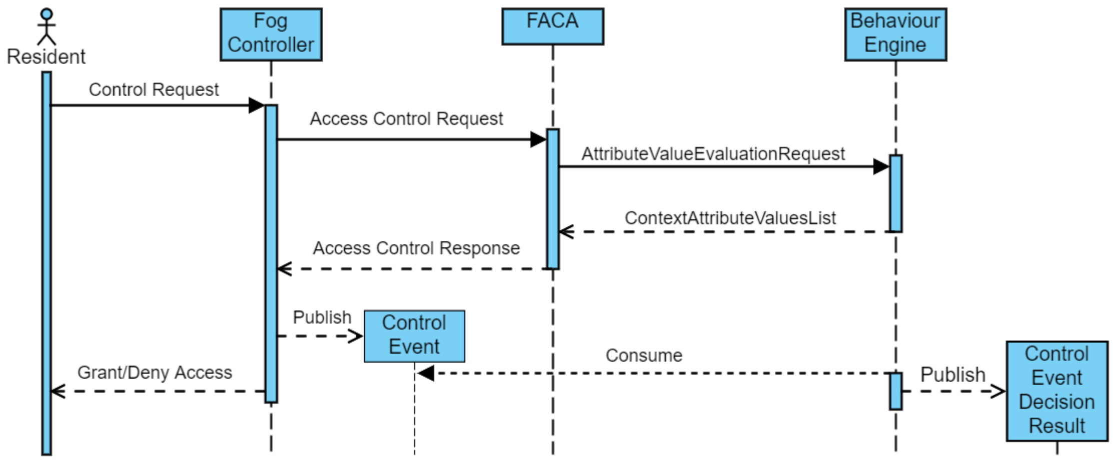


Figure 6.4.: Processing of a control event

An example for an *AttributeValueEvaluationRequest* sent by the FACA to the behaviour engine can be seen in Listing 6.3. This request contains the identifier for the included device and user, a function name, and most importantly, the attribute name. At the end of the attribute name, the unique publishing key is attached as a parameter, which is used by the behaviour engine to identify the pattern it needs to evaluate.

```
{
    "requests": [
        {
            "attributeName": "Context.Behaviour.Pattern.Device.
                Single.Control-Based?publishingKey=8a82f7e4-0541
                -11ec-9a03-0242ac130003",
            "deviceId": "600378187c0435305a19f836",
            "functionName": "turn on",
            "userId": "GBnXifuYc5zVoeGrw7Moe1wGiHw8ieF6WlRimZzct
                9Kfx6GLLumL9Zy7WWG4n5cp",
            "timestamp": 2021-09-10T11:22:56.875276
        }
    ]
}
```

Listing 6.3: JSON message: single pattern evaluation request

Every access control evaluation request is answered with a corresponding *ContextAttributeValueList* and an example for a message in JSON format is displayed in Listing 6.4. The type of context in these responses is always “Behaviour” and the certainty is always indicating a 100% certainty of the result since the behaviour engine’s evaluation is deterministic.

```
{
    "attributeValues": [
        {
            "attributeName": "Context.Behaviour.Pattern.Device.
                Group.Control-Based?publishingKey=8a82f7e4-0541
                -11ec-9a03-0242ac130003&groupId=CXfjkx1m3WAelFC
                KwUyFv3u6t54p1zuHyxxtqyfeyJ0p74li5vZHyPnTJJzQkp
                db",
            "attributeValue": "0.67",
            "certainty": 100.0,
            "timeStamp": 2021-09-24T11:25:56.475976,
            "contextType": "Behaviour"
        }
    ]
}
```

Listing 6.4: JSON message: group pattern evaluation response

One attribute value in the list contains an attribute name, which includes the publishing key and a group identifier in this example and is used by the FACA to determine to which attribute the result is provided. The most important field for the FACA is the attribute value, which contains the numerical value of the decision result and is used to compare whether the received value matches the constraints set in the access rule.

6.6. Behaviour Analysis

The data analysis component is responsible for the creation of every high-level context featured in the reasoning phase, as described in Chapter 5.1.3. The three context types created by this component are behaviour patterns, group suggestions, and pattern correlations and each of these high-level context types needs to be updated once every analysis cycle. The order in which the context types are created are:

- Pattern Generation
- Group Suggestions
- Pattern Correlations

The order in which the context types get analyzed is important because the pattern correlations should be calculated after the new patterns are present, which is why the pattern generation is scheduled first and the pattern correlations are scheduled last. Group suggestions are calculated based only on low-level context, which is why they can be put anywhere in the schedule and therefore they are put in the middle.

At the end of every day, the analysis initiator starts the process of creating the higher-level context. The reasoning phase is not triggered for the current day, but for the previous day instead since no new data can appear for that day anymore. Since the solution design does not specify an analysis cycle time span, the choice of analyzing behaviour every day is an implementation choice in line with the solution design. There is no difference between analyzing all patterns after every day and scheduling the analysis to be done once a week, so the analysis is done every day to divide the computational effort evenly throughout the week. This section presents the implementation details of the analysis of each context type.

6.6.1. Pattern Generation

The process of creating new patterns includes all relevant events matching the patterns metadata and can be considered the newest approximation of the behaviour that the pattern represents. This means, by using the newest set of behaviour patterns, the most information can be included in access control and other analysis components.

One of the reasons for splitting the analysis component and the decision component is the timely task of creating behaviour patterns, which cannot be included at runtime. Some of the reasoning algorithms are especially time-consuming, i.e. K-means, which is an NP-hard algorithm [61]. For K-means specifically, an upper bound of twenty clusters is set for each control event pattern to make sure that the control event pattern generation process does not take up too much computational time.

Besides the upper bound for the K-means algorithm, there are other important boundaries that need to be set to prevent the behaviour engine from malfunctioning. A minimum of two data values is required in every usage cluster in every control event pattern and in every time slot of periodic event patterns for the algorithms to work without errors. A proper minimum of data for both pattern types is investigated as a part of answering the third research question and is presented in Chapter 7.

```
# Control Event Pattern Generation
for each distinct device in all control events
    for each distinct user of the device
        filter events for previous day, user, and device
        for 1...20
            create K-Means clustering
            if cluster is accepted
                result cluster found
        if result cluster could be formed
            store new control event behaviour pattern
```

Listing 6.5: Pseudo-Code: Control Event Pattern Generation

The control event pattern generation process is displayed in Listing 6.5 and the main calculation performed for the pattern generation, the K-Means clustering, is performed with the library commons-math3 of Apache [24]. The algorithm is set up to attempt to create the lowest number of clusters possible and retries until either too many clusters have to be created or the acceptance criteria for a cluster is met. A cluster is accepted if less than 10% of its events have a distance higher than two times the standard deviation of the cluster to the center of their respective cluster. Furthermore, a minimum standard deviation of fifteen minutes is assigned to each peak to prevent small clusters.

```
# Periodic Event Pattern Generation
for each distinct device in all control events
    filter events for day before analysis cycle and device
    for each time slot
        filter remaining events by timestamp
        calculate mean and std
    store new periodic event pattern
```

Listing 6.6: Pseudo-Code: Periodic Event Pattern Generation

The time slots in the periodic event pattern enable a simple generation of a behaviour pattern. All available events are filtered for each time slot and the numeric values in the periodic events are used to calculate a mean and a standard deviation. Listing 6.6 displays the pseudo-code for generating a periodic event pattern.

The analysis component uses the interface of the messaging component to publish notifications about newly created patterns and a message containing all information that is published is displayed in Listing 6.16. The difference between the published message and the example is that no groups or correlations are present since the pattern is newly created.

6.6.2. Group Suggestions

After the patterns are generated, the group suggestion analysis is triggered. There are two types of group suggestions created in the behaviour engine, namely suggestions for individual devices and users, and suggestions for existing groups. Suggestions for devices and users are used to calculate the suggestions for groups, which is why they are not intended to be shared with the environment and serve as an internal calculation aid of the behaviour engine. The creation of suggestions for device groups and user groups is equal, which is why only one example is examined in this section.

```
# User Group Suggestion Creation
devicesTotal = number of all devices target user interacted with
for each device target user interacted with
    find all other users of the device
    count number of commonly used devices for each user
    create suggestion object with
        target user
        suggested user
        commonly used devices
        devicesTotal
```

Listing 6.7: Pseudo-Code: Pseudo-Code: Group Suggestions for a user

Listing 6.7 displays the creation of a group suggestion for one user and the group suggestion for a user group adds all suggestions together for every user outside of the group to create the suggestions. Therefore, the suggestions for a group are based on the suggestions for each individual member of the group.

6.6.3. Pattern Correlations

The pattern correlations are created last so that they can include the newly created patterns of the current analysis cycle. Pattern correlations are calculated as described in Chapter 5.6.3 by determining to what degree two patterns accept the same events. Note that correlations are only calculated between patterns with matching metadata because they are only intended to be used as replacement recommendations and a pattern can only be replaced by a pattern with equal metadata.

```
# Control Event Pattern Correlation Calculation
for each control event peak in pattern A
    for each control event peak in pattern B
        peakBefore = find closest peak in B before peak in A
        peakAfter = find closest peak in B after peak in A
    resultPeakA = calculate coverage of peak in peakBefore
        + calculate coverage of peak in peakafter
        - calculate coverage of peakBefore in peakafter
for each control event peak in pattern B
    resultPeakB = repeat previous loop
resultA = average over all resultPeakA
resultB = average over all resultPeakB
result = (resultA+resultB)/2
```

Listing 6.8: Pseudo-Code: Control Event Pattern Correlation

Listing 6.8 displays the pseudo-code for the calculation of the correlation between two control event patterns. For each peak, the coverage in the closest two peaks in the other pattern is calculated and then the coverage between the closest two peaks is subtracted in case they overlap. The calculation of the coverage follows the general event evaluation as described in Chapter 6.7.3 with the centers of the usage peaks being used in place of the timestamp of the control event.

The correlation is calculated in both directions and the average of the two results is used to create a proper analysis model between two patterns with a different number of usage peaks. If one pattern has only one peak and the subsequent pattern adds two new peaks, then the correlation in one direction would indicate a 100% correlation, but by adding in the 33% of the other direction, the resulting correlation between those two patterns is 66%, which expresses a better picture in both directions.

```
# Periodic Event Pattern Correlation Calculation
for each periodic event pattern value
    resultThisValue = calculate coverage of value A in value B
        + calculate coverage of value B in value A
        / 2
result = average over all resultThisValue
```

Listing 6.9: Pseudo-Code: Periodic Event Pattern Correlation

In Listing 6.9, the pseudo-code for the creation of a pattern correlation between two periodic event patterns is displayed. Periodic event patterns are easily comparable due to their fixed time slots since every time slot can be compared to each other. To calculate the coverage of one periodic event pattern value in another, the calculation method of to what degree a periodic event matches a periodic event pattern value described in Chapter 6.7.3 is used. The calculation is made in both directions and an average of the two results is the coverage of the value, and an average over all coverages is the resulting correlation between the two periodic event patterns.

6.7. Decision-Making

The decision-making component of the behaviour engine utilizes the behaviour patterns created and stored by the data analysis component after retrieving them from the database. The decision-making component acts on two different message types, namely smart home events and access control requests. These two types of messages need to be processed differently due to a limitation of the underlying environment, which requires a device event to be validated by the access control component before the Fog Controller wraps it into a smart home event and therefore before it becomes consumable for all components.

This section discusses the processes behind evaluating an event for access control and for evaluating a smart home event and the distinction between these processes and why they have to be separated. Furthermore, this section presents details about the implementation of the result calculation with code snippets from the behaviour engine.

6.7.1. Access Control Decision

The FACA requires an evaluation decision of the behaviour engine while evaluating the access control request whenever an access policy has a rule with a behaviour access control attribute as described in Chapter 6.5. This decision result is provided by the analysis component and the behaviour engine offers a specific interface for evaluating, but not storing an event. It is important for the event not to be stored at the end of the access control evaluation because the event might still be blocked by a different rule in the FACA, causing the event to never happen. The behaviour engine would include bad events in its analysis process if the event was stored even though it was blocked by the ACM.

Chapter 6.5 discusses the messages exchanged between the behaviour engine and the FACA during an access control request. Once the *AttributeValueEvaluationRequest* is received, the behaviour engine reads out which groups and patterns are to include in the evaluation. The patterns that need to be part of the decision are retrieved and are evaluated against as described in Chapter 6.7.2, the *ContextAttributeValueList* is created according to the results, and the response message is sent.

6.7.2. Smart Home Event Evaluation

The behaviour patterns are statistical estimation models used to determine whether smart home events in the environment are expected. Whenever a new event is created in the smart home, a decision is made whether the event matches the behaviour pattern and the decision result is published. This process is separated from the analysis due to a performance requirement of the second research question.

The evaluation of a smart home event is similar to the access control decision evaluation described in the previous section. The differences are that during the evaluation of a smart home event no information about a specific pattern to evaluate is present and that the event is stored with an indication of whether it is valid. It is necessary to store all events, even if they are declined because the decision might be overridden by a user with an external validation.

All patterns with matching metadata are included in the evaluation process and the highest result is chosen as the decision for the smart home event to determine whether the behaviour engine shall include the event in the analysis cycle. For providing the knowledge of the evaluation to all components subscribed to the event decision information as described in Chapter 6.4.2, each result created in the process is published individually regardless of whether it was the highest result.

```
{
    "deviceType": "Stove",
    "deviceId": "b4792784-092a-11ec-9a03-0242ac130003",
    "userProxyId": "9N0CtJpDMuqz0mx0ujrRuAeyQDyKa0mZeH7
        Ly4nkKcXUvQSnaDUz7MVaVYqhEJuB",
    "function": "turn on",
    "timestamp": 2021-09-10T06:22:51.375476
}
```

Listing 6.10: JSON message: Control Event

```
# Control Event Evaluation
retrieve all control event pattern with equal metadata
for each control event pattern with equal metadata
    check external validation
    if external validation is present
        set max correlation to expected value
    else
        for each control event peak in pattern
            calculate correlation with event
        choose max correlation value
        if max correlation not accepted
            for each control event pattern in groups
                repeat calculation of correlation with event
            choose max correlation
    return max correlation
```

Listing 6.11: Pseudo-Code: Control Event Evaluation

An example of a control event can be seen in Listing 6.10. When a new control event comes in, the Decision Component of the behaviour engine filters all behaviour patterns available in the database having matching metadata with the event, and evaluates the event against all of the resulting patterns. The evaluation process is described in detail in Listing 6.11 and details about the calculation of correlation between event and pattern are described in the next section. Note that the control event peaks are a list where each peak needs to be processed individually since their ordering is not guaranteed.

```
{
    "deviceType": "Thermometer",
    "deviceId": "b0b7dce4-092a-11ec-9a03-0242ac130003",
    "value": "27.5",
    "timestamp": 2021-09-10T10:22:36.875276
}
```

Listing 6.12: JSON message: Periodic Event

```
# Periodic Event Evaluation
retrieve all periodic event pattern with equal metadata
for each periodic event pattern with equal metadata
    check external validation
    if external validation is present
        set max correlation to expected value
    else
        get corresponding periodic event pattern value for event
        calculate correlation with event
        set max correlation value
    return max correlation
```

Listing 6.13: Pseudo-Code: Periodic Event Evaluation

Listing 6.12 displays an example of a periodic event and the pseudo-code of the evaluation of a periodic event is displayed in Listing 6.13. The evaluation of a periodic event is similar to the evaluation of a control event but is two steps shorter. The first step the periodic event evaluation skips is the processing of all periodic event pattern values, which is redundant due to the corresponding time slot being determinable quickly and accessible in constant time. The second step that is skipped is the group evaluation since they are not supported by periodic event patterns.

