

STRUCTURED ARCHITECTURE
EVALUATION OF WELLBEING IOT
SYSTEMS

-

A SAFETY- AND SECURITY-DRIVEN
APPROACH

Julia Rauscher

DISSERTATION

for the degree of
Doctor of Natural Sciences (Dr. rer. nat.)



University of Augsburg
Department of Computer Science
Software Methodologies for Distributed Systems

September 2021

Structured Architecture Evaluation of Wellbeing IoT Systems - A Safety- and Security-Driven Approach

Supervisor: **Prof. Dr. Bernhard Bauer**, Department of Computer Science,
University of Augsburg, Germany

Advisor: **Prof. Dr. Jörg Hähner**, Department of Computer Science,
University of Augsburg, Germany

Defense: 15th December 2021

Copyright © Julia Rauscher, Augsburg, September 2021

Abstract

We are living in a connected world.

In line with this trend, more and more devices have to be connected and communicate automatically. Through these new requirements the Internet of Things (IoT) emerged. The usage in industry, smart cities and agriculture are only a few of the widely spread application fields. Next to these, the healthcare area is able to take advantage of this technology trend in form of the Internet of Things of Wellbeing Devices (IoT-WD). The potential of IoT-WD is recognized by more and more stakeholders. Ambient assisted living (AAL), telecare including remote vital data monitoring and personalized wellbeing approaches, e.g. recommendations of fitness trackers, are only a couple of the possibilities for health-conscious people to take advantage of the IoT-WD. All these functions are made possible by the conjunction and cooperation of so-called things. However, next to above named advantages, IoT is endangered to include new safety and security risks since connected things bring hidden dependencies, and thus, possible dangerous impacts. Especially, IoT-WD has to deal with health-endangering vulnerabilities. Those range from manipulation of intimate data to life-imperiling endangerments of exactly those with a higher need of care, like infants or seniors. Thus, an identification and elimination of safety and security vulnerabilities as early as possible is essential. Since 50% of security flaws arise during the design phase, architecture approaches are needed to offer timely identification and prevention of negative impacts on human beings [VM01].

IoT systems consist of many components, some of which are very small but nevertheless provide important functions, resulting in complex system models. A manual check for dangerous or harmful design decisions in these models is tedious and would include outdated data, because the monitoring process would be highly time-consuming. Since each IoT system is different, a unified and automated way to make review approaches universally applicable is needed. For this purpose, a unified IoT(-WD) meta model and an IoT layered architecture is developed with which IoT architectures can be modeled and analyzed on a reusable level. The meta model contains wellbeing specific components to optimally depict the critical areas, but focuses on modeling of safety and security relevant system aspects and is accordingly based on known IoT safety and security challenges.

As mentioned above, weaknesses often arise in the design phase, in form of so-called design flaws, which could already be noticed at this point but often only become apparent in the fully implemented live system. Since expert knowledge is often not or only temporarily available during the system development lifecycle issues are not caught early on. For this challenge this dissertation develops

a Pattern Recognition Framework (PRF) to present a by design approach for a structured and semi-automated flaw identification process which focuses on pattern respectively anti-pattern based expert knowledge preservation.

Linked to this is the assessment of identified flaws, as without further analyses prevention or mitigation is not possible. Since impacts can be complex and ramified, the process has to be automated as well. An analysis cycle is presented to show the technical and quantitative impact of potential design flaws including wellbeing information for assessment. In addition, the cycle offers a design decision method to weigh up possible countermeasures and to plan an optimized architecture including the required new services. In order to benefit from already successful architecture analysis approaches, IoT usable analysis steps are abstracted and adapted for this purpose.

Finally, the approach of this dissertation is undergoing a three-part evaluation: Related work-based, case study-based and scenario-based. It is evaluated whether the individual approach steps not only cover the current State-of-the-Art but also extend it. Accordingly, it is ensured that no aspects are omitted. An AAL use case, which includes a smart home for the elderly, applies the different steps of the approach by identifying and assessing design flaws to evaluate applicability. A generated performance use case is considered to demonstrate the scalability in large complex models. The scenario-based evaluation includes examination of quality characteristics adaptability, expandability, scalability and reusability.

Zusammenfassung

Wir leben in einer vernetzten Welt.

Einhergehend mit dieser Entwicklung werden immer mehr Geräte verbunden und kommunizieren automatisch miteinander. Durch diese neuen Anforderungen ist das Internet der Dinge (IoT) entstanden. Der Einsatz in der Industrie, Smart Citys und der Landwirtschaft sind nur einige der weit verbreiteten Anwendungsfelder. Zusätzlich kann sich auch der Gesundheitsbereich diesen Technologietrend in Form des Internet of Things of Wellbeing Devices (IoT-WD) zunutze machen. Das Potential von IoT-WD wird von immer mehr Akteuren erkannt. Ambient Assisted Living (AAL), Telepflege inklusive Überwachung von Vitaldaten und personalisierte Wellbeing-Ansätze, wie Empfehlungen von Fitness-Trackern, sind nur einige der Möglichkeiten für gesundheitsbewusste Menschen, das IoT-WD zu nutzen. All diese Funktionen werden durch den Zusammenschluss und die Zusammenarbeit vieler sogenannter Dinge ermöglicht. Neben den oben genannten Vorteilen birgt das IoT aber auch neue Gefahren und Sicherheitsrisiken, da vernetzte Dinge versteckte Abhängigkeiten und damit mögliche gefährliche Auswirkungen mit sich bringen können. Insbesondere IoT-WD muss sich mit gesundheitsgefährdenden Schwachstellen auseinandersetzen. Diese reichen von Manipulation intimer Daten bis hin zu lebensbedrohlichen Gefährdungen, gerade von Menschen mit erhöhtem Pflegebedarf wie Kleinkinder oder Senioren. Eine möglichst frühzeitige Identifizierung und Beseitigung von Sicherheitslücken ist daher unerlässlich. Da 50% der Sicherheitsmängel während der Entwurfsphase entstehen, werden Ansätze benötigt, die eine frühe Identifikation und Vermeidung von negativen Auswirkungen auf den Menschen beinhalten [VM01].

Da IoT-Systeme aus vielen Komponenten bestehen, von denen einige sehr klein sind, aber dennoch wichtige Funktionen bereitstellen, sind Modelle für diese Systeme komplex. Eine manuelle Überprüfung auf gefährliche oder schädliche Designentscheidungen ist mühsam und würde veraltete Daten einbeziehen, da der Überwachungsprozess sehr zeitaufwändig ist. Da jedes IoT-System anders gestaltet ist, wird ein einheitlicher und automatisierter Weg benötigt, um Prüfansätze universell einsetzbar zu machen. Zu diesem Zweck wird ein einheitliches IoT(-WD) Metamodell und eine IoT-Schichtenarchitektur entwickelt. Mit diesen können IoT-Architekturen auf einer wiederverwendbaren Ebene modelliert und analysiert werden. Das Metamodell enthält spezifische Wellbeing-Komponenten, um die kritischen Bereiche optimal darstellen zu können. Der Fokus liegt dabei auf der Modellierung von sicherheitsrelevanten Systemaspekten und basiert daher auf bekannten IoT Safety- und Security-Herausforderungen.

Wie bereits erwähnt, entstehen Schwachstellen oft schon in der Entwurfsphase in Form von sogenannten Design Flaws, die bereits zu diesem Zeitpunkt bemerkt werden könnten, sich aber oft erst im vollständig implementierten Live-System zeigen. Da Expertenwissen während des Entwicklungskreislaufs oft nicht oder nur temporär vorhanden ist, werden Fehler nicht frühzeitig erkannt. Für diese Herausforderung wird im Rahmen dieser Dissertation ein Pattern Recognition Framework (PRF) entwickelt, um einen Ansatz für einen strukturierten und semi-automatischen Fehleridentifikations-Prozess zu bieten, der sich auf eine Pattern- bzw. Anti-Pattern-basierte Konservierung von Expertenwissen konzentriert.

Damit verbunden ist die Bewertung identifizierter Schwachstellen, da ohne eine fachliche Analyse derer keine Vorbeugung oder Abschwächung möglich ist. Da die Auswirkungen komplex und verzweigt sein können, muss der Prozess ebenfalls automatisiert werden. Es wird ein Analysezyklus vorgestellt, der die technischen und quantitativen Auswirkungen von Designfehlern aufzeigt und dabei Wellbeing-Informationen für die Bewertung mit einbezieht. Zusätzlich dazu bietet der Zyklus eine Design-Entscheidungsmethode, um mögliche Gegenmaßnahmen abzuwägen und eine optimierte Architektur, inklusive der benötigten neuen Services, zu planen. Um bereits erfolgreiche Ansätze von Architekturanalysen zu nutzen, werden IoT-taugliche Analyseschritte abstrahiert und an diesen Zweck angepasst.

Der Ansatz dieser Dissertation wird einer dreiteiligen Evaluation unterzogen: Related Work-basiert, Use Case-basiert und Szenario-basiert. Es wird evaluiert, ob die einzelnen Schritte des Ansatzes nicht nur den aktuellen State-of-the-Art abdecken, sondern diesen auch erweitern. Entsprechend wird sichergestellt, dass keine Aspekte ausgelassen werden. Anhand eines AAL-Anwendungsfalls, der ein Smart Home für Ältere beinhaltet, werden die einzelnen Schritte des Ansatzes angewendet, indem Designfehler identifiziert und bewertet werden, um die Anwendbarkeit zu evaluieren. Ein generierter Performance Use Case wird verwendet, um die Skalierbarkeit in großen komplexen Modellen zu demonstrieren. Die Szenario-basierte Bewertung umfasst die Untersuchung von den Qualitätsmerkmalen Anpassbarkeit, Erweiterbarkeit, Skalierbarkeit und Wiederverwendbarkeit.

Acknowledgment

First of all I would like to thank my supervisor Prof. Dr. Bernhard Bauer who gave me the opportunity to write this PhD thesis at the SMDS lab. Thank you, Bernhard, for the many conversations, continuous motivation and best working environment that one could wish for a dissertation. I have always experienced great support, especially in the first years when I changed my topic several times. I also want to thank Prof. Dr. Jörg Hähner for being my advisor of this thesis.

Furthermore, I want to thank my current and former colleagues. The relaxed working atmosphere has made my workplace a place where you can ask for help or advice at any time and are pleased to come back every day. In particular, I would like to thank Sonja. She not only supported me with administrative tasks, but also brought me a lot of joy while crafting several doctor hats together.

I would also like to thank my friends who had to listen to my worries for many years, and thus indirectly had a strong effect on this work. The many activities together have always kept me going and provided me with the energy to continue writing this thesis. I would especially like to mention my best friend Tali. She has been with me in my academic career since day 1, starting with the bachelor's degree all the way to the PhD, and has supported me with numerous late-night conversations and phone calls.

A special thanks goes to my family, who has guided me throughout my life, backed me up and made me the person I am today. My parents have always done everything they could to support me and have constantly encouraged me to go my own way, no matter what I had in mind next. Thank you for everything!

Last but not least, I want to thank my fiancé Thomas. He is the most delightful and encouraging person in my life who always believed in me, especially in the moments I definitely did not. Whenever I wasn't in the best of moods, he always made me laugh with his unique sense of humor and encouraged me with his never-ending positive attitude.

Contents

I. INTRODUCTION AND BASICS	1
1. Introduction	3
1.1. Motivation	4
1.1.1. Increasing IoT Networks	5
1.1.2. Growing Healthcare IoT Networks	5
1.1.3. Safety- and Security-Critical IoT Systems	6
1.1.4. IoT Management Lifecycle	8
1.2. Challenges	9
1.2.1. Management of Complex Systems	9
1.2.2. Neglected Safety and Security by Design	10
1.2.3. Delayed Flaw Identification after Deployment	11
1.2.4. Manual Flaw Assessment	11
1.3. Objectives and Contributions	12
1.3.1. Depictability and Unification of IoT Architectures	13
1.3.2. Early Design Decision Evaluation and Flaw Detection	13
1.3.3. Flaw Assessment and Architecture Optimization by Design	14
1.4. Approach Roundup and Methodology	16
1.5. Thesis Outline	18
1.6. Publications	21
1.6.1. Scientific Publications	21
1.6.2. Project Deliverables	24
2. Foundations	27
2.1. Architecture Terminology	28
2.1.1. Architectures and Reference Architectures	28
2.1.2. Design Phase	31
2.1.3. Architecture Analyses	33
2.1.4. Architectural Pattern	34
2.2. Connected Devices and Systems	36
2.2.1. Internet of Things	36
2.2.1.1. Reference Architectures of IoT Systems	38
2.2.1.2. Wireless and Embedded Systems	40
2.2.1.3. Data Management and Communication Protocols	43
2.2.2. Connected Systems in Wellbeing	45
2.2.2.1. Medical and Wellbeing Application Platforms	49
2.2.2.2. Personalized Healthcare	50
2.3. Approach Requirements	53
2.3.1. Safety and Security	53
2.3.2. Requirements in Critical Areas	54
2.3.3. Risk Management	55

2.3.4.	Safety and Security By Design	56
2.3.5.	Regulations and Standards	57
2.4.	Enterprise Architecture Management	60
2.4.1.	Layered EA Structure	62
2.4.2.	EA Analysis Techniques and Goals	63
2.4.2.1.	Bayesian Belief Networks	64
2.4.2.2.	Extended Influence Diagrams	67
2.4.3.	EAM and IoT Overlap	68
 II. SAFETY AND SECURITY DESIGN APPROACH		71
3.	Safety and Security Architectural Requirements	73
3.1.	Challenge Review	74
3.1.1.	Safety Hazards in IoT Systems	74
3.1.2.	Security Threats in IoT Systems	75
3.1.3.	IoT-WD Specific Challenges	79
3.2.	Challenge Categorizations	81
3.3.	Derived Architectural Requirements	83
4.	IoT-WD Modeling	87
4.1.	IoT Layered Architecture	88
4.2.	IoT-WD Meta Model	91
4.2.1.	Meta Model Requirements	91
4.2.2.	Formalization of Meta Model	93
4.2.3.	Specification of Meta Model	96
4.2.4.	Limitations and Constraints	104
4.2.5.	Modeling Editor	104
4.3.	Related Work	108
 III. ARCHITECTURE OPTIMIZATION		111
5.	Safety and Security Optimization Process	113
5.1.	Improvement Process Requirements	114
5.2.	Process Steps Outline	116
5.3.	Integration Factors and Requirements for Related Approaches and Analyses	119
6.	Flaw Identification	121
6.1.	Pattern Recognition Framework	122
6.1.1.	Formalization of PRF and Patterns	124
6.1.2.	Pattern Definition Approach	126
6.1.2.1.	Generic Specifications	128
6.1.2.2.	Safety and Security Challenge Specifications	129

6.1.2.3.	Safety and Security Assessment Specifications . . .	131
6.1.2.4.	Pattern and Anti-Pattern Implementation Specifications	134
6.1.3.	Pattern Specification Language	136
6.1.4.	Pattern Service Generation	140
6.1.5.	Pattern Identification	147
6.2.	Pre-Defined Pattern and Pattern Database	148
6.3.	Related Work	150
7.	Flaw Assessment	153
7.1.	Transferable Architecture Analyses	154
7.1.1.	Identification of EA Analyses	155
7.1.2.	Categorization of Analysis Approaches	157
7.2.	IoT Architecture Assessment Specifications	173
7.2.1.	Analysis Adaptation Requirements	173
7.2.2.	Formalization of Architecture Analyses	174
7.2.3.	Analysis Selection and Cycle Flow	175
7.3.	Failure Impact Analysis	178
7.4.	Quantitative Impact Analysis	185
7.5.	Countermeasure Decision Support Analysis	190
7.6.	Service Interoperability Analysis	195
7.7.	Related Work	200
IV.	CASE STUDIES AND EVALUATION	205
8.	Evaluation	207
8.1.	Related Work-based Evaluation	208
8.2.	Case Study-based Evaluation	215
8.2.1.	Ambient Assisted Living Use Case	215
8.2.1.1.	System Model	216
8.2.1.2.	Pattern and Anti-Pattern Specifications	218
8.2.1.3.	Sirius Service Application	221
8.2.1.4.	Assessment Cycle Application	223
8.2.2.	Generated Performance Use Case	231
8.2.2.1.	Model Editor Complexity Test	231
8.2.2.2.	Flaw Identification Performance	232
8.2.2.3.	Analyses Performance	235
8.3.	Scenario-based Evaluation	239
8.3.1.	Adaptability	239
8.3.2.	Expandability	241
8.3.3.	Scalability	243
8.3.4.	Reusability	245

V. CONCLUSION AND OUTLOOK	247
9. Conclusion	249
9.1. Design Approach	250
9.2. Architecture Optimization	252
9.2.1. IoT Flaw Identification	252
9.2.2. IoT Flaw Assessment	253
10. Limitations and Future Work	255
VI. Annex	259
Bibliography	261
List of Acronyms	285
List of Figures	287
List of Tables	291
Listings	293
A. Appendix	295
A.1. Meta Model and Enumerations	296
A.2. Model Editor Elements	299
A.3. Architecture Analyses	300

Part I.

INTRODUCTION AND BASICS

1

Introduction

"Internet of Things: A useful innovation or security nightmare?" [SK16]

This dichotomy only can be resolved by the way Internet of Things (IoT) is handled. As both scenarios could become reality, security and with it safety management is one of the most important tasks for a successful deployment of IoT. While the architecture is comprised of many simple elements, the vast majority of the main components, even the smallest, are interconnected, and therefore, carry risks respectively hidden impacts and need to be included in the surveillance. Especially, since the use of embedded devices and sensors is increasingly deployed in IoT-based applications in high-risk sectors, e.g. healthcare to take advantage of remote monitoring functions of vital data, and thus are not allowed to contain vulnerabilities which could harm humans [Thi+18]. Therefore, the goal of this dissertation is to contribute towards making the scenario of useful IoT innovations without safety and security concerns the reality.

This chapter presents the current developments and the associated challenges of IoT, especially in safety- and security-critical areas such as healthcare and well-being. In order to classify these, the developed lifecycle of IoT management is presented and used to place *Objectives* and *Contributions* of this work into context. The methodology is explained in order to highlight the steps of the approach components and to frame the outline and the related publications.

1.1. Motivation

In the age of digitalization there is an increasing number of devices which communicate and interact with each other. This has led to networks including more independent and intelligent devices that can act and react in a uniquely identifiable and automated manner which are known as IoT systems.

These new open embedded and connected networks have emerged through years of development. Until the current state was reached, the industry went through 4 generations. It has started with the mechanization of production facilities which in turn went over to electrification with mass production and assembly line activities. In the early 70s, the industry was fundamentally changed by the automation and use of electronics and IT. This was the foundation for Industry 4.0 and networking based on cyber physical systems. [Var; ER19]

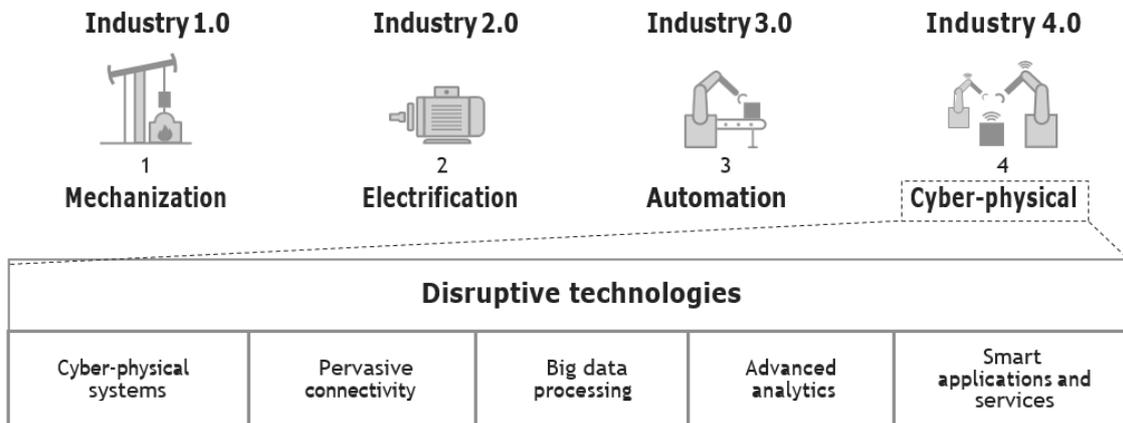


Figure 1.1.: Historic development of industry ([ER19])

Since then, smart things and applications have been interacting with each other, gathering and processing data, and using big data to perform analytics. The data should be used to make intelligent and automatic decisions, and thus, require less human interaction. While IoT originated in simple Radio Frequency Identification (RFID) technologies and industrial applications, it is spreading into more and more areas. [Isc17] Home automation, smart agriculture, fitness and energy conservations are only a few to name. [SS17] All domains want to benefit from the evolution and try to exploit the benefits, such as saving money and time, better monitoring and making intelligent decisions.

IoT systems offer plenty of benefits, thus, also bear challenges. Among other issues, there are two major concerns. First, they are increasing fast which creates complexity including hidden vulnerabilities. Second, they aren't self-contained and hence, are connected to the Internet that leads to additional new possible cyber attacks, new hazards and threats in comparison to conventional computer networks.

1.1.1. Increasing IoT Networks

IoT is one of the fastest and most unstoppable developments of our time. Last year's expected installed 31 billion IoT devices will be more than doubled in 5 years. According to [Dep20] by 2025 there will be 75.44 billion connected devices. The revenue of this estimated IoT device market development will generate \$1.1 trillion. [Sec20]

The IoT trend has been implemented widely from the beginning, but since not only the industry but also other application areas recognize the potential, the number of devices continues to grow immensely. However, not only the so-called connected things increase, but also it is now estimated that 4 billion people are connected through IoT, which offers great potential for data mining. [Com]

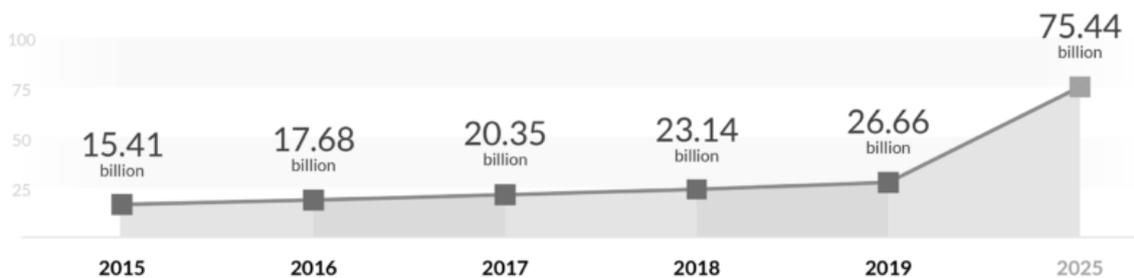


Figure 1.2.: Number of installed IoT devices around the world [Ant21]

The parallel development of cloud services and 6G technologies enables even more data to be collected and stored easily. The gained big data can be used, e.g. for data analytics and machine learning approaches. These results can in turn be used in the IoT for automated actions. This automation or intelligent decisions can, for example, be used in smart factories to facilitate maintenance with sensors. 80% of retailers in the U.S. use these IoT benefits to design store visits and 66% of U.S. cities want to use it for smart city approaches. Always aiming in reducing the workload for people or increasing the quality of life. [Ant21]

1.1.2. Growing Healthcare IoT Networks

One of the main beneficiary of IoT technologies is the healthcare sector. By 2025, the market volume of IoT in the healthcare sector is forecast to grow up to \$188.2 billion, and thus, is one of the strongest IoT consumers after industry. [MM20] Even though this development has already been in the offing in recent years, the Covid-19 pandemic is contributing to it. On the one hand, telecare approaches can avoid face-to-face visits, and thus, enable practicing social distancing guidelines and Covid-19 patients can be supported during quarantine. [Ant21] On the

other hand, wellbeing devices can help maintain the mental and physical health of self-isolated people with recommendations.

The application areas of IoT in the wellbeing or medical field are as diverse as the discipline itself. The vital parameters of patients can be automatically measured and transmitted, in-vitro approaches can perform blood analysis with connected devices, and physiological values such as weight, heart rate and glucose level can be determined. In addition, apps can help with medication, observe sleep and fitness, or monitor internal devices such as implants. Even organizational tasks such as bed management can be implemented with IoT. Regardless of the task, focus is always on improving the user experience, saving money, and creating a continuous disease management. To achieve this, it must always be ensured that the devices work together correctly, data is properly integrated and protected, and expert knowledge is included in spite of everything. [Mos17; Ko+10]

The classical beginnings of the IoT have been in hospitals with RFID technologies to monitor patient locations or movements, and additionally to track equipment [AAA13]. However, the more vital data or surgical IoT technologies are incorporated, the greater the growth in responsibility. IoT systems with medical devices contain health and lifesaving responsibilities and are accordingly subject to even greater trust and acceptance issues. A classic mission-critical IoT system, thus, becomes a safety- and security-critical system.

Accordingly, rapid developments are to be found more in the wellbeing and care sector as they have less dictated requirements. These include smart homes for care of elderly, organizational care processes or voluntary data transfer for studies. However, as rapid growth also often pushes immature and insecure products onto the market, this area is no less safety- and security-critical. [Ant21] Even though these systems and products are designed mainly for prevention, they can also have severe consequences. Due to the fast growing danger, this dissertation focuses mainly on the wellbeing and not the medical areas.

1.1.3. Safety- and Security-Critical IoT Systems

IoT technologies are popular in safety- and security-critical applications as they can monitor critical areas respectively prevent human failures, e.g. monitoring quality in food production by special sensors, early detection of hardware attrition or self-driving cars to prevent accidents caused by inattention. However, the potential benefits are lost if the systems and communication they contain are not designed securely. All communication channels have to be secure and trustworthy. The data generated, sent or received by a device shall not be compromised in terms of integrity, correctness, tampering, manipulation or delays. [Mic18a] Otherwise this data can not be a trusted source.

As mentioned above, IoT systems are still security nightmares. [Isc17] This applies to safety aspects as well. The security market has continued to grow in recent years ([Sec20]), but despite this, the problems continue to increase. Especially in critical application areas, this is a worrying development. In particular as IoT has to handle IoT specific safety and security challenges like data transmission in sensor networks as well as conventional issues like DOS attacks, eavesdropping or virus damages [GLJ11]. According to [OWA18], one of the biggest problems in IoT networks are hardcoded passwords, insecure network services, lack of secure update mechanism, outdated components and insecure data transfer. Often, data can even be intercepted in cleartext. [WAF17] This set of challenges can arise and occur at plenty times and points.

"Many literally run red lights and are surprised when they get run over." [IT-18] This quote comes from Arne Schönbohm, President of Bundesamt für Sicherheit in der Informationstechnik (BSI) and represents the current status of many companies related to their preventive measures. Critical data in IoT networks are sent unencrypted in 98% of traffic which enables attackers to listen and exploit the information ([Pal20]) and according to [Ant21], 70% of IoT devices on the market are easy to attack. From 2018 to 2019, the number of cyber attacks per company doubled, and even if security experts are now able to detect the intruders more quickly, they have caused 43 billion euros in damage in these two years. [Wil19] However, according to [Liu17], only 37% of companies have a response plan. In some cases, these companies work with critical data or their devices are used in critical IoT networks.

The usage of IoT in critical application fields including human beings produces danger of health-endangering vulnerabilities, wrong intimate data or other life-imperiling endangerments. As [Pal20] has shown, most of the connected image devices are operating on OSs without update support leading to threats and hazards that impact the quality of care or privacy. In addition, malware is spreading easily caused by the combination of IoT and IT assets which can lead to exposed or interpreted falsely data.

Private IoT-WD systems face safety and security issues as shown by examples like [SB15] or [FDA17]. Manipulated baby monitors or captured implantable cardiac devices for monitoring purpose represent highly critical elements which are difficult to alter afterwards. A loss of data or control can be severe for residents. However, they are often affected, because many inexpensive and insecure components are used in smart homes. This thesis focuses on the realities of IoT in the wellbeing sector, and thus, IoT-WD systems.

1.1.4. IoT Management Lifecycle

IoT systems require management functions in several areas, for instance the maintenance of components. Since this work focuses on the establishment, validation and optimization of IoT systems, the management cycle of development to deployment is considered here.

The lifecycle (figure 1.3) can be divided into 3 phases. At the beginning, the application area and respective use case have to be considered and their components and conditions discovered. Afterwards, a first architecture can be designed. Following this point, the second phase begins: Evaluation of design. The cycle can go directly back to design phase to revise the architecture or the designed system can be deployed in the next step. Thus, the planned system is allowed to run or simulated. All information collected after deployment can be evaluated in a further evaluation phase. If necessary, the design phase can be repeated. Subsequently, planned design changes have to be evaluated and deployed again.

To apply the management process to an existing system, the cycle starts with deployment and subsequent evaluation of the identified aspects while running the system.



Figure 1.3.: IoT management lifecycle

1.2. Challenges

In the following, four major *Challenges* are presented that can be integrated into the phases of IoT management lifecycle (figure 1.4). Since most challenges do not only affect one phase but also have causes and effects in other phases, they extend several challenges over several phases.

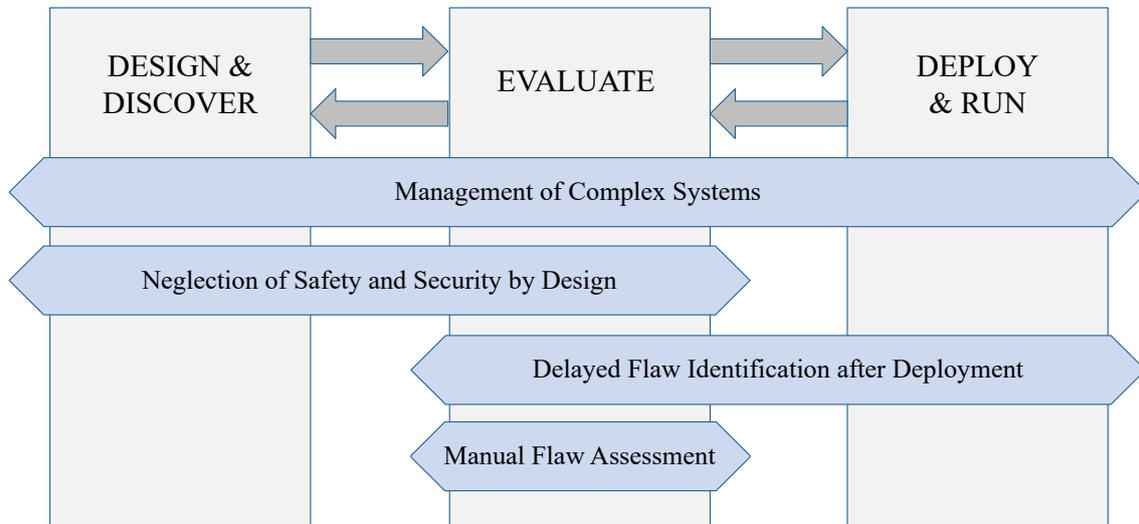


Figure 1.4.: Challenges in IoT management lifecycle

1.2.1. Management of Complex Systems

A problem that extends across all phases is monitoring and coordination of IoT systems. IoT systems can vary greatly in size and complexity depending on the use case and its requirements. However, since a basic characteristic is the connection to the Internet, they are always large, complex networks that include not only the local components, but remote distributed elements or stakeholders. This need for local and at the same time global management is more difficult than pure wireless sensor networks or IP networks, since devices with different properties that are integrated in different networks are context aware [Sil+17]. As the number and connectivity of these challenges steadily increase, so does the scale of challenges [Fu+17].

High amounts of components, relations, stakeholders and other external requirements are leading to the *Challenge* that currently IoT systems cannot be displayed in their entirety. No reliable management decision or analyses can be made on an incomplete representation or data basis. All required information of diverse aspects need to be considered and depicted. For example, if connections are missed

out a meaningful estimation of impacts can not be given or forgotten requirements of stakeholder can entail technical and financial risks.

Missing unification of IoT modeling impedes proper management or further analyses, as the approaches cannot be reused or extended, and thus, countless new individual solution approaches are developed. These are time-consuming and costly, and are therefore, often not taken into consideration.

1.2.2. Neglected Safety and Security by Design

Systems should be checked at all development cycle points, especially for vulnerabilities that could put people or the system at risk. Thus, the larger the networks are, the more vulnerable and inscrutable they become. This is a significant *Challenge* especially when IoT is used in safety- and security-critical areas. Therefore, as discussed, complex safety and security activities are highly needed. However, these aspects are often excluded in several development phases and monitoring is often only considered necessary in the deployed system in order to detect current attacks. Though, the most important point is prevention, preferably directly when they occur. The lack of update mechanisms is one of the big challenges since many IoT devices do not have the possibility to be updated. Reasons range from software design decisions to hardware issues. [Tas02] has stated that it is 30 times less expensive to fix problems in the design phase rather than in the post-product release phase. In addition, he has noticed that problems start to arise during requirements gathering and architectural design. It is estimated that up to 50% of security flaws happen in the design phase [VM01]. However, since most approaches focus on software level there is a lack of model-based IoT approaches. Additionally, the existing model-based approaches are either use case specific or cannot be automated. This can be noticed by the fact that only 26% of surveyed companies implemented their IoT initiatives successfully [Res]. However, if these companies consider all development phases they would be significantly more successful.

Especially, neglection of safety and security by design is critical in the wellbeing sector. Deploying healthcare sensors and utilizing vital data causes a high need for IoT architectures free of flaws. In wellbeing embedded systems, by design approaches are often still rarely used which has created an urgent need. [Fre12] The reasons for this are often a lack of expert knowledge during the design phase and the time consuming aspects.

1.2.3. Delayed Flaw Identification after Deployment

Closely related to the neglected by design approaches is the *Challenge* of flaw detection and assessment during the final management phase when systems have already gone through the evaluation phase and have been deployed. In this phase, weaknesses in the design are often no longer considered or checked. Thus, the real reasons for vulnerabilities sometimes remain undiscovered. An important reason for this is the fact that IoT research is focusing more on evaluating big data and data science approaches, and thus, data analytics. [Ana+18] The strong focus on data puts the deployed systems in the spotlight. The fact that threats and hazards only appear in active systems often leads to that only action is taken place when necessary and the design is not examined for possible flaws at an early stage. A frequent time pressure in the development of IoT devices is a reason for the fast completion of the design phase and the quick deployment of systems [Ant21].

Although it could be seen as a financial consideration when a flaw fix is best timed, it has to be considered, as mentioned, that many IoT devices do not have the ability to update afterwards. Even if [Mic18a] states that there shall be a way to update software and firmware and ensure that bugs can be fixed and rolled back if the update is unsuccessful, often this possibility is not given due to cheap hardware, faulty design or is considered too late after deployment.

1.2.4. Manual Flaw Assessment

When architectural approaches are used, they are addressed either in the design phase or in the evaluation phase. In order to fully evaluate, mitigate or prevent the impact of flaws, a detailed evaluation is required. Flaws that are only identified but not considered further may appear to have been removed, but do not consider far-reaching consequences of new design alternatives. Also, the decision whether to change the design at all can only be answered competently if all affected components have been identified and if several evaluation criteria are included that must be coordinated with each other. In the complex IoT systems described above, this is an almost unmanageable task.

Even if this is not a new insight, still "automated technologies to detect design-level flaws do not yet exist [...]" [McG06]. Complete risk management is to identify problems, to determine probability of occurrence and to estimate severity of damage. [US 07b] Accordingly, there is not only a lack of automated methods for flaw detection in design, but also for automatic assessment of effects. Manual approaches are not only time-consuming but also not up-to-date because other design decisions may already have been made or new problems are known.

1.3. Objectives and Contributions

In the last section, *Challenges* that are addressed in this dissertation were presented. In order to develop solutions for these *Challenges*, *Objectives* and corresponding *Contributions* are presented and assigned to *Challenges* and phases in the following. The selected colors are intended for the assignment of *Objectives* and will be used in the further course of this chapter.

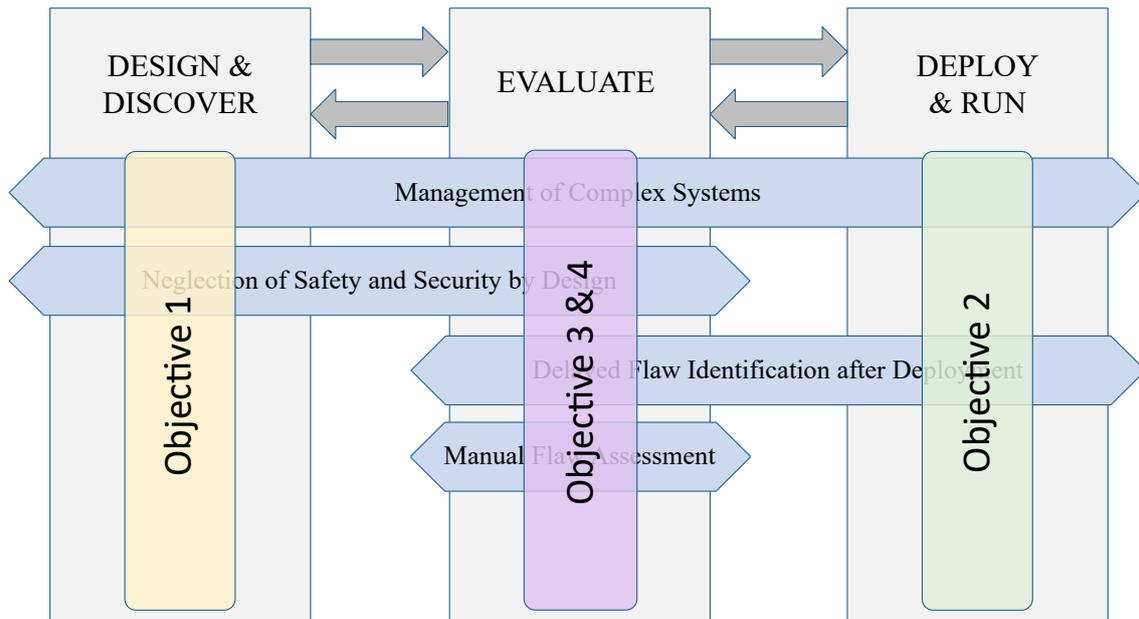


Figure 1.5.: Objectives in IoT management lifecycle

Figure 1.5 shows the mapping of *Objectives* and *Challenges*. *Objective 1* covers aspects from the first phase of the IoT management lifecycle and affected *Challenges*. *Objective 2*, on the other hand, focuses mainly on delayed flaw detection. *Objectives 3 and 4* are interconnected and cover the architectural evaluation phase. Since all *Challenges* are anchored in this phase, more *Contributions* are needed.

Since the *Objectives* are designed to cross multiple *Challenges*, the *Contributions* are applied consecutively. They can be used individually, but an optimal architectural design is only achieved with the complete application of the continuous *Contribution* steps, as they partially perform the required preparatory work.