6.7.3. Decision Constants and Calculation

The result calculation is based on the standard deviations found in both behaviour pattern types for both smart home event types and the acceptable distance to a target value is set to twice the standard deviation. This means, a deviation of up to one standard deviation creates a result with the classification of “Expected”, resulting in a numerical value of at least 75% in the decision result, and a classification of “Uncertain” is assigned to events deviating up to two standard deviations, resulting in a numerical value of at least 50%. A Decision with the classification of “Odd” has always a difference of more than twice the standard deviation to the reference value and has always a resulting numerical value of less than 50%. Note that the code snippets in this section feature the numbers behind project constants like *PEAK_VALUE_ACCEPTABLE_DISTANCE_STD_MULTIPLIER* for better readability.

```
calculateDistance(ControlEventPeak peak, TimeOfDay timeOfDay) {
    distance = TimeUtils.getDistanceSameDay(
        peak.getMean(), timeOfDay);
    result = (distance / (peak.getStd() * 2.0)) * 0.5;
    return Math.min(1.0, result);
}
```

Listing 6.14: Decision Result Calculation for a Control Event

Listing 6.14 displays a code snippet for calculating to what degree a control event matches a control event peak. This is achieved by the distance between the center of the control event peak found in a control event pattern and the time of day value derived from the timestamp of a control event. This distance is then put into relation to two times the standard deviation of the control event peak to determine a result between zero and one, where one is the lowest correlation between peak and event since the distance indicates a 0% correlation.

```
calculateDistance(PeriodicEventPatternNumericValue valueInPattern,
                  String valueInEvent) {
    eventValue = Double.parseDouble(valueInEvent);
    result = Math.abs(eventValue - valueInPattern.getMean())
            / (valueInPattern.getStd() * 2.0) * 0.5;
    return Math.min(1.0, result);
}
```

Listing 6.15: Decision Result Calculation for a Periodic Event

Similar to how control events are calculated, periodic events are using the mean in the periodic event pattern numeric value as target value and its standard deviation to determine how much the periodic event diverges from the pattern. The code snippet responsible for the calculation can be seen in Listing 6.15.

6.8. Configuration

The behaviour engine offers an interface for configuration, which enables the usage of the more sophisticated features introduced in the solution design and is named the "Data Management" interface in Figure 6.2. All of the configuration options on this interface are accessible through the "AMQP Messaging" interface offered by the Messaging Component, which is accessed through the web interface and the Fog Controller in the environment.

All configurable settings which can be updated dynamically via the messaging interface are presented in this section and configurations that are part of general settings chosen during deployment are not included. This section is meant to provide a semantic insight into what can be configured at runtime and not an overview of the specific messages being exchanged on the messaging interface. The configuration messages that are presented are only included to make individual parts of this section more comprehensive.

6.8.1. Pattern Management

Behaviour patterns can be read, updated, and deleted by users, which makes three out of the four CRUD operations possible in the Pattern Manager. It is not possible to create behaviour patterns on the configuration interface because this feature is restricted to the Analysis Component. An example for the retrieval of a behaviour pattern can be seen in Listing 6.16, which is included to display all the information present in the operations of this component. Note that all list fields in this message can have any number of entries.

For read operations, a single pattern can be read, which includes all information related to that pattern as displayed in Listing 6.16, or all patterns with equal metadata can be read, which is based on a list of metadata as input, and has a list of patterns without their correlations in the response message. Note that the example displayed in Listing 6.16 is featuring a control event pattern and the only difference a periodic event pattern has in its respective message are a missing user identifier, missing group associations, and the peak list is replaced with a list of 96 periodic event values containing a mean and a standard deviation. Furthermore, the pattern correlations being part of this message allows them to be available at any component in the environment where patterns are managed.

The users of the smart home can update and delete the behaviour patterns and both of these operations determine the target pattern through a provided publishing key. While the delete request does not require any additional input, an update request furthermore features a string value, which allows for the text field "metaData" to be manipulated. This is the only updatable information in this operation because all other fields are either already handled by a different operation or are immutable. The metadata in the pattern shall not be confused with the set of low-level context, which is also referred to as metadata, because of their different usages. The context metadata is used by the behaviour engine for various processes such as event grouping in the analysis cycle while the metadata in the pattern is a textual field for the users to add information to their patterns.

```
{
    "pattern": {
        "patternId": {
            "userId": "30a38843-27f5-4615-9768-032976bc6c0c",
            "deviceId": "576a8608-0922-11ec-9a03-0242ac130003",
            "day": "WEDNESDAY",
            "localDateTime": ...
        },
        "metaData": "Text entered by user",
        "publishingKey": "73dbf6be-0922-11ec-9a03-0242ac130003",
        "userGroupList": [
            "29770cdc-4dc7-45ac-a677-4e901aab39b5"
        ],
        "deviceGroupList": [],
        "peakList": [
            {
                mTimeOfDay: 29527,
                stdTimeOfDay: 860
            }
        ]
    },
    patternCorrelations: [
        {
            "patternId": "73dbf6be-0922-11ec-9a03-0242ac130003",
            "patternIdOther": "841fe490-0922-11ec-9a03",
            "value": 0.95
        }
    ]
}
```

Listing 6.16: JSON message: Read Control Event Pattern Response

The behaviour engine offers the deletion of behaviour patterns to be handled externally because the responsibility of handling access control policies lies with other components in the environment. This leaves the information about which behaviour patterns are included in the access control outside the behaviour engine and prevents a schedule for automatic determination of obsolete patterns from being realizable meaningfully.

The additional meta-information text aims to make the patterns clearer in the management since usually there are several patterns available for the user, which can become unclear quickly. Furthermore, the meta-information text can help the users to have an overview of all available patterns during the Access Attribute Management described in Chapter 6.8.3.

6.8.2. Group Management

In Chapter 5.4, the design decision of putting the users in complete control over the groups is presented, which allows users to create, read, update, and delete the groups of their smart home. The configuration interface is split into two parts for the group management, in which one handles the device groups and one handles the user groups. All operations that can be performed on device groups can also be performed on user groups and vice versa. Furthermore, the manageable and present information is equal for both group types, which is why displaying only one example of a retrieved group in Listing 6.17 is enough to show all information available in the Group Management component.

```
{
    "deviceGroup": {
        "deviceGroupId": "69a72d5a-092f-11ec-9a03",
        "deviceGroupName": "leisureTimeGroup",
        "devicesInGroupIds": [
            "d4010154-092e-11ec-9a03-0242ac130003",
            "da7ef4c8-092e-11ec-9a03-0242ac130003"
        ],
        "administratorList": [
            "e575d6d0-092e-11ec-9a03-0242ac130003"
        ]
    },
    "deviceGroupSuggestions": [
        {
            "deviceGroupSuggestionId": "6efa9472-092f-11ec-9a03
                -0242ac130003",
            "behaviourId": "69a72d5a-092f-11ec-9a03",
            "suggestedDeviceId": "79007504-092f-11ec-9a03",
            "userTotal": 5,
            "userCommon": 2
        }
    ]
}
```

Listing 6.17: JSON message: Read Device Group Response Message

Listing 6.17 displays a field for a list of administrators of the group. This list refers to the users that are allowed to perform CRUD operations on a group and is set to hold only the creator of a group at the creation of a group. Including this field is a necessary measure since the groups are managed by users and to support group management interfaces in other components, a read operation is provided that retrieves all groups that are manageable by a specific user.

The identifiers of the groups are created outside of and are handed over to the behaviour engine. Therefore, create and update operations are handled by the same process, which simply overrides an existing group if a group id is requested that already exists. This means that if a group is updated, all information present in the update is set as the new information in a group. Deleting groups simply requires their identifier in a request.

The behaviour engine offers Group Suggestions as a high-level context type to support group creation. Once a group is created, the behaviour engine creates updates its group suggestions in every analysis cycle as described in Chapter 6.6.2 and is then made available in every component that works with groups since they are always present when a group is queried. An example of a group suggestion can be seen in Listing 6.17, where the first identifier, which is named "deviceGroupSuggestionId" in this case, refers to the internal identifier this suggestion has within the behaviour engine, the "behaviourId" refers to the entity the suggestion is made for, and the last identifier refers to the suggested entity. The two numerical values follow the description in Chapter 5.4.

6.8.3. Access Attribute Management

The last manageable setting in the behaviour engine involves both groups and behaviour patterns and sets which groups are used when a pattern is used in the evaluation described in Chapter 6.7.2. The two management options in this component are setting a list of user groups and setting a list of device groups for the evaluation of a behaviour pattern. Both operations only require the unique publishing key of the pattern to identify and the list of either device groups or user groups. Note that this management component only works with control event pattern ids due to groups being compatible only with that pattern type.

Each pattern is individually associated with groups and the behaviour engine does not restrict how many groups are included in the decision-making process and allows device and user groups to be used in parallel. An update performed in this component has either a list of device groups or a list of user groups and the list in the update message is overriding any previous list associated with the pattern. This means, that to associate an additional user group with a pattern, all previous user groups and the new one need to be part of the list in the update message. To delete all device group associations on a pattern, an update message with an empty list needs to be sent.

6.9. Testing

Every software component needs to reliably meet its requirements [51], a state that can only be confidently reached after the software has been thoroughly examined and validated [60], which includes a process featuring many layers of testing [91]. The behaviour engine is tested on mostly the two lowest layers, namely the unit testing layer and the integration testing layer. This section provides the most important technical details of the tests on each layer.

6.9.1. Unit Tests

The four internal components of the behaviour engine, as seen in Figure 6.2, have been kept at a nearly 100% test coverage in unit tests [91] during the whole duration of the project, improving the quality of the components and development speed throughout the implementation phase significantly. The tests of this layer are using the open-source java libraries hamcrest [39], mockito [65], and hamcrest-optional [40]. All tests on this layer are performed with artificially generated data for every data object that is relevant for a test to prove that a tested method can work with any form of input.

For this testing layer, all tests have been structured to isolate one software class, while all related classes are simulated. This means that all dependencies within a class are simulated to provide an expected result for a provided input and the tests measure whether the output and the interactions with other components have been done properly. This setup is very useful when an error is encountered, because then only the tests in the broken component indicate an error, while other tests indicate that if the broken component worked as intended, there would be no further issues in the system. Furthermore, due to the unit tests being quickly executed and due to their role of testing correctness of small parts of the implementation [91], unit tests are executed on every compile of the behaviour engine.

```
@Test
handleEvaluateControlEventMessage_validRequest_messageHandled() {
    //Given
    publishingKey = createPublishingKey();
    timeStamp = createTimeStamp();
    expected = createControlEventDecisionResult();
    prepareControlEventDecisionEvaluator(
        publishingKey, timeStamp, expected);
    attribute = createValidAttribute(
        createRequestAttribute(publishingKey), timeStamp);
    request = createAttributeValueEvaluationRequest(
        Collections.singletonList(attribute));

    //When
    received = sut.handleEvaluateControlEventMessage(request);

    //Then
    assertSingleResult(received.getMessage(), expected);
}
```

Listing 6.18: Example unit test.

An example for a unit test can be seen in Listing 6.18, where the general structure that is followed by each unit test can be seen. The naming pattern for each unit test is <tested method name>_<preconditions>_<postconditions>, which enables a quick assessment of what a test does. Furthermore, the tests are divided into a "Given", a "When", and a "Then" part so that the tests are easily readable. The setup and the assertion of the test can have any number of steps, while the when always only has one step, namely the tested method being called on the system under test.

6.9.2. Integration Tests

After the unit tests, which are responsible for isolated testing of functionality [84] to prove the correctness of the components independently [91], it is necessary to investigate whether the components work together properly. This is done with integration testing [17], which is the testing layer in which all components are running during the test. It is still possible to use artificially generated data at this testing step, but even the components that are not being used in a specific test case should be running and no component should be simulated for this type of test. The evaluations found in Chapter 7.1 and in Chapter 7.2.2 are structured as integration tests as well.

```
# Integration Test
for each feature to test
    clean Integration Test database
    generate setup data
    fill database with data
    call method of behaviour engine
    retrieve results
    output results
```

Listing 6.19: Pseudo-Code: Integration Test

A separate integration test script is provided for every feature of the behaviour engine in a supporting project dedicated to testing the behaviour engine. Listing 6.19 displays how an individual integration test is structured and shows how the general script calls them sequentially. All integration tests can be executed independently from each other since the database is always in a clean state. While it is, therefore, possible to execute all integration tests at once with the test script, using the individual test scripts separately might be a good choice since some tests take longer and the output becomes bloated.

Each test has the same general structure, as can be seen in Listing 6.19. First, the database is cleaned in case data from a previous test is still present. Note that the integration test database is separated from the general behaviour engine database, which is why data of a running smart home environment is preserved even when the integration tests are executed. After the database is empty, new data is generated and stored so that the test can be executed and once the scenario is processed, the output of the test is retrieved and printed to the console.

6.9.3. System Tests

The last layer of the testing spectrum [60] this work includes are system tests [97], which are used to validate the behaviour engine to work properly within the smart home environment. In these test cases, every component in the environment is running and the functionality of the behaviour engine is validated by accessing its features as a user or another smart home component would if the system was deployed.

To test the features of the behaviour engine without having to run the environment for several weeks, a script builds up a sufficient database for all features to be examinable. The database is set up according to the CosyHome dataset [25], which is described in detail in Chapter 7.1, with all behaviour patterns that would have been created if the data was gathered in real-time.

6.10. Visualization

A visual representation can generally be used to convey an idea or a concept effectively and easily. Since context-awareness and IoT share the trait of being challenging to grasp for users [69] and since the target user group cannot be expected to have a higher understanding of the concept [27], it is necessary to display the information in a simple and understandable way.

Visual representations of both behaviour pattern types are made available for the web interface of the smart home. As stated at the beginning of the chapter, the web interface is not a part of this thesis and other visual aids for smart home residents are only described in their relation to the behaviour engine. The behaviour pattern visualizations are nonetheless added in this work because they belong semantically to the behaviour patterns and are a helpful display of the data represented by a behaviour pattern. Therefore, the pattern visualizations are presented and explained in this section.

6.10.1. Periodic Event Pattern

The periodic event patterns are visualized in a chart with two dimensions, since the evaluation of a periodic event, representing a measurement of a smart sensor, depends on the measured value and on the time the measurement is taken. Therefore, the y-axis shows the numerical value of the measurement, while the x-axis shows the time and in the graph, the expected value and the confidence interval are shown for every time slot. An example of a visualization of a periodic event pattern can be seen in Figure 6.5.

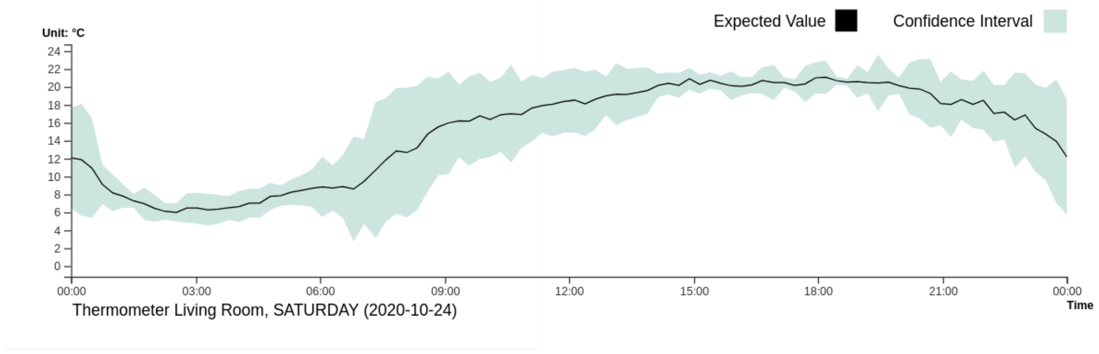


Figure 6.5.: Example visualization of a periodic event pattern.

Note that the confidence interval within periodic event patterns is independent of the certainty in attribute values described in Chapter 6.5.2. The described certainty is always set to 100% due to the event evaluation being deterministic, which means that a periodic event will always give the same result when evaluated against a specific pattern. The confidence intervals in periodic event patterns serve to indicate the range of values in periodic events that are accepted by the behaviour engine.

6.10.2. Control Event Pattern

Figure 6.6 displays an example for a control event pattern generated using the CosyHome dataset [25], which is created as part of this work and introduced in Chapter 7.1. The evaluation of a control event, representing a user-initiated interaction with a smart home device, only depends on the time of the interaction. Therefore, the visualization only needs to displays one axis, and the crucial information presented in the graph is the time spans in which an event is accepted. The displayed pattern is one of the behaviour patterns generated in a test run during the system tests with all associated events in the CosyHome dataset, which means that this is the final pattern for its specific set of metadata.

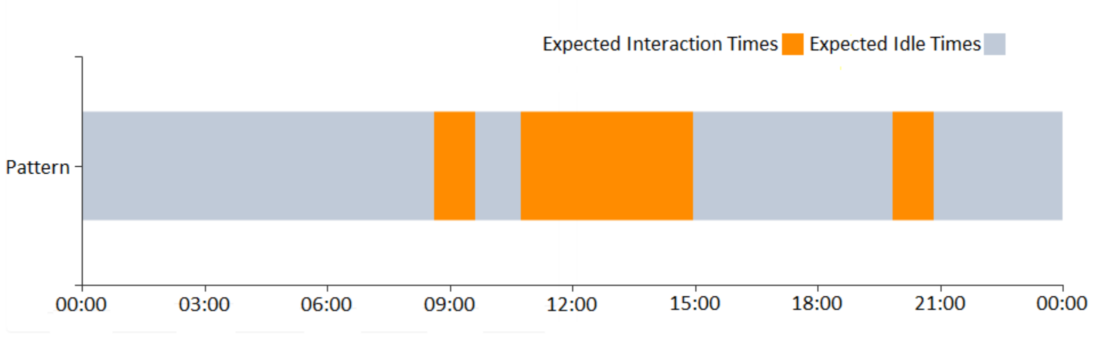


Figure 6.6.: Example visualization of a control event pattern.

The design choice of the visualization is to show only the information the user needs to know to make sense of what the pattern is. Therefore, it is sufficient to know for which times an interaction is expected, and more detailed information like the usage clusters together with their centers and standard deviations are not displayed.

7. Evaluation

So far, the design and the implementation have been discussed in detail in the previous chapters. It is necessary to validate that the first two research questions can be closed by proving that the specified design of behaviour patterns represents behaviour properly and that they can be used in access control. For the third research question, it is necessary to measure how behaviour patterns develop over time. Furthermore, the performance of the introduced methods in regards to accuracy and processing time needs to be evaluated.

In total, four different evaluations are performed to investigate the solution design presented in this thesis from different angles. During the evaluations, the solution design is represented by the behaviour engine and two types of data are used, which furthermore serve as a basis to group the evaluations. The first form of data is synthetic, enabling parameterized test scenarios [38], while the other form of data is created in a case study as a part of this thesis, enabling tests against real-world data. The results of this chapter serve to answer the third research question and to prove that the goals of the first two research questions have been reached.

Chapter 7.1 features two evaluations based on CosyHome [25], the dataset created as a result of the mentioned case study. Therefore, it leads with a section describing the most important details of the dataset for the evaluation, while finer details are extracted in Appendix A. By basing the investigation on data collected in a real-world setting, the results are meaningful, comparable and give a general idea of how well the design and implementation would perform in the real world.

After the analysis based on real-world data, two evaluations based on synthetic data in Chapter 7.2 are examining different aspects of the behaviour engine by exploiting the advantages artificially created data provides. These evaluations determine how well the behaviour engine performs in a parameterized test setup with different configurations.

7.1. CosyHome - Big Data Evaluation

To evaluate the behaviour engine and the solution design of this work, it is necessary to use data, which can be translated into events in a smart home environment. While it is possible and in many ways beneficial to create this data artificially [38], a problem arises if all evaluations are performed on synthetic data since data generated by an algorithm introduces a bias by how the data is expected to be.

Besides the benefits and the usefulness of artificial data, it is not sufficient to base an entire evaluation only on this type of data without including data from the real world. Therefore, the tests presented in this section are based on a dataset, created in the scope of this work to fit the requirements of this thesis.

7.1.1. Test Subjects

In total, there are eight test subjects in 4 different homes. One of the homes houses a dual-income family with two children of school-age and therefore the requirement of having a dual-income family for analyzing their special role in the smart home context [27] in the dataset is fulfilled. The other homes house one elder couple and two young people living alone respectively, which also gives some variety with the inhabitant setups.

7.1.2. Data Gathering

Optimally, the data is gathered in a real smart home with smart devices and sensors the users interact with while the behaviour engine captures and stores all generated events without any additional input. Since real smart homes are necessary for this approach, there are three possibilities to gather the data. The first way is to have people living in smart homes willing to install the behaviour engine in their homes and record themselves for a few weeks. The second way is to upgrade the home of the subjects to be smart by replacing sufficient devices with smart devices for creating a meaningful dataset. The last option is to simulate a smart home in a test lab and put the subjects into this environment like in Orange4Home [26]. None of these options were acceptable, because no real smart home was available, upgrading real homes is very expensive and the test lab would not allow a full simulation with multiple people being present due to capacity issues.

Since the data could not be gathered in a real smart home, the closest possible environment is a non-smart home, since all smart devices that would be interacted with in a smart home have a non-smart counterpart in a non-smart home [80]. The only necessary change to the optimal approach is that the subjects have to manually record their behaviour instead of the smart home doing it automatically for them and instead of the data being extracted from a database in the smart home, the data needs to be translated by the researcher to a digital format for evaluation. This puts only a little effort on the side of the test subjects since they just have to note on a list whenever they use a device in their home, and only a reasonable extra effort onto the researcher.

The data was gathered over the course of nine weeks from the 17th May until the 13th July in 2020. Monitoring the interactions whenever a device is used for nine whole weeks would have demanded too much from the subjects and would have been unnecessary for the evaluation. Since the concept only requires the same day consecutively and to have one weekday and one weekend-day, only Mondays and Sundays were recorded.

7.1.3. Translating Raw Data into Dataset

The events in the smart home are generated with timestamps, but not all devices used by the subjects enable clear identification of singular timestamps since many devices are used over a period of time. Therefore, the devices are divided into single-use devices, like a coffee machine, and continuous-use devices, like a stove. The characteristic of single-use devices is that at the beginning of the usage one button is pressed, triggering the device to do something for the subject. Continuous-use devices are characterized by being used by a subject over a period of time, where the settings of the device can be changed many times in the time period.

The subjects were asked to fill in the starting and end times for the usage of continuous-use devices and the events in the database are translated from these time ranges. The rules for the translation are, that the starting and end times get a timestamp, if there is more than an hour between the timestamps, two timestamps half an hour closer to the middle are added, if there is more than half an hour but less than an hour between the timestamps, another timestamp is added in the middle and if there is less than half an hour between the timestamps, no further timestamp is added.

For the single-use devices, the subjects were asked to write down the usage times plus the number of usages, so that they do not need to write multiple timestamps when they are using a single-use device multiple times, i.e. when they are preparing breakfast and using the toaster multiple times, they just need to write the timestamp of the first usage and the number of usages. The translation rules for single-use devices are simple since the starting time is added as a timestamp and then in an interval of two minutes, timestamps are added to match the number of usages.

7.1.4. Test Execution

The big dataset is used in multiple tests that are executed like integration tests [58]. This means that the behaviour engine is running with all features available while the testing script addresses only the functionality necessary for the current test. The tests are set up this way to show how and which patterns would be created and how many events would be categorized as expected in the real-world setting by the behaviour engine. Every test is repeated multiple times to generate an average result with a low enough standard deviation regarding how many events are accepted compared to how many are declined by the behaviour engine, which is a necessary measure since K-Means clustering [99] is a non-deterministic algorithm.

The test setups are divided by included features of the behaviour engine and all setups include an evaluation based on the complete CosyHome dataset. The created test setups are pattern generation only, pattern generation with user groups, pattern generation with device groups, pattern generation with external validation, and all features combined. The user groups, that are added into the user group test and the combining test only affect two of the four homes, and only the adults in these homes are put into the user group, meaning that the test overall included two user groups with two people each. The device groups added into the device group test and the combining test, are for each home all kitchen devices in one group, and all devices commonly used in free time in another group. The device groups are assigned to all users in their respective homes, while the user groups are assigned to every pattern for the users in the group.

The test runs provide two separate sets of results, one for the Mondays in the dataset and one for the Sundays in the dataset, which is a necessary distinction since patterns created for different weekdays are considered non-related. In this scenario, all events are evaluated against the patterns that a deployed behaviour engine would have evaluated them against based on the dataset.

7.1.5. Results

Figures 7.1 and 7.2 display the decision results of all events in the CosyHome dataset evaluated by the behaviour engine for Sundays and Mondays respectively. These figures show the fraction of the events observed in the real-world dataset that would have been accepted by the behaviour engine in percent. Both figures display five different test setups, which are distinguished by the included features of the behaviour engine.

The evaluation is set up to mimic how the events would have been raised in the smart home at their respective timestamp and with the analysis phases between the days to create the new behaviour patterns, including all events that are gathered between the learning phases. This allows to split the events up by the day they are raised according to the dataset and track how the number of accepted events shifts over the weeks in the evaluation. Each evaluation is repeated 50 times to eliminate inaccuracies introduced by the non-deterministic nature of the involved algorithms.

The most important information, that can be read out of these graphs, is the approximate learning time of the behaviour pattern by examining the fraction of false negatives [75], which is why the results of this evaluation provide a ground to answer the third research question. The rate of true positives in proportion to all positives, regardless of being classified correctly or incorrectly, is referred to as recall [28]. Optimally, the inhabitants are bothered with as few falsely categorized events as possible, since a user constantly being second-guessed or stopped by the system does not feel safe and in control of his smart home, but rather controlled by the smart home [27], which is why the recall value should be as high as possible.

Sunday Pattern

Figure 7.1 displays the test results for Sundays. All events are accepted in the first week since there is no pattern against which the events could be evaluated at this point, which is why all events are naturally accepted by the behaviour engine. The second week has a big drop in the recall [28] by falling below 60% in the test setup with only pattern generation, which shows the usages of devices differed significantly between the first two Sundays. The next week, the patterns recognized a lot of the events correctly, with three of the five test setups even having their overall second highest result in that week. After that, the recall value stays approximately the same for each test setup for some weeks with the two test setups with the lowest recall values at that point showing a positive trend. The second last week show a smaller drop of up to 6% on most test setups, followed by the highest individual result for four out of five test setups at the end of the testing phase.

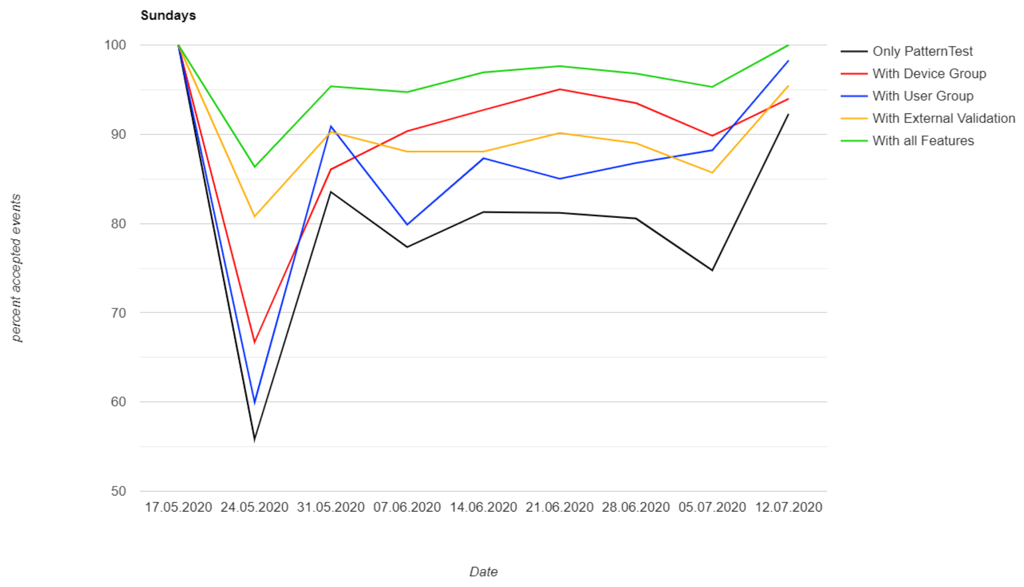


Figure 7.1.: CosyHome: accepted events in percent for Sundays.

There is one major drop and two minor drops for the recall value, which happened on the 24th of May, on the 7th of June, and on the 5th of July, with all of the drops happening for a different reason. The first drop happened as expected in the second week of the testing phase with even the drop to slightly below 60% in the test setup without additional features not being unexpected. Instead, it indicates early unreliability of behaviour patterns, especially when only one week of data is available. The second drop in the graph, which is less significant than the first drop, affects two test setups significantly, which are the setup without additional features and the setup including user groups. The setup with external validation, which eliminates clusters of falsely classified events, does not drop significantly in that week, indicating that there are only a few unexpected activities performed by the subjects. This caused a lot of unexpected events in a short period of time, which could not be caught by some features. The third drop correlates with the first real Sunday of summer, which probably caused the subjects to show differences in their behaviour. The recall going up sharply in the week after the third drop is an indicator that this drop can be interpreted as a correction of the behaviour patterns.

Monday Pattern

In Figure 7.2, the graph with the recall results for Mondays is displayed. Like for the results for Sundays, the first values are all 100% by default, so the first entry for each test setup is by nature always that value. From that point onward, the progression differs a lot from the progress of the previous graph. For the Monday graph, the drop observed in the second week is much smaller than its counterpart in the Sunday graph, with the lowest values being the only pattern test setup and user group test setup, which drop to roughly 70%. Furthermore, all test setups have a higher recall value with a difference of up to 15% compared to the Sunday graph and the second week has only the third lowest value observed in the graph, unlike the Sunday graph where the lowest value could already be seen in the second week. Until the 15th of June, only the external validation setup shows a slight negative trend. On the 15th of June, the lowest value in the graph can be seen in the test setup without any additional features, which drops to slightly above 65%. It is notable that from the remaining setups only the user group setup drops significantly this week while all other setups remain roughly the same as the week before. In the following week, a sharp rise can be seen in all test setups, which is followed by the last drop on the 29th of June. After that, two weeks follow with all setups rising with only one value staying below 90% in the last two weeks.

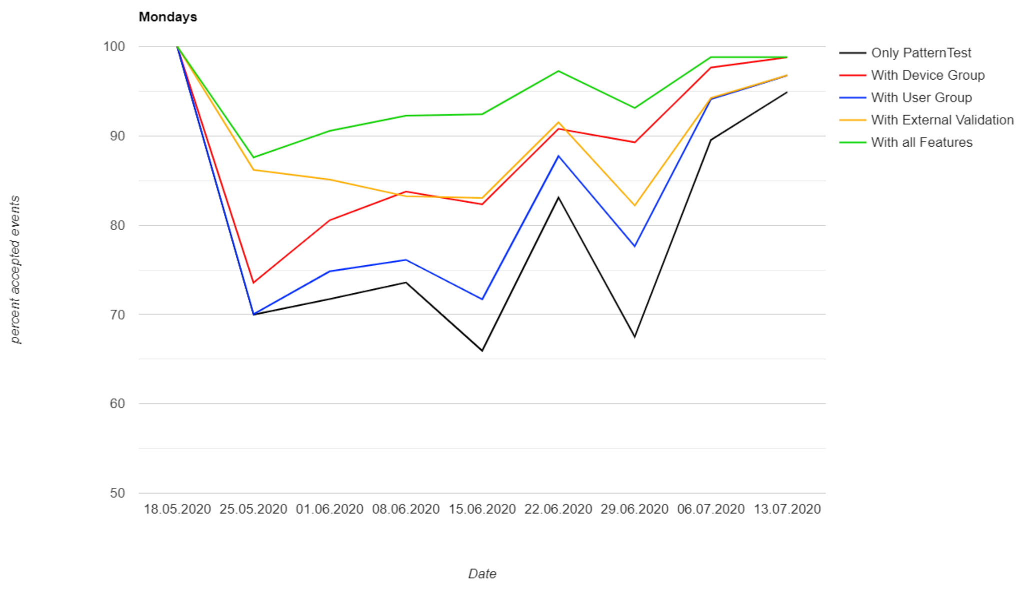


Figure 7.2.: CosyHome: accepted events in percent for Mondays.