1.3.1. Depictability and Unification of IoT Architectures

As described, complex systems come with complex models respectively architectures which are highly component-heavy, ramified, and thus, difficult to secure in early stages. Therefore, a possibility is needed to depict all available architectural information to cover all aspects which are required for further evaluation and analyses. Since architecture requirements and use cases can differ and change significantly, a generic and reusable solution is needed.

Objective 1

Provide a unified and reusable IoT system modeling method to enable use case independent usage including the possibility to analyze safety and security required aspects.

Contributions

To create such a modelable and analyzable IoT architecture multiple steps are proposed:

- Extraction of IoT safety and security challenges, especially in the wellbeing domain, and abstraction of required architectural information of these risks to identify the architectural decisions that enabled these vulnerabilities.
- Development of an IoT(-WD) meta model including abstracted architectural analysis aspects to enable evaluation, flaw detection, distribution and reuse. The meta model can be used generically, but contains wellbeing specific elements to optimally map wellbeing IoT systems. A model editor is providing different filters, views and extensible functions to automate the analysis later on.
- Specification of an IoT layered approach to allow a detailed classification of elements and issues that can address the different use cases of IoT domains, regardless of home or industry applications.

1.3.2. Early Design Decision Evaluation and Flaw Detection

Even when the planned models pass the evaluation phase, many unwise design decisions are overseen or the design is not aware of dangers it poses. This is largely due to the fact that, as described, a manual check misses vulnerabilities as they have hidden effects. That leads to the described delayed flaw detection *Challenges* in deployed systems. Therefore, this *Objective* offers an approach to push the detection in an earlier phase.

Objective 2

Develop an IoT specific flaw identification process to mitigate impacts of possible threats or hazards on an early architectural level through safety and security review by design.

Contributions

To improve the architecture of IoT(-WD) at an early stage this thesis offers:

- Structured conservation of architectural expert experience to create an always available knowledge base accessible regardless of programming skills.
- Development of a Pattern Recognition Framework (PRF) to implement a semi-automated vulnerability or flaw identification process. PRF records generic information that is needed for the subsequent assessment, but also offers the possibility to store wellbeing specific criteria.
- Implementation of a Domain Specific Language (DSL) to define design patterns and anti-patterns based on the preserved expert knowledge to recognize harmful design decisions during design phase in an automated manner.

1.3.3. Flaw Assessment and Architecture Optimization by Design

Objectives 3 and 4 cover parts of all four presented Challenges, as they overlap in the evaluation phase, by developing further approach steps to automatically assess flaws. The identified weaknesses of Objective 2 entail a need for action. However, the mere recognition of a problem does not directly clarify the necessary mitigation or prevention activities. The flaws have to be reviewed in a structured manner with all their effects on the system in order to not fear any new negative influences from design changes. Since IoT systems have special characteristics and differ significantly from conventional systems, the previous approaches cannot easily be reused.

Objective 3

Identify requirements to transfer existing architecture analysis approaches into IoT safety and security management including evaluation and comparison of adaptation and usage limitations.

Contributions

To analyze architecture aspects providing conditions for a suitable selection for an IoT analysis cycle, structured research steps are used:

- A literature review to provide an overview of available architectural analysis approaches. In addition, development of an assessment of maturity of these approaches.
- Definition of requirements and limitations to enable or prevent a transfer, which are based on architectural IoT aspects.
- Development of an approach categorization that considers functional and technical aspects, to select appropriate and adaptable methods.

Objective 4

Develop a structured and consecutive architecture analyses workflow to use flaw assessment methods to conduct a holistic IoT design optimization of To-Be scenarios by estimating impacts in As-Is models.

Contributions

To evaluate identified vulnerabilities consecutively and coordinated on each other, multiple steps are provided:

- Development of a structured cycle that does not terminate after concluded assessment. Partial assessment steps are preliminary steps of the next phase of the analysis cycle. A co-developed tool provides the possibilities of automating flaw assessment.
- Specification of cycle methods that assess current weaknesses of the As-Is architectures and additionally can compare alternatives of To-Be designs. All methods have features to apply stored wellbeing information for assessment.
- Design optimization and prevention of threats and hazards, in order to cover the presented safety and security challenges in critical IoT wellbeing areas, dependent on different views and aspects.

1.4. Approach Roundup and Methodology

To provide a visual overview of content an approach roundup follows. The work of this thesis can be depicted as stacked approach (figure 1.6), divided into three categories: **Concept**, **Tooling** and **Evaluation**. The colors are oriented on the associated *Objectives*.

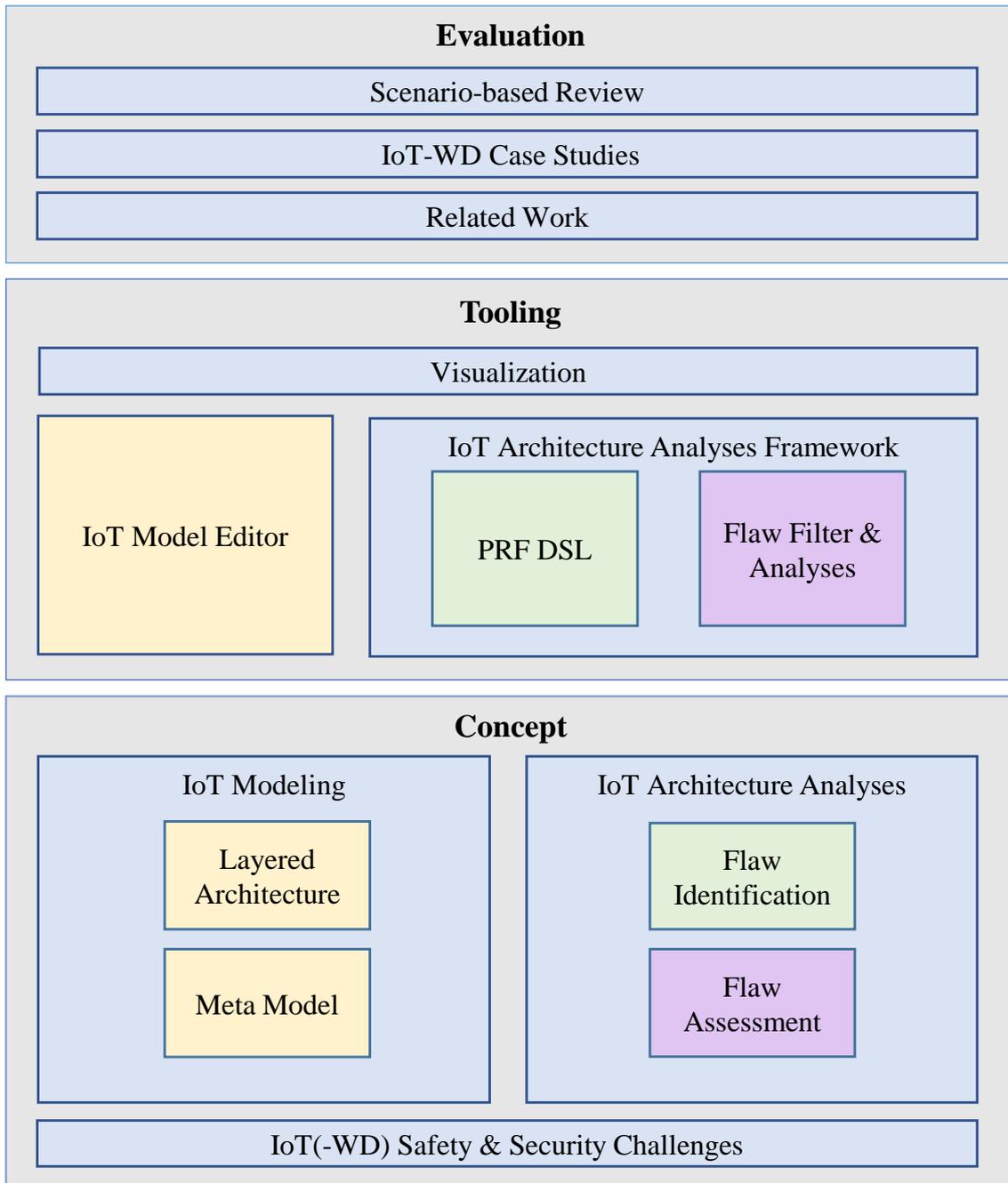


Figure 1.6.: Approach stack

The main part and basis for further implementation and evaluation activities represents the **Concept**. This part is divided into **IoT Modeling** and **IoT Architecture Analyses**. Both sections rely on **IoT-WD Safety and Security Challenges** which are obtained through an in-depth literature review. These challenges are required to identify issues or vulnerabilities of safety- and security-critical use cases especially in wellbeing areas. The characteristics of these issues respectively vulnerabilities can be used to determine architectural requirements for the modeling part and influences the selection of suitable and needed analysis functions. **IoT Modeling** includes a *Meta Model* developed to enable a method to depict IoT systems including safety and security details. Every element of these models requires a layered categorization to assign matching features and to provide information for pattern definitions and analysis activities afterwards, e.g., to chose a suitable countermeasure to mitigate a flaw in a specific layer. Therefore, a *Layered Architecture* with nine layers, including parts for infrastructure, transfer/storage and information, is created. Next to the safety and security design approach of this thesis stands the architecture optimization through **IoT Architecture Analyses**. This part consists of two sections. *Flaw Identification* enables through pattern and anti-pattern definitions and services the automated detection of design flaws or vulnerabilities during the design phase. Therefore, harmful or dangerous architecture decisions can be prevented while past mistakes are not repeated. The *Flaw Assessment* covers the subsequent handling of identified design issues by providing an architecture analyses cycle including four analyses to assess qualitative and quantitative impacts of flaws and to compare required countermeasures to mitigate attacks or accidents.

Each concept part includes an associated **Tooling**. Again, colors are oriented on associated *Objectives*. The described *Meta Model* and *Layered Architecture* are supported through an *IoT Model Editor* that enables to depict an IoT system in layers or containers. Thereupon, an **IoT Architecture Analyses Framework** covers implementation for **IoT Architecture Analyses** to apply the pattern or anti-pattern identification through the *PRF DSL* and the subsequent assessment steps by *Flaw Filter & Analyses*. All **Tooling** parts are united in a tool called ArchiAna which provides the visualization of models and all analyses features.

The **Evaluation** completes this approach roundup and follows the **Concept** and its included **Tooling**. To check several aspects of above mentioned parts there are three sections. **Related Work** reviews the State-of-the-Art of related work and checks whether the approach of this thesis covers respectively replace and mainly extends these other approaches. This is to ensure that the promised generic modeling and analysis approach does not omit any aspects. The **IoT-WD Case Studies** demonstrate the successful application of the approach workflow including all features and the seamless transitions between these. The last method is a **Scenario-based Review** to check several quality characteristics like adaptability or scalability to evaluate how rigid the approach is and how prepared it is for future needs for change.

1.5. Thesis Outline

This section presents a bottom-up stacked outline of all chapters of this thesis. The overview is depicted in figure 1.7 and gives an overview of the chapter distribution of the concept parts. After the introductory chapter 1, the chapter 2 presents foundations for this thesis and can be skipped by readers who are familiar with the concepts of connected systems, safety & security, architectural terminology and Enterprise Architecture Management (EAM). The main concept of this thesis is presented in chapters 3 - 7 and should be read consecutively as they build on each other. Thereby, the thesis is structured in the safety and security design approach, with chapter 3 and chapter 4, and in the architecture optimization including the optimization process (chapter 5) and its architecture analyses approaches of chapter 6 and chapter 7. Chapters 4, 6 and 7 start with the relevant approach, followed by the implemented tooling and conclude with suitable related work. Chapter 8 includes evaluation results of the main chapters. This thesis closes with a summary respectively the revision of *Objectives* (chapter 9) and future work (chapter 10).

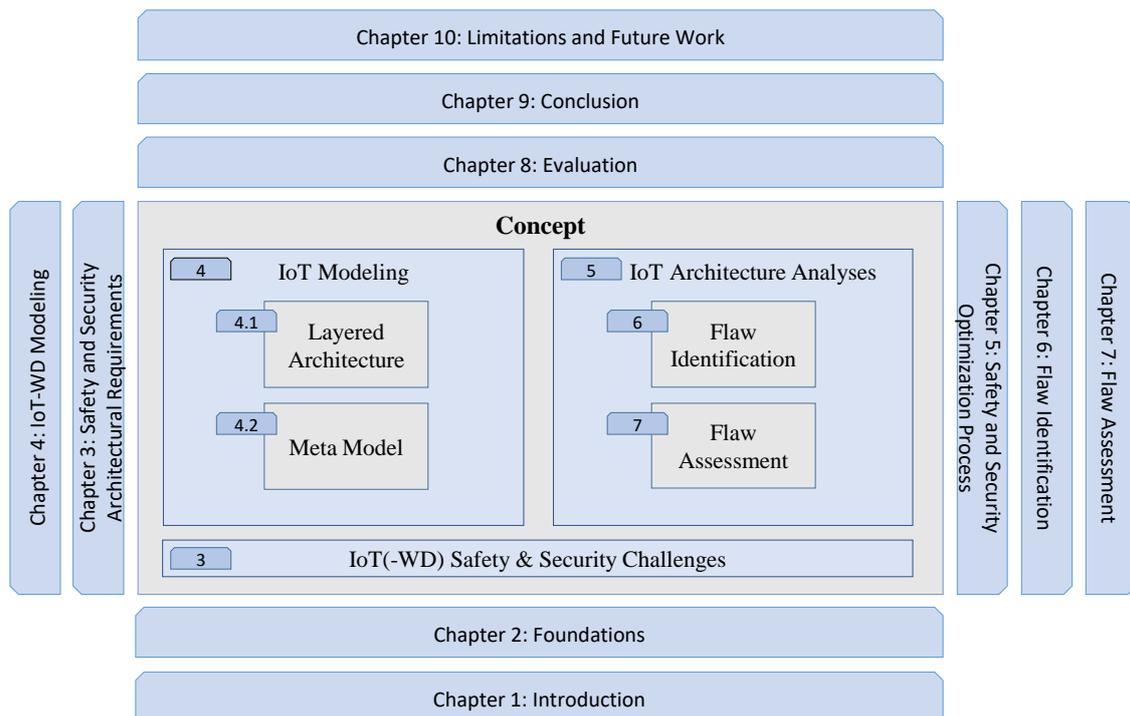


Figure 1.7.: Outline of this thesis

Chapter 1: Introduction

This thesis starts by presenting the main IoT progresses and their associated challenges. Based on this motivation, the *Objectives* and resulting approaches are introduced. In addition, the publications, which already have been published parts of this thesis, are summarized.

Chapter 2: Foundations

This chapter covers the necessary basics for the approaches of this thesis. Therefore, connected systems, including IoT and the application in wellbeing areas, and safety respectively security requirements are considered. As the focus is on "by Design" approaches this chapter includes architectural terminology and EAM.

Chapter 3: Safety and Security Architectural Requirements

As the approaches of chapters 4 - 7 are aiming in safety and security aspects, the architectural requirements must be determined. Therefore, hazards and threats of IoT and particular IoT-WD are explored to categorize the challenges and to derive the requirements to enable modeling of safety and security aspects and identification respectively assessment of the same.

Chapter 4: IoT-WD Modeling

As discussed, the non-existence of standardized IoT architectures blocks unified manners to mitigate flaws as early as possible (*Objective 1*). Chapter 4 presents the approach of a unified IoT(-WD) meta model that is conform to the determined architectural requirements and enable the modeling of generic or specific IoT use cases of the wellbeing area. This chapter offers a detailed specification, limitations and the implemented modeling editor of the ArchiAna tool. The chapter is closed by a related work section about comparable reference architectures and tools.

Chapter 5: Optimization Process

This chapter acts as a link between the design approach (*Objective 1*) and the architectural optimization process (*Objectives 2-4*). A detailed process outline describes required steps to connect the *Objectives* and which steps can be repeated to receive iteratively an improved model. Needed improvement requirements are discussed as well as connecting factors to design analysis interface options.

Chapter 6: Flaw Identification

The first step of the optimization process deals with *Objective 2* and its connected *Challenge* of early flaw identification during the design phase. Therefore, a PRF is developed to enable a structured way to check the model design for vulnerabilities through the definition and usage of patterns and anti-patterns. In addition, the framework offers knowledge conservation capabilities for users regardless of their programming skills. The pattern definition approach is divided into four specifications: *generic, safety and security challenges, safety and security assessment and implementation*. Afterwards, a pattern and anti-pattern DSL is created and a service generation is conducted. To use the services for flaw identification activities, new patterns or pre-defined patterns, stored in a pattern database, can be loaded into the ArchiAna tool to analyze models that are conform to the meta model of chapter 4. A related work section differentiates the approach from other pattern concepts.

Chapter 7: Flaw Assessment

The last chapter of the main part covers *Objectives 3 and 4* by developing an analysis cycle. As architecture analyses are already used successfully in other application fields, existing approaches are categorized and investigated on transferable analyses. Selected approaches must be adapted and brought into a cycle to create a structured manner for assessment of IoT design flaws. Four analyses which are usable for this specific purpose are presented including their ArchiAna tool integration. The goals of these analyses can be divided into flaw impact assessment and design decision support to mitigate assessed flaws. In addition, the analyses can be differentiated by required pre-steps, result types and the associated mission. The chapter is closed by a related work section.

Chapter 8: Evaluation

In order to prove that the approach of this dissertation brings the claimed added value and performs successfully, the main sections are followed by an evaluation. This chapter is segmented as described in the methodology section.

Chapter 9: Conclusion

A concluding chapter summarizes the approach with its IoT modeling, flaw identification and assessment parts. To finalize this thesis, the objectives defined at the beginning are taken up and verified.

Chapter 10: Limitations and Future Work

This thesis closes with approach limitations and possible or planned future work approaches to adapt, change or extend this thesis and its developed tool.

1.6. Publications

During the development of this dissertation, parts of the approach have already been published at conferences or project deliverables. Following subsections present summaries of the publications produced and mark the chapters in which parts of them are used. The afterwards mentioned concepts of the author are not cited again in the associated chapters.

1.6.1. Scientific Publications

Following publications were published in a scientific manner and where produced with several authors. Selective text passages are used during this dissertation.

Titel: *"Characteristics of Enterprise Architecture Analyses"*

Authors: Julia Rauscher, Melanie Langermeier and Bernhard Bauer

Conference: 6th International Symposium on Business Modeling and Software Design (BMSD), 2016

Summary: The author of this thesis and two co-authors conducted a literature review to evaluate Enterprise Architecture (EA) analysis approaches. To enable a classification of these, requirements were defined. A two-dimensional classification approach is developed that allowed division of different technical and functional categories. Therefore, analyses could be sorted by their procedures respectively techniques or by their goals respectively outcomes. A DSL is developed to formalize and evaluate defined characteristics. The designed approach is used in this thesis to identify existing EA analyses and to enable a justified selection for adaptable approaches for the development of IoT architecture analyses. The developed DSL is embedded in the flaw identification of this thesis. The author was also the author of the corresponding bachelor thesis ([Rau13]) and master thesis ([Rau15]) for this paper. Therefore, the main contribution of this paper belongs to the first author.

Reference: [RBL16]

Text passages to be found in: Chapters 2 and 7

Titel: *"Classification and Definition of an Enterprise Architecture Analyses Language"*

Authors: Julia Rauscher, Melanie Langermeier and Bernhard Bauer

Conference: 6th International Symposium on Business Modeling and Software Design (BMSD), 2016, Extended Version

Summary: This publication was written by the first author, as the main contributor, and her co-authors as an extended version of [RBL16]. The approach was

mainly developed by the first author. The DSL was advanced to create a generic analysis language to enable a structured way to describe architecture analyses and their steps. This approach aims in getting easy access to goals and execution environments of classified architecture analyses. Thus, the choice and execution of the most appropriate analysis can be performed. This extended version includes an update of the literature review.

Reference: [RLB16]

Text passages to be found in: Chapters 2 and 7

Titel: *"Smart Data Integration within a Wellbeing Application Platform"*

Authors: Julia Rauscher and Bernhard Bauer

Conference: 4th IEEE International Conference on Biomedical and Health Informatics (BHI), 2017

Summary: The author of this thesis and her supervisor published this position paper to discuss development of big data in medical or wellbeing fields and parallel growth of IoT. A symbiotic pairing of both concepts was proposed by introducing a Wellbeing Application Platform (WAP) to create a possibility to merge multi-sensor data aiming smart data and user-optimized wellbeing recommendations. This concept is not the main focus of this thesis, however, the concepts of this thesis represent the basis that has to be created to enable save development of future WAPs. Thus, this publication is part of the motivation and basics. The first named author has been responsible for the main part of the content.

Reference: [RB17]

Text passages to be found in: Chapter 2

Titel: *"Generic Sensor Framework enabling Personalized Healthcare"*

Authors: Sven Beckmann, Stefanie Lahmer, Moritz Markgraf, Oliver Meindl, Julia Rauscher, Christian Regal, Henner Gimpel and Bernhard Bauer

Conference: IEEE Life Sciences Conference, 2017

Summary: The author of this thesis was the supervisor of the corresponding group seminar paper. Therefore, the first four authors have made the main contribution, while the author of this thesis has acted as an advisory member. This publication focused on the challenge of handling the enormous amount of different IoT devices and sensors. As every provider uses various techniques to collect, save and process data a generic sensor framework was developed which enables personalized healthcare recommendations based on the individual chosen sensors. This publication did not include concepts of this thesis. However, was used to describe sensor frameworks in the basic sections.

Reference: [Bec+17]

Text passages to be found in: Chapter 2

Titel: *"Safety and Security Architecture Analyses Framework for Internet of Things of medical Devices"*

Authors: Julia Rauscher and Bernhard Bauer

Conference: 20th International Conference on e-Health Networking, Applications and Services (Healthcom), 2018

Summary: This publication presented the first concept version of the first author's approach of this thesis including all parts described in chapters 4 - 7. The risks of combining IoT and medical data were discussed and the application fields of this thesis, including new IoT systems, existing IoT systems or IoT systems with a need of change, were presented. However, approach details were not described at this point respectively have changed to the finale approach version presented in this thesis. This publication was cited in multiple papers.

Reference: [RB18]

Text passages to be found in: Abstract, Chapters 4 and 7

Titel: *"Failure and Change Impact Analysis for Safety-Critical Systems - Applied on a Medical Use Case"*

Authors: Philipp Lohmüller, Julia Rauscher and Bernhard Bauer

Conference: 9th International Symposium on Business Modeling and Software Design (BMSD), 2019

Summary: The author of this thesis and two co-authors collaborated to develop a workflow to combine change impact and failure impact analyses to create save systems in safety-critical areas. The approach includes a hierarchical structured model to determine critical goals which are endangered through failures. These failures' effects and required countermeasures are analyzed. The workflow ends by analyzing changes caused by new countermeasures. The author of this thesis was the main contributor for paper sections concerning failure impacts which are part of this thesis.

Reference: [LRB19]

Text passages to be found in: Chapters 2 and 7

Titel: *"Design Optimization of IoT Models: Structured Safety and Security Flaw Identification"*

Authors: Julia Rauscher and Bernhard Bauer

Conference: 10th International Symposium on Business Modeling and Software Design (BMSD), 2020

Summary: This publication is dedicated to the first step of the optimization process of this thesis. The authors discuss the need of flaw identification during the

design phase especially in safety- and security-critical areas. As most of the time systems are not manual controllable, a structured possibility to scan the models is required. Therefore, the publication describes details about the PRF and the application of patterns and anti-patterns to enable the first part of a holistic and automated design optimization. The first named author has been responsible for the main part of the content.

Reference: [RB20]

Text passages to be found in: Chapter 1 and 6

Titel: *"Adaptation of Architecture Analyses: An IoT Safety and Security Flaw Assessment Approach"*

Authors: Julia Rauscher and Bernhard Bauer

Conference: 14th International Joint Conference on Biomedical Engineering Systems and Technologies (HealthINF), 2021

Summary: The authors published this paper to describe the second part of the optimization process of this thesis. Therefore, this paper can be considered as a succession paper of [RB20] and consists mainly of research results from the first author. This publication focuses on the fact that most analysis approaches of IoT assessment analyses are conducted during run time. As the focus is on assessment in the early stages architectural assessment analyses were developed and presented. The assessment cycle of this thesis is described including the holistic flaw assessment steps and the identification of possible countermeasures and its services.

Reference: [RB21]

Text passages to be found in: Chapter 1 and 7

1.6.2. Project Deliverables

Parts of the author's concepts were published in deliverables with multiple project partners. No text passages of these deliverables are used during this dissertation.

Project: CPS4EU - (Grant Agreement Number 826276) [CPS]

Work Package: WP1

Deliverable: D1.1, D1.9

Summary: Among others, the author of this thesis and a co-author published and described a meta model in these deliverables. The elements, relations and attributes were presented in detail. The content is based on the meta model of chapter 4 of this thesis, but is not identical. However, the same concepts are used and aiming in the same flaw identification and assessment process as in chapters

6 and 7 described in this thesis. The results of the meta model are divided equally among the authors.

Project: CPS4EU - (Grant Agreement Number 826276) [CPS]

Work Package: WP5

Deliverable: D5.1, D5.2, D5.3, D5.4

Summary: For this project, the author of this thesis developed a similar optimization process, as described in chapter 5 of this thesis. Some sections of these deliverables were dedicated to this approach. The approach cycle with its steps, diverse calculation rules and goal types are congruent to the approach presented in this thesis. However, adaptations had to be performed to be conform with the changed meta model approach of WP1. In addition, the approach is adjusted to enable the generic application with non-wellbeing use cases.

Project: CPS4EU - (Grant Agreement Number 826276) [CPS]

Work Package: WP8

Deliverable: D8.1, D8.3, D8.5

Summary: The results of this work package aimed at finding use cases and collaborations for approaches of other work packages. Therefore, the author of this thesis and a co-author described the application of selected WP1 and WP5 contents, which mainly can be find in chapters 4, 6 and 7 of this thesis, in a mobile lifting use case. The application in other use cases enabled the Scenario-based evaluation. The results of this deliverable are divided equally among the authors.

2

Foundations

This chapter provides an overview of the most important fundamentals that are relevant for the approaches in the following chapters. It focuses on the components that have a profound impact and contribute to an understanding of the issues. This chapter can be used for referencing.

The basics are divided into four areas. In order to create a uniform understanding, the area of architecture and its activities respectively processes within the associated phase is presented. Further foundations include a definition of IoT and general principles of connected systems and their intended use. In addition, the considered requirements of the *Objectives* are discussed, focusing on safety, security and usage in critical systems. Finally, various concepts of EAM are considered to show the commonalities of IoT and EAM, as this is needed in the course of this work.

2.1. Architecture Terminology

The key development phase for *Objectives* of this work is design phase and its associated architectural expressions and concepts. For a common understanding, the most important terms are defined below.

2.1.1. Architectures and Reference Architectures

The term **architecture** is used in multiple fields and requires a generic definition. A frequently used description originates from ISO/IEC/IEEE 42010:2011. They define architectures as

"fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution." [ISO11b]

Software and System Architecture

Different types of architectures exist within a company. The most important of these are EA, software and system architectures. A system architecture is responsible for representing structures, components and relationships of a system, as it is configured with hardware, software, data, humans, processes and plenty more [ISO15]. In the field of software engineering, if the system is a software component or a software system, it is called a software architecture. Another term, IT architecture, is used when all software and hardware components are considered. [DN02]

Elements

Architectures comprise of components, called elements, which construct a subject of a system. They have specific attributes, behavior and relationships between each other. [Eng18]

Layer

Elements of an architecture are located in layers which are used as horizontally grouping modules. They split code and components in logical units to separate concerns, distribute responsibility and master complexity. In most architectures, layers only have knowledge about their direct lower layers. [Eng18]

Relations

Relations present the dependencies and influences of elements with each other. They can be distinguished between static and dynamic relations, depending on

the behavior of connected elements. [Eng18] In addition, relations present technical dependencies but also logical impacts.

Architecture Description

To express an architecture a description of all containing work products of a system is created to document its mission which should be fulfilled in its environment. These missions are hold by stakeholders that have interests in different concerns and associated objectives of systems. Depending on the stakeholders to whom the description is directed to, a selection of viewpoints are used. [Com00] Stakeholders can be, e.g. users of the system, operators, suppliers or maintainers. [ISO11b] These different descriptions have diverse application fields. They are often used for analyses, business or budget planning, feature specification and communication at contract negotiations. [Com00]

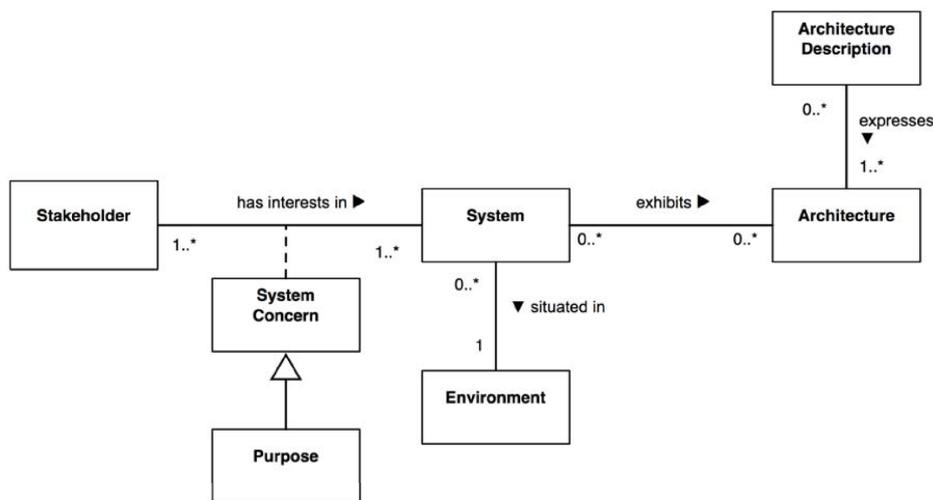


Figure 2.1.: Context of architecture description [ISO11b]

Stakeholder and Concerns

A Stakeholder is a group of people or an entity who has interest or concerns about the system, enterprise or generically architecture. [Eng18] Concerns represent any kind of influence on a system including technological, business, operational, organizational or social influences. [ISO11b]

Views, Viewpoints and Perspectives

Viewpoints and perspectives are both architecture insights. On the one hand, perspectives are collections of activities and guidelines that enable the exhibition of non-functional requirements which are considered by views and their concerns respectively stakeholders. [Eng18] On the other hand, viewpoints establish the purpose and audience for a view [ISO11b]. They contain specifications of templates, conventions and guidelines to create and use diverse views, whereas a view is defined as presentation of a whole system or a structural aspect of an architecture and its associated concerns [Eng18; Com00].

Filter

Incoming data of a source or element in a specific view can be filtered to create a narrowed view or data output. [MT10] These filters are required if selected components have to be considered in detail, and therefore, extract the complexity with views of whole systems. This is solely a matter of presentation and does not change the actual architecture or data set.

Goals and Requirements

A goal can be seen as a future target of a system or stakeholder, which should be achieved. In addition, a distinction is made in the abstraction level of "high-level" strategic goals or technical "low-level" goals. Individual small goals can be seen as requirements and define a goal as a combination of these requirements. Goals aim to achieve the totality of all requirements. [Van01] These requirements can be distinguished in functional and non-functional which represent imposed obligations on systems. Functional requirements define specific conditions of technical solutions that enables the required behavior, services and information. Whereas non-functional requirements define the needed quality of solutions.[Eng18] The considered requirements of this thesis are defined in section 2.3.

Reference Architecture

Reference architectures are often used in application areas that can involve a wide variety of architectures and for which it is difficult to find a uniform specification. [Fre15] A reference architecture can be defined as collection of

"recommended structures and integrations of IT products and services to form a solution. The reference architecture embodies accepted industry best practices, typically suggesting the optimal delivery method for specific technologies." [Ent]

They are designed to help unite the various stakeholders from the outset and facilitate consistent communication. [Ent] Therefore, they can be applied as constraints and a highly abstract template for future architectures.

Model and Meta Model

In order to depict and abstract the described architectures and their components, models and corresponding meta models are used. Depending on the intended use, a suitable model is selected that can display required information. [JNL07] Each of these models is based on a meta model. This represents a superordinate model that defines properties and determines the relationships and structure of individual model parts. [TW14] They can be used to define new languages or new properties. Therefore, meta models are an abstraction of models which in turn can be defined as instances of abstract models. Any number of meta levels can be defined above it to increase abstraction. [BCW17]

Modeling Language and Domain Specific Language

In general, modeling languages are a way to implement specific representation of a system and to obtain an abstract representation of the properties of a meta model. The modeling languages can be distinguished between graphical and textual. Textual languages include advantages that no notation has to be developed, it is easy to read, and thus, requires no training. With the modeling in the representation is to be distinguished additionally between abstract and concrete syntax. Both can also be represented textually. If a language is needed for a certain domain or context a DSL can be designed specifically. [BCW17] Depending on the context, a DSL may include various requirements or constraints and may contain its own appropriate syntax. [FHK09]

2.1.2. Design Phase

Regardless of whether an entire system is produced or only a software component is renewed, the individual steps from idea to finished solution are divided into stages or phases. However, there are different approaches how these can be structured depending on desired objectives. Two classic models are the waterfall model and V-model. [BM12]

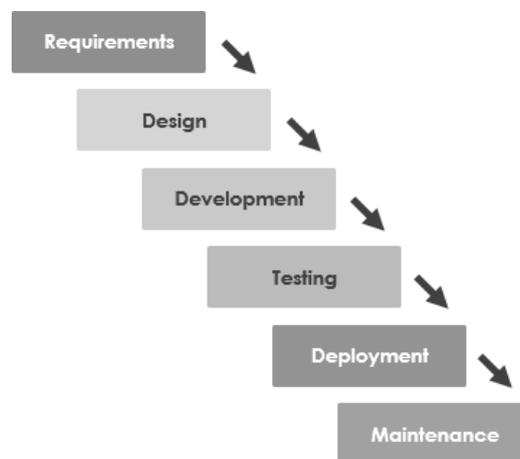


Figure 2.2.: Waterfall model of project activities [Par]

The waterfall model (figure 2.2) can be seen as a sequential development model that is inflexible and non-iterative. [Ins18] It starts with the identification of the requirements before it reaches the design and implementation and the subsequent testing. It is concluded with deployment and possible maintenance. The stages depend on deliverables of the previous phase and always have specialized tasks. [Par] For instance, requirements must be clearly defined and approved before the design phase is started. They cannot be changed later. Only if blueprints of design solutions are fully completed an implementation task can be started. Tests, for example, are specified after the development stage. [BM12]

In contrast, the V-model is divided into two cycles that are connected to each other, because developer and tester work in parallel. Validation and Verification model bends down like a V and is an extension of the waterfall model. [BM12] Accordingly, this model is not linear in its two cycles. The axes represent timeline and degree of project completeness. In the beginning, different types of requirements are defined. These are checked directly with tests. Afterwards, a low and high level is defined, which involves corresponding tests. Transition is coding stage tasks. [Par] Therefore, the main difference to the waterfall model is that all requirements and design proposals are directly reviewed before development starts. [Ins18]

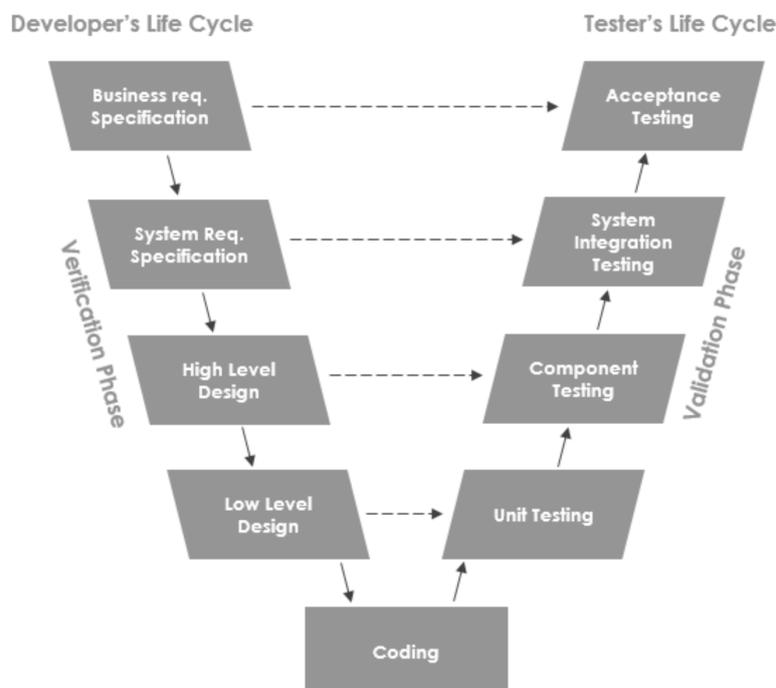


Figure 2.3.: V-model of development process [Par]

New agile methods are in complete contrast to these two models. These feature an adaptive team that is designed to constantly respond to new and changing requirements. Thus, initial requirements or designs do not need to be fully completed. Changes are allowed and desired. Therefore, the typical stages are mixed and repeated if necessary. [BM12]

Even if the stages of different development models are arranged or connected in a different order, the phases involved are always roughly the same. Often, however, not enough attention is paid to the design phase. Although 50% of weaknesses occur in this phase ([VM01]) which makes it one of the most relevant stages. [Inf] This dissertation focuses on design phase.

This phase represents the process of creating underlying basis for code. This process can only be run once stakeholder requirements have been defined and

detailed analyses have been run, as these are needed to select appropriate design for a new system or system parts, depending on project timelines, required use case tasks, resources and available technologies. Therefore, these analyses help to create logical and physical system designs, and thus, determine a manner in which the required can be achieved. Hence, main tasks of design phase are transformation from the desired into detailed specifications that include all aspects of the system. In addition, safety and security risks can be included. [Inf] The requirements identified and associated architectures are described in a system design document which can have diverse formats. This should specify system designs precisely and is used as input and guidelines for next phases. In most cases, several architectures are designed first and afterwards the one that fits the goals best is selected. [Opeb] Thus, it is used as an approval to proceed with next stages. Furthermore, the document facilitates the knowledge transfer, functions as a reminder of the specified tasks and helps with a common understanding for all stakeholders. [Inf]

Flaw vs. Bug

In contrast to the implementation phase, the design phase does not search for bugs, but for so-called flaws. These are faults, weaknesses, vulnerabilities or general system defects which occur during design phase. These have impact on all subsequent stages and tasks. Since flaws cannot currently be searched for automatically, code bugs are being monitored more frequently. As a result, some simple but fatal errors are overlooked, e.g. planning inclusion of authentication. [DM13] Further descriptions of design faults are described in section 2.3.