While there are several observable drops, especially in the test setup without additional features, the recall values never drop below 65%. Notably, the first weeks of the dataset record the device usages of people who switched to home office during the first 2020 lockdown in Austria. This explains the initial drops, which still are not as severe as the drops seen in the Sunday graph. The second notable drop is only observable in the test setup without any additional features and with user groups, indicating that the activities weren't unexpected, but were likely performed on a related object. The last drop on the 29th of June correlates with the first day in which the schools are closed for summer vacation, which can be interpreted as a correction of the behaviour patterns. Notably, the results of the test setup with device groups outperforming the other test setups in later weeks indicates that the activities performed over the course of the data gathering generally follow behaviour that can be estimated very well by including semantic relationships between the objects.

Conclusion

Overall, the patterns seem to be more reliable on Mondays than on Sundays, even if it does not seem like it at first by looking at the graphs. If the results for the 7th and 8th week are ignored, since both graphs display a drop in only one of these two weeks, which makes a comparison difficult, the most important weeks to look at are the 6th and 9th weeks. In these weeks, the Monday test setups generally outperformed the Sunday test setups, even though it is rather close in most cases. Another interesting phenomenon that can be observed is that the Sunday pattern reached a more stable state much quicker than the Monday patterns since the recall value on the Sunday patterns did not change too much after the third week, while the Monday pattern needed four to five weeks. This could be due to the working and school activities were going through a lot of changes in the weeks of the case study, which affected Monday events more than Sunday events.

There is one more insight that can be gained using the results of the tests, which is an estimation of a minimal learning time for patterns to be considered useful. Looking at the lines of the test setups including all features seems the most intuitive due to it consistently displaying the highest result, but groups need to be configured by the users themselves, which makes them an optional feature. Therefore, the three setups containing at least one optional feature cannot be used as the standard for the estimation. The line pattern with external validation can be used, because external validation is not an optional feature, due to it being triggered always when an event gets rejected regardless of the current configuration. In both graphs, the line with external validation hits the 90% mark in week six after having five weeks of data in the system. After that, even a significant event like the school closing for summer holidays does not cause the graph to drop below 80% anymore and for this reason, the estimation for a minimal learning time until the pattern can be considered useful, should be at least five weeks long.

7.1.6. Interchanging Context Test

Another evaluation is performed to determine how much impact exchanging one of the context types in the metadata of the behaviour pattern has on the recall value during the previous evaluation. There are three options for being exchanged, namely the user, device, and time context. Out of the three choices, it is the time context that is the least transparent since it is the most difficult to estimate how much the interactions differ between the weekdays compared to between users and between devices.

The setup for this evaluation is almost equal to the previous evaluation with the only difference being that the behaviour pattern an event is evaluated against is interchanged with its counterpart of the other weekday. This means, that i.e. the Monday events of the third week in the dataset are evaluated against the Sunday behaviour patterns available for the third week. For determining how significant the differences are, the results of this evaluation are compared against the results of the previous evaluation in Chapter 7.1.5.

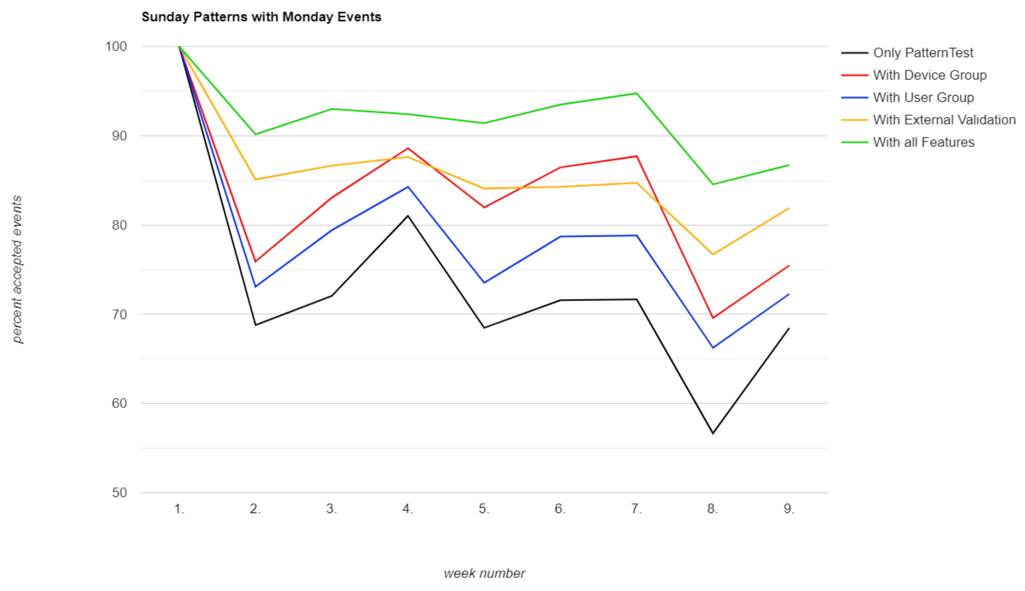


Figure 7.3.: CosyHome: Monday events evaluated against Sunday patterns

In Figure 7.3, the results of the evaluations of Monday events being compared against the respective Sunday patterns. The first insight gained in this chart is that the results do not seem to converge and the progress seems like a triangle downwards move with all test setups, since the later the result is, the worse it is. Interestingly, in the second week, the normal evaluation seems to be outperformed in most test setups by interchanging the time context, which indicates the unreliability of the behaviour pattern if they are based on too little data.

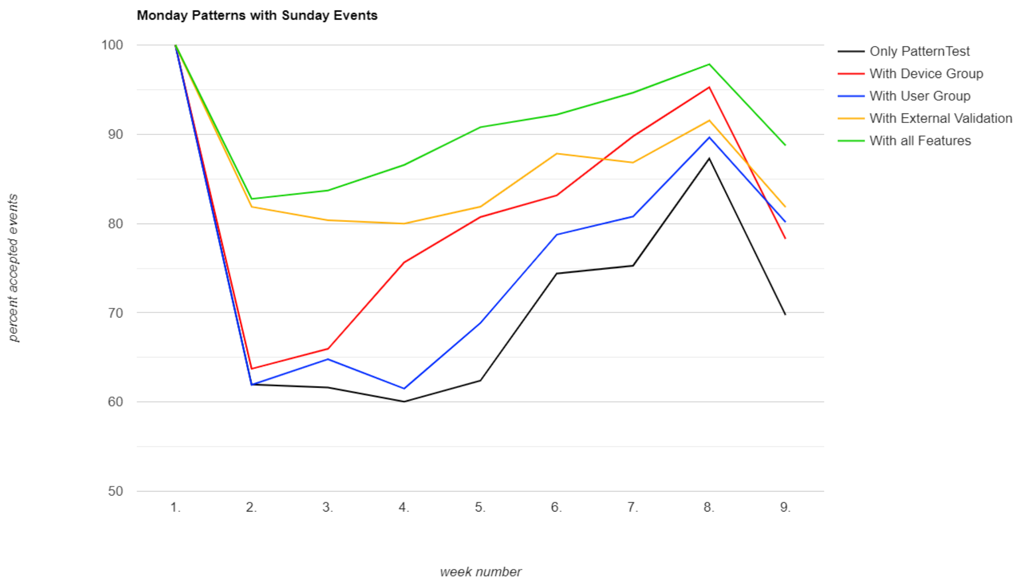


Figure 7.4.: CosyHome: Sunday events evaluated against Monday patterns

The chart in Figure 7.4 creates the impression that comparing Sunday events against Monday patterns produces better results than the other way around, but when compared against its counterpart from the second evaluation, the results of this evaluation are outperformed at almost every point.

Overall, the evaluations prove that changing the time context has a significant influence on the reliability of the estimation models of the behaviour patterns. There are two big phenomena, that are observable in the charts of this evaluation compared to the charts of the evaluation in Chapter 7.1.5 besides the difference in the recall values.

The first observable phenomenon is, that more than half of the test setups in this evaluation, namely six out of ten, never hit the 90% mark, unlike in the previous evaluation setup, where ten out of ten hit the 90% mark. The second insight gained in this test is that in the previous evaluation, the inclusion of groups created a better recall value on average than the inclusion of external validation, while in the interchanging context setup only scattered data points can be found where the device group outperformed the external evaluation, while the user group did not even once outperform the external validation.

7.2. Synthetic Data Evaluation

Generating data for an evaluation can be very time-consuming and expensive, especially when the data has to be gathered over the course of several weeks as is the case for the evaluation of the presented solution design. To a certain extent, it is possible and even useful to quickly generate data following an algorithm, since multiple test scenarios can quickly be executed and their results can be used to improve or validate a design.

Synthetic data is useful for evaluation due to its characteristics of being created fast and cheap [38], which is why a performance evaluation can greatly benefit from being performed with many different test setups. In the scope of this work, a performance evaluation of processing access control rules is necessary to fully close the second research question, since exposing the knowledge of behaviour in the system to an ACM includes the requirement of the processing of this knowledge to be possible at runtime. Therefore, Chapter 7.2.1 is dedicated to examining how well the behaviour engine performs in evaluating the attributes it adds to the ABAC of the smart home environment.

Another characteristic of synthetic data allows the evaluation to be based on the correct reference values, which otherwise have to be provided externally or manually [38]. The evaluation presented in Chapter 7.2.2 provides an estimation of how effective the behaviour engine operates with smart sensors in regards to its correlation feature under the assumption that the measurements of the smart sensor can be described by a normal distribution. Additionally, insight was given into how extreme values affect the outcome of this evaluation.

7.2.1. Policy Evaluation Performance

One of the design requirements set by the second research question is the integration of behaviour patterns into the access control model of the underlying smart home environment. One important aspect of access control is the policy evaluation performance since access control is performed in real-time as part of every user request, which potentially delays the response enough for the users to become impatient [52] and reducing the overall QoE for the user [18].

All attributes introduced in the scope of this work need to be processible in an acceptable time for the user and two of the three attributes are examined in this test. The periodic event pattern evaluation is excluded from this test because an access rule cannot be based only on a periodic event pattern. Furthermore, periodic event pattern evaluation is independent of the two parameters influencing the performance of the evaluation of control events, effectively making its evaluation time always equal to the simplest form of control event pattern evaluation, which includes only one pattern with one usage cluster.

The two parameters relevant for the performance of control event pattern evaluations are groups and the number of usage clusters. Groups allow multiple patterns to be included in the evaluation and each usage cluster needs to be checked to find the closest to the examined event, as described in Chapter 5.5.

Environment Setup

All policy evaluation performance tests are executed in a Raspberry Pi 3 Model B Rev 1.2 [78] environment with a debian version of 10.9. A Raspberry Pi is a cheap alternative to a computer for running a smart home application, while still preserving the ability to run multiple processes, unlike a microcontroller [72].

For this evaluation, two smart home components are running on the Raspberry Pi, namely the FACA and the behaviour engine. These two components were chosen due to their responsibility of performing the logic of the ACM and the goal of this evaluation is to investigate whether the ACM can be performed in a realistic environment with little computational power [72]. The test script is run on an external computational node and is responsible for mimicking the remaining smart home components and measures the time between sending the access request and receiving the response.

Test Setup

The goal of this evaluation is to prove that including behaviour in the access control of a smart home environment does not extend the policy evaluation time enough to reduce the QoE [18]. As a measure for the QoE, it is possible to use the relation between cancellation rate and response time of a web service, which shows that response times ranging from 50 milliseconds to 500 milliseconds have approximately the same cancellation rates [52]. Ideally, the inclusion of behaviour patterns in the ACM does not extend its response time to be bigger than 500 milliseconds.

This evaluation provides an indicator of whether the implicated requirements of the second research question are fully satisfied. Therefore, it is necessary to examine the performance times when evaluating access policies under various conditions, which have to include test setups with the highest reasonable boundaries.

Based on the two parameters impacting the evaluation performance of control events as stated above, twelve test setups are created and divided by the number of involved patterns into groups of three tests. Furthermore, a reference test is performed for each of the groups, where an access policy with an equal number of standard numeric access rules is processed. The three behaviour-related tests in each group include one test where all included patterns have one usage cluster, one test with ten usage clusters in each pattern, and one test with twenty usage clusters in each pattern. The upper bound of twenty usage clusters is chosen due to it being the upper bound in the analysis process in the behaviour engine.

For the test setups regarding how many patterns are processed in one access policy, it is necessary to find reasonable scenarios, with one high but realistic number as upper bound. There are approximately 6.58 connected devices per person [92] in the year 2020 and with this number as guidance, an upper bound of fifty patterns is chosen, which would allow for a device group including all devices in a household of seven people rounded up. While the device group feature is not intended to include all devices of a home and while the average number of connected devices does not give insight into the relationship between the number of devices and the number of people in a smart home, it is a good way of estimating an upper bound in an extreme, but not impossible scenario.

The other test scenarios include one test with only one pattern, which allows the examination of single pattern access policies and gives furthermore an estimation for periodic event pattern evaluations. Another test setup includes ten patterns in a group evaluation, which is a number chosen by rounding up the number of patterns included in the biggest group found in the CosyHome dataset. Therefore, the second test setup simulates groups of patterns based on the CosyHome dataset. The last test setup includes 25 patterns in the policy evaluation, which is chosen similarly to the test with fifty patterns by estimating how many devices a household with four people might have and include all of these devices in the group.

Results

The results of this evaluation are displayed in Figure 7.5. Note that all times are measured from the point of sending the access control request up to the point of receiving the response to the request. This means, that a messaging overhead is included in all tests and additional messaging overhead is added in all tests including behaviour access rules because the behaviour engine is contacted by the access control component over the messaging infrastructure.

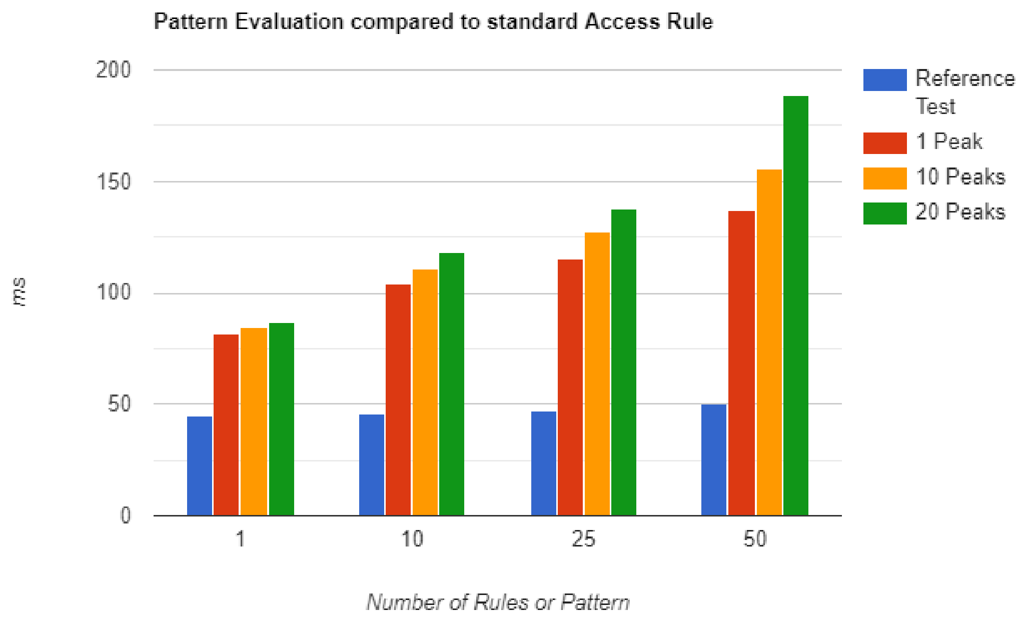


Figure 7.5.: The results of the policy performance evaluation.