2.1.3. Architecture Analyses

Architectural analyses can have a wide variety of goals and come in different types. In general, architectural analyses are procedures for collecting and evaluating data and system properties, using As-Is or To-Be architectures, in order to subsequently interpret these. Depending on the type of analysis certain aspects are emphasized or additionally considered. [Lan05; MT10]

The main common feature of all architectural analyses is that they are performed in the design phase or are based on it. However, they differ in many aspects, such as their goal or concerns, stakeholders, required model types, their scope, such as data or processes, and the automation level. [MT10] In addition, they can be subdivided in terms of their techniques and leading questions. Accordingly, there are inductive bottom-up and deductive top-down approaches, depending on whether an architecture is to be analyzed from above or from below, and thus, whether effects or causes are to be determined. [PV08] For this purpose, various analysis activities are used during execution. These include system thinking to analyze issues, modeling, measuring, satisfying tradeoffs, and com-

paring requirements or alternatives. [WC12] However, most common is evaluation of quality of the architecture. The needs and concerns of the stakeholders are checked to see if they are being met. This includes consistency, completeness and correctness. In addition, the quality of the system can be analyzed for feasibility, efficiency or reliability. Reverse engineering can be used to check the mentioned architectural description. [Com00]

Measurements and Metrics

To check the quality of the architecture, measurements or other comparable sets of values are collected. Metrics are needed to evaluate this information. Metrics should objectively compare the values to enable an interpretation of the quality. [Shi+08] Measurable and comparable metrics for architectural quality can be, for example, understandability, flexibility, reusability, service granularity or structural attributes such as coupling and cohesion. [Ula17] Coupling is a metric to estimate dependencies of individual services or of a whole system. Aim is to achieve a value that is as low as possible. Reusability and understandability are strongly connected with this. In comparison, cohesion can be used to determine how tightly parts belong together. This again influences flexibility. By means of such metrics, the quality of the future system can be estimated at an early stage, in the design phase, on the basis of planned architectures. [Shi+08]

2.1.4. Architectural Pattern

A pattern is intended to identify specific (design) structures and properties of an architecture to aid in creation and optimization of a (system) design. Design patterns identify classes, attributes, roles and dependencies. A pattern can be understood as an implementation-independent template, and can therefore, be reused or adapted. Accordingly, they can be specialized in different use cases or applied generically. Patterns are formed to identify already known or existing problems and to avoid repetition. [Gam95] Accordingly, the focus of patterns is on the connection between a certain context, a problem and a solution. Even though a pattern template can be reused as often as desired, it must be adapted to these connections. [Ale77] This results in the definition for patterns as follows:

"A pattern for software architecture describes a particular recurring design problem that arises in specific design contexts, and presents a well-proven generic scheme for its solution. The solution scheme is specified by describing its constituent components, their responsibilities and relationships, and the ways in which they collaborate."
[BHS07]

However, this definition is related to software architecture patterns. Even if the basic principle is the same, design patterns are specifically designed for the de-

sign phase and catch faults that arise during the architecture specification. In addition, they cover not only software problems, but also misuse of hardware or other aspects defined in this phase. This leads to its own special definition for design patterns:

„A design pattern provides a scheme for refining the subsystems or components [...], or the relationships between them. It describes a commonly-recurring structure of communicating components that solves a general design problem within a particular context.“ [Bus+96]

Design patterns are, thus, designed to look for general, known and possibly recurring structures in systems or their components and to refine or improve them as a result. [Bus+96]

Architecture principles provide rough rules for architectures, which in turn is described in detail by design patterns to ensure that these principles are present or have not been violated. These principles are based on a firmly defined maxim. A balance shall be reached between specified requirements and prescribed principles. Typical principles that can be applied to the design are a holistic consistency, avoidance of redundancy, constructional reusability, fulfillment of standards, loose coupling respectively strong cohesion, open extensibility options and including a way of unique identification. [Eng18]

For this dissertation, design patterns are defined as

a combination of logical and graphical patterns that can be used not only as a template for graphical inconsistencies, but also to verify logical dependencies in the design.

In addition to patterns, anti-patterns are determined to cover the *Objectives*. Anti-patterns help to ensure that known flaws are not repeated [Win+09]. Patterns represent positive and desirable design choices that prevent vulnerabilities, e.g. a highly required authentication mechanism. By contrast, anti-patterns represent negative and avoidable design choices that cause vulnerabilities. Therefore, the absence of a pattern indicates a vulnerability, while the presence of an anti-pattern indicates a problem. Since the focus of this dissertation is on safety and security, these patterns and anti-patterns can be considered as safety and security design rules.

2.2. Connected Devices and Systems

The following section presents the basis for systems consisting of interconnected components, features and their applications, especially in wellbeing areas and their devices.

2.2.1. Internet of Things

The definitions of IoT in the literature are as varied as the application areas. Depending on whether one considers the horizontal possibilities - such as platforms and interfaces - or applications in the various vertical domains - e.g. industrial IoT, enterprises or home use - the building blocks such as software, hardware or their connectivity are defined differently. [CMR14] In order to determine a suitable definition, characteristics and challenges have to be considered in more detail.

One of the most important and obvious characteristics of IoT networks is the high degree of distribution of components and contributing systems or subsystems that are physically separated and remotely located. They are connected through different connectivity methods and communication channels or protocols. The ability to connect to the Internet, directly or indirectly via gateways, is the center of the IoT. This ability to exchange information units, enables objects to communicate with each other and to send and receive data. Different protocols or gateways make it possible to connect different types of networks. The heterogeneity of components and systems must always be taken into account to enable their interoperability. [ISO16b; Dig17]

Another characteristic of IoT is its dynamic nature. On the one hand, the many subsystems and stakeholders mean that new components can be added or others removed at any time. This creates a high degree of uncertainty, which can lead to hidden costs and security gaps. [Fu+17] On the other hand, the flexibility of IoT systems also includes the states of its components. IoT systems are known for their context awareness, and thus, for their sensors that monitor the environment. These sensors can change their location, speed or connection state - wake/sleep or connected/disconnected - and many more attributes. The measured data are either used for immediate action by actuators or are forwarded, processed and used with the help of data management. All these actions and events are executed with support of distribute services. [ISO16b; Dig17]

These characteristics lead to major challenges for scalability and relate to a cost and energy efficient network management [Mar21]. This is accompanied by the challenges of safety, security and reliability, which are attempted to be covered by developed standards and regulations [ISO16b; Dig17]. All mentioned characteristics are considered in the following sections and subsections.

Following these characteristics, based on [Int] and [Dou12] IoT is:

a dynamic global network infrastructure, linking physical and virtual objects using cloud computing, data capture and network communications, with self-configuring capabilities based on standard and interoperable communication protocols where physical and virtual “things” have identities, physical attributes, and virtual personalities and use intelligent interfaces to communicate with each other.

The mentioned virtual personalities, also called digital twins, are a digital representation of IoT devices. They can be used for simulations, apps or services, or to synchronize physical activities with the virtual world. A distinction has to be made between a real IoT device - thing - and a normal hardware IoT component. Only true IoT devices have a digital twin. [Tao+19]

To be an IoT thing, various properties have to be fulfilled. If these are not present, it is for example a non-smart hardware component, low power device or IoT hub. The following characteristics can be abstracted from [ISO16b] and [SS17]:

- **Network connectivity** for communication and transfer through the Internet
- **Unique identification** to recognize things
- **Discoverability** as only detected things can communicate
- **Auto-configuration respectively autonomous actions** of things or events of connected actuators without need of humans
- **Context awareness** to measure or collect autonomously its environment
- **Shareability** to share data across Internet, where it can be processed or utilize, and exchange data autonomously

In summary, IoT devices can be defined as follows:

IoT "things" are defined as objects which have unique identifiers, are able to connect to the Internet and other devices which enables them to communicate and share data across the Internet, where it can be processed or utilized. They have to be autonomic to be able to measure, collect and exchange data and transfer data over a network, without need of humans. Objects with this characteristics, but without the ability to connect are defined as low power devices.

2.2.1.1. Reference Architectures of IoT Systems

As defined above, a reference architecture is intended to recommend a rough structure for building layers, components and services. Best practices are often used in this process. For IoT, there is currently no uniform structure or architectural guidelines. Accordingly, diverse reference architectures were designed. Famous ones are the Azure project [Mic16], IoT-A [Pro15], RAMI 4.0 [Pla16] and IIoT [Con+16].

A frequently applied, but also highly generic reference architecture offers [Fre15]. It is divided into 5 horizontal layers and 2 cross-cutting layers. The device layer, which contains devices that correspond to the typical characteristics of IoT devices, serves as the basis. Accordingly, they must be uniquely identifiable and connected directly to the Internet, or indirectly via special protocols or gateways, such as ZigBee gateways or Arduino Ethernet connections. Above this, the communication layer is located, which is responsible for management of the various communication protocols, and thus, for the connectivity of devices. To aggregate and broker the communication, the aggregation layer is needed. Here, the communication of servers and brokers to interact with the devices is regulated and the different communication channels of the different devices are unified and distributed. This is possible through the bridge and transformation methods for different communication protocols that this layer provides. The event processing and analytics layer is located above and can store and process the data. The last horizontal layer is the external communication layer for interacting with systems outside the IoT network. The two cross-layers are responsible for device, identity and access management, and thus, regulate, for example, the software, security measures and access rights of devices. [Fre15]

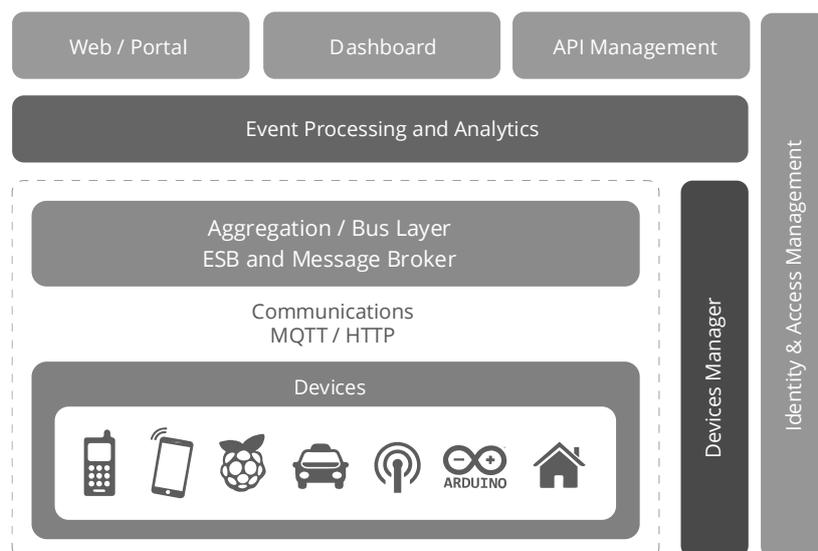


Figure 2.4.: IoT reference architecture [Fre15]

The other reference architectures may differ significantly from this approach. However, the broad contents and functionalities are the same. Devices, or the so-called things, are always the foundation of IoT networks. Same is valid for communication channels and types of data storage and processing. Accordingly, they all use similar layered approaches. They differ in their fine granularity, but could mostly be transformed into other approaches. Figure 2.5 shows three rough layers with an example how to divide these into more specific layers. A classification into more layers can bring advantages, e.g. in analyses as a vulnerability can be located exactly or costs can be split up more accurately. In addition, special application fields sometimes require further intermediary layers if they differ from classical IoT networks. This is considered in more detail in chapter 4.

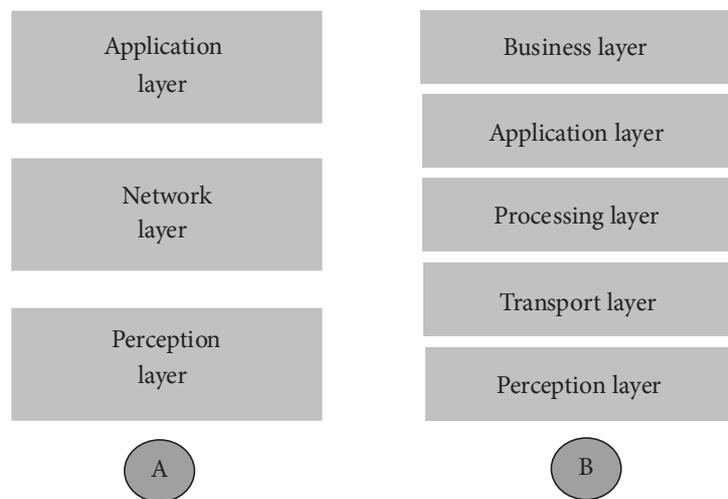


Figure 2.5.: IoT reference architecture [SS17]

Besides the similarities, the various reference architectures have differences. Azure is widely deployed and brings with it many services suitable for IoT. The reference architecture is divided into things, insights and actions. The focus here is not on a layered architecture, but on device management, communication paths such as gateways, and their storage and processing paths. [Mic16] Two reference architectures for IoT in industry provide [Pla16] and [Con+16]. While RAMI 4.0 ([Pla16]) focuses on lifecycle value streams and hierarchy levels, the IIoT ([Con+16]) approach focuses on different stakeholders, viewpoints and model concerns. Aligned with the architectural view of IoT systems are the reference models of [Bau+13] or [ISO16b] and the standard of [Thi19]. This standard is based on [ISO11b] and presents a blueprint for architectural building blocks with connections to multi-tiered systems. The architectural framework defines commonalities of diverse IoT domains and their connection possibilities through a reference model. IoT-A ([Bau+13]) and ISO/IEC 30141 ([ISO16b]) present reference models, that similarly highlight interoperability of IoT domain connections and their requirements. In addition, connected security, communication and functional reference models are included. Further details can be found in section 4.3.

2.2.1.2. Wireless and Embedded Systems

IoT systems are based on several existing structures, methods and functionalities. These mainly include embedded systems, but also Wireless Sensor Network (WSN).

The components of IoT networks are built on embedded devices and systems. These systems consist of small components with hardware and software that are designed for specific, individual tasks, e.g. sensing or acting small jobs, and have hardly any update options. Therefore, subsequent changes are hardly possible. [Spa17] Embedded devices are not directly writable and require a built-in circuit board, micro controller, sensors and connection to a computer to write Assembly code on them, for example. They are connected to input/output devices, storages and an A/D converter. Since they are low cost, low memory and low power devices, they are ideal components of IoT networks. However, to make them an IoT device, they need additional network capabilities to communicate and allow update options. [Mar21] Through Internet connectivity and collaboration with cloud functions, IoT devices can add new functions and perform more complex tasks. However, this property makes them more vulnerable and insecure than classic embedded devices. [Spa17]

However, not all devices or components can be clearly classified. For example, there are different points of view whether a smartphone is an IoT device or more likely an IoT hub, and is therefore, not used for classic data gathering but for data transmission. Smartphones are IP-enabled devices but they need human interaction to perform. Classic IoT devices are supposed to act completely autonomously. However, wearables are counted as IoT devices. It was discussed that they are a superset of machine-to-machine interactions, and can therefore, be influenced by humans, but measure and send data autonomously even if there is no human interaction. [Duf14]

Since these embedded IoT devices are mainly dependent on embedded sensors, WSN and Body Sensor Networks (BSN) are important foundations for IoT systems as well. WSN are self-organized multihop networks composed of large number of sensors whose communication stack includes medium access control (MAC), routing and transport layers. [Sta08] They are connected to the Internet to transmit measured data for processing in back-end services at various locations. There are differences in the processing time, detail depth or analysis reasons. Different data centers can receive the data depending on their application and pass it on to the appropriate data processing resources. In addition, the back-end can be used to control which users have access and control. To make this possible, sensors and components which exist in WSNs must support wireless communication and have to be IP-enabled or can use gateways to transfer their data. [Tok+17]

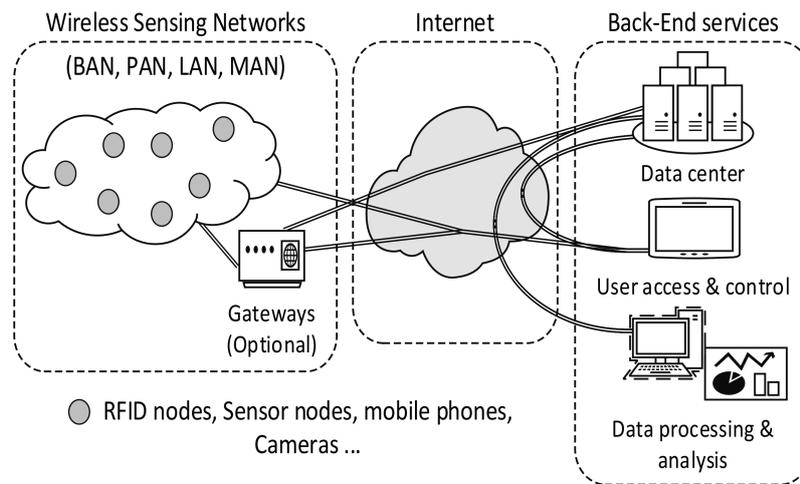


Figure 2.6.: IoT system architecture with WSN [Tok+17]

However, the wireless communication and small sensors are very error-prone. Gateways can help to intercept attacks [Sta08]. A gateway is a hardware or software component that receives and forwards data, and thus, connects different systems with each other. Gateways are possible on a wide variety of layers and protocols. [BJ17] Smart gateways can process or filter data themselves or check for security breaches. [SS17] A distinction can be made between field gateway and cloud gateway, for example. A field gateway enables communication and does local preprocessing, while cloud gateways takes care of the entire transmission routing, protection of the communication path, device authentication and receives data from different edges. [Mic18a]

Typical components include sensors, RFID nodes and mobile devices. These components can communicate and interact with each other [Tok+17]. Sensors consist of a microcontroller for computation, a power source, a small RAM for dynamic data, a wireless transceiver, an antenna, converter and a sensor unit. The sensor unit measures physical values, the controller processes them, and the transceiver transmits them. These components are subject to constraints such as limited energy, memory capacity and processing power. Accordingly, these ad-hoc networks are designed to save power by having the sensors wake up, join the network, and then leave when not needed. [VCS06; Sta08] This characteristic causes, among other things, uncertainties. Besides sensors, RFID or Near Field Communication (NFC) are one of the most common technologies in WSN or IoT networks. RFID is used for identification of hardware components using radio waves. They consist of an identifier or tag including an antenna, a reader and a data handling system. The tag chip can be read and written with help of the reader. No line of sight is required for this. NFC is based on RFID technology and enables short-range wireless information exchange, within 0-20 cm. This short distance makes eavesdropping and reading unsecured tags difficult. Accordingly, NFC is more secure than RFID. [LHJ08]

As seen in figure 2.6 WSN are divided into subnetworks depending on their range and purpose [Tok+17]. BSN were invented mainly for medical purposes and are sensor networks whose nodes are located directly on or are implanted in the body. They measure vital parameters. The main difference between them and a WSN is the different locations where they are used. WSNs can include widely distributed sensors without access, using many and sometimes redundant sensors to mitigate node failures. With BSN it is only possible to use a few selected sensors. Furthermore, the sensor types differ. BSN use for example EEG sensors, blood pressure sensors, gyroscopes or pressure sensors. [Lai+13] Personal Area Network (PAN) are a super category of BSN. They include intelligent, physiological BSN sensors, but extend this with sensors that are in the environment around the human and do not need to be worn directly on the body. An example of this can be a fall detector in a telemedical environment. The perimeter is about 10m away from the human. Furthermore, they enable the communication and data transfer of the processed data to gateways or servers. The security aspects, such as encryption, are particularly important in these networks, as personal and medical data are considered and wireless communication is often insecure. [Jov+00]

The high proportion of embedded devices, sensors and wireless communication cause that IoT systems have a significant delta to classic traditional IT systems. Initially, the high degree of autonomy through deploying IoT devices can be mentioned accordingly [Var+17]. In addition, new layer for autonomic, sensing and acting are required. However, this brings with it a new level of risk, since a high degree of sensor accuracy is required, even though small, inexpensive sensors are used that are associated with failures. [Par+16]

The biggest difference to classic enterprise systems are the characteristics of openness and dynamic change of the participating devices. New components can be added at any time, which brings a high degree of uncertainty and unpredictability. [Fu+17] The large number of autonomous participants can cause scalability problems, and therefore, requires automated information retrieval. Unlike traditional business systems, whose business logic is concentrated in a few components, the logic in IoT systems is widely distributed and uses data and information from different sources what leads to diverse analytic subsystems for data processing [Mic18a].

Another delta is the communication of things. A wide variety of network types are linked together and use or combine communication protocols from traditional systems with IoT specific protocols. [SS17] More about this in the next subsection. Interoperability is a major challenge as not all devices have direct interfaces and still need gateways or hubs. For example, classic embedded devices do not have Internet connectivity, which is now essential [Spa17].

IoT systems are exposed to particularly high risks and require protective measures that other systems do not need to the same extent. One reason for this is the involvement of end users who allow inexpensive products into the system without sufficient security guidelines. Smart homes are particularly vulnerable to this. Accordingly, privacy problems can arise, in some cases revealing impor-

tant information such as Personally Identifiable Information (PII). [SB15] The general access to the Internet brings possible cyber attacks, which are less likely to affect closed networks [Pal20]. The frequent request for lightweight code for IoT devices brings lightweight encryption with it, as it would otherwise be overly energy consuming. [Var+17] However, this makes the devices vulnerable. Especially due to their distributed nature and the risk of being intercepted or manipulated from different places.

2.2.1.3. Data Management and Communication Protocols

The systems and devices mentioned generate or transmit large amounts of data and communicate with it. For big data, the Data 5Vs emerge: Volume, Velocity, Veracity, Variability and Variety. The huge volume of data sources in IoT systems, which are transmitted with high velocity, bring along variety of data formats or quality. Therefore, the veracity of data has to be validated to check whether data are manipulated or vary in any other way. [ISO16b] This leads to an urgent need of data management methods in order to be able to transmit and store, but also to enable a proper usage.

Many relational databases cannot be used in IoT systems because they cannot cope with the volume of real-time data with different formats and the heterogeneity of transmission details or source locations. [AHA13] Whether a local or global storage system is required, depends on the use case. [ISO16b] To meet these challenges of data, a data management system is needed. However, data management in IoT systems enables storing, processing and logging online, and provides concurrently information for offline analysis. [AHA13]

As can be seen in figure 2.7, the IoT data lifecycle can be divided into three paths. Data are produced on IoT system edges and can conduct an autonomous handling of data at the edge. Therefore, they can be processed locally and can be used for real-time queries and actions. In addition, data can be transferred and collected online respectively remote. Filter and aggregation methods can be applied to perform more complex near real-time queries. But also historical queries, based on long term collected data, are available. Therefore, data has to be delivered to preprocessing and afterwards stored long term in an archive. These data collections can be analyzed with complex processes before conducting queries. [AHA13]

To create logical and physical structure for data management solutions that are suitable for IoT systems, diverse architecture principles must be considered. First, data strategies have to be designed to adjust to mobility of devices and to discover respectively collect data. Second, a flexible database model is required, as well as data- and sources-centric middleware, layered storage platforms and

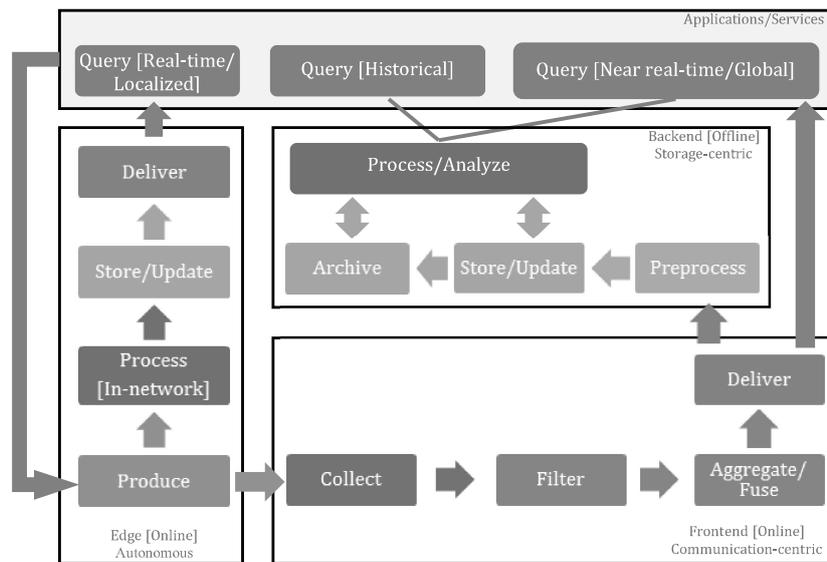


Figure 2.7.: IoT data lifecycle and management [AHA13]

scalable archiving support. Last, to process data, management systems need an aggregation support, query optimization and an access model. [AHA13]

To transport this data, data protocols are needed. However, in IoT systems, many more protocols are necessarily involved to enable communication and coupling of services. They are mainly responsible for sequence control, flow control and retransmission of lost packets, next to defining data exchange formats, data encoding, and addressing or routing schemes. [AV13]

Usually, IoT devices would use the four-layer Internet Protocol (IP) stack to connect to the Internet. However, this stack is not fully compatible with the IoT challenges of communication. One of the challenges is having IoT devices that are battery-powered and have hardly any memory capacity. This means that only routing protocols with low memory requirements can be used, and data transmission shall consume low levels of energy. In addition, devices are mobile and data must not be lost regardless of their location. In addition, IoT devices must be uniquely identifiable everywhere. This means a need for a large addressing space. [SS17]

Accordingly, a non-IP stack is more appropriate as they consume less energy and use smart and local gateways. However, these non-IP communication channels are only usable locally, e.g. BSN, as they send and receive only in short distance. Therefore, combinations of IP respectively non-IP stack, OSI model and IoT stacks are developed. These approaches differ in their classification into layers, but contain same protocols. [SS17; AV13; Pro15] Depending on the connected device, purpose and layer, different protocols exist, such as: Bluetooth, 6 LoWPAN, Zigbee or CoAP and plenty more.

Bluetooth satisfies multi requirements as it is a low power and low cost protocol for data transmission, based on the IEEE 802.15.1 standard. However, this standard only can be used in a range of few meters. Its follow-up is BLE (Bluetooth Low Energy) which was developed especially for embedded systems and requires even less power and costs. BLE performs with a master and slave structure that enables quick transport of small data packages. [AV13; SS17]

Similar to these protocols is 6LoWPAN, which is a low power personal area network protocol combined with IPv6. This enables communication with other IP-based devices. These networks have gateways as intermediate stations before entering the Internet. As low power standard for Wide Area Networks, LowRaWan come into operation that enables communication between connected devices with a 20 miles distance. LowRaWan uses a special gateway and server architecture which enables double encryption. [AV13; SS17]

Even though the energy consumption is 2.5 times higher than with BLE, Zigbee can be counted among the low energy protocols. It is based on the IEEE 802.15.4 and is used for short distance communications which are in return cheap and reliable. It has a star, tree, mesh topology that supports diverse routing schemes, especially multihop routing. This includes discovering of nodes that are joining or leaving the network and maintenance of routes. [AAA13; SS17] SigFox instead, is used for very long distance communication, up to 1000 kms, by performing on narrow bands and long waves. [Isc17; SS17]

Other typical IoT protocols are MQTT and CoAP. While MQTT is used for remote distance communication with a lightweight publish/subscribe messaging model, CoAP applies as an application layer protocol for small resource-constrained Internet devices with multicast operation features. [AV13; SS17]

2.2.2. Connected Systems in Wellbeing

Big Data by monitoring is an omnipresent main challenge of 21st century. Responsible for this is, among other factors, the fact that sensors or other IoT things are spread anywhere which collect a tremendous amount of data. Some of the major profiteers of this Big Data is the medical area with its ever growing number of devices and accompanied number of data. This trend can involve and improve all health-concerning sectors and offers the possibility to be more informed and aware of its own health condition. [Dim16]

But it's not just health-conscious individuals who take advantage of these benefits. Connected healthcare networks are in operation at various locations. This has started with the general digitalization in medicine. This has given rise to the discipline of Medical Information Science. It involves usage of theories and methods, processes and techniques of informatics in various areas of healthcare.

[Soo+07] Closely related to this is e-health. There is still no standard definition for e-health, but it is generally understood to mean the use of electronic, information and communication technologies in healthcare. These include m-health approaches as well as telemedicine, telecare or remote patient monitoring. While Medical Information Science looks at general digitalization, e-health approaches are specific use cases with patients or health-conscious people. [Dee00] M-health refers to the use of mobile devices such as smartphones, tablets or smartwatches in a medical context. Telemedicine or telehealth include consultation and treatment via the Internet and are based on the physical separation of doctor and patient and the use of additional technical aids. [Soo+07]

This development is based on wireless sensor networks. Sensors are used that can record electrical, thermal, optical, chemical, genetic, and other signals. These are subsequently processed to make an assessment of a person's condition. Advances in micro-electromechanical systems and lab-on-chip are having a positive effect as well. At the point of care, evaluations can be made and new forms of chemical, biological, and genomic sensing can be included. In addition, sensors are becoming smaller and cheaper and can be more easily integrated and interconnected in assistive devices and implanted devices. [Ko+10]

From these advances, the IoT with medical devices, healthcare IoT or IoT-WD have emerged, and are thus, based on a healthcare IT system structure of connected components and applications that communicate with each other. [Mar17] Smart materials, fiber-optic sensors, MEMs sensors, such as accelerometer, RFID tags and plenty more are used. These cover mechanical, optical, chemical and general physical measurements. [Tok+17] Various connected devices are used for this purpose: IVD devices for blood analysis, physiological monitors for weight measurements, glucose meters, heart rate sensors, mobile medical apps for medication, wearables for activity tracking or pedometer and intensive devices like implants. [Mos17] This amount of devices and sensors generates various types of big data that need to be collected and stored. [Dim16]

In addition to healthcare facilities, two groups of people may be considered as users: people with a strong interest in their own health or independence, and patients with preexisting conditions or diseases who need to monitor their condition regularly. [Piw+16]

In hospitals, IoT is often used for vital sign monitoring to relieve the nursing staff and to monitor patients during surgery [Ko+10]. If possible, these follow-up services can also be outsourced to the patient's home. Especially during a pandemic and busy hospitals, this creates more capacity. [Sen21] But the main use is bed management or the whole bedside environment communication. Bed locations can be tracked and managed with RFID. The bedside sensors, on the other hand, can communicate with host computers of clinical staff, and thus, interoperate with patient care systems and medical device systems. [EMB96]

In the nursing and pharmaceutical industries, apps and sensors are used to support patients shortly before or after discharge. Medication management can be improved by measuring or recording conditions and tracking medication effects. [Dim16] Smart pillboxes also allow direct reordering. In addition, mobility assessment can be supported by transmitting symptoms and adjusting therapy. If an emergency should occur, all data can be viewed directly in advance in order to be able to exercise essential care upon arrival. [Fla18]

The most promising but also risky application field is home use and the use for wellbeing. As low-cost and insecure IoT devices are connected to sensors that track vital data, there is an increased need for security and safety measures. [Doh+10; Ko+10] These include typical IoT devices and, for example, wearable biosensors, smart thermometers, connected inhalers, and insulin delivery systems [Sen21]. On the one hand, smart medical homes are used by health-conscious people who want continual personalized feedback and guidance on their wellbeing. [Doh+10; Ko+10] On the other hand, Ambient Assisted Living (AAL) approaches are being applied increasingly. [Kri+09] Elderly or chronic ill patients live independently in their homes and are supported by telehealth methods. This includes connection to relatives, telecarer or physicians via various services. [Dim16] For example, movement can be monitored through sensors on crutches and guidance can be given [Ko+10].

Typical benefits of these use cases are lowered cost of care, improved patient outcome, disease management and improved quality of life [Mos17]. In addition, objective reporting, remote monitoring, activity recording, automation of processes and precision medicine approaches are advantages that make IoT in healthcare a mighty tool. [Mar17]

IoT-related technical challenges are mainly the generic problems such as energy efficiency, latency, scalability and throughput. [Dim16] Other challenges are amplified by deployment in healthcare. These include safety, privacy protection, and device diversity with medical devices on the one hand and cheap private devices on the other hand. [Ko+10; Mos17].

In order to clearly separate these application cases, medical and wellbeing devices or sensors have to be defined.

[EMB96] defines medical devices as components or systems to diagnose and treat diseases with support for electronic communication, and thus, is roughly defined in the same way as the German Medical Devices Act. According to [Dev12], a medical device is a component that is related somehow with diagnosis, prevention and treatment of diseases and injuries. In addition, they define that any support for physical processes and life sustaining measures are included. Since then, several norms and standards have appeared for products that are a medical product or a part of it. The standard for medical device software and software

lifecycle processes ([IEC06]) is applicable to embedded and standalone software. This refers to software that has been classified as a medical product or is part of a medical product, and therefore, is an under subgroup of medical devices. An example of this is software in surgical robots. More general software is health software, which, however, is not covered by the German Medical Products Act. [Ins] In addition, this standard defines guidelines for each development phase and has IT security requirements to be certified. [IEC06] This is a big difference of wellbeing devices, as these do not have to be certified. Similar to that the medical device regulation ([Par20]) demands State-of-the-Art security methods during development, the IEC 60601-1 standard ([DIN13]) requires countermeasures against attacks and data manipulation. Accordingly, the ISO 13485:2016 standard ([ISO16a]) also includes data protection measures. In addition, a standard defines risk management for medical IT networks [IEC11]. More details of these security standards can be found in subsection 2.3.5.

Many of the IoT devices, especially in the home, do not meet these standards, notably security guidelines. [Pal20] Accordingly, they are not certified and are not allowed to be used as medical devices. This leads to wellbeing things and IoT-WD. They include next to real medical devices, also things, sensors or any hardware respectively software components which are used for self-determined prevention that has no direct influence on the body or are similar to medical products but just for analytic data gathering reasons. [Mit+18] However, safety and security aspects are no less important in this context respectively are even more important. Wellbeing devices are often cheap and insecure, and therefore, often contain security gaps [All20; Pal20; Var+17]. This can affect other components of the network, but can also lead to manipulated data or breached privacy.

This leads to a definition for sensors or other devices for IoT-WD:

Wellbeing or pseudo-medical devices are physical components which enable the measurement of vital data or other physical conditions that can be used for preventive health management, symptom tracking or fitness improvement. This can include medical devices excluding interfering features with a limited set of functions, made for prevention and health-conscious monitoring .

Accordingly, the focus of this work is on wellbeing devices, sensors and systems. The use cases are in home use with connections to other stakeholders. The users are health-conscious people who want personalized feedback on their fitness or wellbeing level, or elderly residents who want to maintain their independence or are no longer mobile for medical visits. Often these wellbeing devices are fitness wristbands or other wearables that are becoming more and more popular and track various information. They can be used for different areas, e.g. health education, symptom tracking, and collaborative disease management and care coordination. [Dim16] But also classic medical products can be included such

as a glucose measurement system that automatically measures and transmits the values. Whether insulin is administered directly or manually depends on the use case. [LRJ11] Pacemakers can also be used for remote monitoring if only heart signals are measured and transmitted for evaluation. This can be used for long term analysis, but also for technical control of the device. [Lop+16]

Medical Application Platform (MAP) and personalized health is presented in more detail in sections 2.2.2.1 and 2.2.2.2.

2.2.2.1. Medical and Wellbeing Application Platforms

Increasing amount of sensors, connected medical devices and the accompanying amount of data pose a significant problem in hospitals and other medical areas. In order to utilize this quantity of data, [Hat+12] proposes a MAP which serves as “safety- and security-critical real-time computing platform” for integrating and managing medical devices and data. Another area with similar development of increasing data and its usage is private health monitoring, where people rely on health-apps, -sensors and increasingly IoT devices, to monitor and support their health-conscious living.

However, two main problems occur. Firstly, a large part of those sensors and devices is not accurate enough regarding methods of measurement and results of sensors. Secondly, most health applications are not personalized enough. General-purpose advices, like targeted amount of steps, are not advanced enough recommendations, and thus, only of limited value for improving individual wellbeing - a combination of medical and general physical health and fitness. This lack of results-producing recommendations is a possible reason for the behavior of wearers of health-sensors observed by [LM14]: At the end of their study half the attendees were not using their health-sensors any more. One third even stopped using them after 6 months. In order to receive the needed information for results-producing recommendations, all available data has to be processed further, to create smart data like MAPs already do.

The above named issues are addressed from [RB17] by introducing a WAP, as a symbiotic pairing of IoT and medical devices, as well as an associated process of multi-sensor data fusion to create smart data and user-optimized recommendations. A WAP provides a new possibility of combining the mentioned areas and is able to deliver customized advices whereby focusing on wellbeing and domestic health monitoring. MAP’s concepts are used by transferring it into the wellbeing area, including new requirements of a non-clinical domain. WAPs meld together non-professional devices, like wearables, already used by consumers, and single sophisticated medical devices which provide highly accurate results, in any sort of combination. On the one side, these combinations compensate disadvantages

of non-professional devices regarding vague or false results, for instance by preventing an erroneous pedometer during ironing. On the other side, they reduce the cost of professional equipment by only deploying selected ones.

To involve a WAP on a daily basis, existing IoT-Platforms are used to integrate ordinary devices as new sources of wellbeing data. While smartphone apps and wearables can be used as non-professional devices, BSN or Wireless Sensor Networks are already applicable as an option of including professional medical devices [Ull+08]. A WAP represents a centralized data integration platform for context-aware multi-sensor data fusion of ordinary IoT devices, consumer-grade wearables and medical-grade equipment. This big and various data collection capacitate WAP's ability to create Smart Data. By applying medical knowledge and machine learning algorithms customized recommendations are received in the next step. The main added value of a WAP represents the skill of improving the quality and usage of non-professional medical devices and their results by reducing measure errors through knowledge of professional ones. [RB17]

Because of the wide range of used devices, use cases of WAPs can be located in diverse wellbeing application fields. Nutrition counseling and monitoring with smart-fridges using RFID combined with BSN and clinical accurate devices represent a powerful wellbeing scenario which can help with weight problems or medical conditions like diabetes [Dim16; Tok+17]. Sportsmen using WAPs for optimizing their performance by getting recommendations about their needed behavior changes are also possible [Gla+03]. Further WAPs can be used in already existing technical environments like cars. The combination of wearables and implemented medical accurate sensors can be used to monitor and advise a driver [Kat+11].

2.2.2.2. Personalized Healthcare

By the usage of wearables or e-textiles further personalized monitoring can be conducted. This data acquisition is a mighty tool to create a customized rehab plan, individual care needs or wellbeing recommendations for health-conscious users. As people carry more and more devices and sensor technologies allow more accurate and precise measurements, this data can be used for these recommendations. In addition, tracking of health status is easier to handle and advanced by machine learning and the higher amount of data and a more efficient use of it. [RAC17; Gla+08] These recommendations may have medical advices, however, they are mostly designed for private fitness or wellbeing. Combined with apps, mental health, physical condition changes, appointment scheduling or personal health information can be entered or managed. For example, breast cancer patients who need support in addition to their treatment regarding health information or mental health care may have the opportunity to receive personal-

ized recommendations based on their family and disease history. These recommendations are based on a variety of data sources. [JJM18]

All this collected data can accomplish an important knowledge for health improvement, but the potential is foregoing frequently. Through sensor and data fusion this deficit can be compensated, as combined data exhibit added value [VS00]. However, the heterogeneity of data prevents a simple fusion of them. Especially privately collected data are often unused, whereas this data are suitable for smart living or to improve personalized health of an individual by the usage of Internet of Things and its connected data. Here sensor fusion represents the possibility of personalized healthcare, based on the variable choice of used devices.

In the medical sector sensor fusion can be deployed in three different application fields. Most common area is clinical sensor fusion in hospitals within MAPs ([Hat+12]) for concurrently smart monitoring patients' various health values. In addition, sensor fusion can be used in domestic telemedicine or telerehab in combination of IoT. Closely associated is the wellbeing and prevention area, which appears in combination with IoT and Smart Homes. This area represents the most potential and also the most risks of personalized healthcare, since prevention is highest affected by individual's health needs. A personalized medical Smart Home, which is using sensor fusion, requires different devices, for instance wearables, nearables and sensors for each patient's medical precondition. However, this fact represents one of the most complex issues of sensor fusions as devices are heterogenic and difficultly combinable.

A possibility to merge these data and prepare them for further processing is offered in [Bec+17]. The named issue was addressed by developing a generic medical sensor framework which is able to combine sensors and collect data independent of devices. Through this solution sensor fusion was made generic, thus enables personalized health needs and the usage of desired sensors. Furthermore, continuous measuring is offered including collecting data and enables further processing. The solution includes physiological data of patients and combines these with ordinary sensors and medical sensors, and minimizes limitations of personalization through unbounded sensor choice. Therefore, the JDCF is capable to assess and process data and information from different types of sensors and provides this value resources for further systems by storing this information in various data spaces, called persistence forms. The three middle phases of figure 2.8 represent JDCF. The main task of the sensor controller is binding of sensor components and JDCF. The sensor module manages the connection of them by activating and deactivating the connection. Every time the controller is executed, a new sensor record is created, which is a central object of JDCF, that includes the whole received data of one sensor at one receiving moment with a unique identifier and time of creation. The received data are coming unsorted and unstructured. Therefore, the record preparation processes the data, performs

a sensor data check and creates structure data. This structure includes the regulation for data sequence and data types for every sensor. The checked and correctly filled record is send to the logger. Since there are different forms of persistence, it is important to allow a generic saving of the data. Due to the information contained in the structure, each logger has all the necessary information to store the data. The JDCF finalize with data fusion. [Bec+17]

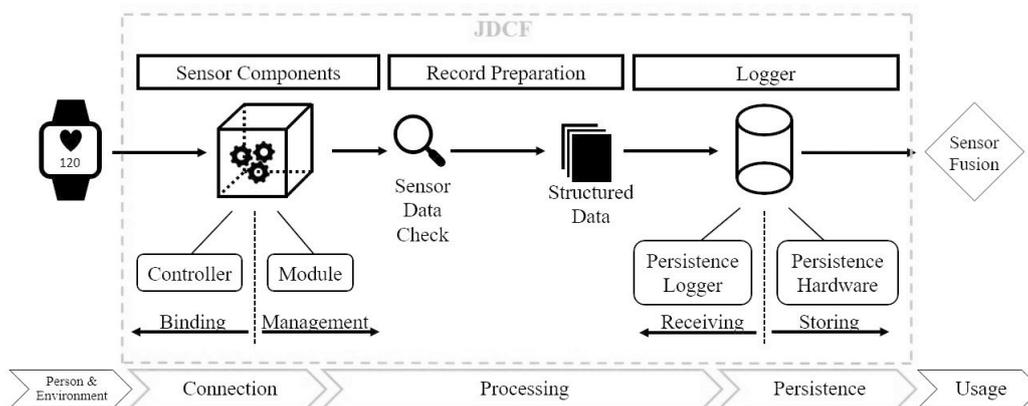


Figure 2.8.: JDCF components [Bec+17]

Once a kind of data store is created which contains, e.g. wearable, smart device or generic patient data, that are merged together and have a unified data structure, can be used to generate knowledge and recommendations. This can be performed, for instance, with AI technologies. These are long term evaluations and not short term events which are performed by actuators. That kind of personalized recommendations are based on the knowledge and experience of clinicians who are deciding based on past experiences and the collected data of the user. [Dim16]

2.3. Approach Requirements

Design optimization is built based on different requirements. Following, considered requirements and related foundations for this approach are pointed out.

2.3.1. Safety and Security

The considered system requirements have direct impact on the system design, and thus, on the architecture. [Som11] Therefore, the main requirements of this approach are safety and security, and related risk assessment features. These are interconnected and even overlap partially. However, in contrast they are facing multi-concerns issues, and can therefore, obstruct each other. [Loh20] For instance, secure authentication methods and the need of fast emergency calls can be a dilemma.

Safety is defined, referred to IEC 61508 as

"the detection of a potentially dangerous condition resulting in the activation of a protective or corrective device or mechanism to prevent hazardous events arising or providing mitigation to reduce the consequence of the hazardous event." [IEC10] Furthermore, safety "is the condition of the system operating without causing unacceptable risk of physical injury or damage to the health of people, either directly or indirectly, as a result of damage to property or to the environment." [Con+16]

In this regard, the rate of tolerable risk that represents the acceptable risk for a use case must always be considered. Whereas risk is a product of probability of occurrence and severity of a harmful event. [ISO99] Harms and accidents are caused by hazards and result in death, human injuries, damage to property or to the environment. Hazardous conditions can have multiple types, but are mainly based on failures that represent inability of a system unit to provide a functionality in its required result. Faults are incorrect conditions which can be inherent weaknesses of the design or implementation that cause errors respectively a deviation between desired values and measured values. Systems require a specific fault tolerance to enable correct functionalities in a defined range, even if faults are present. If this range is passed, errors cause failures. [ISO99; IEC10; Som11]

Whereas safety focuses danger caused by systems, security faces danger for a system itself and is a pre-requisite for further requirements like availability or reliability. Furthermore, security has an impact on safety aspects, as attacks can have an in-built side-effect on humans. [Som11]

Security is "a system property that reflects the system's ability to protect itself from accidental or deliberate external attack" ([Som11]) and "is the condition of the system being protected from unintended or unauthorized access, change or destruction". [Con+16]

This danger is based on weaknesses in the system, and the resulting vulnerabilities, which can arise in every development phase and may be exploited. If an exploitation happens, denoted as attack, the system is facing a threat which is a circumstance to cause loss of an asset. Assets can be anything that must be protected and is valuable for a system. The asset endangered by threats and vulnerabilities is a kind of risk. [Som11] To ensure security for a system, the CIA triad is applied. Confidentiality prevents sensitive, private data against unauthorized access of people or entities and controls device and message access, including encryption technologies. If data are accurate, complete and consistent, and therefore, free of data loss or tampering of messages and all its related information, Integrity is preserved. Whereas Availability is a property to enable to provide information, services and functions at any time which relates to robust systems. [Con+16; BSI12] In healthcare, this triad is not evenly distributed as confidentiality is highly important, however, not necessary to treat patients. Therefore, in this case availability has the highest priority. [Mur15] In addition, the triad is related to accountability to assign people to actions ([Li17]), authenticity for checking entities identity ([BSI12]) and non-repudiation to prevent denial of changes or access [ZG97].

Even if the focus is on safety and security requirements, there are always other related requirements to consider, e.g. privacy and reliability, as none of these requirements is independent and can be achieved isolated. Reliability, which is related to availability, means the ability to perform its assigned function for a specific time between failures reliability [Con+16]. Whereas privacy is a side-effect of security and represent the right of human or other entities to decide which of their data are collected and who can interact with them [Con+16; SC16].

2.3.2. Requirements in Critical Areas

The described requirements have to be considered modified depending on the application field. In particular, benchmarks, metrics or Key Performance Indicators (KPI) are more critical. A distinction can be made between business-critical, mission-critical and safety/security-critical systems. While business- or mission-critical systems consider risks of financial loss or failures to achieve goals, safety/security-critical systems are the most critical and vulnerable ones. IoT systems can also be used in critical systems. As a result, non-critical IoT networks can become critical networks. [Lan17] Considering the requirements of these systems

can also be done in the design phase. [CIS]

Safety-critical systems involve the wellbeing of people and include high reliability, availability and security ([Lan17]) and can be specified as

"a system is referred to as safety-critical when the consequences of its failure can lead to loss of life, or to significant property or environmental damage." [PR13]

Reinforced by safety-critical systems are the new connected medical devices and the fact that electronic protected health information is collected. Weaknesses in the network can, thus, put patient confidentiality and safety at risk. [Voc12] Safety-critical systems must have a high reliability, and therefore, a low acceptable probability of failures to be approved. Depending on the use case, there are allowed between 10^{-4} and 10^{-10} failures per hour. Important points are a redundancy management, a fast fault detection respectively isolation and that a system still works when a certain number of components fail. The redundancy management should create the right balance between redundant components and their maintenance requirements. [LH94] As already defined, safety-critical systems always include security-critical requirements, since attacks can have consequences for safety aspects. Cyber-critical systems - systems that are critical for attacks from the Internet - are a subset of security-critical systems and occur frequently in IoT networks as they use services via the Internet. For instance, data from health-care that is collected or processed over the Internet requires special protection, as theft or unauthorized access to sensitive data could have serious consequences for the people concerned. [Szy17; SC16] Thus, a definition follows from this:

"Security-critical systems deal with the loss of sensitive data through theft or accidental loss." [HC10]

2.3.3. Risk Management

To consider generic steps of risk management, its tasks are

"coordinated activities to direct and control an organization with regard to risk." Related to risk as an "effect of uncertainty on objectives". [ISO09]

Specifically, risk management can be defined as the establishment of the methods required for risk processes in order to identify, analyze and counteract risks. This includes risks to the organization, business units, subsidiaries, related infrastructure and stakeholders. Accordingly, risk management requires various

safety and security evidences to cover different types of risks, ranging from metrics to KPI monitoring tasks to verify compliance with requirements. Based on this, appropriate decisions should be made, including priorities, policy and budget decisions to the further implementation process. However, since risks are not static, the risk management processes have to be dynamic and continuous. Changes in risks can be caused by concept changes, new critical system areas, new hardware components or features, new potential attackers, but also changed risk assessment activities can occur. [Con+16]

For risk management, based on [AR10] and [MLY05], three main steps can be specified: risk acceptance, risk assessment and risk reduction. Ever step contains multiple sub-steps:

- **Risk identification and acceptance:** Different risk carriers perceive different risks or view them as having different degrees of severity. Accordingly, all risks must first be identified and communicated. Risk management cannot make a system completely risk-free, but it can contribute to risk acceptance and recognition. On the one hand, the existence of risks have to be accepted by all stakeholders. On the other hand, if the cost of prevention exceeds the risk, it must be accepted as tolerable.
- **Risk assessment:** Risk assessment procedures differ widely depending on risk type, use case and assessment perspective, e.g. technical vs. business. Probabilistic approaches are often used. However, the focus is always on characterizing risks and assessing their impact and severity. [Con+16]
- **Risk mitigation:** Risks that have been assessed as critical have to be removed or mitigated. A distinction can be made between removing entire features if their risk is not worth it, and adapting features. The countermeasures must be communicated and accepted again. If no acceptable countermeasures can be found, there can be no acceptable system.

In chapter 5 the applied risk assessment approach of this thesis is discussed.

2.3.4. Safety and Security By Design

As already described, the development of products or the deployment of entire systems consist of several phases. As a result, the necessary requirements must be taken into account in all phases. This applies to safety and security measures. [DM13; BM12; GB20] Thus, safety and security requirements can be checked in early development phases and refinements can be applied on models if necessary. This prepares implementation steps that adhere to the requirements from the beginning. [GB20]

Accordingly, a safety lifecycle includes

"necessary activities involved in the implementation of safety-related systems, occurring during a period of time that starts at the concept phase of a project and finishes when all of [...] safety-related systems and other risk reduction measures are no longer available for use" [IEC10]

Same applies to security measures. Although the basic principles are the same, there are many names for this approach. Architecture analysis, by design approaches or model-driven safety and security are some of them. [GB20; CD13; DM13] Even though individual steps are different, they are same in the idea that vulnerabilities must be identified when systems or products are only models and have not yet been implemented. By design approaches, architectural mistakes can be uncovered or general risks can be identified that are already known at the time of analysis. These approaches have been successfully used in the past to detect and fix weaknesses. Often these methods are based on expert knowledge and have to be performed manually. Frequently, threat models are used for this purpose. [DM13] These models define possible threats to the system and should represent as many different threats as possible and be able to include new threats. [GB20]

In addition, to the various analysis approaches that check the design phase, by design is often understood as principles that should be observed in the architecture phase. These can be highly abstract but also quite specific. Among other things, this includes minimizing the attack surface area, and thus, avoiding unnecessary functions, the principle of least privilege to prevent unauthorized persons from gaining access, and the separation of duties, and thus, the distribution of roles. Another important principle is avoiding security by obscurity as this is in direct contrast to security by design and means that a system is only secure as long as an attacker cannot gain knowledge about it. [Fou16] A more detailed consideration of these principles is given in chapter 3.

2.3.5. Regulations and Standards

As already mentioned above, there are various standards relating to safety, security or risk management. They range from general rules to domain specific regulations. A distinction has to be made between guidelines and laws. Here, the fields of application differ. This leads to uncertainties, particularly in the case of cross-border deployments.

The IEC 61508 standard already used above is an international series of standards for functional safety requirements in safety-related systems and serves as a basic

standard for further standards in special domains. The objectives are mainly to provide guidelines for the manufacturing process in order to develop products that are safe for the user and its environment. A lifecycle is used to cover all phases of production. Thus, the guidelines start from product architecture and include organizational structures for documentation purposes. This is to ensure that all steps are traceable. Safety Integrity Level (SIL) is used to define the correct level of safety requirements. Devices, sensors or controls must, therefore, be given a SIL rating or other trust metrics. These four levels determine the necessity and efficiency of safety functions. In addition, this standard focuses on software requirements and uniform definition of terms and abbreviations. [IEC10] This standard is widely recognized and used. Even the BSI refers to its guidelines. The BSI uses SIL in particular for risk assessment. Although the BSI mainly considers hazards for government systems and system-critical applications, it deals with consumer protection and security, and thus, is relevant for IoT systems. [BSI12] In addition, a new law to increase the security of information technology systems was recently passed, which entails an expansion of the BSI's powers. This law is to be the root for cyber security strategy for Germany. This contains basic principles for the orientation of IT security measures for operators of critical infrastructures. [Bun21] A classic norm to keep further risks under control is [ISO09]. It provides guidelines for principles and implementation of risk management which are valid worldwide. This general standard is based on the understanding that risk management is at the focus of attention, and is therefore, a management task. A top-down approach is to be taken in order to discuss the interrelationships. The focus is on risk prevention rather than mitigation. Therefore, a special risk management system is to be integrated into the existing measures in order to supplement risk management gaps. [ISO09]

In addition to the general standards, there are also specific standards and regulations for almost every domain.

DIN EN 60601 is based roughly on IEC 61508. However, this standard extends it to include medical-related countermeasure requirements against attacks and data manipulation. The focus is on essential performance and safety characteristics of medical electrical equipment and medical electrical systems. These systems are intended for diagnosis, treatment or monitoring of a patient. In addition to alarm systems and security measures of medical technology emergency operations, measures in the physical environment are also considered. Extensions to these standards consider precise measures tailored to individual specific products. [DIN13]

A related standard is ISO 13485:2016 that represents requirements for quality management systems for the design and manufacture of medical devices and includes data protection measures. The clear focus of this standard is to be able to implement the constantly changing requirements of legislators or customers without losing any quality. Here again, a lifecycle is applied that is intended to include all phases, and thus, stakeholders. [ISO16a] Along with this, the stan-

standard [IEC06] can be used, as it defines guidelines for each development phase including security requirements. In addition, three safety classes for severity categorization were determined for this standard. Further details are discussed in chapter 3.

The medical sector has its own standards and regulations for risk management. Two of the best known are ISO 14971 and IEC 80001. Risk management that includes risk analysis, risk acceptance criteria and risk reporting is provided by [ISO20]. Risk analysis involves identifying hazards posed by medical devices, evaluating probabilities and severity, and deciding on acceptability of risks. Risk acceptance criteria should help to estimate which countermeasures are needed and are suitable. An important point of these standards is the continuous monitoring of new risks, implemented measures and their acceptance. [ISO20] In contrast, [IEC11] focuses on medical IT networks and their risk management. The steps to be performed are very similar, but three additional protection goals are specified. Safety, effectiveness, and data and system security. These are all inter-related and should be taken into account in all processes. [IEC11]

Most of these standards are not required by law, even though they must ultimately be applied in order to comply with the law. However, what exactly is required by law often differs immensely. As an example, HIPAA is one of the most important legal regulations in the USA. The content of this law is the security and privacy of patient data and access to medical records. All stakeholders involved in the handling of personal medical information are required by law to comply with various requirements. These include ensuring the confidentiality, integrity, and availability of the data, integrating safeguards against threats to or disclosure of that data, and taking steps to verify compliance with these requirements. A special HIPAA security series provides guidelines for this. [US 07a]

2.4. Enterprise Architecture Management

The most important type of architecture in companies is EA that consider many aspects at once. The focus is on individual components and their connections. In the case of EA, this can include hardware, software, business processes, applications and the technical infrastructure of an organization, but also personnel parts. The connections can also include further characteristics. Thus, this architectures are used to describe all elements of a holistic enterprise system and the relationships between them. The term also comprises the process of creation and maintenance of the architecture work. [Lan05]

Therefore, compared to general definition of architecture, EA is defined more precisely:

"An EA is a conceptual framework that describes how an enterprise is constructed by defining its primary components and the relationships among these components. " [Roo94]

In addition, [Roo94] defines multiple aspects which have to be fulfilled by an EA to reflect an enterprise:

- Consists of people, information, and technologies
- Performs business functions
- Has a defined organizational structure that responds to internal and external events
- Has a purpose for its activities
- Provides specific services and products to its customers

Business processes or functions can be viewed as a sequence of activities within a company. This includes, for example, sequences of activities in service or production. In particular, sequences that have to be repeated regularly must be mapped and coordinated. Modeling approaches are often used for this purpose. [Yen09; VSP08] These are not limited to business processes within a single company. For example, the approach of [BMR04] offers a possibility to model business processes that are distributed across several companies.

Enterprise architectures and its processes can be depicted in many different ways with different modeling techniques, but most approaches include a subdivision into several small architectures. [WF06] These sub-architectures are first modeled

independently and afterwards connected to include dependencies. These dependencies are critical to the business goals. [BIT11; WF06] The Open Group ([Grob]) justifies this by stating that it reflects underlying business and IT structures. A successful IT strategy, exploitation of IT advantages and defined technical processes enable achievement of business goals. Hence, it is useful to analyze the goals of each architecture to enable their interfaces to be adjusted to each other in a more effective way [WF06].

Therefore, EAM was introduced to deal with assessment and development of business processes and IT components. In general, the term can be defined such that EAM

"maintains and uses a coherent set of guidelines, architecture principles and governance regimes that provide direction for and practical help with the design and development of an enterprise's architecture in order to achieve its vision and strategy." [Ahl+12]

This definition has become widely accepted in EAM and is used by large frameworks such as TOGAF. [Grob] In general, EAM is a strategic, conceptual and organizational framework for EA and implements and maintains an architecture with the help of the integrated principles and tools. The focus is always on trying to align the design with the requirements and goals and to be able to react to the constantly changing IT landscape and market conditions. [BIT11] The well-founded description of the current situation and correlations within an information system helps to achieve a common understanding of all stakeholders involved who have an influence on the company's success. [Grob] Thus, this holistic overview of IT design and business landscape enables knowledge about individual small architecture models with the different layers within the EA domains. [BIT11] Thus, typical EAM tasks are develop, describe, communicate, analyze, and configure. [BMS10]

The current company status, and thus, an abstraction of reality, is referred to as As-Is status. This status can be retained in very few cases. Conditions or goals change and the architecture shall be adapted accordingly. A future architecture that represents intended plans is called a To-Be architecture. Both models or states contain uncertainties and influence achieving To-Be architectures. [Buc+11]

To transform enterprise architectures, a plan must be developed that reconfigures all components and relationships. This is part of EAM tasks. For this purpose, a goal-oriented adaptation of IT landscape based on changed framework conditions is performed. [BIT11] However, success of this undertaking always depends heavily on the completeness of knowledge about As-Is architectures. The more documentation, the more reliably plans can be implemented. [Saa10a]

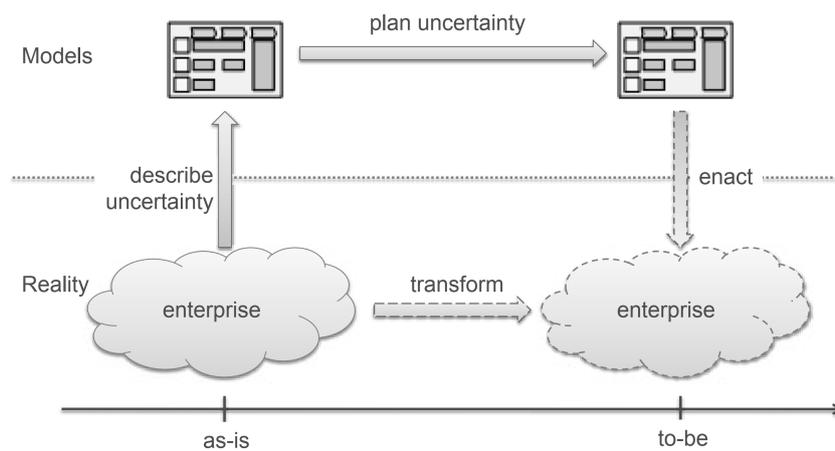


Figure 2.9.: Different types of uncertainty [Buc+11]

2.4.1. Layered EA Structure

A typical EA is divided into layers in order to be able to assign its components. Since enterprise architectures are model-based, they can be described with diagrams. Because of this feature, their different layers are often represented graphically. [Joh+07a] The division into different levels is intended to reduce complexity. The components of the individual levels can be considered within and across levels. This feature is important for the function of analyses in EAM, as it allows the effects of architectural changes to be analyzed vertically as well as horizontally in an enterprise. [BIT11]

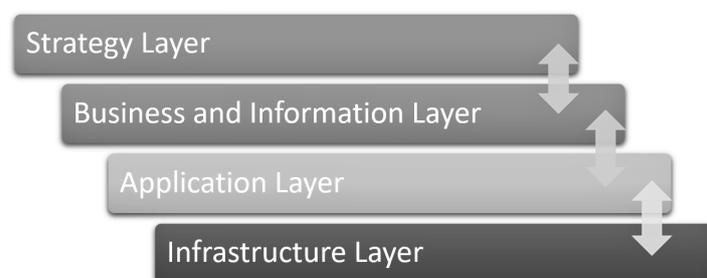


Figure 2.10.: Four enterprise architecture layer

There are several approaches to layering EA. Often used layers are strategic layer to represent the organization's strategy with its goals, business layer describing business processes and products, information layer with information objects, and application layer as well as infrastructure layer describing soft- and hardware components [Lan05; WF06]. Despite the examination of different layers the focus of an EA are the dependencies between layers, i.e. how business and IT relate to each other. Layers are dependent according to the Align-Enable-Principle. The lower layers are the foundation for the upper ones, and the upper ones adjust the lower ones [WF06; Krc15; IJ06].

2.4.2. EA Analysis Techniques and Goals

According to [Wir] a goal is a

"target value which a current state is compared with that is to be worked on until it corresponds to the target state."

This general definition can be applied to EAM and its analyses. The target value corresponds to the To-Be state, and is thus, the EAM goal to be reached. This goal is to be achieved through adjustments and scenario comparison after an EAM analysis has been performed. [Saa10b] Through analysis the information flow in organizations is optimized. Therefore, analyses are one of the most important artifacts integrated in EAM and are indispensable in the EAM cycle. This cycle contains five phases [Nie06]: Document, Analyze, Plan, Act and Check. This cycle is based on uncertainties as all As-Is states. [Nie06]

Thus, analysis is an essential part in order to create and implement future plans. As described, it supports decision making through an evaluation of the current architecture [SK11]. The result of analysis and planning actions is finally the creation of a target architecture. Those actions enable planning, acting, controlling and documenting through all layers. The execution of an analysis decomposes the analyzed object in its components. Those single elements are examined and evaluated as well as the relationships and interactions between them. Furthermore weak points can be revealed, new advantages be discovered and various design alternatives be evaluated [ZAA11]. However, creation of an EA target model is time and cost consuming. Thus, analysis support is essential in order to generate value from an EA model. Support for decision making and planning generates value and increases the acceptance of the EA analyses initiative in an organization [Lan05].

The focus of every analysis depends on the analysis type and the required goal type. [SK11; Lan13] If the analysis is not developed with the right requirements or goals and then is used, misjudgments occur and failed projects may result. [BIT11] Here, the decision maker has several alternatives available and must weigh them against each other to determine which analysis to use. [Ull+10] Additionally the questions of what is feasible and what is desirable are crucial [Joh+07b].

The process of analysis can be segmented in different phases and activities. The parts *system thinking*, *modeling*, *measuring*, *satisfying*, *comparing with requirements* and *comparing alternatives* are used in [WC12] to identify the characteristics of analysis categories. Another approach proposes, for example, the division into quantitative, functional, analytical and simulation. [BMS09]

In current literature a great variety of analysis possibilities is described. These approaches are mainly isolated and integrated ones are rare. The well-known EA frameworks deal only secondary with EA analysis and a common understanding of the term is not established yet [BMS09; NBE12]. Nevertheless due to the importance of EA analyses, methods are required to specify them consistently [BMS09; JNL07]. The analyses rely on different technologies, like ontologies ([SKR13]), probability networks ([När+08]) or expert interviews ([KMP11]), have different preconditions and provide different kinds of results. For example, preconditions can be required properties for model elements or specific data structures. Typical results are quantitative ones like an overall metric for the architecture, measures for specific architecture elements, but also a determined set of elements. Several analysis approaches focus on the dependencies between the elements of an architecture. For example, they are used to determine the impact of changes (e.g. [Boe+05a]). Other analyses focus on specific attributes of elements. For instance, an availability analysis predicts the availability value for an element, dependent on several other factors modeled in the EA [NBE12]. Accordingly metrics based on attribute values can be calculated. They can be used as KPI for the evaluation of the architecture and for decision making [Mat+12]. In those calculations the relationships between the elements as well as the attributes of dependent elements can be considered. The analysis of timing aspects is very important for EAM. Therewith the evolution of the architecture can be monitored [Mat+12]. Every analysis supports a different goal, and thus, for a sound evaluation of the architecture different kinds of analyses are required.

2.4.2.1. Bayesian Belief Networks

Many architecture analysis techniques use logic and formulas of a Bayesian Belief Network (BBN) which is an often used and proved concept as it is not restricted to a specific field. Therefore, approaches of plenty research fields use or describe BBN, like [KN11], [SSH+15] or [CHX17]. But also EA uses this technique for its analyses, as e.g. [Hol+09]. BBN are applied for analyzing highly complex networks, because they are probabilistic, graphical models which are used to represent and calculate the conditional probabilities and dependencies of model elements and its variables. [Nea+04]. Therefore, they focus on making impacts of branched networks visible and traceable.

Important characteristics are directed and acyclic edges, representing dependencies between ancestors and descendants nodes which depict random variables (X) with discrete states. The direction of edges present their causality of connected nodes, and therefore, their relations. [KN11] Depending on the state of ancestors, conditional distributions are defined for each node respectively variable: $P(X_i \mid \text{ancestors}(X_i))$. Its probability that a value occurs is called belief. A Directed Acyclic Graph (DAG) can be defined as Graph $G = (V, E)$ where V is a set

of variables and E is a set of edges. [Nea+04]

(G,P) is defined as Bayesian Network if it complies with the Markov condition. This assumption describes relations between graphs and probability distributions and aims in handling efficiency of large instances. It states that each variable is conditionally independent of its non-descendants given the set of all its parents. [Nea+04]

Conditional independence is defined with A, B and C as example variables of V :

$$(A \cap B | C) = P(A | C) * P(B | C), \text{ A and B are independent given C with } P(C) > 0 \quad (2.1)$$

With Conditional Probability (CP) defined as:

$$P(A | B) = P(A \cap B) / P(B), \text{ probability of A given B and } P(B) \neq 0 \quad (2.2)$$

Whereas dependent nodes are defined with:

$$P(A \cap B) = P(A | B) * P(B) \text{ or } P(A \cap B) \neq P(A) * P(B) \quad (2.3)$$

And independent nodes in conditional situations:

$$P(A \cap B) = P(A) * P(B) \quad (2.4)$$

This results in the Bayesian Theorem which states that if there are two events of nodes, the probability of A under the condition that B has occurred can be calculated by the probability of B under the condition that A has occurred:

$$P(B | A) = P(A | B) * P(B) / P(A), \text{ if } P(B) > 0 \text{ and } P(A) \neq 0 \quad (2.5)$$

These variables of nodes have possible sets of values. To determine the corresponding conditional probability distribution of every node and its values in the graph, a Conditional Probability Table (CPT) has to be defined. This table contains all possible combinations of diverse states of ancestor nodes to determine the probability of these combinations. Thus, to determine the likelihood of being in a specific state whereas the ancestors are in another specific state. [Joh+07b]

As described, these concepts are often used for large and complex networks. This is mainly based on the fact that not all connections have to be considered at once. Only states and values of ancestors have to be considered or estimated. Therefore, only small models are concerned in order to reduce the complexity. To consider the entire network, one needs global Joint Probability (JP) distribution. [Nea+04] This is obtained by factorization of CPTs:

$$JP(X) = \prod_{i=1}^n P(X_i | \text{ancestors}(X_i)) \quad (2.6)$$

Figure 2.11 shows an example of such a model based on this concepts. The model

shows the health status of a patient and its influence on a test result.

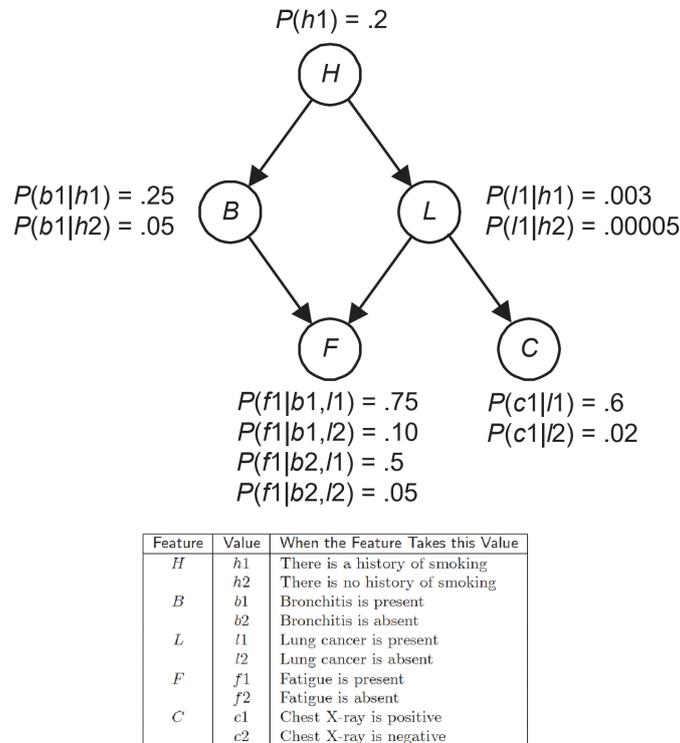


Figure 2.11.: Example of Bayesian Belief Network [Nea+04]

For example, the history of smoking influences the presence of lung cancer. This direct influence is represented by a directed edge. The same applies to result of the X-ray which is determined by lung cancer. However, there is no edge from history of smoking to X-ray results as it is only an indirect influence. The figure also shows multiple states of each node and conditional probabilities of each state given every combination of values of ancestors. These options are often presented in a CPT. The probabilistic inference can be calculated by the Bayesian Theorem. For example, the conditional probability that the patient has bronchitis can be determined if it is known that the patient smokes, is fatigued, and has a positive X-ray. [Nea+04]

The majority of use cases for BBN are networks with questions about conditional probabilities of nodes with different possible states, e.g., the determination of correctness of a disease test [Nea+04]. Depending on the leading question a model can be analyzed with different approaches and formulas, as long as the dependent probabilities or independences of graphs are to be considered. Therefore, the BBN concepts can be applied to different architectural analyses with varying objectives.

2.4.2.2. Extended Influence Diagrams

Extended Influence Diagram (EID) is another methodology used in many analyses in EA at the architectural level. These diagrams require logic which is based on BBN fundamentals and is a graphical and mathematical representation of probabilities that contribute to visualization and interpretation. [Joh+07b] In the meantime, there are several applications for EID, but first approaches to use EID for architectures in EA areas are work of [Joh+06; LJN07]. The approaches can check different variables, but are always used as a decision support to weight different alternatives.

These diagrams are directed graphs with connected directed edges and nodes. Each node corresponds to an attribute with a probability value that depends on the values of its predecessor nodes. [Joh+07b]

As EID is an extension of simple influence diagrams, their notation is reused. Figure 2.12 presents this notation. There are three different node types. Goals of decision makers are depicted by a utility node (*V*). All elements in the diagram that have a causal influence on each other are chance nodes (*C*). For this purpose, the causal relations are additionally used. [Joh+07b] To compare different alternatives decision nodes (*D*) are added. Arrows that indicate time precedence are called informational relation. [LJN07]

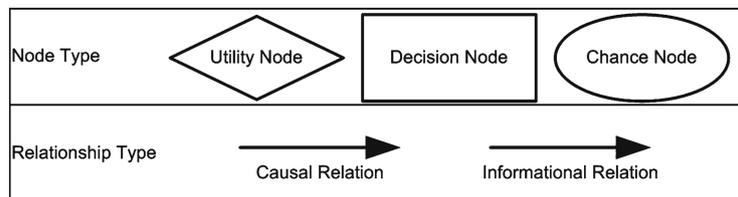


Figure 2.12.: Generic influence diagram notation [Joh+07b]

As described, each node (*i*) corresponds to a variable. The probability distribution (*Pr*) of these variables are stored in an attached CPT. [Joh+06] This lists all the possibilities of what the probability distribution can be, depending on predecessor: $Pr\{x_i|x_{CP(i)}\}$ [Joh+07b]



Figure 2.13.: CPT of two nodes of an EID [Joh+07b]

Ordinary influence diagrams are merged, and thus, allow node collapse or node expansion to obtain an EID. Thus, certain facts can be considered more precisely

or a higher abstraction can be determined. [Joh+06] Accordingly, EID is defined as

"a set of graphs, $E = \{G^1, \dots, G^n\}$, related by expansions and collapses of the definitional structures, where the expansions and collapses are dictated by specific transforms, $G^j = coll(G^i)$ and $G^i = exp(G^j)$ " [Joh+06]

In figure 2.14, G^1 presents a fully expanded diagram which is collapsed in G^2 on vertical level $a_1 \dots a_m$. This is a minimalist and generic example. However, EID also include other specific notations. A distinction is made between defined and undefined chance nodes. The list of defining nodes is specified by $DF(i) = \{j \in N : (j,i) \in \Delta\}$ where $i \in \{V, C\}$ and Δ is the semantics for definitional relations. [Joh+07b] These relations are simple aggregation of their elements and use the same conditional probability distributions as causal relations: $\Pr\{x_i|x_{DF(i)}\}$ [Joh+06; Joh+07b]

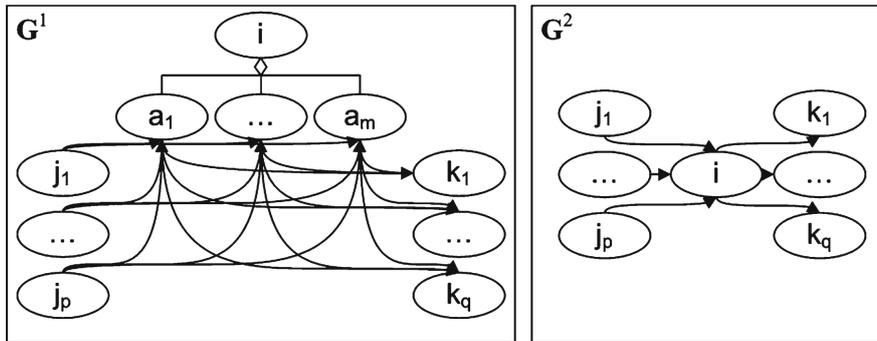


Figure 2.14.: Node collapse [Joh+06]

In addition, chance nodes can be divided between controllable, semi-controllable and uncontrollable. This distinction is used to show whether the decision node is independent or dependent. [Joh+06] A further distinction can be made for chance nodes as to whether the predecessors are causal or informational ones. Causal predecessors are defined by $CP(i) = \{j \in N : (j,i) \in K\}$, where $i \in \{V, C\}$. Informational predecessors are covered by $IP(i) = \{j \in N : (j,i) \in I\}$, where $i \in D$. [Joh+07b]

2.4.3. EAM and IoT Overlap

The digital transformation of enterprises is forcing many units and departments to rethink their strategies. [Zim+18] IoT is increasingly being used not only in industry parts, but in all areas of an enterprise and its management tasks. Due to the numerous overlapping points, the combination of both concepts is feasible. [Age16]

Often, IT systems now have to cope with small distributed structures which have to be adapted often, as frequently found in IoT applications. As the world and associated systems have become open-model minded, enterprises will have to become more flexible in the future in order to keep pace with digitalization. In particular, depending on the use case, customers will interact with IoT devices to interact with specific business processes. Corresponding digital services and business models must be created that create financial value for companies. [Zim+18] IoT approaches are expected to help keep pace with fast-changing markets by enabling service-oriented systems to migrate to IoT services and collect more data. This data is generated through cloud computing to knowledge. In detail, these approaches are to be integrated into EAM by creating and monitoring small EA descriptions, each of which is assigned to an IoT object. [Zim+18; Zim+15] The further development of ArchiMate, an open and independent modeling language for EA, is helpful in this context. ArchiMate enables the mapping of technology behavior concepts, and thus, the representation of sensors and other connected devices. [Groa]

To enable usage of IoT devices in EAM, device management functions must be adapted. [Zim+15] Mainly, compute nodes need to be involved to process data from the IoT devices in corresponding analytic systems to match the distribution of business logic of IoT systems which is not located in one layer as in traditional business systems. [Mic18a] Guidelines were developed on how technical IoT benefits can be translated into business goals and which platform is best for the individual EA-IoT initiative. [Age16]

To make EA IoT-enabled, there are IoT architecture approaches that directly provide a pillar for business components, enterprise domain specific applications and decision support systems. These include cross-platform interfaces for end users such as UIs for big data, BPM, SOA platforms, government applications and operations. Such connected enterprises enable IoT operations like monitoring and deployment via an IoT layer that is directly connected to the customer experience and application layer. With additional enhancements in infrastructure, network and sensing layers, costs can be saved, employee productivity be improved and customer requests handled. [Dig17]

TOGAF offers possibilities to connect enterprises and IoT. Through this framework, all phases of EAM are covered, such as design, planning up to maintenance. Among other things, structured methods are used to determine requirements needed for architecture development. [Grob] IoT standards Open Data Format (O-DF) and Open Messaging Interface (O-MI) have been introduced to connect IoT data and components with existing enterprise components to collect data easily. Among other things, focus is on reducing costs of equipment monitoring. These data standards are intended to ensure that information is presented in a way that is meant to be understood in IoT-affected systems and to transport payloads in any format, such as data about lifecycle events. [Sim21]

In addition to all the commonalities and approaches in research, there is also a delta between IoT and EAM that needs to be dealt with. The most important points to consider are the open IoT systems, which do not always fit in with corporate security policies, and the associated new threats from the Internet. Furthermore, traditional layers in rigid enterprises are not always compatible with new IoT layers and do not provide an interface to new IoT components. In addition, there are usually more stakeholders involved in an IoT system than in traditional systems due to the remotely distributed, connected systems.