The two parameters examined in this evaluation have shown that the number of usage clusters only have a significant influence in case enough behaviour patterns are involved in the evaluation and the number of patterns in the group seems to cause a linear growth in evaluation time. Overall, the results are satisfying since even the response time from the access control component in the upper bound scenario stays below 200 milliseconds, which is far below the 500 milliseconds that were aimed for. Note that the response times in the result are measured by repeating the test 1000 times to create an average with minimal fluctuation from external factors such as network issues.

7.2.2. Correlation Test

The last feature, which is presented as part of the solution design and has yet to be included in an evaluation, is the correlation between two patterns introduced in Chapter 5.6.3. To evaluate this feature, periodic event patterns are being used in a parameterized test scenario to investigate how the correlation values behave in different scenarios.

For the test, data is produced sequentially, so that the repeated analysis process creates patterns over a course of twenty simulated weeks. By setting the test scenario up this way, the amount of data present in each generated pattern is proportionate to their associated week number, meaning that i.e. pattern number five is based on five weeks of data. Since the periodic event pattern divides its periodic events into time slots, one week of data equals one periodic event for each time slot.

The aim of this evaluation is to compare how extreme values affect the periodic event pattern by using the correlations between the patterns as an indicator of how much a certain pattern deviates from other patterns. This evaluation is performed only for periodic event patterns because periodic event patterns feature always the same number of comparable values based on numerical measurement of smart sensors. Control event patterns are based on a clustering algorithm, which would require many assumptions about the characteristics of how many clusters are present, their standard deviations, and how extreme values are defined in their case.

General Setup

The simulated scenario in this evaluation is one pattern being created every simulated week starting with the second week, which has two data values in each time slot of the periodic event pattern and ending with the twentieth week, which has twenty data values for each time slot. The simulated data values follow an initially defined reference pattern, assuming a normal distribution of the numerical values in the periodic events.

The reference pattern has a randomized average and standard deviation for each time slot and since the results are not dependent on the actual numerical values of the average and standard deviation in the time slots, the actual numbers are not included in the result. The lower the resulting correlation between two patterns is, the more they deviate from each other and the most interesting correlation values are the ones comparing a pattern with his preceding or succeeding pattern. Each test is repeated 100 times to create an average resulting correlation value with a low enough standard deviation to provide a meaningful result.

Results

This section displays the results of the correlation evaluation and features three tables, namely Table 7.1, Table 7.2, and Table 7.3. These tables are read by looking for the correlations between two patterns, with each pattern's correlation being displayed to every other pattern created in the evaluation. Each cell provides two percentage numbers, with the upper number indicating the average correlation value and the lower number indicating the standard deviation.

For example, to find the correlation of the pattern of the second week and the pattern of the sixth week, the values can be read out in the 2nd row and 6th column or in the 6th row and 2nd column. Both cells have the same numbers, which is why both ways are a valid way of finding the result, making each week's pattern correlation results equally readable. Note that the fields with the correlation a pattern has with itself are always gray due to the correlation always being 100% in that case.

The first test setup follows only the general setup and the results are displayed in Table 7.1. The results show that the first pattern deviates consistently the most from each other pattern and the correlations including the first generated pattern generally have the highest standard deviations. The result table of this test can also be used to estimate how many data values and therefore how many weeks are usually necessary to get a meaningful periodic event pattern.

Ideally, more values are available before the periodic event patterns are used in access control, but the feature should also be made available at a reasonable time. Looking at the results, after five data values are present, the correlation to newer patterns does not drop below 80%, which means that the patterns stay relatively similar after having five data values. In Chapter 7.1.5, a minimal learning time of five weeks is suggested for control event patterns to stabilize, resulting in an equal learning time for both pattern types.

Table 7.1 shows the results of the first test run, in which no extreme values are included in the data. This means, that in the first test run all the data values are falling into the ranges of one, two, or three times the standard deviation of the reference pattern. The second and third test setups are adding extreme values every fourth week instead of the normal data value. In the second test, which is displayed in Table 7.2, the extreme values are generated by using a ten times higher standard deviation than in the other weeks.

#	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	▨	73.67 2.33	69.85 2.4	68.02 2.53	66.97 2.7	66.22 2.73	65.74 2.64	65.32 2.68	65.03 2.82	64.79 2.82	64.51 2.87	64.24 2.79	64.08 2.79	63.99 2.74	63.89 2.82	63.78 2.83	63.73 2.86	63.6 2.81	63.52 2.85
3	73.67 2.33	▨	84.69 1.51	81.22 1.83	79.16 1.9	77.91 1.92	77.11 1.93	76.37 2.02	75.9 2.12	75.48 2.16	75.12 2.07	74.85 2.05	74.59 2.07	74.43 2.04	74.23 2.09	74.1 2.1	73.96 2.07	73.85 2.12	73.74 2.11
4	69.85 2.4	84.69 1.51	▨	89.18 0.89	86.12 1.09	84.36 1.19	83.2 1.25	82.33 1.33	81.68 1.37	81.16 1.46	80.68 1.49	80.27 1.52	80.03 1.55	79.82 1.61	79.63 1.66	79.44 1.68	79.32 1.7	79.14 1.65	79.02 1.69
5	68.02 2.53	81.22 1.83	89.18 0.89	▨	91.64 0.69	89.15 0.79	87.56 0.87	86.44 0.95	85.65 1.09	85.02 1.21	84.5 1.24	84.05 1.3	83.73 1.29	83.45 1.29	83.21 1.37	83.01 1.39	82.86 1.41	82.68 1.38	82.54 1.38
6	66.97 2.7	79.16 1.9	86.12 1.09	91.64 0.69	▨	93.12 0.47	90.9 0.7	89.46 0.73	88.49 0.8	87.69 0.88	87.11 0.92	86.67 1.04	86.31 1.09	86.0 1.02	85.69 1.06	85.47 1.11	85.26 1.08	85.08 1.12	84.92 1.15
7	66.22 2.73	77.91 1.92	84.36 1.19	89.15 0.79	93.12 0.47	▨	94.13 0.42	92.19 0.67	90.95 0.78	90.01 0.83	89.31 0.9	88.78 0.96	88.33 0.98	87.99 0.99	87.64 1.05	87.36 1.06	87.13 1.09	86.94 1.14	86.76 1.15
8	65.74 2.64	77.11 1.93	83.2 1.25	87.56 0.87	90.9 0.7	94.13 0.42	▨	94.91 0.4	93.18 0.51	92.02 0.65	91.2 0.73	90.61 0.79	90.09 0.76	89.72 0.83	89.34 0.89	89.01 0.92	88.72 0.94	88.49 0.94	88.28 0.96
9	65.32 2.68	76.37 2.02	82.33 1.33	86.44 0.95	89.46 0.73	92.19 0.67	94.91 0.4	▨	95.49 0.35	93.92 0.56	92.83 0.59	92.14 0.64	91.51 0.64	91.12 0.65	90.69 0.77	90.33 0.79	90.02 0.84	89.79 0.89	89.58 0.88
10	65.03 2.82	75.9 2.12	81.68 1.37	85.65 1.09	88.49 0.8	90.95 0.78	93.18 0.51	95.49 0.35	▨	95.97 0.33	94.57 0.47	93.61 0.55	92.9 0.58	92.4 0.6	91.91 0.68	91.54 0.7	91.19 0.75	90.92 0.78	90.71 0.77
11	64.79 2.82	75.48 2.16	81.16 1.46	85.02 1.21	87.69 0.88	90.01 0.83	92.02 0.65	93.92 0.56	95.97 0.33	▨	96.32 0.32	95.01 0.4	94.17 0.43	93.55 0.48	93.0 0.57	92.56 0.61	92.16 0.67	91.84 0.73	91.59 0.74
12	64.51 2.87	75.12 2.07	80.68 1.49	84.5 1.24	87.11 0.92	89.31 0.9	91.2 0.73	92.83 0.59	94.57 0.47	96.32 0.32	▨	96.65 0.26	95.43 0.37	94.62 0.44	93.96 0.47	93.42 0.53	92.99 0.53	92.66 0.58	92.4 0.58
13	64.24 2.79	74.85 2.05	80.27 1.52	84.05 1.3	86.67 1.04	88.78 0.96	90.61 0.79	92.14 0.64	93.61 0.55	95.01 0.4	96.65 0.26	▨	96.89 0.22	95.75 0.35	94.96 0.41	94.32 0.46	93.83 0.49	93.46 0.5	93.16 0.5
14	64.08 2.79	74.59 2.07	80.03 1.55	83.73 1.29	86.31 1.09	88.33 0.98	90.09 0.76	91.51 0.64	92.9 0.58	94.17 0.43	95.43 0.37	96.89 0.22	▨	97.13 0.23	96.08 0.3	95.34 0.38	94.77 0.44	94.31 0.47	93.96 0.48
15	63.99 2.74	74.43 2.04	79.82 1.61	83.45 1.29	86.0 1.02	87.99 0.99	89.72 0.83	91.12 0.65	92.4 0.6	93.55 0.48	94.62 0.44	95.75 0.35	97.13 0.23	▨	97.32 0.2	96.33 0.28	95.66 0.34	95.13 0.4	94.71 0.39
16	63.89 2.82	74.23 2.09	79.63 1.66	83.21 1.37	85.69 1.06	87.64 1.05	89.34 0.89	90.69 0.77	91.91 0.68	93.0 0.57	93.96 0.47	94.96 0.41	96.08 0.3	97.32 0.2	▨	97.53 0.19	96.63 0.29	95.99 0.33	95.44 0.35
17	63.78 2.83	74.1 2.1	79.44 1.68	83.01 1.39	85.47 1.11	87.36 1.06	89.01 0.92	90.33 0.79	91.54 0.7	92.56 0.61	93.42 0.53	94.32 0.46	95.34 0.38	96.33 0.28	97.53 0.19	▨	97.65 0.21	96.8 0.28	96.16 0.32
18	63.73 2.86	73.96 2.07	79.32 1.7	82.86 1.41	85.26 1.08	87.13 1.09	88.72 0.94	90.02 0.84	91.19 0.75	92.16 0.67	92.99 0.53	93.83 0.49	94.77 0.44	95.66 0.34	96.63 0.29	97.65 0.21	▨	97.78 0.17	96.93 0.23
19	63.6 2.81	73.85 2.12	79.14 1.65	82.68 1.38	85.08 1.12	86.94 1.14	88.49 0.94	89.79 0.89	90.92 0.78	91.84 0.73	92.66 0.58	93.46 0.5	94.31 0.47	95.13 0.4	95.99 0.33	96.8 0.28	97.78 0.17	▨	97.88 0.17
20	63.52 2.85	73.74 2.11	79.02 1.69	82.54 1.38	84.92 1.15	86.76 1.15	88.28 0.96	89.58 0.88	90.71 0.77	91.59 0.74	92.4 0.58	93.16 0.5	93.96 0.48	94.71 0.39	95.44 0.35	96.16 0.32	96.93 0.23	97.88 0.17	▨

Table 7.1.: Correlation Integration test: no extreme values.

#	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	▨	73.77 2.08	49.44 1.86	52.6 2.03	55.17 1.98	57.13 2.01	56.19 2.18	57.73 2.19	59.12 2.18	60.26 2.16	59.83 2.07	60.79 2.09	61.65 2.07	62.38 2.07	61.78 2.05	62.49 1.95	63.07 1.98	63.66 1.98	63.5 2.2
3	73.77 2.08	▨	51.5 1.77	55.18 1.88	58.34 1.9	60.84 1.95	59.16 2.15	61.06 2.1	62.77 2.05	64.2 2.05	63.04 1.94	64.25 1.98	65.37 1.98	66.36 2.01	65.44 2.01	66.33 1.94	67.11 1.91	67.86 1.91	67.55 2.04
4	49.44 1.86	51.5 1.77	▨	93.66 0.45	90.12 0.66	87.34 0.76	80.36 1.28	80.16 1.23	79.83 1.2	79.41 1.14	78.53 1.24	78.39 1.25	78.19 1.22	77.97 1.21	77.73 1.44	77.59 1.42	77.41 1.41	77.22 1.38	77.33 1.36
5	52.6 2.03	55.18 1.88	93.66 0.45	▨	95.19 0.39	92.39 0.44	82.37 1.3	82.7 1.19	82.76 1.13	82.64 1.07	80.94 1.25	81.07 1.23	81.08 1.19	81.05 1.17	80.43 1.34	80.45 1.31	80.43 1.3	80.37 1.27	80.19 1.24
6	55.17 1.98	58.34 1.9	90.12 0.66	95.19 0.39	▨	96.04 0.31	83.06 1.33	83.8 1.24	84.23 1.2	84.42 1.1	82.25 1.24	82.58 1.18	82.77 1.1	82.91 1.06	82.0 1.25	82.16 1.23	82.26 1.2	82.32 1.17	81.9 1.22
7	57.13 2.01	60.84 1.95	87.34 0.76	92.39 0.44	96.04 0.31	▨	83.19 1.33	84.26 1.21	84.99 1.17	85.45 1.08	82.93 1.24	83.43 1.17	83.79 1.09	84.07 1.05	82.93 1.26	83.2 1.22	83.39 1.19	83.55 1.15	83.01 1.2
8	56.19 2.18	59.16 2.15	80.36 1.28	82.37 1.3	83.06 1.33	83.19 1.33	▨	97.73 0.17	96.21 0.26	94.9 0.33	89.9 0.73	89.76 0.73	89.55 0.73	89.31 0.73	88.01 0.88	87.91 0.86	87.78 0.85	87.63 0.82	87.17 0.88
9	57.73 2.19	61.06 2.1	80.16 1.23	82.7 1.19	83.8 1.24	84.26 1.21	97.73 0.17	▨	98.0 0.14	96.68 0.22	90.49 0.72	90.54 0.72	90.48 0.71	90.35 0.7	88.81 0.88	88.8 0.86	88.75 0.84	88.67 0.81	88.04 0.85
10	59.12 2.18	62.77 2.05	79.83 1.2	82.76 1.13	84.23 1.2	84.99 1.17	96.21 0.26	98.0 0.14	▨	98.21 0.16	90.77 0.73	90.98 0.73	91.06 0.7	91.06 0.68	89.34 0.87	89.42 0.85	89.44 0.83	89.43 0.8	88.67 0.84
11	60.26 2.16	64.2 2.05	79.41 1.14	82.64 1.07	84.42 1.1	85.45 1.08	94.9 0.33	96.68 0.22	98.21 0.16	▨	90.82 0.72	91.19 0.71	91.41 0.66	91.51 0.63	89.65 0.83	89.81 0.81	89.91 0.79	89.97 0.76	89.1 0.81
12	59.83 2.07	63.04 1.94	78.53 1.24	80.94 1.25	82.25 1.24	82.93 1.24	89.9 0.73	90.49 0.72	90.77 0.73	90.82 0.72	▨	98.72 0.11	97.84 0.17	97.06 0.22	93.32 0.5	93.21 0.5	93.05 0.49	92.88 0.49	91.68 0.64
13	60.79 2.09	64.25 1.98	78.39 1.25	81.07 1.23	82.58 1.18	83.43 1.17	89.76 0.73	90.54 0.72	90.98 0.73	91.19 0.71	98.72 0.11	▨	98.83 0.1	98.03 0.15	93.63 0.52	93.61 0.51	93.53 0.49	93.42 0.49	92.08 0.64
14	61.65 2.07	65.37 1.98	78.19 1.22	81.08 1.19	82.77 1.1	83.79 1.09	89.55 0.73	90.48 0.71	91.06 0.7	91.41 0.66	97.84 0.17	98.83 0.1	▨	98.92 0.08	93.79 0.53	93.87 0.51	93.87 0.49	93.83 0.49	92.36 0.64
15	62.38 2.07	66.36 2.01	77.97 1.21	81.05 1.17	82.91 1.06	84.07 1.05	89.31 0.73	90.35 0.7	91.06 0.68	91.51 0.63	97.06 0.22	98.03 0.15	98.92 0.08	▨	93.84 0.55	93.99 0.52	94.06 0.49	94.1 0.49	92.54 0.63
16	61.78 2.05	65.44 2.01	77.73 1.44	80.43 1.34	82.0 1.25	82.93 1.26	88.01 0.88	88.81 0.88	89.34 0.87	89.65 0.83	93.32 0.5	93.63 0.52	93.79 0.53	93.84 0.55	▨	99.14 0.06	98.56 0.11	98.02 0.13	94.99 0.41
17	62.49 1.95	66.33 1.94	77.59 1.42	80.45 1.31	82.16 1.23	83.2 1.22	87.91 0.86	88.8 0.86	89.42 0.85	89.81 0.81	93.21 0.5	93.61 0.51	93.87 0.51	93.99 0.52	99.14 0.06	▨	99.2 0.06	98.64 0.1	95.17 0.4
18	63.07 1.98	67.11 1.91	77.41 1.41	80.43 1.3	82.26 1.2	83.39 1.19	87.78 0.85	88.75 0.84	89.44 0.83	89.91 0.79	93.05 0.49	93.53 0.49	93.87 0.49	94.06 0.49	98.56 0.11	99.2 0.06	▨	99.23 0.06	95.28 0.39
19	63.66 1.98	67.86 1.91	77.22 1.38	80.37 1.27	82.32 1.17	83.55 1.15	87.63 0.82	88.67 0.81	89.43 0.8	89.97 0.76	92.88 0.49	93.42 0.49	93.83 0.49	94.1 0.49	98.02 0.13	98.64 0.1	99.23 0.06	▨	95.31 0.38
20	63.5 2.2	67.55 2.04	77.33 1.36	80.19 1.24	81.9 1.22	83.01 1.2	87.17 0.88	88.04 0.85	88.67 0.84	89.1 0.81	91.68 0.64	92.08 0.64	92.36 0.64	92.54 0.63	94.99 0.41	95.17 0.4	95.28 0.39	95.31 0.38	▨

Table 7.2.: Correlation Integration test: every fourth value deviating ten times the standard deviation

#	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	▨	73.79 2.07	38.63 0.73	40.17 0.79	41.38 0.9	42.39 0.95	43.61 0.99	44.33 1.02	44.96 1.08	45.51 1.08	45.82 1.1	46.27 1.12	46.67 1.14	47.06 1.16	47.37 1.12	47.7 1.13	48.02 1.13	48.31 1.17	48.0 1.17
3	73.79 2.07	▨	38.87 0.77	40.44 0.78	41.69 0.87	42.71 0.92	43.76 0.99	44.51 1.03	45.18 1.1	45.76 1.12	45.96 1.05	46.43 1.08	46.86 1.12	47.27 1.12	47.58 1.18	47.92 1.19	48.26 1.2	48.58 1.22	48.32 1.23
4	38.63 0.73	38.87 0.77	▨	94.61 0.17	90.68 0.2	87.54 0.24	78.6 1.37	78.45 1.29	78.15 1.24	77.74 1.2	76.87 1.44	76.74 1.4	76.55 1.37	76.32 1.34	76.53 1.44	76.42 1.41	76.28 1.39	76.12 1.37	76.14 1.42
5	40.17 0.79	40.44 0.78	94.61 0.17	▨	96.02 0.12	92.98 0.18	79.94 1.48	80.35 1.39	80.49 1.32	80.44 1.26	78.82 1.52	78.95 1.46	79.0 1.42	78.98 1.38	78.71 1.48	78.77 1.44	78.79 1.41	78.77 1.38	78.43 1.43
6	41.38 0.9	41.69 0.87	90.68 0.2	96.02 0.12	▨	96.87 0.12	80.15 1.6	81.0 1.49	81.52 1.41	81.81 1.34	79.69 1.62	80.05 1.55	80.28 1.49	80.43 1.44	79.84 1.54	80.03 1.5	80.18 1.46	80.28 1.43	79.7 1.49
7	42.39 0.95	42.71 0.92	87.54 0.24	92.98 0.18	96.87 0.12	▨	79.8 1.66	81.01 1.59	81.84 1.5	82.4 1.41	79.98 1.71	80.52 1.64	80.93 1.57	81.23 1.51	80.37 1.59	80.69 1.55	80.94 1.51	81.15 1.48	80.39 1.56
8	43.61 0.99	43.76 0.99	78.6 1.37	79.94 1.48	80.15 1.6	79.8 1.66	▨	98.12 0.1	96.53 0.17	95.16 0.23	89.46 0.98	89.38 0.95	89.2 0.92	88.95 0.89	87.68 1.02	87.59 1.0	87.46 0.98	87.31 0.96	86.71 0.99
9	44.33 1.02	44.51 1.03	78.45 1.29	80.35 1.39	81.0 1.49	81.01 1.59	98.12 0.1	▨	98.41 0.07	97.05 0.14	89.94 0.97	90.05 0.95	90.04 0.92	89.93 0.89	88.42 1.01	88.42 0.99	88.38 0.97	88.31 0.95	87.51 0.99
10	44.96 1.08	45.18 1.1	78.15 1.24	80.49 1.32	81.52 1.41	81.84 1.5	96.53 0.17	98.41 0.07	▨	98.63 0.07	90.09 0.97	90.39 0.94	90.53 0.91	90.57 0.89	88.84 1.0	88.96 0.98	89.01 0.96	89.01 0.95	88.05 0.99
11	45.51 1.08	45.76 1.12	77.74 1.2	80.44 1.26	81.81 1.34	82.4 1.41	95.16 0.23	97.05 0.14	98.63 0.07	▨	90.01 0.97	90.47 0.93	90.76 0.91	90.93 0.88	89.06 1.0	89.26 0.98	89.4 0.95	89.49 0.94	88.39 1.0
12	45.82 1.1	45.96 1.05	76.87 1.44	78.82 1.52	79.69 1.62	79.98 1.71	89.46 0.98	89.94 0.97	90.09 0.97	90.01 0.97	▨	98.99 0.07	98.1 0.13	97.3 0.18	93.24 0.58	93.17 0.57	93.04 0.57	92.87 0.57	91.66 0.64
13	46.27 1.12	46.43 1.08	76.74 1.4	78.95 1.46	80.05 1.55	80.52 1.64	89.38 0.95	90.05 0.95	90.39 0.94	90.47 0.93	98.99 0.07	▨	99.11 0.06	98.31 0.12	93.48 0.59	93.51 0.56	93.47 0.55	93.38 0.55	92.02 0.64
14	46.67 1.14	46.86 1.12	76.55 1.37	79.0 1.42	80.28 1.49	80.93 1.57	89.2 0.92	90.04 0.92	90.53 0.91	90.76 0.91	98.1 0.13	99.11 0.06	▨	99.2 0.05	93.57 0.59	93.7 0.57	93.75 0.55	93.73 0.54	92.25 0.64
15	47.06 1.16	47.27 1.12	76.32 1.34	78.98 1.38	80.43 1.44	81.23 1.51	88.95 0.89	89.93 0.89	90.57 0.89	90.93 0.88	97.3 0.18	98.31 0.12	99.2 0.05	▨	93.55 0.61	93.77 0.58	93.9 0.55	93.95 0.54	92.39 0.66
16	47.37 1.12	47.58 1.18	76.53 1.44	78.71 1.48	79.84 1.54	80.37 1.59	87.68 1.02	88.42 1.01	88.84 1.0	89.06 1.0	93.24 0.58	93.48 0.59	93.57 0.59	93.55 0.61	▨	99.35 0.05	98.76 0.09	98.22 0.13	94.97 0.4
17	47.7 1.13	47.92 1.19	76.42 1.41	78.77 1.44	80.03 1.5	80.69 1.55	87.59 1.0	88.42 0.99	88.96 0.98	89.26 0.98	93.17 0.57	93.51 0.56	93.7 0.57	93.77 0.58	99.35 0.05	▨	99.41 0.04	98.87 0.08	95.12 0.39
18	48.02 1.13	48.26 1.2	76.28 1.39	78.79 1.41	80.18 1.46	80.94 1.51	87.46 0.98	88.38 0.97	89.01 0.96	89.4 0.95	93.04 0.57	93.47 0.55	93.75 0.55	93.9 0.55	98.76 0.09	99.41 0.04	▨	99.46 0.04	95.18 0.39
19	48.31 1.17	48.58 1.22	76.12 1.37	78.77 1.38	80.28 1.43	81.15 1.48	87.31 0.96	88.31 0.95	89.01 0.95	89.49 0.94	92.87 0.57	93.38 0.55	93.73 0.54	93.95 0.54	98.22 0.13	98.87 0.08	99.46 0.04	▨	95.17 0.39
20	48.0 1.17	48.32 1.23	76.14 1.42	78.43 1.43	79.7 1.49	80.39 1.56	86.71 0.99	87.51 0.99	88.05 0.99	88.39 1.0	91.66 0.64	92.02 0.64	92.25 0.64	92.39 0.66	94.97 0.4	95.12 0.39	95.18 0.39	95.17 0.39	▨

Table 7.3.: Correlation Integration test: every fourth value deviating one hundred times the standard deviation

The result of the second test run aims to discover how fast a pattern recovers from the extreme value. After the extreme value is added, the newly created pattern has a much worse correlation to the preceding patterns, but the succeeding patterns stabilize immediately. The drop in correlation also becomes smaller the more values are present in the data analysis phase, which causes the twentieth pattern to only drop to 95% correlation to its predecessor.

The last test run, which is displayed in Table 7.3 uses a 100 times higher standard deviation for the creation of the extreme values and aims to provide insight into how much extreme differences between the data values impact the patterns in the short and long run. For the short-term influence, the correlation between the neighbouring patterns can be examined to get an indicator of how much the extreme value is influencing the pattern, similar to how the second test was evaluated. The long-term influence can be measured by comparing the results of the third test with the results of the previous two tests.

In the short term, the pattern drop in correlation significantly once a new extreme data value is added, but like in the previous test, the patterns adapt quickly to the new value. The last extreme value causes a drop about the same size as the drop in the previous test, even though the initial drop after the first extreme value is much more severe than in the previous test. The long-term effect is visible in the result table, indicating less significance after enough data values are added. The biggest long-term impact caused by the extreme values seems to be that the early patterns have a bigger difference to the more stabilized block of patterns, which is a phenomenon only observable in the last test.

8. Conclusion

In this work, a context-aware approach is taken to analyze the measurable behaviour in a smart home environment, resulting in the creation of behaviour patterns. These patterns serve as prediction models for further events in the environment and are integrated into the access control model of the smart home, allowing its ACM to base the access control decisions on the knowledge of expected behaviour. This chapter summarizes the key contributions presented in this thesis and revisits the research questions from Chapter 1.3 to discuss how the goals of this work have been achieved.

8.1. Key Contributions

This thesis starts with an in-depth context analysis of the underlying smart home environment, resulting in a comprehensive context model on which the design is built. The creation of the context model is a crucial step to gain insight into what can be used to extract information about behaviour in the smart home [4].

The context model features two event types received from the smart home, namely user inputs and smart sensor measurements, and a respective behaviour pattern type has been introduced for both event types. These patterns serve as prediction models for the observable behaviour in the smart home and therefore patterns are designed to be capable of evaluating smart home events to indicate whether an event matches expected behaviour or is suspicious.

Control event patterns and periodic event patterns are introduced as prediction models for user behaviour and system behaviour respectively. Both pattern types are created with different algorithms and evaluating smart home events works differently for both pattern types. The context model also defines relationships between control event patterns to preserve the knowledge about semantic relationships between users and between devices.

Integrating the knowledge of behaviour into the ABAC model of the smart home implicates that evaluating a smart home event needs to be performable at runtime. Therefore, behaviour patterns are designed to quickly evaluate an event, are created in a periodic analysis process, and persistently stored so that the knowledge is quickly retrievable. This decouples the behaviour analysis from the ACM and allows them to be used in access control despite their time-intensive creation, which fulfills this important requirement.

The behaviour engine is introduced as a smart home component, which is responsible for data gathering, analysis, and dissemination of behaviour patterns and all related features. The behaviour engine is integrated into COSYLab, a smart home application with an ABAC model, and provides the knowledge of behaviour to the ACM of COSYLab by associating decision results with attributes during its decision-making phase.

One important contribution is CosyHome [25], a big dataset generated in a case study as part of this work. The case study includes eight participants in four smart homes and runs over the course of nine weeks. This dataset is necessary for the evaluation of this thesis to provide insight into how the behaviour engine performs in a real-world setting, allowing for better comparability.

Different methods of testing [91], which focus on different aspects and come with their respective advantages and disadvantages, are utilized to give the most accurate picture of the results of this thesis. A total of four evaluations are performed against the behaviour engine for proper examination with two of the evaluations being based on CosyHome and the other two being based on synthetic data. The results of the evaluations prove the reliability of behaviour patterns once sufficient data is available, the meaningfulness of the underlying context model, and that integrating the knowledge of behaviour into the access control model is successful.

8.2. Research Questions Revisited

The three research questions presented in Chapter 1.3 are processed throughout this work. Chapter 5 sets requirements for the two design-related research questions to be satisfied and discusses them in detail while the third research question is processed in Chapter 7. This section gives a short overview of how each research question is dealt with and what results prove that the respective question is answered.

1. How can the daily routine and the habits of an inhabitant of a smart home be translated into a context-aware behaviour pattern?

The first requirement to create a context-aware application is a thorough context analysis to examine the existing context in the environment. Therefore, this work presents a context model for a smart home environment based on an analysis of the environment and uses this context model as the basis to translate the behaviour in the system into behaviour patterns.

Two types of behaviour patterns are introduced to represent the user behaviour and the system behaviour separately. Both types of patterns are designed to be prediction models so that they can be used to evaluate whether newly created smart home events match the expected behaviour. The context model further includes relationships between patterns and external validations by users, which allow the behaviour to be predicted more precisely.

The design of behaviour patterns and all other behaviour-related features is implemented in the behaviour engine. The correctness of the implementation is validated in the lower levels of testing [91][60], for which quickly and cheaply generated artificial data is used [38]. Furthermore, the big dataset CosyHome is used in several evaluations, proving that the presented concepts work as intended.

2. How can context-aware behaviour pattern recognition be used in a smart home environment for access control?

The access control in the underlying smart home environment is based on attributes and therefore, three attributes are defined to expose the behaviour evaluations to the ABAC [113] of COSYLab. The three attributes are responsible for evaluating a single control event pattern, a group of control event patterns, and a single periodic event pattern.

One crucial requirement implicitly set by this research question is that the evaluation process is usable during an access control request, which is why behaviour patterns are prepared regularly so that they only need to be retrieved during access control. This is an important step to overcome the problem of behaviour being only estimable over time [10] and access requests needing to be handled at a certain point in time. In Chapter 7.2.1, a performance analysis is carried out to prove that the behaviour evaluations are usable during access control and the results indicate that behaviour is successfully integrated.