Part II.

**SAFETY AND SECURITY
DESIGN APPROACH**

3

Safety and Security Architectural Requirements

This chapter represents the initiation of the optimization process by observing architectural conditions to create IoT models suitable to depict and analyze safety and security aspects. Findings and results of this chapter form the basis for chapters 4, 6 and 7.

An IoT model can be represented in different ways, depending on viewpoints and possibilities offered by its associated meta model and modeling tool. In anticipation of the optimization process and its included flaw identification and assessment steps, a meta model has to be designed that incorporates appropriate aspects. To identify these aspects, typical safety hazards and security threats are considered in the following to determine their architectural roots. From this, it can be concluded which information a model must be able to contain and depict in order to recognize and prevent flaws. The main focus is on hazards and threats in IoT-WD systems and their specific challenges. In order to evaluate challenges in groups rather than individually, and thus to identify common measures and aspects, various categorization options are considered, which are taken up again in the flaw identification process. Finally, architectural requirements are derived from this chapter, which are used in the subsequent IoT-WD modeling approach.

3.1. Challenge Review

The challenge review is divided into three parts around safety hazards, security threats and their specific concepts in wellbeing systems. To enable the detection of architectural aspects, vulnerabilities' roots are the focus of this review to create the basis for section 3.3.

3.1.1. Safety Hazards in IoT Systems

Classic safety hazards are unintentional and occur, for example, due to misconfigurations or overheating. However, a safety hazard can also be a byproduct of a security attack that has consequences for living beings. Especially in IoT systems, hazards can be versatile, as they are not bounded by physical distance limitations and can be affected by the outside. [Zal19] In the following, unintentional failures are considered.

One way to distinguish safety hazards is by type. Physical, electrical, biological but also service failure hazards are possible among others. [Som11] However, the most important aspect is the origin of failures. In many cases, various fault types that exceed their limit value are responsible for a failure and its associated accident. A study by [WK01] identified multiple fault classes and their probabilistic distribution of fault sources:

Fault Classes: Calculation, Logic and Algorithms

The most common fault types are calculation factors and related algorithmic problems. These include overflow problems, improper handling of boundary conditions, data structures and abnormal conditions such as electrical noise. But also truncation or rounding errors. Closely related and also a common reason are logic problems. Incorrect data control logic, data validation, exception handling like power failures or wrong behavior of component interactions are all reasons for this fault type.

Fault Classes: Software Changes and Countermeasures

Further hazards can be caused by change impacts if, for example, no verification is performed against the original design specifications. But also functional losses due to updates and incorrect conditions due to incompatibility of different versions or relationships can be causes. Similarly related are countermeasures as a fault type, as this can cause incorrect configurations especially in remote areas, incompatibility of new components or a reduced upgrade scope. In addition, software can impact devices or other software components if interfaces are not able to communicate anymore. One hazard that causes this is device interoperability. While simple construction errors can lead to mismatched hardware components,

software incompatibilities are often caused by updates. For instance, new services or standalone functions may change and subsequently be unable to process data. [WK01] The update errors can stop entire devices or system areas ([Zal19]) or lead to software malfunctions, e.g. hazardous communication interruption which leads to missed message transmission in emergency situations.

Fault Classes: Data and Documentation

Other issues can be caused by invalid or inconsistent data and database corruption, excessive use or human misuse. But also omission faults due to missing documentation or missing important functions due to no requirements traceability or non-compliance with standards are critical.

These different types result in specific hardware and software failures that can lead to serious accidents. In the hardware area, there are battery failures of important components, electromagnetic interferences, device incompatibility and sensor malfunctions, among many examples.

Fault Classes: Hardware Issues

IoT devices are mostly equipped with rechargeable batteries, using batteries with a reduced size, weight and power consumption compared to conventional energy sources. [Har19] However, there is risk of capacity drop, less energy supplied than planned or increased consumption. Therefore, a battery management unit is required which is part of a rechargeable battery system together with the cell and charging circuit. [Exp18] Electromagnetic interference, on the other hand, is an even more complex safety problem. If many devices are used in one place with inconsistencies in their frequencies, their communication capabilities or other functions may be affected, even to the point of failure. Responsible factors can be electromagnetic fields, their compatibility and distance. [Alb19] However, equipment failures are not always due to energy or interference problems. Components manufactured at too low a price, neglected maintenance or overuse can lead to wear and tear. Even if not whole components fail, even small parts such as sensors or controllers can become a major hazard if, for example, measurements fail or are distorted. [KR09] Sensor malfunction is one of the common safety issues in IoT systems, as they occur in large quantities and offer many distributed error sources. [Zal19] In addition to attrition, overheating is a frequent trigger.

Not all of these fault types and examples can be prevented in the design phase, but can potentially be detected and mitigated on this point by planing suitable countermeasures. Derived architectural requirements are listed in section 3.3.

3.1.2. Security Threats in IoT Systems

The opposite are intentionally exploited vulnerabilities to create security threats to harm the system. The range of options is broad and varied, as offline threats

are just as possible as cyber attacks. While attackers had a hard time in the past, the connection to the Internet offers numerous opportunities to launch an attack. A selection of these threats is considered for this review.

The numerous attack opportunities are often based on major violations of security policy or recommendations. OWASP identifies the most prominent problems each year to create awareness. Weak, hardcoded or easy-to-guess passwords that are publicly known, rarely updated or determined by light bruteforce attacks are always involved. Especially if the system is connected to the Internet, the circle of offenders becomes larger and the risk of unauthorized access rises. [BSI12; OWA18] Further unauthorized remote access can be caused by insecure network services. Insecure services or redundant functions that provide unnecessary vulnerabilities are to blame. [BSI12] But not only insecure services, also interfaces are an entry point for attacks. Web, backend API, cloud or mobile interfaces can allow unauthorized access to devices. This is often due to authentication or encryption problems. Affected are mostly insecure or outdated components with customized areas or already compromised third-party software. [BSI12] However, when these vulnerabilities are identified, they cannot always be fixed immediately. Reasons are a lack of secure update mechanisms which lead to updates that are not delivered securely. In addition, anti-rollback options and firmware validation is missing. Often, updates are simply applied too late or are unknown. Privacy violations, insecure data transfer and storage can happen, if devices are compromised that do not include a device management with security support. This is mainly caused by lack of encryption or access control of sensitive data. These vulnerabilities are often introduced by bad default settings that do not provide any possibility to raise system's security level. [BSI12; OWA18]

These security problems lead to the numerous attacks mentioned above. A distinction can be made as whether they are directed against physical objects, protocols, data or software components. Physical objects including hardware components, are attacks on the lowest architectural layers and refer to objects with restricted and unrestricted access. Protocol attacks are directed against communication between different components and entire networks. Thus, multiple layers can be compromised and include active data. Data at rest, which is permanently stored on a computer or cloud storage, is threatened in data attacks. All other attacks are based on exploiting software vulnerabilities and attempting to breach them. IoT software includes any application, services, firmware, operating systems, gateways and other software parts used in IoT networks. [AKM18] However, this is only a rough division. Threats can never occur in isolation and always require multiple aspects and often have multiple targets.

Physical Object Attacks

IoT physical objects are as diverse as possible attacks and threats are numerous and varied. A common attack procedure is tampering. Thus, data streams or entire devices are manipulated. Real physical modifications can be made or a com-

munication link can be exploited that lead to accessed hardware elements or violating CIA security requirements by stealing identities. This enables injection attacks that allow distribution of malicious software and disclosure of pre-installed encryption keys or other security relevant information. [Var+17; BTM15] Node capture or manipulation is mainly performed on low power devices with missing security standards or encryption lacks, vulnerable to attacks and easy to steal their identities [Has+19]. Node injection, on the other hand, introduces a completely new node and places itself between two nodes in order to interfere with, intercept or change their communication. The new node should appear as part of the system. Since several manipulated or newly injected nodes are used for this purpose, identification is often complex. The subsequent code injection aims in obtaining control or information. [DV17] Furthermore, duplicates of devices can be used to gain trust and subsequently enter confidential networks as well. In this process, nodes are cloned and presented to other nodes as known communication partners. [Var+17; BTM15] The required data can be obtained via further eavesdropping methods by intercepting data in different phases [Has+19; MJ16]. The linked tampering of communication is often done in connection with Denial of Service (DOS) attacks. Communication is distorted and traffic is increased until availability of the system or services can no longer be guaranteed. [Var+17; BTM15] In WSN node jamming is a common method for DOS attacks as well as sleep deprivation [DV17]. It is tried to drain battery life by running infinite requests which seem to be legitimate. In comparison, there are attacks that do not directly target nodes or their communication. So-called side-channel attacks are directed against information that the device discloses besides. This can be electromagnetic signatures, energy consumption or other private information that can be used to draw conclusions about the user or state of the device. [Has+19; MJ16] Lack of encryption of data to be transmitted allows these attacks to occur. This can have far-reaching consequences. For example, if IoT networks with high energy consumption are identified, attacks could be launched to take over the system and abuse it for MadIoT attacks. These are attacks to shut down entire power grids with increased energy demand. [SMP18]

Protocol & Active Data Attacks

Further attacks are increasingly directed against protocols to attack network communication, middleware aspects and gateways. Active data is intercepted in data transit attacks when it is being transferred from one device to another or from one IoT application to others. Data can be stolen or manipulated in the process. These thefts are often preceded by access attacks in which intruders gain access to the network and can remain undetected for a long time. The focus here is on identity theft and authentication misuse. However, wide-ranging SQL injection attacks can be used to access or manipulate data in applications. [Has+19] In addition, DOS attacks are also a suitable method for intruders to threat network aspects. Collision attacks are similar to jamming and aim to reduce the goodput of a system or prevent communication in wireless communication. Another way to manipulate communication can be spoofing of routing information. In

this case, unencrypted routing and header information is intercepted and used to produce routing loops, incorrect routes or fake messages. Similarly, selective forwarding involves dropping individual messages in order to prevent the entire message or all the information from arriving. Sinkhole attacks can be used for this. Some nodes appear to be preferable for routing, and therefore, arrive at a sinkhole node that drops the message. Other DOS attacks can be wormhole or sybil attacks. Wormholes intercept messages, take them to other entry points in the network and replay them. Sybil attacks, on the other hand, fake multiple identities for a node in order to have more rights in voting or to get more resources allocated. These are often starting points for further flooding attacks. [Var+17] If the attack is not primarily aimed at overloading the system, but instead focuses on access to data, this is also referred to as man-in-the-middle attacks. Data communication between two nodes is intercepted in various ways. Eavesdropping intercepts data, modifies it and replays it back into the network. In this way, routing attacks can be carried out. In some cases, data that could not be decrypted is also intercepted and replayed, since the replay can later be mistaken by the receiving nodes as trust evidence and authorize the sender. [Var+17; Has+19] Attacks on networks, communications and their protocols also have an impact on gateways. Man-in-the-middle and eavesdropping attacks are particularly dangerous, as encryption keys can be captured. Only complete end-to-end encryption can guarantee data confidentiality. [Has+19]

Attacks on Data in Rest

Data at rest that is stored in clouds for later analysis is target for attacks if cloud service providers do not implement sufficient protection measures. Back-door attacks or password guessing can lead to access to the entire cloud infrastructure. But also unencrypted data, compromised virtual machines, accessing devices from an insecure terminal and portability of data can lead to threats. [Var+17]

Software Attacks

Threats for IoT applications or software can occur on all spots of applied software or services, and therefore, are scattered and never isolated. In particular, data theft is closely related, as applications collect, share or further process software. Thus, encryption measures, privacy management and access controls are needed. [Has+19; Var+17] Authentication attacks are part of almost every IoT software threat and enable not only access but also false repudiation, unauthorized interception and falsification of trust levels. [ISO11a] If access is compromised, further attacks can follow. For example, DOS attacks can trigger service interruption attacks and prevent legitimate users from running needed services. [Has+19; Var+17] Furthermore, applications can be modified by injected code or services by launching phishing attacks, viruses or spyware [DV17] .

3.1.3. IoT-WD Specific Challenges

The presented safety hazards and security threats can occur in any IoT system. However, even if the attack types are roughly the same, IoT-WD systems include other devices. Thus, further extended or additional threats or hazards can appear in IoT-WD networks. In this area, safety hazards are often associated with vital sensor malfunctions. Important data for subsequent analysis processes may not be fully collected, e.g. missing heart rate sensor measurements for long-term atrial fibrillation monitoring. In addition, actuators may cause serious consequences due to failures. For example, an insulin pump could take false readings due to wear and tear and suggest its actuator the daily maximum dose. This means that the user can no longer rely on the pump and must set up the dose via remote control. [Som11] A safety hazard can arise as a by-product when a deliberate attack is launched on an IoT-WD component. Spoofing attacks can tap or alter, e.g. node measurements based on the saturation level of sensor input and output, which lead to control over infusion rate and possible under- or over-infusion. [Par+16] Neglected maintenance could also affect the wellbeing devices' monitoring unit [Som11]. Hazards are particularly critical when alarm signals are affected because too many false alarms cause them to be ignored [BT10]. However, wellbeing systems, as described, are designed for preventive and self-monitoring devices rather than for treatment of diseases. Thus, security threats in IoT-WD systems are more pronounced and considered, since they focus on sensitive, identifiable (PII) and secret data which are highly valuable for intruders.

Degree of Separation

The severity of threats in IoT-WD systems depends on the degree of separation. Several levels can be distinguished. The least dangerous levels for human health are attacks directed against components that are outside the user, do not touch and are only concerned with the required environment of the system. More dangerous are attacks against components that do not touch the user, but still have vital functions. Most critical are devices that touch the outside of a human body or are even attached to it. For example, an implanted hearing aid is critical for DOS attacks if the user cannot hear anything in traffic. Important or valuable data can be tapped at all degree of separation levels. [All20]

Attack Targets in Wellbeing Systems

A study by TÜV Rheinland dealt with this topic and examined the danger posed by connected infusion systems. Unprotected networks enabled authentication and subsequent manipulation of the dosage. The reasons for this were public and private keys stored as plain text. [SC16] Transmitters of cardiac devices can also be a target of attack. Data could be remotely queried without authorization, rapid battery depletion could occur or device functions could be changed. [FDA17] However, more traditional components of a wellbeing system, such as wearables and smart pillboxes, are also affected. Wearables mainly track data

for private use. However, they are no less critical, as the data could be used for further consultation with physicians or the data could be transferred to health insurance companies for bonus systems. Moreover, an IoT system is only as strong as its weakest link that, thus, has to be protected [RV16]. Wearables can be the gateway to far-reaching threats due to their weak encryption or inadequate authentication methods. Data can be intercepted or modified during transmission to the cloud, and illegal device pairing is possible. This enables further gateway attacks, sending false queries to servers or obtaining sensitive data from devices. [Lee+16] Video monitor systems are often used in smart homes, explicitly to monitor babies. However, these video systems are often cheap and lack security measures. The local and remote communication is not encrypted and in addition light passwords are often used. This allows attacks to follow the video streams and not only violate privacy. Information about the house could also be leaked. [SB15] If these monitoring systems are connected to other smart objects, such as a smart mattress, the attacks can spread even further. Another example can be smart pillboxes. They are designed to remind residents to take their medication and prevent overdoses. In addition, tablets can be reordered automatically. If these devices are attacked, damage to health and finances can result. [MA18] Many of these threats are enabled by the use of unprotected gateways or not updated mobile phones in smart homes. Often, users of these smart systems are not aware and allow malicious access to their devices, which then spreads to their sensitive wellbeing devices. In most cases, the presented software attacks are used. [Gen+17]

3.2. Challenge Categorizations

Since meta model adjustments (Chapter 4) cannot be made for each individual vulnerability, commonalities have to be determined in order to group hazards or threats. Since it depends on the point of view which characteristics should be used for the groupings, different categorizations exist. This overview is taken up in chapter 6 to identify challenges depending on their categories.

Point of Origin and Attack Goal

The first step in detecting attacks or hazards is to determine the point of origin. It can be distinguished whether it starts from the outside, and therefore, is mostly an issue through an Internet connection, or whether it happens locally or internally. Here again, a distinction has to be made whether internal employees or the company's own components are responsible, or whether external units are infiltrated. For externals, it can be considered whether clouds are the place of occurrence or it happens on edges. To further distinguish the attack goal, a division can be made between information disruption by disturbing the flow of information, host attacks by exploiting the properties of host hardware and software, and network attacks by targeting the transmission media. [Isl+15] On the other hand, the attack goal can be a manipulation attack to change the decision cycles of the system, a capture attack to gain control over hardware, software or data, and disrupt-degrade-deny-destroy attacks to compromise or damage the attacked system. [CC13]

Tolerance Level

Regardless of whether safety or security challenges are considered, a distinction can be made according to the tolerance level, and thus, how challenges are handled. Based on [Sar16], three levels can be defined. Devices with a high tolerance level are the least critical for hazardous situations. Longer disturbances are tolerable for the system and they can be restarted occasionally. Devices with a temporary tolerance level can withstand short-term faults, but must be repaired or restarted as soon as possible. Devices with a no-tolerance level are critical for the system or health of users. Therefore, they must never fail. The potential danger posed by an attack or hazard on an IoT-MD system is directly dependent on the type of device involved.

Element to protect

In addition, the IoT-A project proposes a consideration and subdivision of the elements to protect. The physical person is named as the most important element. This is followed by a consideration of the user's privacy in connection with it. Communication channels and leaf of devices are named for the more technical considerations, as these are often the weakest and most dangerous elements. Further classification can be made into intermediate devices, backend functions, general infrastructure elements, services and facilities. [Bau+13]

Severity

Another important subdivision is severity. The safety classification of the IEC 62304 standard defines three classes. Class A: No injury or damage to health is possible. Class B: No serious injury is possible. The most critical class C includes hazards that make serious injury possible. [IEC06] Pure IoT-WD systems can use this severity classification to exclude devices whose safety cannot be guaranteed as a result of a hazard. Other approaches are used to categorize security threats. In comparison to safety, a separate severity classification is used for threats. Based on [Hou16], five levels are classified. Catastrophic threats include wide-scale threats that are critical for the entire infrastructure. Critical threats, on the other hand, have a significant impact but don't paralyze the entire system. Marginal threats are noticeable but may be tolerable in the short term, while negligible threats have little impact and require less urgent countermeasures. Finally, threats can occur that are unlikely to have an impact.

Safety Classification

Moreover, categorizations explicitly for safety challenges can be defined. These include the fault classes of [WK01] already described above. The subdivision helps to distinguish how faults could arise. For this purpose, 342 faults were considered and divided into classes. Medical devices were explicitly used, since these are a highly critical consideration for the danger to humans. In addition, a classification of the resulting hazards can be derived from the fault classes. Five broad and overlapping categories were considered useful for this dissertation. System corruption includes all hazards that results in the inability to deliver the basic functions of a system. Loss of service considers hazards for individual services and areas of a system. In this context, outages are another hazard category, in addition to external problems that cannot be directly influenced. Here, hazards are considered which contain different reasons for failures. No matter if hardware or software related. Last, data corruption includes hazards that can cause accidents based on faulty data. [WK01]

Security Classification

All the threats presented can be active or have an effect at different points in the system. However, the basic ideas and methods of attacks are similar, and can therefore, be divided into groups in order to make countermeasures generally definable. One of the best known approaches is the **STRIDE** classification, which is used in large projects such as IoT-A [Bau+13]. **Spoofing** is disguising or faking an identity to deceive in local attacks. **Tampering** is responsible for manipulation of data in transit, and can therefore, change databases by storing modified data. **Repudiation** is the denial of an act in order to make it more difficult to determine where instructions came from. Unauthorized persons who can view data streams or databases represent a threat type **Information disclosure**. **DOS** threats occur when an attacker can degrade or deny services. The last subdivision is **Elevation of privilege** threats. These include attacks that can unjustly increase privileges, enabling e.g. anonymous user same rights as a local user. [Mic07]

3.3. Derived Architectural Requirements

After threats and hazards have been examined, suitable specifications in the design phase in order to avert challenges can be inferred. These aspects are, thus, basis for the meta model in chapter 4. Furthermore, derived requirements are also necessary for the subsequent architecture optimization through architecture analyses of chapter 6 and 7. Only architecture-based safety and security requirements are included. Other requirements and meta model specifications are discussed in the next chapter. The requirements are divided into six areas.

Concept	Description
Layer	Each component must be able to be assigned a layer
Location	Components require an indication of their location
Redundancy	Multiplicities should not have constraints
Redundancy	Internal and external units should have redundancies

Table 3.1.: Generic architectural requirements

The requirements from table 3.1 are generally valid for all elements of a suitable meta model. By assigning layers, layered protection can be enabled and each component can be assigned to specific security measures. Devices with a special protection need may only occur in suitable locations or require location tracking to prevent unauthorized physical access [BSI12; Cor21]. Furthermore, certain components must not be found in the same location as they negatively influence each other. A certain degree of redundancy can compensate for failures of critical elements, e.g. storage systems, services and physical components.

Concept	Description
Entities	Different physical entities must be compartmentalized
Entities	Hardware and software information must be modelable equally to deposit all known information
Wellbeing entities	IoT-related wellbeing entities must include at least the same protection as IoT things
Sensors	Sensors require separate protection in IoT systems
Foreign components	Entities need a registration facility and trust level
Component attributes	Documentation has to be attached at the component
Component attributes	Components need uniquely assignable identifiers
Component attributes	Default password information must already be available in the model

Table 3.2.: Component architectural requirements

Components in IoT systems are subject to special architectural requirements (table 3.2). Physical entities have a key role in IoT systems. Therefore, a modeling segmentation must be possible, depending on whether they rank as an IoT device or as a small element. Each component must be modelable, as each can have a role in a hazard or threat. In addition, its hardware and software related information - as type, configuration, firmware - must be filled in as this can be weakness related [Cor21]. Wellbeing devices must contain all these information as well to guarantee same protection level or in special cases even more protection. Sensors, with their high significance, must also have special modeling capabilities. Foreign, un-registered devices must be able to be modeled. For this purpose, a meta model must consider registration activities and a trust level against man-in-the-middle attacks. To provide as much protection as possible, it should be possible to attach documentation and standards to elements to have their requirements available [BSI12]. But also clear identifiers against node cloning and secure password default requirements must be specified already in the design phase [Fou16].

Concept	Description
Service type	Individual operations for each component must be modelable
Service quality	Services must provide interfaces and interoperability capabilities
Service communication	Services are connected by physical connections
Service attributes	Services must be assigned trust attributes
Software condition	Rules for different states should be defined as early as possible
Active data	Anonymization or pseudonymization methods of the services must be included
Data in rest	Permanently stored data must be protected at the storage location
Data records	Recorded data must guarantee integrity from measurement to storage and usage

Table 3.3.: Software, service and data architectural requirements

Many threats and hazards are software-based, and therefore, cannot be fully addressed in the design phase. Table 3.3 shows architectural requirements to provide services and their software with mitigation options. Each component in IoT systems must be assigned a service and made modelable. Different service types must be supported in order to cover different operations. Service quality may also be verified in the design phase by modeling different quality attributes, interfaces and interoperability properties. Connections must be modeled, including encryption and authentication of the physical connections. Various trust attributes such as trust level and Service Level Agreement (SLA) shall be included as well as

timelimit information to prevent DOS attacks. [Fou16; Bau+13; Cor21]. By specifying rules and metrics for different software states or boundary conditions in the model, later false activities are prevented. An example can be the use of timestamps or time synchronization [Bau+13]. Anonymization and pseudonymization methods for data used by the services must already be specified. Regardless of whether data are active or in rest. [Cor21] In addition, data measurements and data records require attributes, like timestamps and data units, to prove data integrity.

Concept	Description
User management	For separation of duties, users must assigned a role
Access control	Every resource must be part of access control
Access control	Authorization and rules must be connected
Right management	Authentication methods and authorization rights must be specified in models already

Table 3.4.: User and authentication architectural requirements

In models with different users, user management must be included to control the rights, tasks and roles of the users [Cor21]. The least privilege principle applies. Thus, everyone can do only at most what he should. [Fou16] No resource should be freely accessible, and therefore, only accessible with a suitable authentication method and matching users or roles [AKM18; DD18; Bau+13]. The assignment should already be done in the model (table 3.4).

Concept	Description
Connections	Connections between components and services are part of the model
Secure communication	Secure communication details must be modeled
Networks	Trust level of included networks must be integrated
Networks	Network related attributes must be modeled
Clouds	Cloud accesses from different networks must be made via a secure access point
Encryption	Encryption methods must be part of the model
External communication	External communication options or connections have to be architecturally secured
External communication	External connections must be represented by special components

Table 3.5.: Communication and encryption architectural requirements

Table 3.5 contains conditions for depicting secure communication. Most important aspect is to include all connections in order to capture dependencies and

possibilities of impact. Communication protocols, authentication and encryption methods and other security relevant communication attributes have to be defined [Bau+13]. Used networks must also be modeled including a trust level and all related information, like IP addresses or DNS server [Cor21]. The meta model must provide means of recording the encryption methods used, including relevant components [Bau+13; AKM18]. To the outside world, systems should be limited as far as possible. However, a connection to external resources and components is often important and indispensable [Bau+13]. These connections must also be modelable. If there is no direct connection, gateways, trusted platforms or other portals used should be part of the model. [BSI12; AKM18]

Concept	Description
Requirements	Requirements of the respective components must be assignable
Requirements	Estimates for CIA should be attached to components in order to be able to perform subsequent evaluations
Assessment	Flexible assessment attributes enable later controls
Assessment	Evaluation of requirements can be used for later error detection, and must therefore, be stored in the model

Table 3.6.: Assessment architectural requirements

Models can already be the cornerstone for later assessments, fault detection or requirements controls [LH94]. For this, special properties are required in the model (table 3.6). This includes the possibility to store different requirements for the individual components and to compare them later with assessment values. Special attributes for the security CIA requirements are also feasible to be specified in the model [Bau+13]. Flexible assessment properties in models facilitate later analyses.

4

IoT-WD Modeling

Once the architectural requirements have been determined in the previous chapter, a suitable meta model can be designed, thus, laying the foundation for further analyses. In the process, not only the safety and security aspects should be taken into account, but also IoT and wellbeing specific requirements. This chapter focuses on *Objective 1*.

To introduce the modeling approach of this work, this chapter first presents an IoT layered architecture, which is necessary for the classification of elements of IoT systems and creates the basis for layered protection. Further architectural requirements are presented before the meta model is formalized and described. The associated ArchiAna tool components are demonstrated by means of a running example. A related work part concludes the modeling approach.

4.1. IoT Layered Architecture

The separation of models and their elements into layers is by no means novel. However, the previous approaches are either highly specialized in use cases or are designed in such a generic way that they can be applied to any network. Therefore, a layered architecture is developed that is suitable for various IoT use cases, and thus, covers all aspects and layers of an IoT system and its required elements and concepts. This is intended to allow IoT elements and their special properties to be considered in particular. For example, the important autonomous and analytical areas can be pointed out or sensing activities can be presented and analyzed separately. Furthermore, the assignment of a layer to each element enables layered protection approaches and monitoring of the appropriate measures for each element depending on its layer. This also allows conclusions to be drawn about entire layers at risk and their effects. Accordingly, the provided layered architecture forms the fundamental point of view for further analysis results.

The IoT layered architecture is subject to the conventional align enable logic and is the first step towards the generic IoT(-WD) meta model. There are nine layers defined, which can be divided into three areas. The color scheme for each layer will be maintained throughout this work and will be used in the later running example. The architecture is roughly based on [Con+16], [Bau+13], [Wu+10], [Mic18a], [AHA13], [Dig17] and [Kha12].

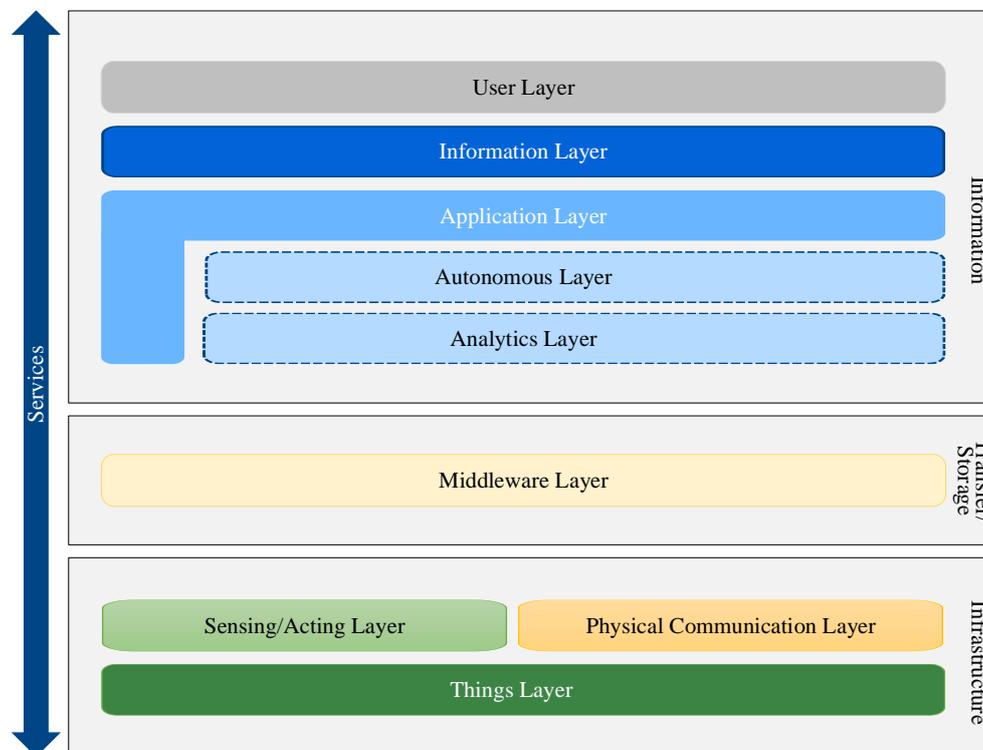


Figure 4.1.: Layered architecture for IoT Systems

Infrastructure

Things Layer

The things layer is the most important layer in an IoT system. Physical objects can simply be classified here, but smart and autonomous entities are also included. Classic IP-based IoT things or smart wellbeing devices also belong in this layer, as do their smaller components like batteries. It is supplemented by the sensing/acting and communication layers.

Sensing/Acting Layer

The heart of the IoT is measuring and responding to measurement results. Therefore, physical entities that measure, like sensors, or respond, like actuators, are placed in a special layer. This means that separate arrangements can be made for this vulnerable layer. But also components for identification, action control or triggers as well as further collecting entities belong to this layer.

Physical Communication Layer

The physical communication layer, or abbreviated communication layer, is assigned to the infrastructure, since the services for the communication sequences are outsourced in the architecture. However, communication is based on physical connections and is, therefore, still part of the infrastructure. In some cases, physical components can also be used for secure transmission, such as a Physical Unclonable Function (PUF).

Transfer / Storage

Middleware Layer

Middleware can be split into service management and data management but is defined as one layer since they are closely related. Services must be coordinated to aggregate, store and forward data sent through appropriate procedures. Therefore, data storage elements are assigned to this layer as well. Typically for middleware, these components are the interface between physical aspects and further logical applications. Through aspects such as authentication methods, this layer can also be crucial for safety and security measures.

Information

Layers that are assigned to the information area are defined in the most detailed way in order to make clear separations. Components of the autonomous layer and analytic layer are optional and can only be defined in connection with entities in the application layer. This implies that components from this layer do not necessarily have to be present. For example, if a use case is modeled that only involves local and temporary decisions and does not include long-term analyses.

Application Layer

Components of this layer are smart applications distributed throughout the IoT network. Applications can be on the end devices, but also in remote components.

Autonomous Layer

Autonomous decisions can take place on the end device if the local measurements are sufficient. This is included in the sensing/acting layer, since no complex calculations are required. However, components with decision units aimed at future actions and long-term analyses are represented in this layer. Appropriate services are crucial thereby.

Analytics Layer

Directly related to autonomous actions or components, is the analytics layer. Also recognizable by the same coloring, since these layers can only act meaningfully in combination. Cloud computing, complicated query execution, and thus, the generation of further insights happen in the analytics layer.

Information Layer

IoT system providers have different business models and goals that influence different aspects of a network. To include these impacts in a model, a suitable layer is created to include these decision components. This idea is based on EAM models.

User Layer

Stakeholders and users are important components of an IoT model, as each stakeholder has different goals and users have different needs and require different security measures. All components in a system that are needed to manage these areas belong to this layer.

Services

Services cannot be assigned to a specific layer, since they usually work across layers or require information from multiple layers. However, there are different service types that can be allocated to specific layers as start or result point. The subdivision can be found in the meta model specification. Based on the color assignment in the ArchiAna tool, the assigned layer is highlighted.

4.2. IoT-WD Meta Model

The depictability of systems is based on a suitable meta model that can represent the appropriate aspects. In this approach, these are architectural IoT-WD aspects and their safety and security information. As described in chapter 1, there is a lack in such a meta model, which is needed for use case modeling and later optimization.

The meta model can be used in three different scenarios:

- 1) A new IoT system which is not deployed yet and shall be analyzed in advance to prevent safety and security vulnerabilities.
- 2) An IoT system extension which is modeled and analyzed beforehand to prohibit new issues caused by potential changes.
- 3) An already existing IoT system is analyzed regarding its vulnerabilities in an As-Is scenario.

Scenario one and two represent the concept of safety/security by design and meet the aforementioned issues by eliminating vulnerabilities during design phase.

The approach presented below lean partial on inputs from the following sources: [Mic18a], [Bau+13], [ISO16b], [Groa] and [Cic+17].

4.2.1. Meta Model Requirements

The basics of meta model development have already been presented in chapter 2 and are applied below. Several requirements shall be met to design an IoT-WD and safety/security specific meta model. The specific safety and security requirements are already introduced in chapter 3 and apply to the following meta model development. Further functional and non-functional requirements, principles and conditions still need to be defined. They are based on the discussed properties of IoT systems and devices from subsection 2.2.1. Some aspects of these requirements are adapted from [Dig17] and [ISO16b].

Whether the requirements have been met and are sufficient is discussed in chapters 8 and 9.

Table 4.1 contains generic concepts the meta model must fulfill. When modeling an IoT system, its various networks should be considered as a whole, no matter how large and complex the system is. It should always be noted that IoT systems are rarely closed. The required management units must be mapped, in order to cover all the needs of various stakeholders. Different levels of abstraction are required, depending on which goals are targeted by the model.

Concept	Description
Context-awareness	Meta model must be able to accommodate IoT context
Modularity	Networks should be able to be viewed in modules
Scalability	The meta model must be suitable, no matter how large the IoT system is
Open architecture	Interfaces must be contained
Manageability	Management modules shall be included
Stakeholder	Different stakeholders must be able to use the meta model
Abstraction level	Different abstraction levels shall be expressible

Table 4.1.: Architectural requirements for IoT(-WD) meta model part 1

In addition, the meta model must be able to represent hardware and software elements so that all important and relevant parts of an IoT system can be included in further analyses. Due to their significance, sensors must be modeled separately. Wellbeing components must be modeled separately as well to capture their specific attributes and requirements [Dev12]. For example, it must be feasible to specify a UID and the manufacturer [WAF17]. IoT devices must be able to represent their described properties in the physical and digital world, and special IoT services must be included.

These requirements are listed in table 4.2.

Concept	Description
Sensors	Sensors must be depicted as separate physical entities
Identification	Devices must contain their unique identifier
Identity	Identity registration components shall be in IoT models
Wellbeing	Wellbeing devices must be depicted as separate physical entities
Virtual Twin	Every IoT device must have a digital equivalent
Autonomy	Autonomous components must be included
Shareability	Models need a possibility to depict Internet connection
Interoperability	Standard IoT communication methods must be modelable
Services	Services must be mapped to IoT activities

Table 4.2.: Architectural requirements for IoT(-WD) meta model part 2

4.2.2. Formalization of Meta Model

Before describing elements of the developed meta model in detail, a formalization, which is based on [KA09], is provided. The formalization covers meta model level and model level of the modeling approach. This distinction is crucial for future definitions of design patterns or anti-patterns.

The meta model (MM) and its element types (ET) are formalized as tuples:

$$\text{MM} = (\text{ET}, \text{RT}, \text{LT})$$

and

$$\text{ET} = (\text{CT}, \text{AT}, \text{R})$$

such that

CT is a set of *class types*

AT is a set of *attribute types*

$\text{LT} \subseteq \text{AT}$ is a set of *layer types*

$\text{RT} = \{\text{Association}, \text{Bi-directional}, \text{Aggregation}, \text{Composition}, \text{Generalization}\}$
is a set of *relation types*

$\text{R} \subseteq \text{ET} \times \text{RT} \times \text{ET}$ is a set of relations of *element types*, *relation types* and *element types*

In addition, there are two functions:

EL: $\text{ET} \rightarrow \text{LT}$ is a function that returns the *layer type* of an *element type*, such that $|\text{LT}|=1$

EA: $\text{ET} \rightarrow \text{AT}$ is a function that returns the *attribute types* of an *element type*

These functions are used for the assignment and return of the respective layer type and attribute types. Only one layer type can be returned, but multiple attribute types.

For an improved understanding, a minimal example of a formalization conform meta model is illustrated in figure 4.2 to show instances of the different types and tuples.