The users are in charge of configuring their access policies and the groups, which means that they might be overwhelmed by the management and maintenance of their system. This thesis introduces a visual representation of patterns and pattern correlations as indicators for when a pattern needs to be exchanged to aid the users in the pattern and access policy management. Furthermore, group suggestions are created to indicate possible relationships in the environment during group management.

3. How long is the learning time for a behaviour pattern?

A real-world dataset is required to provide a meaningful indicator of how well the presented concept and implementation would perform in a non-simulated setting. CosyHome is a dataset created in this work to evaluate the behaviour engine because no suiting dataset was available.

The learning time of control event patterns is determined based on CosyHome in an evaluation that is configured to mimic how the observed actions would be interpreted by the behaviour engine. Overall, the results were very satisfying and indicate that the behaviour patterns become reliable after being based on five weeks of data.

To determine the learning time of periodic event patterns, synthetic data is used to evaluate how periodic event patterns adapt in different scenarios. The results are based on how much the patterns change over the course of twenty simulated weeks, using pattern correlations as indicators, and generally show that even extreme values are adapted towards quickly. Periodic event patterns do not change much after five weeks of data is present, which is why the minimum learning time suggested in this thesis is five weeks for both pattern types.

8.3. Future Work

The huge scope of this thesis allows for many future work projects, from extending the context model to improving the evaluation process. Any part of this work can be isolated and used as the basis for another thesis, which focuses on improving or extending it. In this section, a few examples are presented as suggestions.

Reasoning Methods

The mathematical model used in the analysis process of periodic event patterns, which represent the numerical values measured by smart sensors, uses a calculation of an average and a standard deviation for further events to be compared. Further statistical approximations can potentially enhance the resulting behaviour patterns.

Another way in which periodic event patterns could be improved is by replacing the static nature of their time slots. In the final version presented in this work, the time slots are evenly split throughout the day, which could limit their meaningfulness by not properly representing every smart sensor. When the measurements of a smart sensor usually remain relatively constant throughout a long time span in the day, splitting those time spans up into smaller time slots is inefficient. Furthermore, if the split is unlucky between the time slots, there could be an unnecessarily high standard deviation, which could be better represented by rearranged time slots.

Control event patterns only include the time of the interaction and not the frequency. Extending control event patterns this way would change the control event usage peaks from simply having one dimension, the time of the usages with an average and a standard deviation, to having a second dimension for comparison, which indicates how often the smart home resident interacts at a certain time with a device.

An example for the use of frequency in control event patterns could be how often the coffee machine is used by the inhabitant, which in the state of this work would be treated as separate events. The resulting events are very close to each other on the time axis but have no further semantic information or relationship to each other. If the frequency is also tracked, the behaviour engine could furthermore notice additional forms of deviating behaviour such as devices being used unexpectedly often, which could be a sign for a malicious outsider.

Non-Numerical Values in Periodic Event Pattern

Smart sensors periodically publish events including numerical values, which are represented in the solution design by periodic events. Other smart devices also periodically send events to the system such as updates about their states, which are events that are not handled by the design. Extending periodic event patterns to include non-numerical values is a possible future task to extend the behaviour engine.

Extending Group Feature

Groups currently only work for the user and device context, expressed by user and device groups respectively. Behaviour patterns also include time context, which is currently not supported by the group feature and while time context is defined by interesting metadata, the day of the week, there is no group allowing for the decision-making process to include the patterns of a different weekday.

The time context can also be used as the basis for relationships between patterns such as introducing relationships between weekdays and weekend days respectively. Groups supporting time context are not included in this work, because the dataset in the evaluation only features one working day and one weekend day, which show lower similarities and the result would not be meaningful enough.

Bigger Dataset

The most difficult future work can only be achieved by generating a bigger dataset and is about the maintenance of the solution design and its scalability. It is very important to not fall for the misconception that the deployment of a context-aware system is the end of its life cycle and the design and implementation need to strife towards being suitable in the long run [4].

The measures taken to reduce the maintenance effort cannot be asserted as sufficient, since the system is not deployed with a significant enough amount of data and is not running long enough so that a proper estimation can be made of whether the solution design scales well for real-world smart home environments over a long time or if the implementation is sufficiently maintainable. The bigger dataset for a smart home environment in itself could also be seen as an appropriate future work.

A. CosyHome Dataset Details

In the scope of this work, a dataset had to be created in a real-world setting for the evaluation to be more meaningful since the behaviour engine and therefore the concept of this work need to be compared to a real-world scenario. Before the decision of conducting a case study with a resulting dataset, already existing datasets were examined for the possibility of obtaining fitting data without having to gather them, but unfortunately, all examined datasets were either insufficient or unobtainable.

This section provides an overview of the related datasets and a more in-depth view of CosyHome [25], the dataset created in the case study, that ran over the course of nine weeks in the scope of this work. It is necessary to point out that only the most basic relevant information of the people who participated as subjects in the case study are given due to privacy reasons, which means that no information is provided that can identify any subject participating in the study.

The case study started on 2020-05-17 and ended on 2020-07-13, which means that the dataset was gathered approximately at the end of the first 2020 lockdowns in Austria and ended with the first weeks of summer. The time span of the case study introduces mainly two points of expected higher irregularity in the interactions with the devices in the home.

Note that some days do not have a single entry for a subject, indicating that not a single interaction with any device in the home was performed during a whole day by a subject. This is not an error or a missing data entry but indicates a participant being absent from home for the whole day, i.e. because of being on holiday.

A.1. Related Datasets

Instead of creating new data, it is recommended to try finding an existing dataset, which matches at least the required criteria since finding a fitting dataset saves the time and resources necessary to gather the data. Furthermore, basing the evaluation on existing data allows for more comparability, including to previously published work. Even though no dataset fulfills the criteria of having multiple users being monitored over at least eight weeks, there are still some datasets that stood out as being potentially useful in the evaluation of this work.

The most interesting dataset, Orange4Home [26], monitors one person over the course of four consecutive weeks during the working hours in one apartment. For the purposes of the dataset, this apartment is an environment with deployed smart sensors to simulate a smart home and the test subject was put into the environment to work there for the duration of the experiment. The reason the dataset cannot be considered to monitor the user behaviour in a real-world scenario set in a smart home is due to the person not usually interacting with this environment. The second problem comes from only the working hours being monitored and the third problem is the relatively short time span the experiment takes place in. The upside of this dataset is, that it provides everything from the timestamps to the multiple devices in the home and the interaction with one person, which suffices to generate the behaviour patterns.

Another dataset created in a sensor-rich environment is called Opportunity [82] and features twelve subjects with 25 hours of collected data. Furthermore, the data was gathered in only one room with body sensors worn by the testing subjects and environment sensors deployed in the room. The data being generated over a shorter than required period of time and the lack of timestamps disqualify this dataset for being used in the evaluation. Otherwise, the upsides of this dataset would have been the number of test subjects and the measured complex task in a smart environment.

The last interesting dataset was created to evaluate a method for analyzing daily routines by equipping one test subject for sixteen days with two wearable sensors and monitoring the daily life of the subject [46]. The resulting dataset shows which complex activities have been performed at which time of day for the method to break these complex activities down into simpler parts. Similar to Orange4Home, there is a lack of multiple test users and the criteria of sufficient learning time are not met in this dataset. One advantage this dataset has over Orange4Home is that the sensors were measuring the behaviour in the environments the subject would usually interact with and the data is not only set in the working hours. Furthermore, the wearable sensors only gather information about the motion of the test subject, there is no indication of which devices the subject is interacting with within the home.

A.2. Subjects

The first home, called H1 in the dataset, is inhabited by a dual-income family with two children, with the parents being in their late 30s and early 40s, while the children are 13 and 17 years old. The parents are listed as P1 and P2 in the dataset, the older child is listed as P3 and the younger child is listed as P4. Both children are going to school, but only the younger child has a fixed schedule given by the school during the time the data is gathered since the older child is in the final school year and does not have to attend regular classes anymore. P1 is working full time as a technician, which requires a part of office work and a part of on-site work, which leads to some working days in the dataset having interactions over the day because this work was done in home office. P2 is working at a grocery store, which requires their presence, and therefore the person does not have any tracked interactions in the dataset during the working hours on working days.

The second home, which is listed as H2 in the dataset, has two inhabitants in their mid to late 50s. The person listed as P1 is working an office job sporadically working from home on the Mondays in the dataset. The other person works as a social worker with a fixed timetable, which changes every month and requires sometimes breaks during working hours, in which the person goes home and possibly interacts with the devices in the home. Note that in this home one exceptionally long interaction between P1 and the TV overnight is tracked, which has to be omitted as an error because the person forgot to turn the device off overnight, which leads to the person not having any interactions with the device in that night.

In H3, the third home that is part of the case study, only one person lives and this person is in their late twenties. The participant of the study is working an office job, but due to the dataset being gathered around a time where home office was encouraged, a lot of devices have many interactions during the weekdays.

The last home, H4, has also only one resident who is in the mid-twenties and works an office job. During the case study, the first weeks the participant worked from home and then switched to going to the office until the end of the data gathering.

A.3. Objects

Besides the subjects, which are representing the people living in the smart homes in the dataset, the objects are representing the smart devices with which the inhabitants interact. These devices are controlled with a remote, usually, the smartphone of the inhabitants, and the information the dataset is providing is expressed by the tracked interactions between subjects and objects. In the scope of this work, the objects of each smart home have been categorized and put into groups, but these groups do not explicitly appear in the dataset.

In the first home, named H1 in the dataset and housing the dual-income family, there are thirteen objects for the subjects to interact with. The smart devices, that are part of the kitchen devices group, are the barbecue, coffee machine, oven, refrigerator, stove, and toaster. The other device group of the home includes the devices, categorized as free time events, which are the respective notebooks of the children, the tablet, TV, and radio. The washing machine and the dishwasher, are not considered part of any group.

The home with the most objects is the second home, which is named H2 in the dataset and has a total of fourteen smart devices. The kitchen group in this home features the coffee machine, microwave, oven, refrigerator, stove, toaster, and water boiler, while the free time device group encompasses the radio and the two TVs. The devices not considered as parts of any group are the washing machine, dishwasher, printer, and notebook.

The least number of objects are featured in the third home, which is recorded as H3 in CosyHome. In total, there are only seven smart devices in this home, with five of them being grouped as kitchen devices, which are the microwave, oven, refrigerator, stove, and water boiler. The dishwasher and the computer, are not part of any group, which makes the third home the only home without a group of free time devices.

The fourth home, referred to as H4 in the dataset, has only one more object than the third home but has enough free time devices to create a group for these devices. The kitchen devices in this smart home are the oven, the refrigerator, the stove, the toaster, and the water boiler, while the notebook and the TV are grouped as the free time devices.

The dishwasher is not part of any group, which makes it the only device in the home not being related to any other object.

B. Acronyms and Abbreviations

ABAC	Attribute Based Access Control
ACL	Access Control List
ACM	Access Control Mechanism
EMS	Energy Management System
FACA	Fog Access Control Agent
IBAC	Identity Based Access Control
IoT	Internet of Things
LBAC	Lattice Based Access Control
PAP	Policy Administration Point
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PIP	Policy Information Point
QoE	Quality of Experience
QoS	Quality of Service
RBAC	Role Based Access Control
WoT	Web of Things
WWW	World Wide Web
XACML	Extensible Access Control Markup Language

Bibliography

[1] Gregory D Abowd et al. ‘Towards a better understanding of context and context-awareness’. In: *International symposium on handheld and ubiquitous computing*. Springer. 1999, pp. 304–307.

[2] Jose Aguilar, Marxjhony Jerez and Taniana Rodriguez. ‘CAMeOnto: Context awareness meta ontology modeling’. In: *Applied computing and informatics* 14.2 (2018), pp. 202–213.

[3] Tanweer Alam. ‘A reliable communication framework and its use in internet of things (IoT)’. In: *International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT)* 3.5 (2018), pp. 450–456.

[4] Unai Alegre, Juan Carlos Augusto and Tony Clark. ‘Engineering context-aware systems and applications: A survey’. In: *Journal of Systems and Software* 117 (2016), pp. 55–83.

[5] Abdul-Rahman Al-Ali et al. ‘A smart home energy management system using IoT and big data analytics approach’. In: *IEEE Transactions on Consumer Electronics* 63.4 (2017), pp. 426–434.

[6] Luigi Atzori, Antonio Iera and Giacomo Morabito. ‘The internet of things: A survey’. In: *Computer networks* 54.15 (2010), pp. 2787–2805.

[7] Lars Backstrom et al. ‘Group formation in large social networks: membership, growth, and evolution’. In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2006, pp. 44–54.

[8] Matthias Baldauf, Schahram Dustdar and Florian Rosenberg. ‘A survey on context-aware systems’. In: *International Journal of Ad Hoc and Ubiquitous Computing* 2.4 (2007), pp. 263–277.

[9] Horace B Barlow. ‘Unsupervised learning’. In: *Neural computation* 1.3 (1989), pp. 295–311.

[10] Alison M Bell, Shala J Hankison and Kate L Laskowski. ‘The repeatability of behaviour: a meta-analysis’. In: *Animal behaviour* 77.4 (2009), pp. 771–783.

[11] D Elliott Bell and Leonard J LaPadula. *Secure computer systems: Mathematical foundations*. Tech. rep. MITRE CORP BEDFORD MA, 1973.

[12] Victoria Bellotti and Keith Edwards. ‘Intelligibility and accountability: human considerations in context-aware systems’. In: *Human–Computer Interaction* 16.2-4 (2001), pp. 193–212.

[13] Kamal Benzekki, Abdeslam El Fergougui and Abdelbaki ElBelrhiti ElAlaoui. 'A context-aware authentication system for mobile cloud computing'. In: *Procedia Computer Science* 127 (2018), pp. 379–387.

[14] Matt Blaze et al. 'The role of trust management in distributed systems security'. In: *Secure Internet Programming.* Springer, 1999, pp. 185–210.

[15] Alexandru Boicea, Florin Radulescu and Laura Ioana Agapin. 'MongoDB vs Oracle–database comparison'. In: *2012 third international conference on emerging intelligent data and web technologies.* IEEE. 2012, pp. 330–335.

[16] Mariusz Bojarski et al. 'End to end learning for self-driving cars'. In: *arXiv preprint arXiv:1604.07316* (2016).

[17] Lionel C Briand, Yvan Labiche and Yihong Wang. 'An investigation of graph-based class integration test order strategies'. In: *IEEE Transactions on Software Engineering* 29.7 (2003), pp. 594–607.

[18] Kjell Brunnström et al. 'Qualinet white paper on definitions of quality of experience'. In: (2013).

[19] Hans Ulrich Buhl et al. *Big data.* 2013.

[20] Michael Burrows, Martin Abadi and Roger Michael Needham. 'A logic of authentication'. In: *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences* 426.1871 (1989), pp. 233–271.

[21] Jorge Cardoso et al. 'Quality of service for workflows and web service processes'. In: *Journal of web semantics* 1.3 (2004), pp. 281–308.

[22] Marie Chan et al. 'A review of smart homes—Present state and future challenges'. In: *Computer methods and programs in biomedicine* 91.1 (2008), pp. 55–81.

[23] Mingte Chen et al. *Asynchronous message push to web browser.* US Patent App. 10/033,146. Nov. 2003.

[24] *commons-math3.* Version 3.6.1. 17th Mar. 2016. URL: `https://mvnrepository.com/artifact/org.apache.commons/commons-math3`.

[25] *CosyHome Dataset on Git.* `https://gitlab.cs.univie.ac.at/cosylab/cosylab-fog/fog_behaviourengine/-/tree/develop/src/main/resources/cosyHomeDataset`. Accessed: 2021-10-30.

[26] Julien Cumin et al. 'A dataset of routine daily activities in an instrumented home'. In: *International Conference on Ubiquitous Computing and Ambient Intelligence.* Springer. 2017, pp. 413–425.

[27] Scott Davidoff et al. 'Principles of smart home control'. In: *International conference on ubiquitous computing.* Springer. 2006, pp. 19–34.

[28] Jesse Davis and Mark Goadrich. 'The relationship between Precision-Recall and ROC curves'. In: *Proceedings of the 23rd international conference on Machine learning.* 2006, pp. 233–240.

[29] Dorothy E Denning. ‘A lattice model of secure information flow’. In: *Communications of the ACM* 19.5 (1976), pp. 236–243.

[30] Paul Dourish. ‘What we talk about when we talk about context’. In: *Personal and ubiquitous computing* 8.1 (2004), pp. 19–30.

[31] Janick Edinger et al. ‘Fault-avoidance strategies for context-aware schedulers in pervasive computing systems’. In: *2017 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE. 2017, pp. 79–88.

[32] Sélinde van Engelenburg, Marijn Janssen and Bram Klievink. ‘Designing context-aware systems: A method for understanding and analysing context in practice’. In: *Journal of logical and algebraic methods in programming* 103 (2019), pp. 79–104.

[33] Patrick Th Eugster et al. ‘The many faces of publish/subscribe’. In: *ACM computing surveys (CSUR)* 35.2 (2003), pp. 114–131.

[34] Joel L Fernandes et al. ‘Performance evaluation of RESTful web services and AMQP protocol’. In: *2013 fifth international conference on ubiquitous and future networks (ICUFN)*. IEEE. 2013, pp. 810–815.

[35] Fielding Hudson Garrison. ‘The history of heating, ventilation and lighting’. In: *Bulletin of the New York Academy of Medicine* 3.2 (1927), p. 56.

[36] Simon Godik and Tim Moses. ‘Oasis extensible access control markup language (xacml)’. In: *OASIS Committee Secification cs-xacml-specification-1.0* (2002).

[37] Dominique Guinard and Vlad Trifa. ‘Towards the web of things: Web mashups for embedded devices’. In: *Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009), in proceedings of WWW (International World Wide Web Conferences), Madrid, Spain*. Vol. 15. 2009, p. 8.

[38] Ankush Gupta, Andrea Vedaldi and Andrew Zisserman. ‘Synthetic data for text localisation in natural images’. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2315–2324.

[39] *hamcrest*. Version 2.1. 20th Dec. 2018. URL: `https://mvnrepository.com/artifact/org.hamcrest/hamcrest`.

[40] *hamcrest-optional*. Version 2.0.0. 9th June 2017. URL: `https://mvnrepository.com/artifact/com.github.npathai/hamcrest-optional`.

[41] Richard Harper. *Inside the smart home*. Springer Science & Business Media, 2006.

[42] Trevor Hastie, Robert Tibshirani and Jerome Friedman. ‘Overview of supervised learning’. In: *The elements of statistical learning*. Springer, 2009, pp. 9–41.

[43] Abdeltawab Hendawi et al. ‘Benchmarking large-scale data management for Internet of Things’. In: *The Journal of Supercomputing* 75.12 (2019), pp. 8207–8230.

[44] George Hsu. *Modular RF communication module for automated home and vehicle systems*. US Patent 6,374,079. Apr. 2002.

[45] Vincent C Hu et al. 'Guide to attribute based access control (abac) definition and considerations (draft)'. In: *NIST special publication* 800.162 (2013).

[46] Tâm Huynh, Mario Fritz and Bernt Schiele. 'Discovery of activity patterns using topic models'. In: *Proceedings of the 10th international conference on Ubiquitous computing*. 2008, pp. 10–19.

[47] Kim Hyun-Wook et al. 'Development of middleware architecture to realize context-aware service in smart home environment'. In: *Computer Science and Information Systems* 13.2 (2016), pp. 427–452.

[48] Xin Jin, Ram Krishnan and Ravi Sandhu. 'A unified attribute-based access control model covering DAC, MAC and RBAC'. In: *IFIP Annual Conference on Data and Applications Security and Privacy*. Springer. 2012, pp. 41–55.

[49] Satu Jumisko-Pyykkö and Teija Vainio. 'Framing the context of use for mobile HCI'. In: *International journal of mobile human computer interaction (IJMHCI)* 2.4 (2010), pp. 1–28.

[50] Won Min Kang, Seo Yeon Moon and Jong Hyuk Park. 'An enhanced security framework for home appliances in smart home'. In: *Human-centric Computing and Information Sciences* 7.1 (2017), pp. 1–12.

[51] Joachim Karlsson and Kevin Ryan. 'A cost-value approach for prioritizing requirements'. In: *IEEE software* 14.5 (1997), pp. 67–74.

[52] Stas Khirman and Peter Henriksen. 'Relationship between quality-of-service and quality-of-experience for public internet service'. In: *In Proc. of the 3rd Workshop on Passive and Active Measurement*. Vol. 1. 2002.

[53] Wei-Chi Ku and Shen-Tien Chang. 'Impersonation attack on a dynamic ID-based remote user authentication scheme using smart cards'. In: *IEICE Transactions on Communications* 88.5 (2005), pp. 2165–2167.

[54] Kevin Kunzelman and Sterling Hutto. *Common session token system and protocol*. US Patent 6,041,357. Mar. 2000.

[55] Charles A Kunzinger. *Integrated system for network layer security and fine-grained identity-based access control*. US Patent 6,986,061. Jan. 2006.

[56] Andres Laya, Vlad-loan Bratu and Jan Markendahl. 'Who is investing in machine-to-machine communications?' In: (2013).

[57] Ying-Tsung Lee et al. 'An integrated cloud-based smart home management system with community hierarchy'. In: *IEEE Transactions on Consumer Electronics* 62.1 (2016), pp. 1–9.

[58] Wu-Hon Francis Leung. *Methods and apparatus for preventing software modifications from invalidating previously passed integration tests*. US Patent 6.769.114. July 2004.

[59] Bojun Li, Piyanuch Hathaipontaluk and Suhuai Luo. 'Intelligent oven in smart home environment'. In: *2009 international conference on research challenges in computer science*. IEEE. 2009, pp. 247–250.

[60] Lu Luo. 'Software testing techniques'. In: *Institute for software research international Carnegie mellon university Pittsburgh, PA* 15232.1-19 (2001), p. 19.

[61] Meena Mahajan, Prajakta Nimbhorkar and Kasturi Varadarajan. 'The planar k-means problem is NP-hard'. In: *Theoretical Computer Science* 442 (2012), pp. 13–21.

[62] John McHugh and Carrie Gates. 'Locality: A new paradigm for thinking about normal behavior and outsider threat'. In: *Proceedings of the 2003 workshop on New security paradigms*. 2003, pp. 3–10.

[63] Saul McLeod. 'Maslow's hierarchy of needs'. In: *Simply psychology* 1 (2007), pp. 1–8.

[64] Alexey Medvedev et al. 'Waste management as an IoT-enabled service in smart cities'. In: *Internet of Things, Smart Spaces, and Next Generation Networks and Systems*. Springer, 2015, pp. 104–115.

[65] *mockito-core*. Version 2.28.2. 29th May 2019. URL: `https://mvnrepository.com/artifact/org.mockito/mockito-core`.

[66] Michael Mozer et al. 'The adaptive house'. In: *IEE Seminar Digests*. Vol. 11059. IET. 2005, pp. v1–39.

[67] Zhaolong Ning et al. 'Vehicular social networks: Enabling smart mobility'. In: *IEEE Communications Magazine* 55.5 (2017), pp. 16–55.

[68] Christena E Nippert-Eng. *Home and work: Negotiating boundaries through everyday life*. University of Chicago Press, 2008.

[69] Bettina Nissen et al. 'Geocoin: Supporting ideation and collaborative design with smart contracts'. In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 2018, pp. 1–10.

[70] Sylvia Osborn, Ravi Sandhu and Qamar Munawer. 'Configuring role-based access control to enforce mandatory and discretionary access control policies'. In: *ACM Transactions on Information and System Security (TISSEC)* 3.2 (2000), pp. 85–106.

[71] Andrei Palade et al. 'Middleware for Internet of Things: A quantitative evaluation in small scale'. In: *2017 IEEE 18th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE. 2017, pp. 1–6.

[72] Vamsikrishna Patchava, Hari Babu Kandala and P Ravi Babu. 'A smart home automation technique with raspberry pi using iot'. In: *2015 International conference on smart sensors and systems (IC-SSS)*. IEEE. 2015, pp. 1–4.

[73] Charith Perera et al. 'Context aware computing for the internet of things: A survey'. In: *IEEE communications surveys & tutorials* 16.1 (2013), pp. 414–454.

[74] Charith Perera et al. 'Sensing as a service model for smart cities supported by internet of things'. In: *Transactions on emerging telecommunications technologies* 25.1 (2014), pp. 81–93.

[75] Stan Pounds and Stephan W Morris. ‘Estimating the occurrence of false positives and false negatives in microarray studies by approximating and partitioning the empirical distribution of p-values’. In: *Bioinformatics* 19.10 (2003), pp. 1236–1242.

[76] J. Ross Quinlan. ‘Induction of decision trees’. In: *Machine learning* 1.1 (1986), pp. 81–106.

[77] Rasmus V Rasmussen and Michael A Trick. ‘Round robin scheduling–a survey’. In: *European Journal of Operational Research* 188.3 (2008), pp. 617–636.

[78] *Raspberry Pi 3 Model B Rev 1.2*. `https://www.raspberrypi.org/products/raspberry-pi-3-model-b/`. Accessed: 2021-09-26.

[79] Leonard Richardson and Sam Ruby. *RESTful web services*. " O’Reilly Media, Inc.", 2008.

[80] Vincent Ricquebourg et al. ‘The smart home concept: our immediate future’. In: *2006 1st IEEE international conference on e-learning in industrial electronics*. IEEE. 2006, pp. 23–28.

[81] Rosslin John Robles et al. ‘A review on security in smart home development’. In: *International Journal of Advanced Science and Technology* 15 (2010).

[82] Daniel Roggen et al. ‘Collecting complex activity datasets in highly rich networked sensor environments’. In: *2010 Seventh international conference on networked sensing systems (INSS)*. IEEE. 2010, pp. 233–240.

[83] Patrick Rosenberger and Detlef Gerhard. ‘Context-awareness in industrial applications: definition, classification and use case’. In: *Procedia CIRP* 72 (2018), pp. 1172–1177.

[84] Per Runeson. ‘A survey of unit testing practices’. In: *IEEE software* 23.4 (2006), pp. 22–29.

[85] Ameena Saad al-sumaiti, Mohammed Hassan Ahmed and Magdy MA Salama. ‘Smart home activities: A literature review’. In: *Electric Power Components and Systems* 42.3-4 (2014), pp. 294–305.

[86] Ravi Sandhu. ‘Role hierarchies and constraints for lattice-based access controls’. In: *European Symposium on Research in Computer Security*. Springer. 1996, pp. 65–79.

[87] Ravi S. Sandhu. ‘Lattice-based access control models’. In: *Computer* 26.11 (1993), pp. 9–19.

[88] Ravi S Sandhu and Pierangela Samarati. ‘Access control: principle and practice’. In: *IEEE communications magazine* 32.9 (1994), pp. 40–48.

[89] Ravi S Sandhu et al. ‘Role-based access control models’. In: *Computer* 29.2 (1996), pp. 38–47.

[90] RS Sandhu et al. ‘Access Control Models’. In: *IEEE computer* 29.2 (2013), pp. 38–47.

[91] Abhijit A Sawant, Pranit H Bari and PM Chawan. 'Software testing techniques and strategies'. In: *International Journal of Engineering Research and Applications (IJERA)* 2.3 (2012), pp. 980–986.

[92] Kamlesh Sharma and T Suryakanthi. 'Smart system: IoT for university'. In: *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*. IEEE. 2015, pp. 1586–1593.

[93] Stefan Soucek, Gerhard Russ and Clara Tamarit. *The smart kitchen project-an application of fieldbus technology to domotics*. Citeseer, 2000.

[94] *spring-amqp*. Version 2.2.11.RELEASE. 16th Sept. 2020. URL: `https://mvnrepository.com/artifact/org.springframework.amqp/spring-amqp`.

[95] *spring-data-mongodb*. Version 3.0.4.RELEASE. 16th Sept. 2020. URL: `https://mvnrepository.com/artifact/org.springframework.data/spring-data-mongodb`.

[96] *spring-rabbit*. Version 2.2.11.RELEASE. 16th Sept. 2020. URL: `https://mvnrepository.com/artifact/org.springframework.amqp/spring-rabbit`.

[97] Hema Srikanth, Laurie Williams and Jason Osborne. 'System test case prioritization of new and regression test cases'. In: *2005 International Symposium on Empirical Software Engineering, 2005.* IEEE. 2005, 10–pp.

[98] Biljana L Risteska Stojkoska and Kire V Trivodaliev. 'A review of Internet of Things for smart home: Challenges and solutions'. In: *Journal of Cleaner Production* 140 (2017), pp. 1454–1464.

[99] Ting Su and Jennifer Dy. 'A deterministic method for initializing k-means clustering'. In: *16th IEEE International Conference on Tools with Artificial Intelligence*. IEEE. 2004, pp. 784–786.

[100] Wencheng Sun et al. 'Security and privacy in the medical internet of things: a review'. In: *Security and Communication Networks* 2018 (2018).

[101] Harald Sundmaeker et al. 'Vision and challenges for realising the Internet of Things'. In: *Cluster of European Research Projects on the Internet of Things, European Commision* 3.3 (2010), pp. 34–36.

[102] Lu Tan and Neng Wang. 'Future internet: The internet of things'. In: *2010 3rd international conference on advanced computer theory and engineering (ICACTE)*. Vol. 5. IEEE. 2010, pp. V5–376.

[103] Francois Trans. *Means and method for a synchronous network communications system*. US Patent 6,377,640. Apr. 2002.

[104] Ciprian-Octavian Truica et al. 'Performance evaluation for CRUD operations in asynchronously replicated document oriented database'. In: *2015 20th International Conference on Control Systems and Computer Science*. IEEE. 2015, pp. 191–196.

[105] Hamed Vahdat-Nejad, Kamran Zamanifar and Nasser Nematbakhsh. ‘Context-aware middleware architecture for smart home environment’. In: *International journal of smart home* 7.1 (2013), pp. 77–86.

[106] Pravin Varaiya. ‘Smart cars on smart roads: problems of control’. In: *IEEE Transactions on automatic control* 38.2 (1993), pp. 195–207.

[107] Ovidiu Vermesan et al. ‘Internet of things strategic research roadmap’. In: *Internet of things-global technological and societal trends* 1.2011 (2011), pp. 9–52.

[108] Richard L Villars, Carl W Olofson and Matthew Eastwood. ‘Big data: What it is and why you should care’. In: *White paper, IDC* 14 (2011), pp. 1–14.

[109] Jiafu Wan et al. ‘Context-aware vehicular cyber-physical systems with cloud support: architecture, challenges, and solutions’. In: *IEEE Communications Magazine* 52.8 (2014), pp. 106–113.

[110] Fei-Yue Wang, Daniel Zeng and Liuqing Yang. ‘Smart cars on smart roads: an IEEE intelligent transportation systems society update’. In: *IEEE Pervasive Computing* 5.4 (2006), pp. 68–69.

[111] Martin Wollschlaeger, Thilo Sauter and Juergen Jasperneite. ‘The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0’. In: *IEEE industrial electronics magazine* 11.1 (2017), pp. 17–27.

[112] John Yen and Reza Langari. *Fuzzy logic: intelligence, control, and information.* Vol. 1. Prentice Hall Upper Saddle River, NJ, 1999.

[113] Eric Yuan and Jin Tong. ‘Attributed based access control (ABAC) for web services’. In: *IEEE International Conference on Web Services (ICWS’05).* IEEE. 2005.

[114] Andrea Zanella et al. ‘Internet of things for smart cities’. In: *IEEE Internet of Things journal* 1.1 (2014), pp. 22–32.