- $\text{CT} = \{\text{DigitalTwin}, \text{PhysicalEntity}, \text{Gateway}, \text{System}, \text{Cloud}, \text{LowPowerDevice}, \text{ConnectedDevice}, \text{Sensor}\}$
- $\text{AT} = \{\text{boolean}, \text{Enum:trustlevel}, \text{Enum:type}\}$
- $\text{LT} = \{\text{Application}, \text{Infrastructure}\}$

- $RT = \{\text{Composition, Generalization, Bi-directional}\}$
- An example of EL might be: $\text{PhysicalEntity} \rightarrow \text{Infrastructure}$
- An example of EA might be: $\text{PhysicalEntity} \rightarrow \text{trustLevel}$

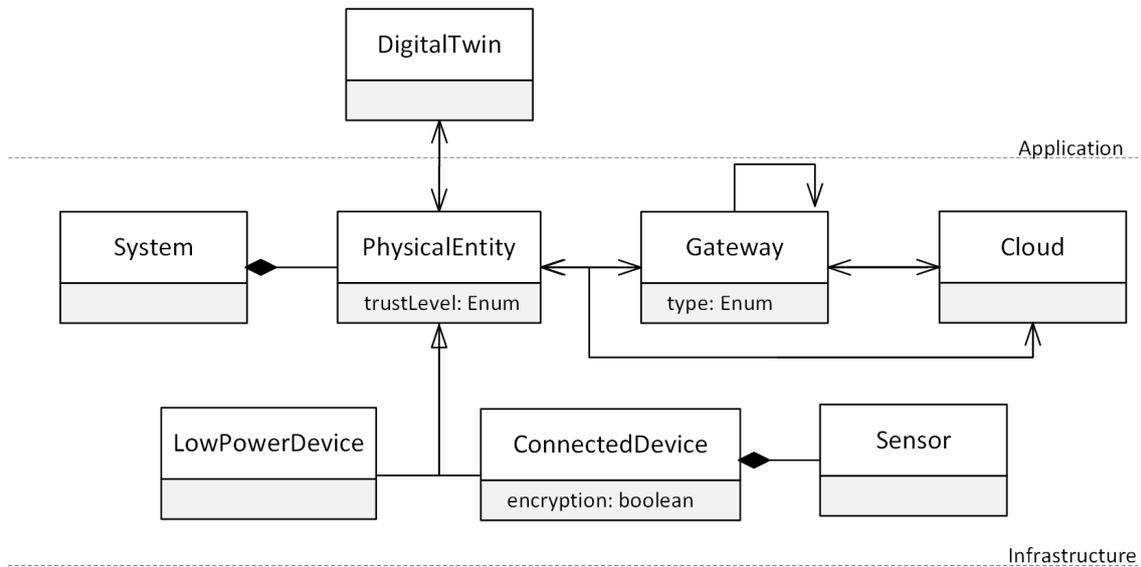


Figure 4.2.: Example of a formalization conform meta model (MM)

An IoT-WD model (WM) and elements (E) are described as tuples:

$$WM = (E, RT, L, SE, WE) \text{ being an instance of MM} \quad (4.1)$$

and

$$E = (C, A, R') \text{ being an instance of ET} \quad (4.2)$$

such that

C is a set of *classes* having a value, each being an instance of CT

A is a set of *attributes* having a value, each being an instance of AT

$L = \{L_0, \dots, L_n\}$ is a set of *layers*, each being an instance of LT and $L \subseteq A$

$R' \subseteq E \times RT \times E$ is a set of relations between *elements, relation types* and *elements*

$SE \subset E$ is a set of *smart elements*, each being an instance of ET and $|SE| \geq 0$

$WE \subset E$ is a set of *wellbeing elements*, each being an instance of ET and $|WE| \geq 0$

In addition, there are two functions:

$EL': E \rightarrow L$ is a function that returns the *layer* of an *element*

$EA': E \rightarrow A$ is a function that returns the *attribute* values of an *element*

Derived from the functions of MM, these functions are responsible for assigning and returning actual layer and attribute values of an element.

Constraints (C) are necessary to ensure compliance with essential conditions. An IoT-WD model is a valid instance of MM if the following criteria are fulfilled:

(C1): Every IoT-WD model has exactly one root element:

$\exists! re \in E. PE(re) = \emptyset$ where re is *root element* and PE a set of *predecessor elements* of re

(C2): Every element is assigned exact one layer:

$\forall e \in E \Rightarrow \exists! l \in LT \wedge EL'(e) = l$

Parallel to the formalization of the meta model, an example (figure 4.3) for an IoT-WD model is shown. It should be noted that the example is at model level and not at instance level.

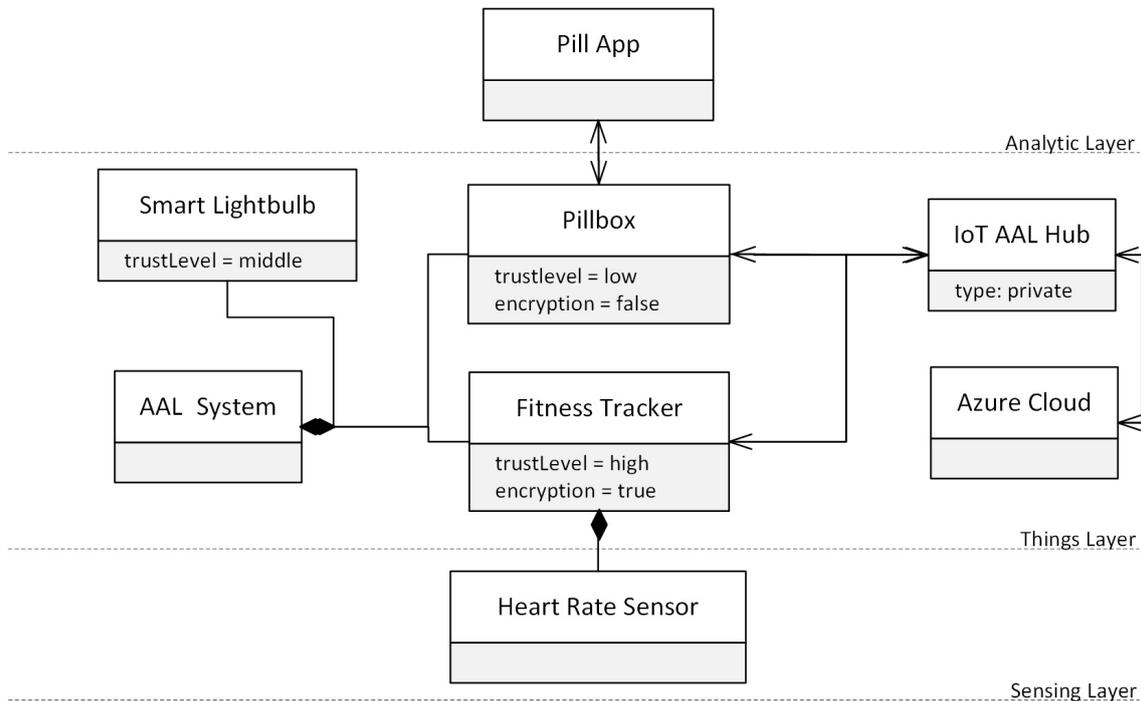


Figure 4.3.: Example of a formalization conform IoT-WD model (WM)

- $C = \{\text{Pill App, Pillbox, Fitness Tracker, Smart Lightbulb, AAL System, IoT AAL Hub, Azure Cloud, Heart Rate Sensor}\}$
- $A = \{\text{trustlevel:low/middle/high, encryption:false/true, type:private}\}$

- $L = \{\text{Analytic Layer, Things Layer, Sensing Layer}\}$
- An example of EL' might be: Pillbox \rightarrow Things Layer
- An example of EA' might be: Pillbox \rightarrow trustLevel=low

4.2.3. Specification of Meta Model

The provided meta model of this thesis is specified in the following section. An overview of the entire meta model and a complete list of enumerations can be found in appendix A.

For the description of specification of elements, attributes and connections, the meta model is divided into logical groups to simplify understanding. Some inheritances have been hidden in the meta model for clarity, although they are mentioned textually. Entries of specified Enums are to be understood exemplary, since they can be extended at any time.

The first section shows the starting point of the models and general attributes.

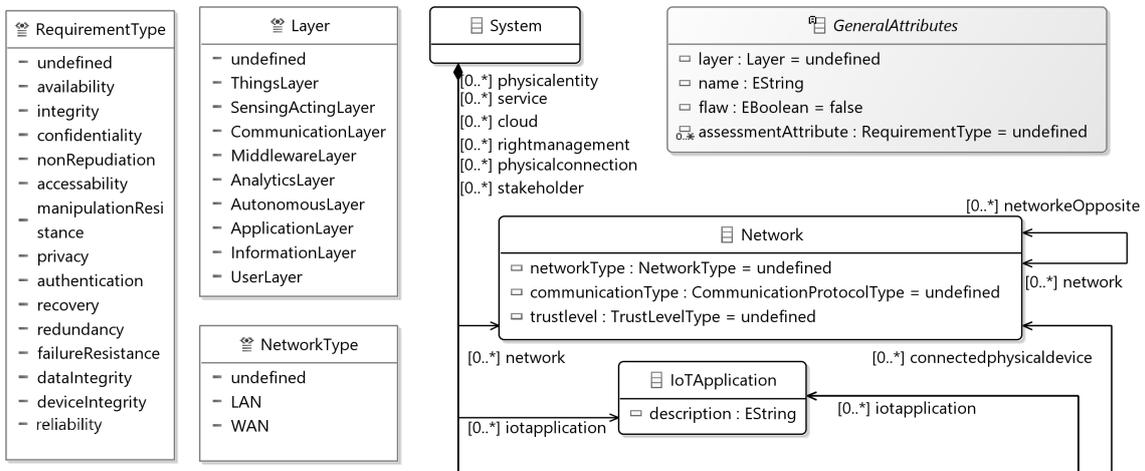


Figure 4.4.: Meta model part general elements

An IoT System consists of several Networks that have further specifications underneath. Some attributes must be inherited to all elements in the meta model, and are therefore, outsourced. Thus, presence of a flaw can be indicated by an attribute and requirements per element can be set for analyses in chapters 6 and 7. The attribute layer is used for mentioned layered protection. Thus, the system can contain everything from physical signatures to high level access controls for safety and security measures.

Element	Description
System	Root element and starting point for further components
GeneralAttributes	Contains attributes which are inherited to all classes including attributes for their layer and further analyses and flaw identification
RequirementType	Enum for further assessment of its element
Layer	Enum which contains all layers of layered architecture
Network	Enables multiple networks depicted in a system, including details of its trustworthiness, contained network and communication types
NetworkType	Enum for further network details
IoTApplication	Depicts end user applications and complex process applications

Table 4.3.: Meta model part general elements

Figure 4.5 and tables 4.4/4.5 show the services and connections required for typical processes in an IoT system.

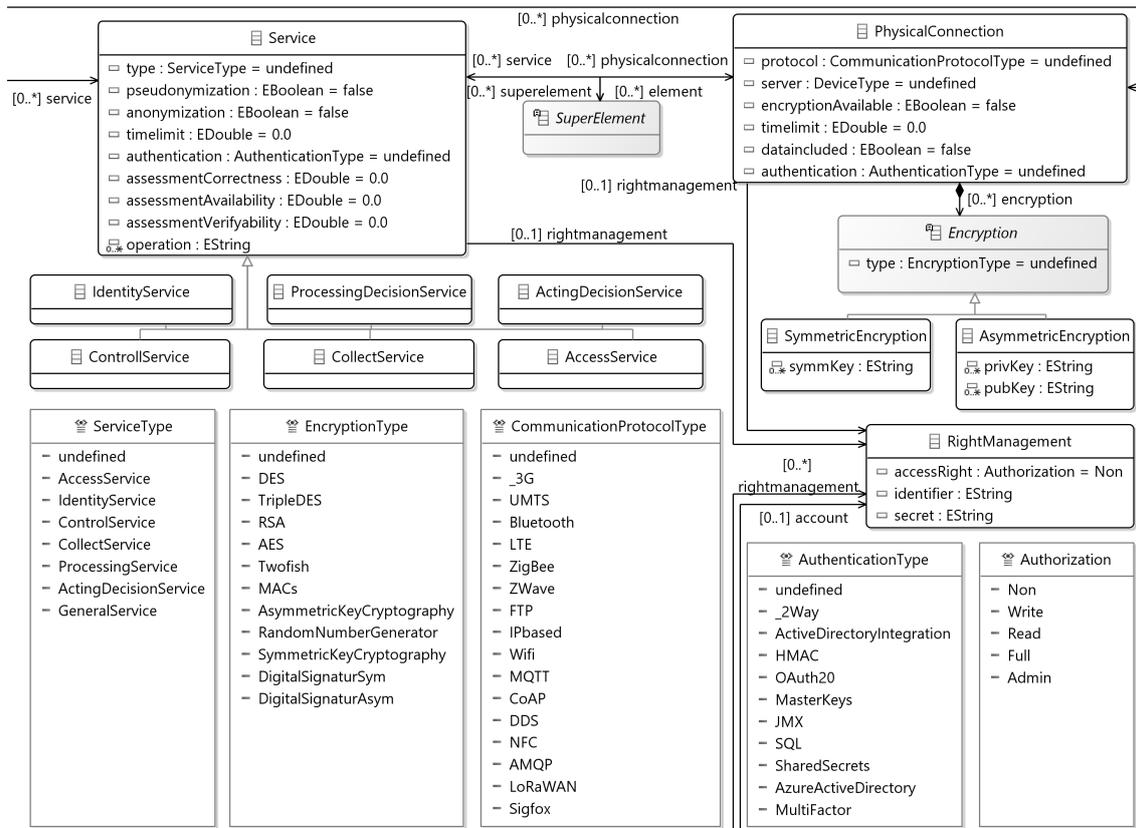


Figure 4.5.: Meta model part services and physical connections

All actions in IoT Systems take place via *Services*. These can run on one component or act between two or multiple components. For this purpose, it should always be possible to assign a *PhysicalConnection* to a service. Connections can be aligned with this by choosing a suitable protocol. In order to be able to check the quality of services for various properties, evaluation criteria can already be specified at this point. The attributes of services also support the layered protection measures.

Element	Description
<i>Service</i>	Distinguished between generic or typical IoT services with operation defined software tasks. Critical services require authentication and pseudonymization/anonymization. Further evaluation is enabled by multiple assessment attributes
<i>ServiceType</i>	Enum for service specification
<i>AuthenticationType</i>	Enum for defined authentication methods
<i>PhysicalConnection</i>	Component to depict connections between elements including optional and essential attributes. EncryptionAvailable and authentication are essential for critical connections
<i>Comm.ProtocolType</i>	Enum for possible protocols in IoT systems

Table 4.4.: Meta model part services and physical connections part 1

A *SuperElement* combines both concepts and depicts their linkage. Additional components can be added to the model to define further safety and security measures.

Element	Description
<i>Encryption</i>	Distinguished between symmetric and asymmetric encryption for <i>PhysicalConnections</i> including required keys
<i>EncryptionType</i>	Enum for encryption methods
<i>RightManagement</i>	Management component for right assignment and login checks
<i>Authorization</i>	Enum for authorization rights
<i>SuperElement</i>	Inheritance of relations to <i>Service</i> and <i>PhysicalConnection</i> to enable connectivity and operations of all elements

Table 4.5.: Meta model part services and physical connections part 2

Physical entities and related components are specified in figure 4.6 and tables 4.6 - 4.8.

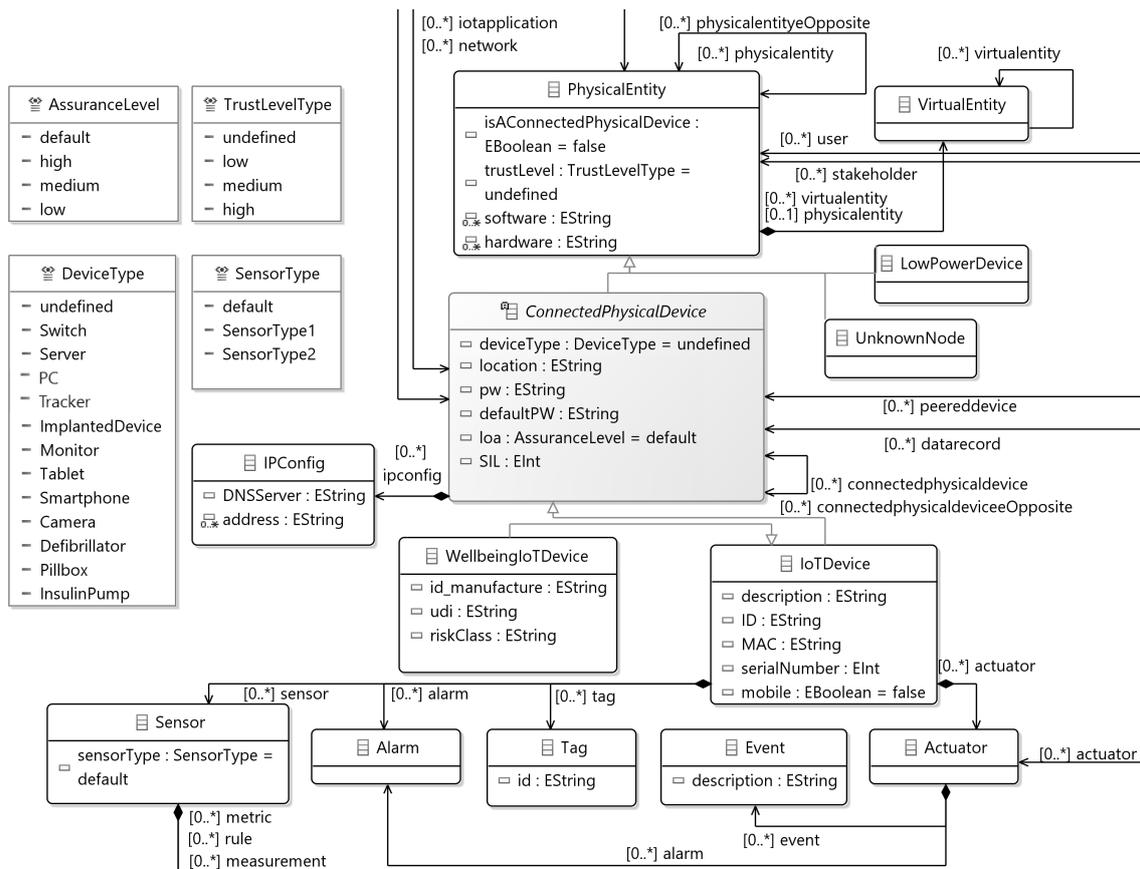


Figure 4.6.: Meta model part physical entities

Each system has **PhysicalEntities** in different variants that can interact with each other. These can be large, complex units, but also the smallest physical components can be mapped. In addition, stakeholders or users can be assigned to depict belongings and responsibilities. Further properties must be inherited if they have connectivity as a property.

Element	Description
PhysicalEntity	Superclass for all physical components despite <code>isAConnectedPhysicalDevice</code> is true or false. <code>TrustLevel</code> classifies into trustworthiness of entities
TrustLevelType	Enum for confidentiality
VirtualEntity	Digital Twin for every physical component
LowPowerDevice	Non-smart devices as defined
UnknownNode	Uncertainty component to depict new and unidentified entities with high level of risk

Table 4.6.: Meta model part physical entities part 1

Connectable PhysicalEntities are not necessarily IoTDevices. LowPowerDevices can also connect, even if they do not contain smart functions. Connections are possible to a Network, IoTApplication or as a peerdevice to Gateways. If Measurements are required, a connection to DataRecords is necessary. Accordingly, the specified level of assurance is also dependent [ISO]. IoT devices inherit the same attributes and connections and can extend them with their attributes and containments. This corresponds to the properties described. On the other hand, WellbeingIoTDevices must be defined as an extra class as they are subject to legal attributes, such as references to manufacturers [Leg17].

Element	Description
ConnectedPhysicalDevice	Superclass for all connected devices despite smart or not. Contains multiple safety and security attributes, like loa (level of assurance) or SIL (Safety Integrity Level), which are mandatory depending their location, deviceType, trustLevel and device services
DeviceType	Enum for device selection
AssuranceLevel	Enum for authentication criteria
IPConfig	Contains network relevant information for connections
IoTDevice	Component which defines as IoT device and requires mandatory information about its mobile state and identification
WellbeingDevice	Component which defines as wellbeing device and contains official required attributes for identification and riskClass classification

Table 4.7.: Meta model part physical entities part 2

In order for IoTDevices to fulfill their defined tasks, additional subcomponents are required. These are directly subordinated to a device.

Element	Description
Sensor	Main component of an IoT device to enable measurements
SensorType	Enum for assurance of diverse sensor selection
Alarm	Represents occurrence signal of a specific action or event
Tag	Component for identification of IoT devices
Actuator	Component for autonomous actions of an IoT device
Event	Result of an conducted action

Table 4.8.: Meta model part physical entities part 3

Measurements and storage elements are described in the following figure and table.

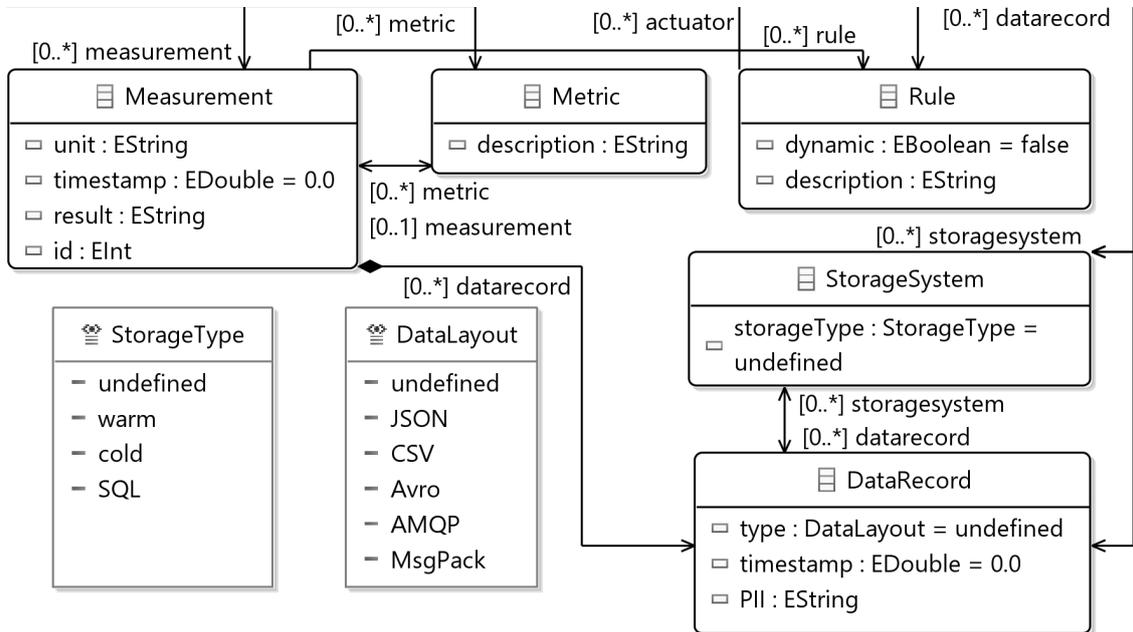


Figure 4.7.: Meta model part measurements and storages

IoTDevices provide the physical components for Measurements. However, these still have to be specified themselves, as they are subject to conditions. Not only Metrics and Rules are crucial, but also the right time and measurement unit. Thus, values can first be stored locally and then be processed in a Cloud. Therefore, a StorageSystem also needs a connection to the outside. Each DataRecord entry is connected to its ConnectedPhysicalEntity for later inference and decision making.

Element	Description
Measurement	Allocated component to sensors for measurement activities and their specification and results
Metric	Associated metric for a measurement
Rule	Measurement depending dynamic or non-dynamic rule for an <i>actuator</i>
DataRecord	Detailed result of measurements including its timestamp and possible critical information
DataLayout	Enum for data types
StorageSystem	Allocated storage for measurement persistence
StorageType	Enum for storage definition

Table 4.9.: Meta model part measurement and storage

Tables 4.10/4.11 and figure 4.8 present components and relations of system gateways and external cloud options.

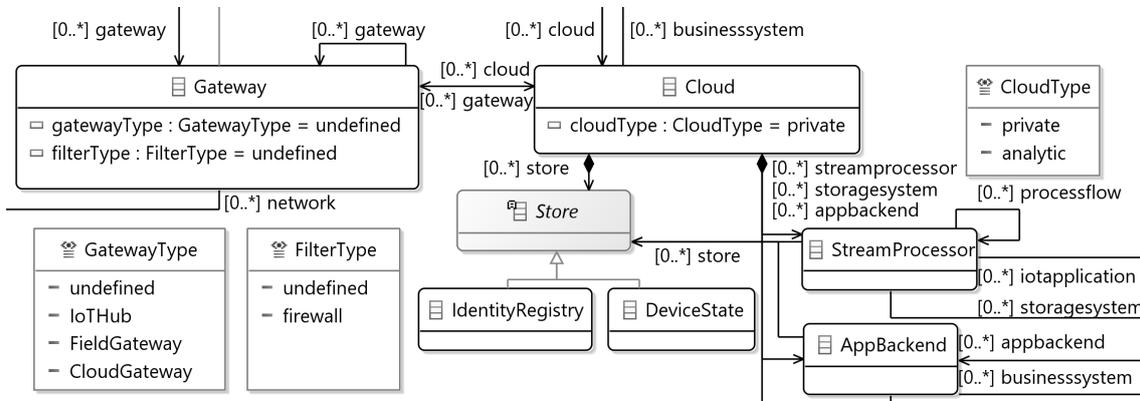


Figure 4.8.: Meta model part gateways and clouds

Gateways are not only connectors to Networks or Clouds, but can also be used as intermediaries requiring communication from several Gateways.

Element	Description
Gateway	Interface for ConnectedPhysicalDevices that want to connect to another network or components outside their local network
GatewayType	Enum for gateway selection
FilterType	Enum for assigned filter to its gateway

Table 4.10.: Meta model part gateways and clouds part 1

In IoT Systems, different types of Clouds can be found with different purposes. Either locally operated or external clouds for complex calculations for future autonomous decisions. The respective BusinessSystem is connected to this to adjust rules on stakeholder goals. The structure is oriented on Azure [Mic18a].

Element	Description
Cloud	Local or external processing and decision unit
CloudType	Enum for selection of publicity of used cloud
Store	Distinguished between identification and state depending cloud components
StreamProcessor	Further processing component of measured DataRecords from a StorageSystem
AppBackend	Equivalent to IoT applications which are performed on cloud side

Table 4.11.: Meta model part gateways and clouds part 2

4.2.4. Limitations and Constraints

When using the meta model for depicting an IoT-WD system, some restrictions and conditions occur, which are based, among other factors, on the constraints of the formalization in subsection 4.2.2:

- An IoT-WD system is only modeled if proper wellbeing devices are included. If these are only IoT devices, a pure IoT system is present.
- In critical systems wellbeing devices must have its specified attributes filled in.
- A newly modeled element should never have the value `flaw=true`.
- The layer and assessment attribute of `GenericAttributes` presented must be specified in the model in order to be able to use the approaches in the next chapters.
- Interoperability of services can be checked only if their assessment attributes are filled in.
- The abstraction level of the displayed model shall match the views and filters of the modeling editor.
- Depicted models have to be service-oriented as software can only be defined in the form of services.

4.2.5. Modeling Editor

To build a visual representation of a specific IoT-WD system, as formalized in subsection 4.2.2, a modeling editor is needed which is based on the provided meta model. A running example is introduced to present the modeling editor and its features. This example is also used for the further approach steps in the next chapters.

The Eclipse Modeling Framework (EMF) and Eclipse Sirius are used to develop the model editor. EMF describes a domain model that represents a problem area to be mapped in the form of a class diagram and can, thus define meta models in a so-called Ecore model [EMF]. For the graphical representation of the model, customized model viewpoints are defined with Sirius in a graphical modeling workbench [Sir]. The model editor is part of the developed ArchiAna tool.

In the graphical workbench representation of elements and their connections can be defined. For one of the viewpoints, the shapes of ArchiMate are used as a rough guide, since they have included IoT elements [Groa].

The color scheme of the elements is based on the IoT layered architecture as described above to highlight the affiliation of elements and their layers. Since services can be assigned to different layers, they can take on different colors. They have an oval shape as their only distinguishing feature. IoT devices or wellbeing IoT devices are specially highlighted with a 3D effect. Their subcomponents have the same symbol in their shape included. All other components are designed in a two-dimensional way. In the case of a bi-directional connection, this is combined into one connection in the representation. In addition, a palette of tools can be found in the modeling editor to apply filters, change viewpoints or create new elements.

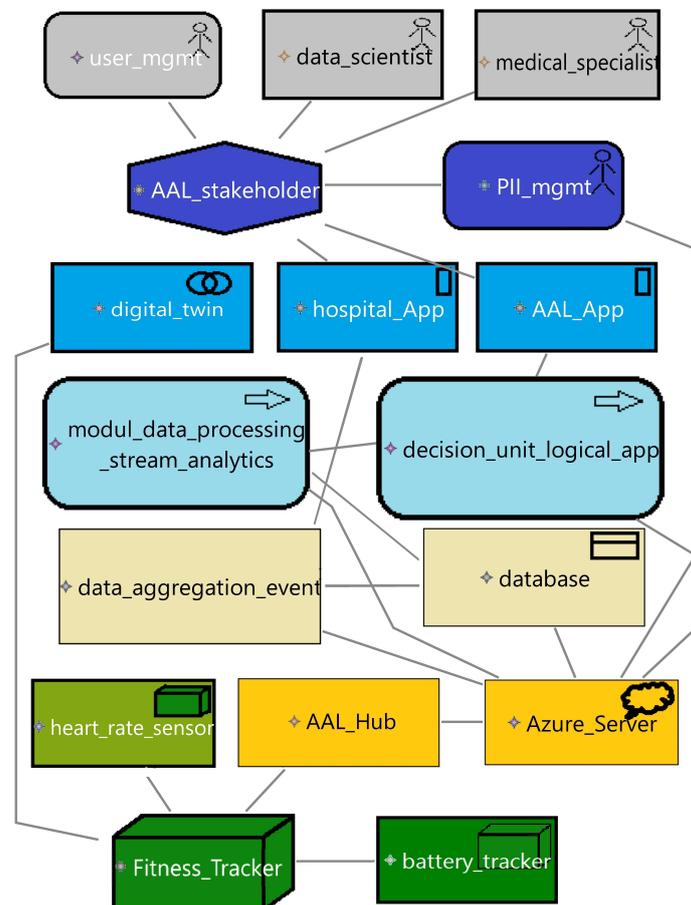


Figure 4.10.: Exemplary usage of the ArchiAna model editor to model an AAL example

A small AAL system is defined for a running example (figure 4.10). The corresponding legend can be found in figure A.4. A resident of the smart home has a battery-powered, smart Fitness_Tracker that collects information via a heart_rate_sensor that is used for a long-term examination of heart activities,

and thus, serves purely for prevention and wellbeing. It has a `digital_twin` for simulation purpose and is connected to an `AAL_Hub` of the smart home and sends data from there to an `Azure_Server` cloud. This can store and aggregate data and then process it in `modul_data_processing_stream_analytics`, while a `decision_unit_logical_app` cloud component plans logical future actions. This information can be accessed by various apps. Multiple users are involved to analyze data and incorporate medical knowledge. In addition, PII can be handled.

The running example is a representation of an IoT-WD model at a high abstraction level. If the modeling is more detailed and less abstract, the associated service and its physical connection can be modeled for each action. Figure 4.11 shows a short example of how the running example can be extended. Here, the `AAL_Hub` tries to connect to another device from the smart home. The hub requests via `Tracker_Gateway_Bluetooth_Connection` data of a `Fitness_Tracker` with an `Data_Access_Service` for synchronization.

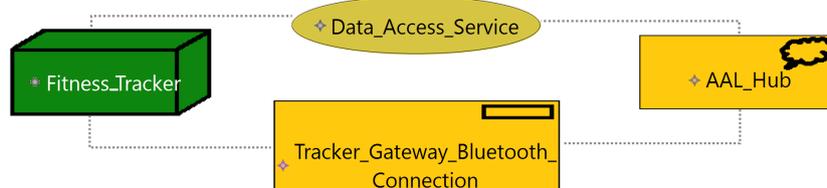


Figure 4.11.: Exemplary service-physical connection depiction with ArchiAna

Filters and Viewpoints

Filters are available to limit the display of included elements. This can be necessary or helpful if only individual aspects are to be considered or large complex architectures are to be examined. Figure 4.12 shows a selection of filters, such as `FilterNoServicesAndPhConnections`, which takes the system to a higher level of abstraction and hides `Services` and their `PhysicalConnections`. For the same reason there are layer specific filters. Filters only cause hiding and do not permanently remove elements.

```

FilterOnlyServiceAndPhConnectionEdges
FilterOnlyServices
FilterOnlyPhysicalConnections
FilterOnlySensors
FilterNoServicesAndPhConnections
FilterThingsLayer
FilterSensingActingLayer
FilterCommunicationLayer
  
```

Figure 4.12.: Extract of list of available filter in IoT-WD systems

In addition to filters, different viewpoints can be displayed graphically. Figure 4.13 shows the same running example, but from a different viewpoint. Here, the focus is on the assignment of elements to their corresponding areas or components, represented in containers. Depending on the goal of the modeling, one or the other viewpoint may be more meaningful and provide easier insight into the aspect to be investigated.

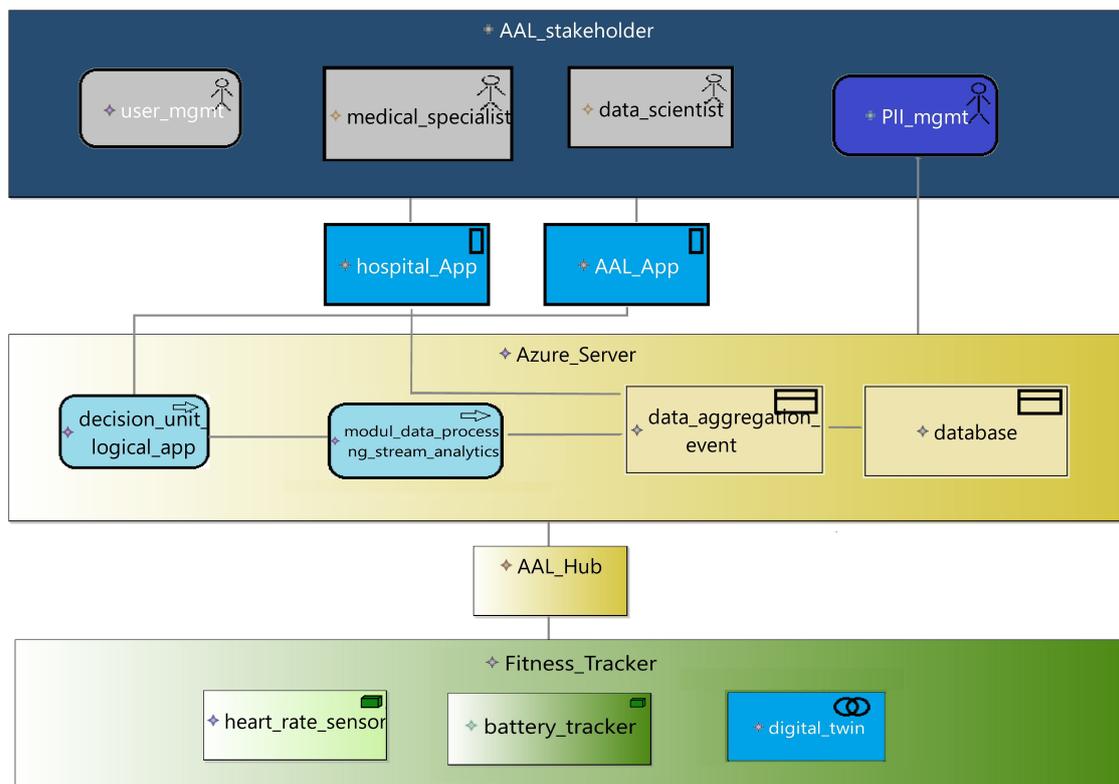


Figure 4.13.: Exemplary usage of the ArchiAna model editor to model an AAL example with container view

4.3. Related Work

This work is distinguished from related approaches as follows:

IoT Modeling Concepts

There are already some approaches that try to consider IoT at the model level. Subsequent projects have developed the most promising reference architectures and based on them further modeling concepts.

One of these is Microsoft's Azure. They offer a reference architecture for IoT structures that is divided into three parts. A classic IoT system process is divided into things, insights and actions. This should make it easier to set up an IoT system that can work with Azure servers. In addition, the structure of physical devices, services and intelligent edge devices is prepared in nodes and edges. With the help of Microsoft's threat modeling, the IoT model can be checked for dangers [Mic18b]. Even if their reference architecture already contains classes and edges, they do not provide a complete meta model for system modeling. In addition, they do not foresee the inclusion of other stakeholders. Furthermore, their modeling proposal is not suitable for wellbeing claims, since all devices are treated equally. They do offer a retrospective analysis of their models, but do not offer the option to model safety and security aspects immediately. [Mic16]

The EU project IoT-A ([IoT13]) is focused on the provision of an IoT architecture reference model. This consists of several models such as a domain model, information model and a communication model. The models differ greatly from one another as some are designed as simple reference layered architectures and others as meta models. There are two meta models for the domain model and information model. The domain model is divided into users, augmented entities, services, resources and devices. Hence, the meta model is more abstract and covers fewer areas, since these have been outsourced to other models for which there are no meta models. Accordingly, not all aspects can be modeled in one model as in the approach of this dissertation. Like Azure, the use of wellbeing devices has not been considered separately. However, they have covered security aspects directly with their security model. Yet this does not allow a direct link to the domain model, and thus, no direct modeling of safety or security aspects of individual components. [ISO16b] proposes a similar reference architecture. However, this architecture has been extended with various architecture views and a concept model. The architecture views, such as communication view or information view, are described by guidelines. However, no possibility is offered that a model can switch flexibly between views. The concept model provides specific and detailed meta model excerpts. However, again safety and security are only provided as best practice and guidelines. This prevents further use for model analyses of vulnerabilities, since not all defined measures can be modeled.

One promising approach comes from [Con+16]. However, it is aimed at industrial IoT, and is therefore, only related to the approach of this dissertation to a limited extent. They are also working on a reference architecture and want to enable systems to be created uniformly with their architectural template and methodology. They use the architectural description of [ISO11b], which is very high level and abstract. The promised architectural template is designed accordingly. Components and their presentation are only described by guidelines and recommendations. However, no meta model is offered as a template and direct modeling. It would be possible to adapt the project to the field of wellbeing, as this project, in comparison to other approaches, also explicitly addresses safety aspects. However, the direct link to safety and security aspects as in the other approaches is also missing, and therefore, makes the following analysis approaches in the next chapters not possible.

No projects could be found that cover all areas of an IoT system in one model in the design phase or that also model safety and security information. The connection of wellbeing devices and their special treatment during modeling could also not be identified. Thus, the existing approaches would not be suitable for the further course of this dissertation.

Layered Approaches

The layered structure of IoT systems has already been mentioned in the basics. Mostly, approaches such as [Kha12] and [SS17] are used for this in the IoT area. These are two classic layer models which are either three or five layers, as described in subsection 2.2.1.1. However, these layers or subdivisions are often not sufficient when a detailed review of the system is required. It is also not possible to assign specific measures if the layers are chosen too broadly.

Subsequent approaches provide a more use case specific or fine granulated layered selection.

As mentioned above, the IoT-A project is divided into several models. For the functional model, a layered architecture is presented. It is based on a device layer and ends with applications. In between are horizontal and vertical layers for management, services and their orchestration, as well as for virtual entities and communication components. A security layer can also be included, as it has a connection to the IoT-A security model. This is the biggest restriction. Since not all IoT-A models are interconnected, a matching of layers and domain model is not possible, and thus, no complete meta model can be derived from the layered architecture. [Pro15]

Even though it is designed for industrial IoT, RAMI 4.0 ([Pla16]) can be considered as a related concept. This project provides a 3D model for all elements and components, including security aspects, so that layers can be combined with a lifecycle value stream and hierarchy levels. The layered approach starts with physical assets and their integration and builds upwards to business via communication, information and functional components. The other two dimensions relate to the product development phase and the deployment of devices. As designed for IoT, the important integration of IoT services and sensing or acting layers are missing. In addition, the 3D architecture model is focused on the product lifecycle and is not generic enough for an independent layered architecture.

Likewise, a 3D layered model for industrial IoT systems is presented by [Con+16]. In addition to the classic layers, this application includes important sensing, acting and control aspects. Furthermore, an extended align-enable principle offers the possibility that horizontal effects can also take place in IoT systems. One dimension is completely designed for system characteristics such as reliability or scalability, while the third dimension includes crosscutting functions such as distributed data management. It is a very promising approach, but not suitable for this dissertation, as no layer for stakeholders or users is included. These may be negligible in the industrial domain, but not in the wellbeing area.

A very detailed layered approach from [Dig17] describes the use of a dedicated IoT layer in a connected enterprise. The focus is on an enterprise architecture with eight layers including enterprise system and integration components, as well as business and information services, but also including various user and consumer component layers. Other management and security layers run in parallel. Next to this architecture is the IoT layer, which is divided into a classic IoT layered architecture that also includes IoT operations. However, it is not clear how the IoT layer integrates with the other layers and can interact with them. Besides, the IoT layer is not the center of attention. Additionally, the IoT layer does not contain a layer for processing operations or autonomous actions.

All identified layered approaches in the IoT area have either used the classic three to five layer models, which are not sufficient for the approach of this dissertation, or are designed for specific areas, and thus, partly do not include all necessary layers.

Part III.

ARCHITECTURE
OPTIMIZATION

5

Safety and Security Optimization Process

This chapter introduces the second part of the main part of this thesis. After defining in the previous chapters how IoT systems can be modeled in a uniform way and how their safety and security aspects can be included, the concurrent optimization process of these systems is now introduced. It should be mentioned that the optimization focuses on safety and security aspects. Other optimization goals are possible, but are out of scope of this dissertation.

Since this chapter is the introduction of the optimization process, the following sections should be read before chapters 6 and 7. The optimization targets to identify problematic design decisions and consider them in retrospect, suggesting another possible architectural design if necessary. In order to define certain acceptance criteria for the achievement of the target, this chapter first specifies improvement requirements. This is intended to define points or properties that must be considered during the optimization. Then, the process steps are described in detail. Finally, integration factors of the process are named in order to make the process open and to enable extensions.

5.1. Improvement Process Requirements

In order to design a system or model optimally, it should be determined in advance which conditions are required for it to be an optimal model. However, no matter how carefully a system is planned and modeled, poor or critical design decisions can occur. Therefore, a process is needed to improve created models. Attention has to be paid to the fact that these conditions are generic and fit the use case or the underlying meta model. The following conditions are accordingly aligned with the meta model from chapter 4. If this is adapted, the improvement requirements should be reconsidered. The criteria include not only design decisions, but also general non-functional ones to determine which aspects are needed to be checked on the one hand and which steps the process must contain on the other hand. For this purpose, the following conditions should be used to create a process which checks whether a system is optimally designed according to predefined aspects.

Requirements for IoT models can be considered by multiple views. On the one hand, a distinction can be made between functional and non-functional. On the other hand, more detailed differentiations can be made, such as criteria for safety and security, but also for reliability, availability or interoperability. Furthermore, conditions in different domain areas can be considered too, such as user interactions, virtual entities or devices. In this work, improvement requirements focus on safety and security in the modeling domain for IoT systems, whereby some criteria overlap. The already mentioned reference architecture for IoT systems of IoT-A [Bau+13] provides an overview of various criteria. The following requirements are based on the results of this project [Pro15; Bau+13]:

- Virtual elements including digital twins must be checked and optimized since they can be tampered and endanger integrity.
- Especially in safety-critical areas availability of devices is essential and must be optimized.
- Event-based elements, rule-based functions and actuators must be free of design flaws as they guarantee event-based and autonomous communication which is essential for IoT systems.
- The system shall provide quality of service including reliability.
- Interfaces must be checked to guarantee that systems run in operable manners to enable flawless vital functionalities.
- If external information sources are part of the system, they must be considered as possible threat or hazard.

- Data must remain encrypted and anonymous respectively data spreading is not to be extended by error to prevent data exposure or manipulation.
- Business scenario aspects shall be taken into account through optimizing systems as they are an essential part of design decision processes.
- Resolution functionality of systems must be checked to guarantee correctness.

As IoT-A is not specialized for safety and security aspects in wellbeing areas and their associated analyses during design phase, more requirements must be added:

- After the optimization process has been completed, there shall be less vulnerabilities that could lead to an accident or attack.
- Risk management shall be included, accordingly vulnerabilities must not only be identified.
- If the use case is in the wellbeing area, the optimization must take place before the implementation.
- Every element and every architecture layer must be considered in improvement steps.
- Optimization must be designed generically, as each element is different and constantly evolving.
- Wellbeing elements must comply with their prescribed guidelines and data protection before and after the improvement proposals.
- An optimized model must include elements and stakeholders regardless of whether they are accessible on-site or remotely.
- An alternative design proposal must not bring any new flaws into the architecture.
- Fault elimination must be in proportion to time, danger, cost and probability.

These defined improvement requirements are used to validate the approach of chapter 6 and 7 to evaluate (chapter 8) whether the optimization process is suitable to improve IoT models and help to create optimal models.

5.2. Process Steps Outline

The optimization process is part of the holistic IoT safety and security management approach of this thesis which is divided into steps contained in a sequential workflow. This workflow is based on two related approaches: [IB06] and [NIS12]. NIST SP800-30 [NIS12] defines a risk assessment process which is used as a rough template for risk handling in plenty approaches, e.g. [US 07b] for risk analyses in medical applications. Following steps are part of the approach:

- Prepare for assessment
 - Identify purpose, scope, assumptions and constrains of the assessment.
 - Identify sources of information to be used as inputs, the risk model and analytic approaches which are be used during assessment.
- Conduct assessment
 - Identify threat and vulnerability sources.
 - Determine likelihood of occurrence, scale of impacts and risk.
- Maintain and communicate results

These three main steps are highly generic and can be adopted in almost every use case for risk assessment. Only the individual sub-steps need to be adapted, as each use case requires different information as input and wants to query different results with the assessment. The manner in which the results are communicated and maintained depends on whether the assessment is the final step or whether defined follow-up steps are planned. The second approach ([IB06]) was developed for a security management process in EAM and is illustrated in figure 5.1. This approach is designed more like a workflow and is specifically defined for security in EAM. However, the individual sub-steps are congruent with [NIS12]. The first 4 steps can be matched with *Prepare for assessment*, whereas the fifth and last steps are equivalent to *Conduct assessment* respectively *Maintain and communicate results*.

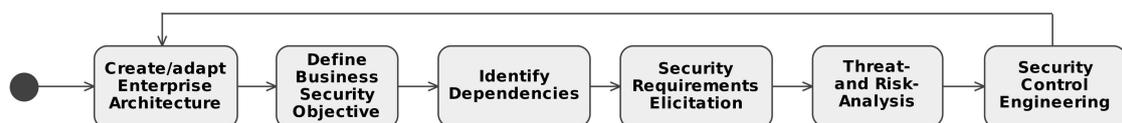


Figure 5.1.: Security management process of [IB06]

Since both approaches are very similar and work in a generic as well as in a specific way, these steps are used to create a workflow for the optimization process of this work. The steps presented here are combined and adjusted to make a workflow suitable for safety and security approaches and to meet the IoT challenges presented.

The workflow is divided into two phases: *System Architecture* and *Safety and Security Management*. The workflow begins in the first phase, and thus, contains the first preliminary steps for optimization. First, the scope of the respective use case and the underlying IoT system is identified and data are gathered. Since this includes all subsystems, components and connections, an architecture mining approach has to be carried out. This can be done manually or automatically. Particularly in the case of large and complex systems, an automated identification of the components may be necessary in order to subsequently enable a complete identification of weaknesses and their elimination. In addition, safety and security (S^2) requirements are identified to cover constraints of the assessment. After all information are collected, the modeling of the IoT-WD architecture can be performed. In order to cover the S^2 aspects, the meta model and the layered architecture from chapter 4 are used.

As proposed, the second phase of the workflow includes first the identification of threat and hazard sources and afterwards the implementation of different analytical approaches based on these sources. *Flaw Identification* is based on a framework (PRF) that is used to define architecture patterns and anti-patterns which represent harmful design decisions that are not allowed to be in IoT architectures. Templates are provided for safety hazards and security threats. These templates are based on a S^2 categorization that is used to define the respective objectives. The actual definition is carried out with the help of a developed DSL. Subsequently, flaws can be identified by running a pattern recognition method. This uses pattern services that are stored in a database. Afterwards further evaluation of identified flaws is possible through *Flaw Assessment*. These steps are based on a performed analyses categorization with different filters and analysis types. The analyses included built a closed cycle: Failure Impact Analysis (FIA), Quantitative Impact Analysis (QIA), Countermeasure Decision Support Analysis (CDSA) and Service Interoperability Analysis (SIA). The cycle initiates with FIA to estimate the effects of the identified flaws in the model, based on probabilities. These results are used in QIA to calculate potential costs in the event of an attack or accident in order to estimate how expensive countermeasures can be on a financial and business level. Subsequently, CDSA and SIA provide a small internal loop to plan countermeasures and their services to prevent or mitigate the identified flaws, and to assess their impact on the As-Is system. After concluding *Flaw Assessment* and communicating results, the optimization process can be closed. Though, to conduct control engineering the process offers an iterative methodology, and can thus, go back to modeling and retest the new solutions.

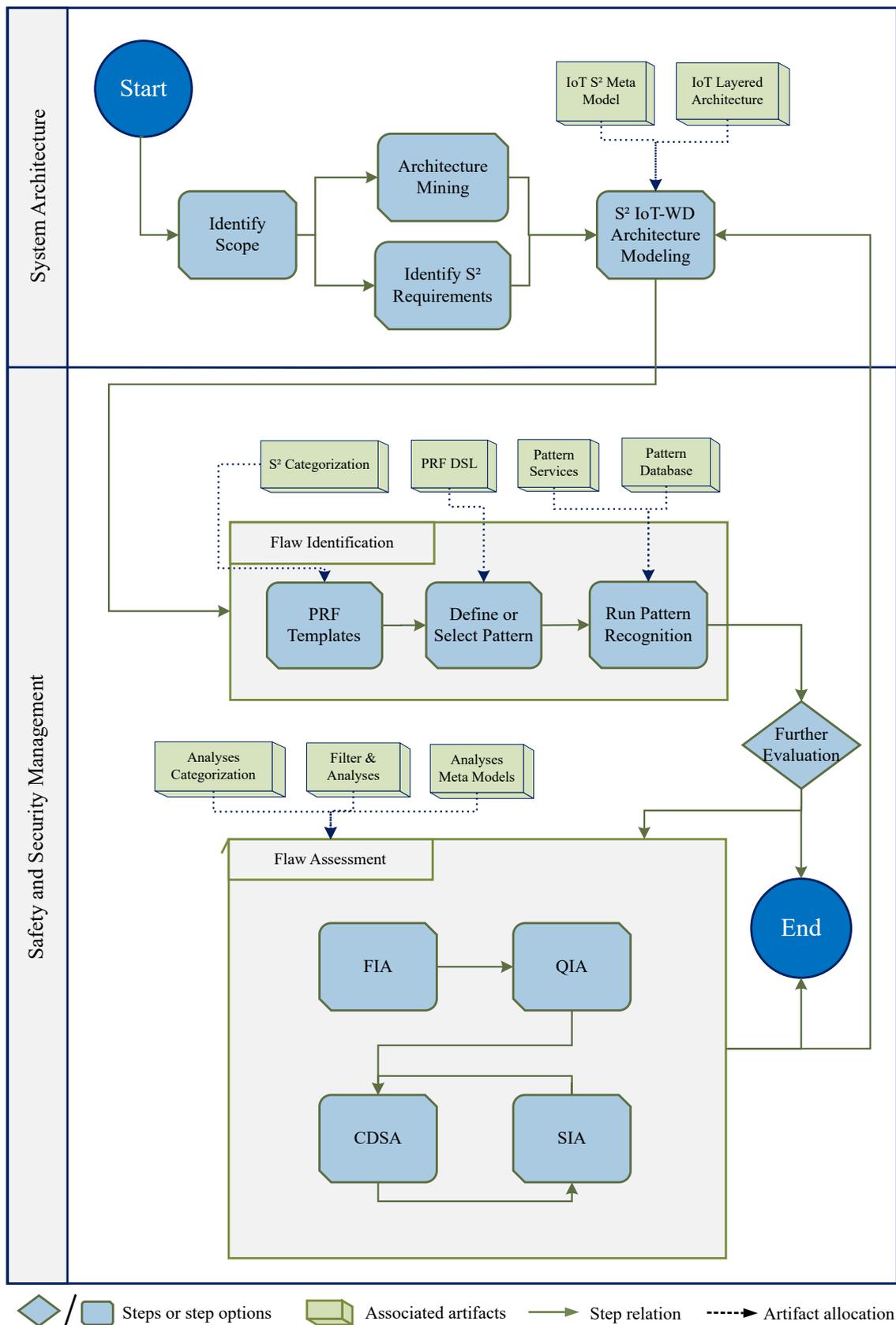


Figure 5.2.: Optimization process workflow

5.3. Integration Factors and Requirements for Related Approaches and Analyses

The optimization process presented is a holistic self-contained cycle. However, as noted in the IoT-A project, it is important that IoT systems provide ways to secure corporate security guidelines and regulations or recommended standards. [Bau+13] Therefore, the optimization approach of this thesis is designed in a manner that allows extensions and openings to enable connections to related analyses or other external information by including integration factors.

The first possible extension, anchored in flaw identification, is connecting the pattern database to other databases, e.g. [NVD], that have already noted safety or security vulnerabilities. Due to the fact that the PRF database is based on textually defined patterns, patterns defined in a different format can be exported and transferred and stored by a connector. By using the DSL and its connected code generation, an automatic query of external patterns and anti-patterns in the model is possible. However, the externally defined vulnerabilities must be at architectural level to be transferable and be able to provide the information which are required for the PRF (chapter 6).

Another option is to combine flaw assessment with related analyses. This may be necessary if additional points in risk management are required, such as the use of FMEA ([Sch96]) or fault trees ([FFJ09a; När+11a]) (compare related work section of chapter 7). In addition, further architectural analyses may arise in the future which are then to be subsequently added to the circuit. The connection to new or external analyses is made possible by the design of the assessment cycle. Each analysis is subject to its own meta model which contains connection components to the other analyses' meta models. This ensures that analyses can pass on their results and continue to be used in the subsequent analysis. If new analyses are added, these meta models can be extended by further connection components. Even if analyses build on each other, the cycle does not necessarily have to be run through completely. A partial assessment is also possible. Accordingly, analyses can be integrated that do not require any pre-steps or only match one of the developed analyses. For example, a SIA does not have to be the last step in the cycle as it can be finished with another analysis. For external analyses to be connected, however, they must comply with the structure and functions of categorized analyses of chapter 7.

A detailed overview of these options is provided in chapter 10.

6

Flaw Identification

After modeling IoT systems, the second part of the presented optimization process workflow can be started. It begins with flaw identification in order to check the modeled systems, uncover weaknesses that could develop into vulnerabilities or be starting points for accidents or attacks. For this purpose, the developed PRF is presented to define design patterns and anti-patterns and to use them for flaw identification at an early stage. The focus is on vulnerabilities based on design decisions and their subsequent impact on services, connections and processes. In the following chapter and in the further course of this work, anti-patterns are always included when referring to patterns.

For the presentation of the PRF, the definition process, components and functionalities are examined in more detail. A formalization is also carried out and the individual components are explained using a running example. Finally, pre-defined patterns and the corresponding database are presented. The chapter ends with a related work section. Thereby, this chapter is based on the already described basics of patterns from chapter 2.

6.1. Pattern Recognition Framework

To detect flaws with the help of patterns in the design phase, a simple definition of these patterns is not sufficient. In addition, grouping, mapping and final execution must be ensured. These steps are made possible by the PRF. Even though the main goal is to define and query patterns in a structured way independently of use case and vulnerability areas, the PRF focuses on several areas. The main goal is on the architecture level, and therefore, focuses on design patterns and not code patterns to detect code smells. However, design patterns may contain clues to later code problems. In addition, the framework focuses not only on flaw identification, but also on subsequent further assessment and mitigation of these. For this purpose, categories are built in that allow this assessment, but also a categorization for the pattern selection. This involves the integration of IoT and IoT-WD specific respectively safety and security specific categories and elements. On the one hand, this allows safety and security problems to be targeted, and on the other hand, identification can be seen as a pre-step before the flaw assessment (chapter 7). This can be used, for example, to quickly decide which design flaws need to be eliminated and at what priority. By focusing on IoT-WD models, specific aspects of the requirements of these special models can be defined and checked.

Goals of PRF

When planning the design or considering design changes, two main issues occur which are described in chapter 1. First, designing details highly depend on expert experience. However, experts are not always available during design phase or afterwards if changes are required in the model. Therefore, a possibility to preserve the knowledge is highly significant. The second issue addresses the complex dependencies, interoperability and requirements of the IoT system components. Hence, a manually verification of all included elements, connections and features is not possible in a reasonable time. The PRF approach covers both challenges, leading to multiple goals. To address the first challenge, a selection of required information about flaws, risks, avoidable design decisions and possible impacts is offered to create a structured chance to enable knowledge conservation. The experience of the experts can be saved for later observations. Therefore, misplanned design mistakes, which already happened before, can be prevented in the design phase during the design check. As the knowledge is conserved in the first PRF step in textual readable form, every team member is able to follow it independent of programming language knowledge. The second goal captures the second challenge by executing the automated flaw identification based on the defined patterns or anti-patterns. The automation is ensured by the development of a tool that can take as input the depicted models from the model editor of the ArchiAna tool. Entire models or sub-ranges, selected by views and filters, can be analyzed.

Patterns and Anti-Patterns

As described the knowledge conservation and flaw identification respectively recognition is conducted through design patterns. They are based on the characteristics described in the basics. They are divided into positive patterns and negative anti-patterns. Patterns describe desirable design decisions that promise abstinence of vulnerabilities. To recognize possible flawed modeling decisions, model components that do not match these patterns are searched for. During this search, the model components are examined for conditions of the unambiguous pattern definitions. Only if all pattern conditions are violated a match is displayed. With anti-patterns, dangerous design decisions are defined that must not exist or have led to problems in the past. At anti-patterns the automated flaw identification looks for model components that match them.

The framework enables to define generally applicable patterns which are suitable to all IoT models. Though specifically designed patterns for individual IoT models are possible and required. To cover different abstraction levels, patterns can be defined on meta model or on model level. Depending on whether generic patterns or instance specific patterns are to be defined.

Workflow

The links of concept parts within the PRF workflow can be seen in figure 6.1.

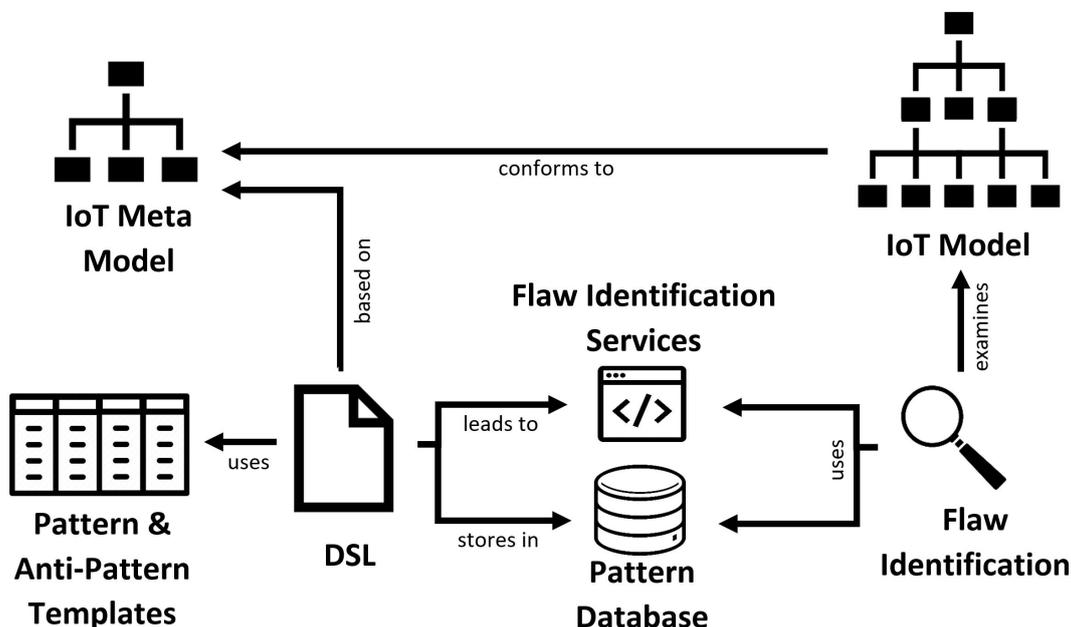


Figure 6.1.: Concept and workflow of flaw identification with PRF

The four concept parts are pattern and anti-pattern templates, DSL, flaw identification services including the pattern database and the flaw identification. As described the creation of a model requires a meta model independently of the domain. Therefore, PRF uses the presented IoT(-WD) meta model which can be used to depict IoT networks. This meta model is in turn the basis for the PRF, and

thus, enables flaw examination in IoT models. The first step of PRF are the templates to define patterns and anti-patterns for each safety or security flaw. Details of this are described in section 6.1.2. The DSL, which is based on the meta model, uses the fulfilled templates including their categories and values to configure the pattern specification language (section 6.1.3). This language helps to translate the textually defined patterns and anti-patterns into computer readable services in the next step to enable an automated analysis of the flaws. With the help of Xtend, flaw identification service generation is, thus, performed in section 6.1.4. The translated services respectively patterns are stored in a pattern database. In the last step (section 6.1.5), the flaw or pattern identification is performed to examine the IoT model. The database is accessed for this purpose. To find suitable patterns in the database that can be queried, the defined values of the templates from step one can be used.

6.1.1. Formalization of PRF and Patterns

Before describing the individual parts of the developed PRF in detail, a formalization, which is based on the formalization of the IoT(-WD) meta model and IoT-WD model in section 4.2.2, is presented. This section covers a generic formalization of PRF, its patterns respectively anti-patterns and the description of their generic or instance specific structure.

A generic pattern recognition framework (PRF), element types (ET) and part types (PT) are formalized as tuples:

$$\text{PRF} = (\text{MM}, \text{ET}, \text{RT}, \text{PFT})$$

$$\text{ET} = (\text{CT}, \text{AT})$$

$$\text{PT} = (\text{CGT}, \text{VT})$$

where

MM as defined in chapter 4 to enable usage of its ETs

$\text{RT} = \{\textit{Association}, \textit{Bi-directional}, \textit{Aggregation}, \textit{Composition}, \textit{Generalization}\}$
is a set of *relation types*

$\text{PFT} = \{\textit{Pattern}, \textit{Anti-Pattern}\}$ is a set of *pattern function types*

CT is a set of *class types*

AT is a set of *attribute types*

CGT is a set of *category types*

VT is a set of *value types*

$\text{R} \subseteq \text{ET} \times \text{ET}$ is a set of relations of *element types* with reltype: $\text{R} \rightarrow \text{RT}$,

i.e. reltype $(et_1, et_2) \in \text{RT}$ with $et_1, et_2 \in \text{ET}$

Patterns (P), elements (E) and parts (PT') are described as tuples:

$P = (MM, E, RT, PF)$ being an instance of PRF

$E = (C \cup I, A, R')$

$PT' = (CG, V)$

where

MM as defined in chapter 4

$PF = \{SecurityPattern, SafetyPattern, SecurityAntiPattern, SafetyAntiPattern\}$ is a *pattern function*, being an instance of PFT

C is a set of *classes* having a value, being an instance of CT

I is a set of instances of C

A is a set of *attributes* having a value, being an instance of AT

$R' \subseteq E \times E$ is a set of relations between *elements*

CG is a set of *categories* being an instance of CGT

V is a set of *values* being an instance of VT

Anti-patterns (AP) are formalized and defined analog to patterns.

For a pattern respectively an anti-pattern to be a correct instance of PRF to identify flaws the following has to be fulfilled:

$\forall (e_1, rt, e_2) \in R_P \Rightarrow \exists (e'_1, rt, e'_2) \in R_{WM}$

e_1 compatible e'_1

e_2 compatible e'_2

WM as defined in chapter 4

with compatible defined as (c_1, A_1, R_1') compatible (c_2, A_2, R_2')

iff

- $c_1 \equiv c_2$, if $c_1, c_2 \in I$

OR c_1 instance-of c_2 in terms of object-oriented implementation

- $A_1 \subseteq A_2$ with $\forall a \in A_1. \sigma_1(a) = \sigma_2(a)$

with $\sigma_1: A_1 \rightarrow VT$ and $\sigma_2: A_2 \rightarrow VT$

- $R_1 \subseteq R_2$

Two examples of formalization conform patterns are shown in figure 6.2. The pattern on the left is a generic pattern, conform to the IoT(-WD) meta model. Therefore, only meta model elements without specific characteristics are set. It is based on the AAL running example (further specified in the next section) and on the meta model example of MM formalization. The pattern queries whether ConnectedDevices in the model are connected to a Cloud through a Gateway. The

pattern on the right is an instance specific pattern and shows the same example, but on model level, conform to the defined exemplary IoT-WD model. Thus, specific elements with defined attributes are set. It is checked by the query whether a specific ConnectedDevice type Fitness Tracker, reaches Azure clouds via a Field gateway. If not fulfilled, a flaw is reported.

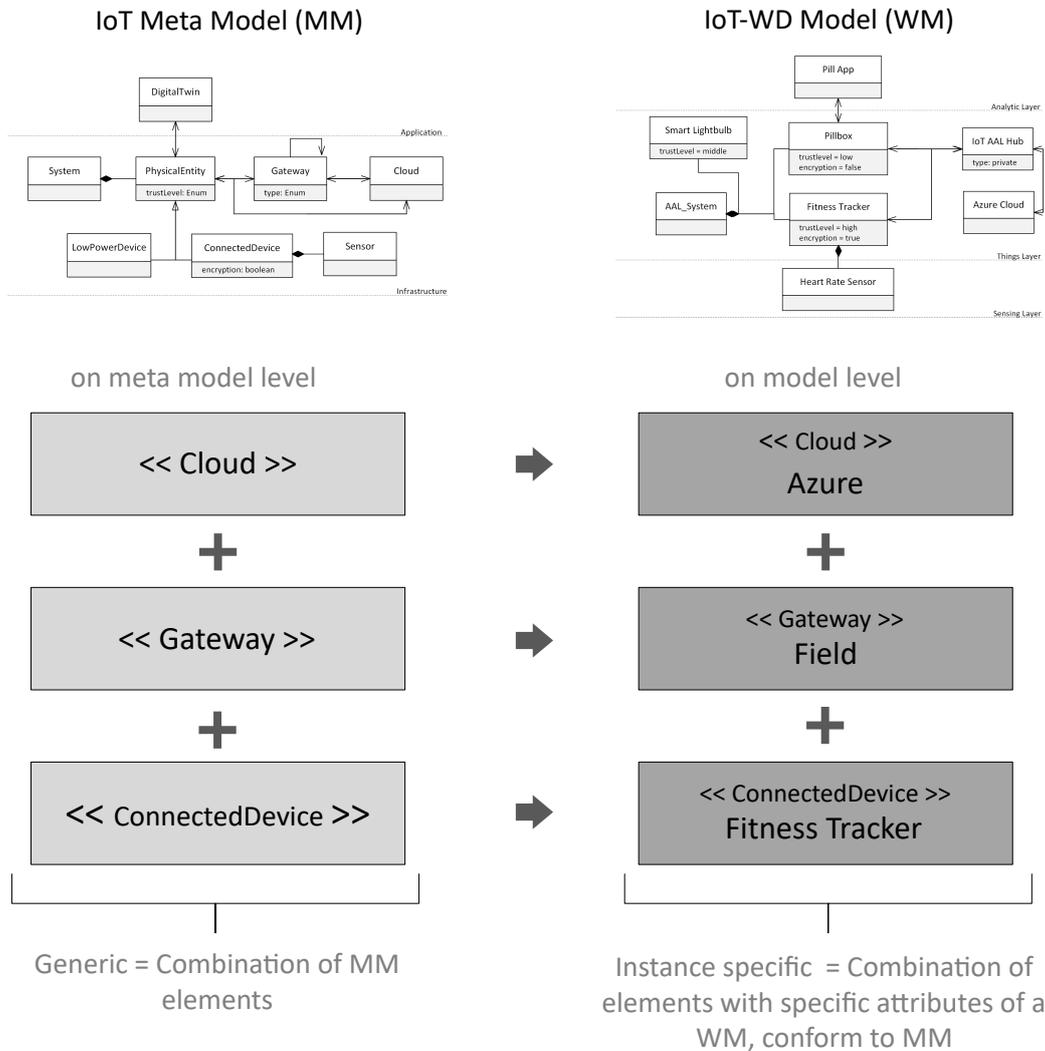


Figure 6.2.: Example of formalization conform generic and instance specific pattern/anti-pattern structure

6.1.2. Pattern Definition Approach

As described, the basis of pattern and anti-pattern definitions are the PRF templates. The presented formalization provides a highly generic definition of pattern templates. The following templates are geared towards IoT-WD and safety/security as described above. As the PRF covers different issues, different

safety and security PRF templates are needed which can be distinguished further depending on whether a pattern or an anti-pattern is to be defined. The rough structure of every template consists of four part types. Depending on which template is required, the composition of these parts and their categories varies. The structure of the different PRF templates consists of

1. a Generic Part,
2. a Safety **OR** Security Challenge Part,
3. a Safety **OR** Security Assessment Information Part,
4. a Pattern **OR** Anti-Pattern Implementation Part.

The first three parts are used for knowledge conservation and for later flaw and risk categorization or assessment. The fourth part is used for the implementation. Categories of the different parts are either of enumeration style or unrestricted textual value style. Enumerations are predefined sets of possible values. The categories of the included parts are based on architectural requirements from chapter 3 and meta model elements from chapter 4. In addition, they are supplemented by more categories necessary for the further course. The filling of the categories of the first three parts is optional, since not all information is always be available. However, the last part must always be entered completely, since it is used for the later implementation, as the algorithm functionality can only be guaranteed in this way.

The various possible parts and the composition of the templates are presented in the following tables. Table 6.1 provides an overview. The four different templates Security Pattern (SEP), Safety Pattern (SAP), Security Anti-Pattern (SEAP) and Safety Anti-Pattern (SAAP) are equipped with the ticked parts:

	SEP	SAP	SEAP	SAAP
Generic: Table 6.2	X	X	X	X
Security Challenge: Table 6.4	X		X	
Safety Challenge: Table 6.5		X		X
Security Assessment: Table 6.7	X		X	
Safety Assessment: Table 6.8		X		X
Pattern Implementation: Table 6.10	X	X		
Anti-Pattern Implementation: Table 6.11			X	X

Table 6.1.: Composition of PRF templates structure

For the AAL running example, an instance specific security pattern is defined be-

low, and thus, the SEP template is used. Afterwards, the running example can be recognized by green filled tables.

Challenge: Tampering, eavesdropping or data manipulation of lightweight devices without encrypted communication.

Architectural Inquiry: Device with weak encryption algorithm must use a field gateway to connect with a cloud gateway.

6.1.2.1. Generic Specifications

Every PRF template starts by using a *Generic Part*, which is presented in table 6.2 to specify the conditions that are independent of safety or security specific characteristics.

Generic Part	
ID	*unrestricted value*
Name	*unrestricted value*
Component	Supercategory: Element to Protect
Element Type	*restricted meta model or model element*
User Group	*unrestricted value*
Element Category	Enum: Physical Person, Privacy, Communication Channel, Leaf of Devices, Intermediate Devices, Backend, Infrastructure, Service, Facilities
Hardware	Enum: CommunicationProtocolType
Software	Enum: ServiceType
Layer	Enum: Layers of IoT Layered Architecture
Location	Enum: Local or Cloud
Disruption Tolerance	Enum: Tolerant, Temporary tolerant, Zero tolerant

Table 6.2.: PRF Generic Part

To be able to identify the defined patterns in the pattern database an ID and name is required. Apart from this, a supercategory for the protected element is included. This supercategory is used for pattern categorization within the pattern database and defines the element type and category, as well as the user group type. The element to protect categories are extracted of the IoT-A project ([Pro15]), whereas the elements and user groups are IoT(-WD) meta model conform. The element categories attempt to cover all aspects of an interactive IoT system and are elaborated of a special IoT security architecture approach. Therefore, users, different kind of devices, software and hardware aspects are covered. Since the

kind of hard- and software is decisive of the needed actions, types of physical connections and services are specified as well. To conduct analyses on diverse model levels, e.g. layered protection analysis, the affected layers must be set. For this purpose the introduced layered architecture approach is used. To determine the point of origin and responsible stakeholders the location of vulnerability is specified. This category helps to divide the architecture decisions. The last category views the disruption tolerance to categorize the sensitivity of the affected element, as described in section 3.2

Completed - Generic Part	
ID	876543
Name	Lightweight device without encryption
Component	Supercategory: Element to Protect
Element Type	WellbeingIoTDevice
User Group	Patient
Element Category	Enum: Leaf of Devices
Hardware	Bluetooth Connection
Software	Collect Service
Layer	Enum: Thing Layer, Physical Communication Layer, Middleware Layer
Location	Enum: Local, Cloud
Disruption Tolerance	Enum: Temporary tolerant

Table 6.3.: AAL example Generic Part

For the running example SEP, table 6.3 is filled in. For the pattern an ID and name is assigned for identification. The name should indicate the problem. Since this is an instance specific pattern, a specific lightweight connected device is specified. This pattern is used to check wellbeing devices, since they contain critical data and data manipulation would be serious. These are leaf elements of the meta model and mainly affect patients. The hardware setup provides for a Bluetooth connection for data collection and transmission services. Accordingly, the point of origin is distributed locally or remotely and distributed over several layers. This pattern corresponds to a temporary tolerant issue as it does not lead to an immediate, complete failure of functions.

6.1.2.2. Safety and Security Challenge Specifications

The *Challenge Part* is divided into the safety and security template version as described. In these parts specific categories are summarized which are dependent

on the challenge. Table 6.4 defines the specific characteristics security challenges bring along to vulnerabilities.

The intent and risk represent the aim and risk of loss of possible attacks. It is needed enable estimation of later consideration of countermeasure costs. To classify the type of attack the STRIDE categorization ([Mic07]) is used. As described before, it can be used to decide which prevention method is suitable for a weak point. The attack goal category corresponds to the categorization from section 3.2 to assess which attack methods are used, and thus, which elements are particularly at risk and must be included in countermeasures. All these categories are used for assessment and database sorting, too.

Security Challenge Part	
Intent	*unrestricted value*
Risk	*unrestricted value*
Classification	Enum STRIDE: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege
Attack Goal	Enum: Capture, Disrupt-Degrade-Deny-Destroy, Manipulation, Information Disruption, Host Attack, Network Attack

Table 6.4.: PRF Security Challenge Part

As described above, every security PRF template part has a corresponding safety part which includes safety specific categories (table 6.5).

Safety Challenge Part	
Fault	*unrestricted value*
Hazard	*unrestricted value*
Classification	Enum: Outage, External Problem, Loss of Service, System Corruption, Data Corruption
Fault Class	Enum: Attrition, Energy, Calculation, Change Impact, Configuration Management, Data, Interface, Logic, Omission, Timing, Initialization

Table 6.5.: PRF Safety Challenge Part

Within this part the fault and fault class are determined. The fault category describes the possible origin of failure. Whereas the fault classes define the possible type of failure. These classes range from hardware causes like attrition to software or logical causes like interface issues or miscalculation. In addition, the hazard describes the consequences of possible faults which can also be classified. For instance, the safety classification distinguishes between simple failures, complete outages, single service losses or a system corruption. External problems

can be mentioned as well, however, these issues are difficult to prevent. Though, countermeasures or safety measures can be taken into consideration. Both enumeration categories are extracted of section 3.2.

Completed - Security Challenge Part

Intent	Capture private information on communication way
Risk	Theft of PII, Manipulate data and change analysis results
Classification	Enum Information disclosure, Tampering
Attack Goal	Enum: Capture, Manipulation

Table 6.6.: AAL example Security Challenge Part

Table 6.6 continues the example SEP. As second part the *Security Challenge Part* is needed to define the intent and risk of the example. The risk in this case is seen in the interception and manipulation of PII. Accordingly, far-reaching analysis results can also be falsified. It is classified as information disclosure and tampering through possible capture and manipulation attacks.

6.1.2.3. Safety and Security Assessment Specifications

The assessment information of security flaws is shown in table 6.7. Assessment can have a wide variety of objectives and considers a wide variety of aspects. Accordingly, a suitable assessment feature must be defined for each pattern or anti-pattern, to which all further values of this template part relate. Privacy is often related to security and is taken into consideration during assessment as well. Therefore, PII can be mentioned to ensure attention to this aspect. The *Assessment Information Part* has a supercategory as well, to describe the possible direct impacts and their estimated consequences. These values are based on probability values and are subject to estimations or previous experiences. Two subcategories must be defined. First, a direct impact of a node that affects other nodes can be defined as causal type. Linear chains, branching forks and merging colliders represent the junction patterns of impacts on related nodes [PM18]. Second, the causal relation defines the severity of impacts. Next to direct impacts, indirect impacts exist. These kinds of impacts are more difficult to estimate and often not obvious. Therefore, the type of indirect impacts can be chosen freely and offers hints for system architects for further design decisions. Related to impacts is severity of attacks to assess the necessity of eliminating flaws. This category uses described categorization of section 3.2. The likelihood of occurrence can be divided into frequent, probable, occasional and improbable, and is intended to estimate how likely an attack is. This estimation is needed for later cost calculations of best and worst case scenarios and is based on [NIS12] and [Dep12]. As last category of the *Security Assessment Information Part*, the security requirements

are chosen. These requirements include the typical security aspects presented in chapter 2.

Security Assessment Information Part	
Assessment Feature	Enum: RequirementType
PII	*unrestricted value*
Probable Direct Impact	Supercategory: Estimated Consequences
Causal Types	Enum: Fork, Chain, Collider
Causal Relation	*unrestricted probability definition*
Indirect Impact	*unrestricted value*
Severity	Enum: Catastrophic, Critical, Marginal, Negligible, None
Likelihood of Occurrence	Enum: Frequent, Probable, Occasional, Improbable
Security Requirements	Enum: Confidentiality, Integrity, Availability, Manipulation Resistance, Privacy, Authentication, Non-Repudiation

Table 6.7.: PRF Security Assessment Information Part

The counterpart of the security assessment displays the *Safety Assessment Information Part* (table 6.8). Some aspects are equal to the security assessment. However, these aspects are not included in the *Generic Part* as the values of equal category still depend on safety or security specific issues. First category of the safety assessment also defines information details depending the assessment reason. Since there are several possible features, also multiple information can be set. Compared to the security counterpart of this template part, no PII aspects are defined. Equal, on the other hand, is the supercategory for elements to protect for estimating the consequences of direct impacts, as well as indirect impact definitions. However, this has a higher priority in this context as indirect impacts are more critical as these can cause new safety-critical aspects that can harm human beings. Equally, this challenge is often dependent on experience as well. Safety patterns require other categories for assessment like security since they cover the wellbeing of users. While the same categorization applies to the likelihood of occurrence as in the security counterpart, other distributions must be used for the definition of severity as the severity concerns patient health and not system functionalities. This is also presented in section 3.2. Since safety brings along its own needs, safety requirements are specified. For instance, typical requirements are the ability of recovery of devices or functions, redundancy of sensitive elements and data/device integrity to ensure right service results.

Safety Assessment Information Part

Assessment Feature	Enum: RequirementType
Probable Direct Impact	Supercategory: Estimated Consequences
Causal Types	Enum: Fork, Chain, Collider
Causal Relation	*unrestricted probability definition*
Indirect Impact	*unrestricted value*
Severity	Enum: Death, Serious Injury, Nonserious Injury, No Injury
Likelihood of Occurrence	Enum: Frequent, Probable, Occasional, Improbable
Safety Requirements	Enum: Recovery, Redundancy, Failure Resistance, Availability, Data/Device Integrity

Table 6.8.: PRF Safety Assessment Information Part

The next step in the AAL example is the definition of the *Security Assessment Information Part* of its SEP. The assessment feature availability is chosen in order to check whether data and devices are still available in the event of manipulation through missing encryption. This mainly concerns cardiac data. For the calculation of direct impacts, a chain causality is defined that would result in a decrease of availability in the event of an attack. This SEP is critical and has a high probability of attack. Accordingly, several security requirements must be included in subsequent analyses and countermeasure planning. For example, a new design must be tamper resistant and also a high level of data integrity must be ensured or not degraded.

Completed - Security Assessment Information Part

Assessment Feature	Availability
PII	Cardiac measurement values
Probable Direct Impact	Supercategory: Estimated Consequences
Causal Types	Enum: Chain
Causal Relation	Availability of A decreased 30%: P(B)-(1/10)
Indirect Impact	Impact on other health recommendations
Severity	Enum: Critical
Likelihood of Occurrence	Enum: Probable
Security Requirements	Enum: Confidentiality, Integrity, Manipulation Resistance

Table 6.9.: AAL example Security Assessment Information Part

6.1.2.4. Pattern and Anti-Pattern Implementation Specifications

Until this point, the template parts create the basis for knowledge conservation and later assessment. Therefore, following parts are responsible for the implementation and automated flaw detection. Additionally, the *Implementation Parts* are not safety or security specific. However, they are pattern or anti-pattern intrinsic.

The next table presents the *Pattern Implementation Part* that is used to check the presence of a protective design decision.

Pattern Implementation Part	
Protective Design	*unrestricted value*
Implementation	Supercategory: Artifact Combination
Filter Conditions	Subcategory: Concerned Elements
Node2Node	Enum: Available Node (Types)
Node2Relation	Enum: Available Relation (Types)
NodeAttribute	Enum: Available Attribute (Types)
Pattern Requirements	Subcategory: Recognition Details
Node2Node	Enum: Available Node (Types)
Node2Relation	Enum: Available Relation (Types)
NodeAttribute	Enum: Available Attribute (Types)
Flaw	*unrestricted value*

Table 6.10.: PRF Pattern Implementation Part

For documentation reasons a textual solution for a protective design initiates the implementation details. On this occasion a short description of the element, relation and attribute types should be given. This description is displayed after the conducted flaw identification to explain discovered details and to prevent misconceptions. A supercategory includes the specification of required combinations of artifacts like nodes, attributes and their relations. This is divided into two further subcategories. Filter conditions must be defined to filter out concerned elements during the identification process. Therefore, specific node, relation and attribute types are set. The enumerations of this supercategory depend on the used IoT(-WD) meta model and whether it is a generic or instance specific pattern. For example, generic patterns chose node types whereas an instance specific pattern set specific nodes. All depictable elements and characteristics must be selectable to check the whole IoT model and to define patterns for all aspects. In the second step of this supercategory, pattern requirements are defined. These are recognition details for the protective design that the concerned and filtered

elements must fulfill. A detailed algorithm explanation is given in the next sections. To highlight the flaw, the last point of the PRF is a textual documentation of the exact flawed feature.

Table 6.11 shows the counterpart of the pattern implementation, the *Anti-Pattern Implementation Part* that is similar to the pattern definition. A security specific anti-pattern explains a vulnerable design, while a safety specific anti-pattern documents a hazardous design. Accordingly, the category of protective design description is changed. Both variants are used for documentation and explanation after the flaw recognition in the IoT model.

Anti-Pattern Implementation Part	
Vulnerable/Hazardous Design	*unrestricted value*
Implementation	Supercategory: Artifact Combination
Element Filter Conditions	Subcategory: Concerned Elements
Node2Node	Enum: Available Node (Types)
Node2Relation	Enum: Available Relation (Types)
NodeAttribute	Enum: Available Attribute (Types)
Anti-Pattern Requirements	Subcategory: Recognition Details
Node2Node	Enum: Available Node (Types)
Node2Relation	Enum: Available Relation (Types)
NodeAttribute	Enum: Available Attribute (Types)
Flaw	*unrestricted value*

Table 6.11.: PRF Anti-Pattern Implementation Part

Furthermore, there is a supercategory for the required combination of node, attribute and relation (types) for element filter conditions and anti-pattern requirements as stated above. The process starts in the same way as mentioned before with an anti-pattern algorithm definition. However, in the anti-pattern requirements, details are defined that are not recognized in the optimal case. Once again, the flaw characteristic description of the element to protect finishes the PRF anti-pattern part.

To complete the SEP definition of the AAL example the implementation details are defined (table 6.12). To control that lightweight devices with a weak encryption uses a gateway of a specific type, the concerned elements are filtered first. For this purpose, the conditions are defined in the SEP. Only nodes that are of type `WellbeingIoTDevice` and have a `PhysicalConnection` are concerned. Additionally, the `Encryption` nodes of `PhysicalConnections` are needed to control the strength of its encryption. To define this example more concrete and specific, additional conditions are included. The node attributes are additionally

filtered to include only Trackers with Bluetooth connections and weak encryption methods. Nodes that do not meet these conditions should not be controlled within this pattern. The last step is to define the pattern requirements. Accordingly, the filtered element list is examined for recognition details. The elements of the list represent the presence of peerdevice connections to a Gateway of type FieldGateway. The absence of these is defined as a flaw.

Completed - Pattern Implementation Part	
Protective Design	Device with weak encryption algorithm has to use a field gateway to connect with a cloud gateway
Implementation	Supercategory: Artifact Combination
Filter Conditions	Subcategory: Concerned Elements
Node2Node	Enum: WellbeingIoTDevice/ PhysicalConnection/ Encryption
Node2Relation	Enum: physicalconnection/ encryption
NodeAttribute	Enum: deviceType:Tracker/ protocol:Bluetooth / type:weak
Pattern Requirements	Subcategory: Recognition Details
Node2Node	Enum: Gateway
Node2Relation	Enum: peerdevice:WellbeingIoTDevice8765432
NodeAttribute	Enum: gatewayType:FieldGateway
Flaw	gatewayType != FieldGateway

Table 6.12.: AAL example Pattern Implementation Part

6.1.3. Pattern Specification Language

However, at this stage the defined patterns and anti-patterns cannot yet be used for automatic flaw detection. Therefore, the templates are used to create a pattern specification language as a DSL which transforms the defined patterns and anti-patterns to computer readable patterns. The patterns are stored in the database in this form.

To realize the approach different technologies are needed. Therefore, the DSL and service code generation are conducted with related concepts of Xtext and Xtend. Xtext is a framework that comprises a powerful language for the description of textual languages and generates the model, parser, linker, type checker and compiler. [Xteb; Xtea]

The pattern specification language contains all described template parts and is structurally oriented towards them. The presented categories' enumerations are

realized with Xtext respectively Ecore enumerations. These can be extended at this point as well, if the analyzed IoT model requires changes. To enable the full configuration of patterns and a complete review of IoT systems, the DSL contains all IoT(-WD) meta model elements including all node types, attribute types, relation types and type enumerations. Following, the structure and rules of this DSL are presented with rule excerpts.

After a template type has been selected in the DSL, the *GenericPart* (listing 6.1) rule is navigated to. This corresponds to table 6.2 and all presented categories. Accordingly, all generic values from the templates are accepted in this rule. The arguments of rules are always defined by a keyword and a type assignment. These can be simple types, but also other rules. An example is component for definition of elements to protect. The *Components* helper rule is used for this and contains redirections to the rules for *Element*, *StakeholderType* and *ElementCategory*. In turn, these have access to all elements of the IoT(-WD) meta model. Thus, the use of the DSL provides an automatically specified selection of available meta model artifacts. The Xtext syntax can also be used to restrict the multiplicity of pattern components. For example, multiple elements can be declared as element to protect.

```

//Generic Part definition
2 GenericPart:
   'Generic Part' '{'
4   'id' id=ID
   'name' name=ID
6   component+=Components
   'hardware' physicalConnection=STRING
8   'software' software=STRING
   'layer' ':' layer+=Layer
10  'location' '{' (location+=Location (','?))* '}'
   'disruptionTolerance' ':' tolerance+=Tolerance
12  '}'
;
14
Components:
16  'Component' name=ID '{'
   'elementType' '{' (element+=Element)* (','?)* '}'
18  'userGroup' '{' user+=StakeholderType (','? user+=
   StakeholderType)* '}'
   'elementCategory' '{' category+=ElementCategory (','? category
20  +=ElementCategory)* '}'
   '}'
;

```

Listing 6.1: PRF DSL generic and components rule

Rules are also defined for the template parts for challenges and assessment information. These are not considered in more detail here, but correspond to the same structural rules presented above. The following listing corresponds to table 6.10, and thus, to the pattern template for implementation details. The rule *ImplementationPatternPart* contains the template categories as keywords. For the specific im-

plementation details the helper rule *Implementation* is needed to define expressed filter conditions and requirements. They can be designed in more detail by connecting them with helper words like **if**, **then**, **or** and **equals**. In addition, the *Node* rule is used which has access to the IoT(-WD) meta model elements.

```

2 //Implementation Pattern Part
3 ImplementationPatternPart:
4   'ImplementationPattern Part' '{'
5   'solution' solution=STRING
6   implementation=Implementation
7   'flaw' (('and')? flaw+=[Node|STRING] ('=' flawValue+=STRING)?)
8   '*'
9   '}'
10 ;
11 ...
12 Implementation:
13   'PatternRule' rule=ID '{'
14   //possible flawed element
15   'algorithmElement' element=[Components|STRING]
16   //element filter condition
17   ('filterCondition' filterNum+=ID '{'
18   (('if')? ('and')? ('then')? ('then not')? ('equals')? ('or')?
19   filterNode+=Node)* '}'*)
20   // anti-/pattern requirement
21   ('Requirement' reqNum+=ID '{'
22   (('if')? ('and')? ('then')? ('then not')? ('equals')? ('or')?
23   ReqNode+=Node)* '}'*)
24 '}'
25 ;

```

Listing 6.2: PRF DSL implementation pattern rules

Node rule (listing 6.3) is a junction to other rules for IoT(-WD) meta model elements. Each element has its own rule with keywords and features depending on the attributes defined in the IoT(-WD) meta model. Cardinalities are according to the meta model as well. If connections are to be defined, the respective identifier of a rule to the connected element is used, as for example in lines 12-14 of *Sensor* rule to define a relation to an element defined with rule *PhysicalConnection*.

```

2 Node:
3   PhysicalConnection | Service | PhysicalEntity | Sensor |
4   Actuator | Measurement | Network | IoTApplication | ...
5 ;
6 Sensor:
7   'Sensor'
8   name=STRING
9   '{'
10  ('layer' layer=Layer)?
11  ('sensorType' sensorType=SensorType)?
12  ('service' '(' service+=[Service|STRING] ("," service+=[
13  Service|STRING])* ')')?

```

```

12 ('physicalconnection' '(' physicalconnection+=[
    PhysicalConnection|STRING] ( "," physicalconnection+=[
    PhysicalConnection|STRING])* ')' )?
14 '}'';

```

Listing 6.3: PRF DSL element rules

The defined SEP of the AAL running examples is transferred into the pattern specification language to make the pattern computer readable and analyzable. The defined DSL is used to express the contents of all template parts with Xtext. Listing 6.4 shows the description of the implementation, according to table 6.12. After noting the protective design solution approach, the rules and keywords are used as presented to express the filter conditions and requirements. In addition, in line 12 an element is selected that corresponds to the element to protect determined in lines 4-6. The excerpt demonstrates that each filter condition must be implemented separately, but must be queried together using the condition keywords (**if**, **and** and **then**) in combination. For example, in lines 22 and 23 the filter for Bluetooth connections is created. In the same schema requirements are created. In this example only one requirement has to be set.

```

GenericPart{
2   id SEP8765432
   name LightweightDeviceEncryptionPattern
4   Component SEP8765432_ElementsToProtect{
   elementType{
6     WellbeingIoTDevice
   ...}}}
8
ImplementationPatternPart{
10  solution "Device with weak encryption algorithm has to use a
   field gateway to connect with a cloud gateway"
   PatternRule SEP8765432_1{
12   algorithmElement "LightweightDeviceEncryptionPattern.
   SEP8765432_ElementsToProtect"
   filterCondition FC1{
14     if
   WellbeingIoTDevice WellbeingIoTDeviceNameSEP8765432{
16     deviceType Tracker
   physicalconnection ("PhysicalConnectionNameSEP8765432")
18     gateway ("GatewayNameSEP8765432")
   }}
20   filterCondition FC2{
   and
22     PhysicalConnection PhysicalConnectionNameSEP8765432{
   protocol Bluetooth
24   }}
   filterCondition FC3{
26     and
   PhysicalConnection PhysicalConnectionNameSEP8765432{
28     encryption{
   AsymmetricEncryption
   AsymmetricEncryptionNameSEP8765432{
30     type undefined

```

```
    }}}}
32   Requirement PR1{
      then
34     Gateway GatewayNameSEP8765432{
          gatewayType FieldGateway
36         peerdevice(
              "WellbeingIoTDeviceNameSEP8765432")
38     }}}}
flaw
40   "GatewayNameSEP8765432" = "not FieldGateway"
      and
42   "PhysicalConnectionNameSEP8765432.
          AsymmetricEncryptionNameSEP8765432" = "undefined"
      }
44 }
```

Listing 6.4: Running example excerpt of Xtext

6.1.4. Pattern Service Generation

After concluding the first two manually conducted steps, the theoretical pattern definition is completed. To enable the automated flaw identification, applicable pattern services are required. To transform the structured, defined patterns and anti-patterns into executable code a code generation is required. Thus, after a pattern or anti-pattern is defined and saved in the pattern database, the automated code generation is conducted. Therefore, every specified pattern can be used for design optimization immediately. The transferred details of the PRF DSL are translated automatically into pattern services through the functions of Xtend. Xtend is a template based, flexible and expressive Java dialect to generate compilable source code. It is mainly used to write easy readable code. Thus, it is a tool for code generation. [Xtea] This code generation process is provided for all kind of patterns.

There are two kinds of code generation types. Structural and logical code generation. Whereas a logic code generation is based on the logical, semantical meaning of a DSL and depends on the sense of defined language parts, the structured code generation is focused on the syntactical construction of the DSL parts. The pattern and anti-pattern services are always structured the same way. Therefore, an explicit and structural code generation method is used. However, a split generation algorithm is needed to cover pattern and anti-pattern services as they use different flaw identification structures. To clarify the structure of the algorithm, pseudo code, presented in listing 6.5, explains the composition of pattern and anti-pattern services.

```

start:
2   M - A predefined IoT model
   P - The concerning anti-/pattern defined with the PRF which
       contains the details of the algorithm being applied
4   CL - An initially empty output set of filtered elements
   RL - An initially empty output set of flawed elements
6
input:
8   E - A set of element types to be protected such that E is part
       of P and M
   F - A set of filter conditions such that F is part of P
10  R - A set of pattern requirements such that R is part of P
12 begin:
   //fill CL with all elements
14  add elements of E to CL

16  //perform filter process
   while F set is not empty do
18    take one condition from F
    for each element of CL do
20      if element properties match filter condition then
        keep element in CL
22      else
        delete element from CL
24
   //choose of pattern type
26  if P is a pattern do
    //perform pattern requirement check on filtered list
28    add elements of CL to RL
    while R set is not empty do
30      take one requirement from R
      for each element of RL do
32        if element properties NOT MATCH pattern requirement then
          set element attribute "flawed" from undefined to true
34        keep element in RL
        else
36          set element attribute "flawed" from undefined to false
          delete element from RL
38  else
    //perform anti-pattern requirement check on filtered list
40    add elements of CL to RL
    while R set is not empty do
42      take one requirement from R
      for each element of RL do
44        if element properties MATCH anti-pattern requirement
          then
          set element attribute "flawed" from undefined to true
46        keep element in RL
        else
48          set element attribute "flawed" from undefined to false
          delete element from RL
50
end:
52  RL - A filled output set with flawed elements

```

Listing 6.5: Algorithm flaw identification process

Before flaw identification can be started, a predefined IoT model conforming to the meta model is required. In addition, the previous steps for pattern definition must be completed and lists for filtered and flawed elements must be created. Afterwards, the process requires an element to protect type, defined filter conditions and pattern requirements as input. Line 14 describes the preparation of filtered lists with elements which are traversed in lines 16-23. Elements that do not match the filter condition are removed from the list. After this first sorting, the algorithm splits depending on whether a pattern or anti-pattern is to be checked. Lines 25-37 include the algorithm for a pattern check. For this purpose the filtered elements are transferred into the flawed list. A second sorting is performed in a loop that examines each of the pattern requirements. The counterpart for anti-pattern is done in line 38-49. The result of this algorithm is a list that contains only elements that are labeled as flaws.

As the condition and requirement check acts dependent on the meta model and the concerning relation combinations, the pseudo code presents the outer conceptual structure. However, additionally the check depends on whether an attribute or a connected element is checked. This is done in a hidden inner structure that is located in lines 31 and 44. To reach an element or attribute in the model multiple paths are possible as the meta model is not a simple tree structure. Classes are connected with multiple classes through diverse kind of relations. For example, in the presented IoT(-WD) meta model, the class *User* can be reached through *Stakeholders* and an associated containment relation *user*. However, the element *User* also can be reached through *IoTDevice* with a simple bi-directional relation as this class inherits from *PhysicalEntity*. Therefore, diverse combinations of classes and relations are possible to reach a specific class dependent on the starting point of the pattern. However, as not every pattern is structured the same, multiple code generation rules for the inner structure have to be defined to cover all pattern cases. A few case examples of inner rules are:

- Attribute check of element to protect: No connection check or loop required
- Attribute check of connected element: Additional loop check and existence check required
- Attribute check of multiple branched connection: Additional multiple loop checks plus prior rules required
- Element list vs. single element: Connected elements to check must be distinguished in their multiplicity to add additional required loops
- Attribute list vs. single enumeration: Difference must be made between single or multiple verification
- Attribute check of multiple attributes or attribute check of attributes in de-

pendence: If requirements are designed conditionally, further branches must be included

Rules for the outer structure and multiple rules for the inner structure are required to be combined. This challenge has to be taken into consideration in the code generation, described afterwards. For this purpose, pseudo code is used as a template for the structure.

The code generation is realized by **def methods** and permanent text modules. For code parts that are always constant and not dynamic, text modules can be used. Dynamic parts which must be formed depending on the just mentioned inner and outer structure rules, according to the defined pattern, use **def methods**.

Listing 6.6. shows a Xtend snippet for code generation of a filter method for checking filter conditions. This method is used dynamically often, depending on the number of conditions. After the filter method is created in line 5, the *elementToProtect* is dynamically taken from the DSL and inserted. This creates the list of correct elements that is processed by the filtered method. In lines 8-10 further **def methods** are called to create dynamically and dependently the remaining components of the filter method.

```

...
2  def compileFilterCheck(Implementation impl){
   ',,'
4  <<FOR filter : impl.filterNum>>
   public void filter<<filter>>(){
6     List<<<elementToProtect>>> removeList = new ArrayList<>();
     if(filteredList!=null && (!filteredList.isEmpty())){
8         for (<<<elementToProtect>>> element : filteredList) {
             <<compileFilterMiddlePart(impl, impl.filterNode.get(impl.
                 filterNum.indexOf(filter)).class.simpleName.replace("
                 Impl", ""), impl.filterNode.get(impl.filterNum.indexOf(
                 filter)))>>
10        }
        filteredList.removeAll(removeList);
12    }
    else {
14        return;
    }
16 }
   <<ENDFOR>>
18 ',,';
   }

```

Listing 6.6: Xtend excerpt of code generation - filter check

The def method *compileFilterMiddlePart* is explained in listing 6.7. This is invoked depending on the respective filter condition. In this **def method** the different inner rules can be seen, e.g. between lines 3-14 and 16-28. Further inner rules are not included in the excerpt. For each inner rule different actions are taken. For

example, direct attribute check leads to direct inspection of elements in lines 8-10 including cases for each element from the IoT(-WD) meta model. As shown in the second inner rule, the first query is whether the relationship is a simple or a containment relationship. Afterwards, code is generated that creates a loop for each of the connected elements. Finally, code is generated to mark the flawed elements.

To generate code for checking the elements, additional **def methods** are needed for each meta model element, depending on the checking type of the inner rules. Thus, more **def methods** must be implemented, as indicated in lines 9 and 21.

```

1 def compileFilterMiddlePart(Implementation impl, String nodeName
2   , Node node){
3   '''
4     <<<<<< if direct attribute
5     <<IF nodeName == elementToProtect>>
6       //no additional if condition and loop needed if element
7       attributes are checked
8       //concerning element -> keep in list
9       <<switch node {
10        case node instanceof WellbeingIoTDevice:
11          attributesCheckWellbeingIoTDevice(node as
12            WellbeingIoTDevice)
13          ... <<<<<< etc.
14        default : ""
15      }>>
16    <<ENDIF>>
17    <<IF nodeName != elementToProtect>>
18      boolean check = false;
19
20    <<<<<< if attribute of connected element
21    <<IF impl.filterNode.get(0).eCrossReferences.contains(node)
22      >>
23    for (<<nodeName>> node : element.get<<nodeName>>()) {
24      <<switch node {
25        case node instanceof WellbeingIoTDevice:
26          attributesCheckWellbeingIoTDevice2(node as
27            WellbeingIoTDevice)
28          ... <<<<<< etc.
29        default : ""
30      }>>
31      node.setFlaw(true); //required to colorize connected impacted
32      elements in last step
33      check = true;
34    }
35  }
36  <<ENDIF>>
37  ...
38  '''
39  }

```

Listing 6.7: Xtend excerpt of code generation - filter check middle part

Listing 6.8 shows an example of excerpts from the **def method** *attributesCheckWellbeingIoTDevice*. This method helps to generate code to compare the properties of the current node with the filter condition. Using the node supplied by the method parameter, the defined values from the DSL are later transferred to the finished pattern service, as in lines 4, 7 and 10.

```

2   def attributesCheckWellbeingIoTDevice(WellbeingIoTDevice node)
3   {
4     <<IF node.deviceType!=null>>
5     if(element.getDeviceType()==<<node.deviceType.literal>>) {
6     <<ENDIF>>
7     <<IF node.layer!=null || node.layer.toString!="undefined">>
8     if(element.getLayer()==<<node.getLayer()>>) {
9     <<ENDIF>>
10    <<IF node.riskClass!=null>>
11    if(element.getRiskClass==<<node.riskClass>>) {
12    <<ENDIF>>
13    ...
14  }

```

Listing 6.8: Xtend excerpt of code generation - filter check attributes

The defined example pattern is continued in listing 6.9 and 6.10. A section of a finished pattern service code is shown. Based on the presented Xtend **def methods**, several filter methods are created. Listing 6.9 shows the first filter. A list of *WellbeingIoTDevices* that is created in advance is iterated through and checked for elements that are of type *Tracker* as specified in the SEP definition in advance. Negative checks result in exclusion from the list, as seen in lines 13-18. The defined pattern components can be recognized by the green coloring. The remaining code components are not dynamic, and thus, structurally always defined the same way. These elements are defined as text modules in Xtend.

```

public class JActionPatternSEP8765432 implements
  IExternalJavaAction {
2
3  ...
4
5  public void filter1() {
6    List<WellbeingIoTDevice> removeList = new ArrayList<>();
7    //uses current filtered list
8    //updates list
9    if(filteredList!=null && (!filteredList.isEmpty())){
10   for (WellbeingIoTDevice element : filteredList) {
11     //no additional if condition and loop needed if element
12     //attributes are checked
13     //concerning element -> keep in list
14     if(element.getDeviceType()==DeviceType.TRACKER) {
15       element.setFlaw(true); //required to colorize connected
16       impacted elements in last step
17     }
18   }
19   else {

```

```

18         //not concerning element -> remove from list
           removeList.add(element);
           }
20     }
           filteredList.removeAll(removeList);
22     }
           else {
24         return;
           }
26 }

```

Listing 6.9: Sirius service Java code excerpt - filter

The running example is continued in listing 6.10. After remaining filter methods, the requirements methods are applied. The defined pattern example contains only one requirement to be fulfilled. As defined in the templates, a connection to the cloud through a FieldGateway is required. Thus, starting at line 6, the filtered list of WellbeingIoTDevices of type Tracker is checked to see if such a connection exists for each Tracker. From line 9, trackers without such a connection are marked as flawed, while in line 17, elements with such a connection are excluded from the filtered list. This step is necessary in case of further requirement methods.

```

public void requirementCheck1() {
2   List<WellbeingIoTDevice> removeList = new ArrayList<>();
   //uses current flawed list
4   //updates list if there are elements to protect
   if(flawedList!=null && (!flawedList.isEmpty())){
6       for (WellbeingIoTDevice element : flawedList) {
           boolean check = false;
8           for (Gateway gateway : element.getGateway()) {
               if(gateway.getGatewayType() != GatewayType.FIELDGATEWAY)
10                  { //does NOT MATCH
                       gateway.setFlaw(true); //required to colorize
                           connected impacted elements in last step
                           check = true;
12                  }
                   }
14           if(check) {
               element.setFlaw(true);
16           }
           else { // match pattern -> not flawed -> remove from list
18               removeList.add(element);
                   }
20       }
           flawedList.removeAll(removeList);
22     }
           else {
24         return;
           }
}

```

Listing 6.10: Sirius service Java code excerpt - requirement

6.1.5. Pattern Identification

The last step represents the final identification of vulnerable or hazardous design flaws in IoT models. The created pattern services are used to analyze the design automatically. The results of the pattern recognition process depend on the kind of defined pattern. In case of identification of an anti-pattern all elements and relations, which match the definition, are highlighted as they aim to reveal a negative design decision. However, if a pattern identification is conducted to recognize flaws, only the elements that are not matching the definition are highlighted, as the definition represents desirable design. Every IoT model that complies with the developed IoT(-WD) meta model can use the content of the pattern database to detect flaws.

For this purpose, the ArchiAna tool is extended with Sirius-based functions. It has access to the pattern database, and can thus, call the services and analyze the depicted models. Figure 6.3 shows the modeled AAL example. The exemplary pattern is tested on this model. A `Fitness_Tracker` is identified that met the filter condition and is tested for the pattern requirements. Since the `AAL_Hub` is an IoT smart hub without encryption capabilities and is only used for data collection and aggregation, the security pattern is violated since no field gateway between the `Fitness_Tracker` and an `Azure_Server` is modeled. Thus, the affected elements and relationships are colored red and marked as flawed. In addition, the pattern details are output for explanation. This example shows that pattern or anti-pattern violations can be hidden and would not be noticed by visual inspection. Especially such problems would not be detected by a manual examination. In the next chapter, the discovered flaw can be further investigated, assessed and mitigated.

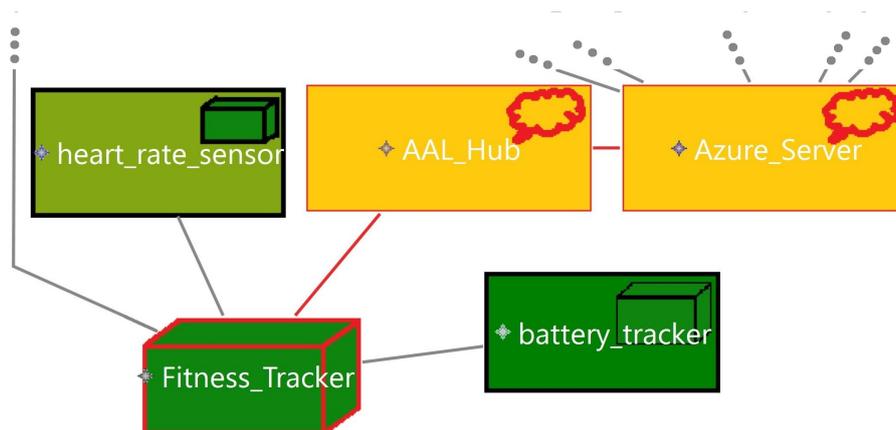


Figure 6.3.: Identified flaw in AAL running example

6.2. Pre-Defined Pattern and Pattern Database

As described, the goal of the PRF is to preserve knowledge even when experts are not currently available. Therefore, the designed patterns and anti-patterns are persisted in a pattern database. They are stored in their Xtext form. As soon as they are called, Xtend is triggered to create generated Java code and starts the pattern services.

In the following some pre-defined pattern and anti-pattern examples are shown. For this, the safety hazard or security threat is mentioned and its according architectural mitigation is shown which is basis for a pattern or anti-pattern definition. Complete pattern examples can be found in chapter 8.

Safety Challenges

Hazard: Medical devices with a direct influence on the wellbeing of humans have a zero tolerance to failures.

Architectural Mitigation: There must be a design diversity in the system, which is guaranteed by a different backup device. In addition, zero tolerance devices can only be connected to the Internet via gateways to reduce the amount of possible dangerous impacts.

Hazard: Devices with responsibilities for drug doses work incorrectly after changed configuration.

Architectural Mitigation: Infusion systems that can be changed subsequently must also contain a control service, including alarm actions, in parallel with the change service.

Hazard: Products to which people may be allergic can cause permanent damage.

Architectural Mitigation: Implanted devices must contain information on side-effects in their description, risk class and hardware information. In addition, the associated user must be connected to check compatibility.

Hazard: Measured values are subject to fluctuations caused by different environmental conditions.

Architectural Mitigation: The system must not perform measurements without device location identification.

Hazard: Improper handling of data units and metrics.

Architectural Mitigation: For each data record a compatible data unit and metric must be modeled.

Security Challenges

Threat: Bluetooth-enabled device is manipulated and prevents the correct execution of services.

Architectural Mitigation: Devices with a high SIL must only allow Bluetooth 4.0 or newer, as this does not allow an unlimited number of authentication challenge requests or short PINs.

Threat: Eavesdropping by communication from external to internal devices.

Architectural Mitigation: High distance communication must be implemented with LowRaWan as communication protocol to ensure secure coupling.

Threat: Physical attacks conducted through malicious node injection.

Architectural Mitigation: Each critical node must be registered in the identity store and connected as a peered device to its respective gateway. Additional installation of Universal Integrated Circuit Card (UICC) to prevent data breach.

Threat: Unauthorized access to a resource.

Architectural Mitigation: Access services must have a connection assigned to a user who has appropriate roles and rights.

Threat: Critical services can be triggered by remote control.

Architectural Mitigation: Services on resources with high risk and low trust level must not be remotely controllable. Services may only be exercised from a specific location or network segment of the resource.

6.3. Related Work

The presented flaw identification process differs from related approaches as follows:

Especially in safety- and security-critical areas, there are many approaches for detecting weak points. The SESAMO project, (**S**ecurity and **S**afety **M**odelling), focuses on safety and security requirements, aiming "to develop a component-oriented design methodology based upon model-driven technology, jointly addressing safety and security aspects and their interrelation for networked embedded systems in multiple domains" [Ses15]. This project aims in identifying safety and security hazards in order to calculate a trade-off between contradicting safety and security issues. Even though embedded systems and IoT systems are related, this approach is very different from the approach of this dissertation, since no design patterns are used. Accordingly, only related approaches that focus on patterns are considered. These can be divided in software patterns and design patterns.

Software and Code Patterns

One example of software patterns is the SonarQube platform. This offers various modules for analyzing source code. Static analyses as well as technical quality aspects of source code can be checked. For this purpose a kind of code pattern is used. Redundant code, complexity, potential errors and guideline violations can be determined or checked. In addition to highlighting problems, metrics can be used to evaluate and identify the source of the problems. However, this approach is not safety or security specialized and only includes code-level root causes. [Son]

Very specific software patterns, so-called code bad smells, were first identified by [Fow18]. These are to be used to refactor code. Since then, more and more software patterns have been detected. [ZHN11] have analyzed 319 papers that have presented such patterns. These range from simple coding problems to complicated structural problems whose origins can be difficult to identify. These include too large classes, use of lazy classes, incorrect switch statements, middle man concepts, or inappropriate intimacies. Safety and security relevant aspects were also explored in this context. However, these patterns are all defined in an unstructured way and no approach to a pattern development scheme is established. Thus, the quality of new patterns is strongly dependent on the use case and the creator. [ZHN11]

Since software patterns refer only to code artifacts, they do not include areas such as hardware components. In addition, they are often only used in fully deployed systems, and therefore, do not cover the design phase and early flaw detection.

Design Patterns

Design patterns can be implemented in different ways, but mostly they are still focusing on software details, even if they are defined or used in the design phase.

[SA14] present an IoT specific approach that introduces a dedicated pattern recognition layer into its reference model. With this, they enable that the pattern recognition is used in the lower levels of the reference architecture in order to lower the demand in higher levels. The pattern layer is connected with a semantic layer and a security layer. A separate pattern manager in the middleware provides a schema for the creation of patterns that can access information from the other layers. However, in this approach, only the pattern definition is performed at design time. The pattern query is done at run time and asks, e.g. for live data or big data to classify it. [SA14]

A similar approach by [Zan+14] focuses on data-intensive systems and provides a scheme for pattern detection. Among other things, the schema is linked to the domain model in order to incorporate design aspects. However, again the focus is on software and data design. Therefore, patterns are created with the help of domain models, however, are used at a later stage and not during design phase. In addition, information security is covered, but safety aspects are not included, which makes it inappropriate for critical systems.

In addition to the approaches that have developed a pattern schema for the creation, there are some approaches that offer specific design patterns, among others in the IoT or safety/security area, but do not present a structured schema for further pattern definition. These include the IoT design patterns from [Com14]. They want to offer use-case appropriate end-to-end solutions with their design patterns. They suggest how applications should be connected to things via networks, virtualization should be designed or middleware platforms in general should perform. They also try to describe the design patterns on system level, and thus, to design high level design patterns. Patterns for security, but also for information model aspects are presented. These are used to design an IoT system in the absence of a reference architecture. [Com14]

Another approach develops security patterns specifically for cloud vulnerabilities to preserve system and data security. These specific patterns are first divided into high level patterns that can be introduced at different abstraction levels and development stages. In addition, software best practices and design advices are also defined. The patterns are divided into areas such as secure architecture, secure development and identification/authentication. Each pattern is divided into its problem, context and solution. Again, no generic schema for further patterns is presented and no automated query of the patterns is proposed. [Rat+19]

Schumacher et. al ([Sch+13]) have published an entire summary on the evolution and use of security patterns. Therefore, the domain specific pattern type of security patterns and their characteristics are presented. They introduce a pattern landscape and the use of certain patterns in areas such as identification, access

control and firewall control. Patterns for different development phases are shown and their application is demonstrated. Although a pattern structure is presented for comparison, no structured development approach for new patterns is shown. In addition, no safety relevant patterns are included.

A case study on the application of security design patterns in IoT systems is presented by [LL17]. For this purpose, an IoT system is defined and patterns such as secure adapter, secure directory and secure logger are used to prove the security improvement through patterns. However, no further patterns are defined. Even though they are patterns on the design level, not all areas of an IoT system architecture are covered.

These approaches usually do not include, or combine, hardware problems or only indirectly, and therefore, do not cover the entire system architecture. In addition, they also do not provide automated query options of their defined patterns.

Design patterns for safety-critical embedded systems are presented by [Arm10]. Again, the focus is on fixed patterns and a catalog for tools is provided, including a pattern decision support. Depending on the available resources and the pattern requirements, this decision is made. The aim is to cover known problems in critical areas, such as handling of random and systematic faults. As an exceptional approach, non-functional requirements are covered by implications of non-functional properties in the pattern concept. After the decision support, this approach offers an additional evaluation of impacts of the pattern. The patterns are created using a template that roughly covers the same areas as the template of this work. However, this template is only used as a textual template and is not used for further creation of pattern services and automated flaw detection.

Another related approach originates from [Mem+14]. The SECTISSIMO framework is developed for security modeling and configuration of target platforms. A security vocabulary and patterns are used for this purpose. This approach is also used in the healthcare sector and incorporates corresponding requirements. A functional model, security model and a run time platform are defined. The used meta model differs significantly from the meta model of this dissertation. Their meta model is focused on activities, processes and their roles, and not on the contents of IoT models. For the pattern refinement process, a workflow is developed to refine defined patterns and adapt them more closely to the target platform. A defined pattern catalog is proposed but no schema for pattern development is included. In addition, this approach also includes code generation, but in contrast to this dissertation, model code is generated for the defined target model and not performable pattern code to identify flaws.

After the literature review of related approaches, no approach could be identified that focuses purely on the design phase, including all architectural aspects next to software details. In addition, the related approaches do not focus on safety and security in combination and neither provide a general framework to define further patterns or anti-patterns.

7

Flaw Assessment

This chapter covers the final steps of the presented optimization process introduced in chapter 5. Accordingly, the following subsections focus on the assessment of flaws. Hereby, the previously identified flaws that could be determined through the application of the PRF (chapter 6) are considered.

The main goal of this process step is covering *Objectives 3* and *4*. Accordingly, a structured methodology is pursued to transfer existing architecture approaches into IoT safety and security management in order to subsequently form an assessment workflow. This allows impacts or severity to be determined and countermeasures to be evaluated and proposed to prevent or mitigate serious problems in the design.

This chapter is structured as follows: It is initiated by the methodology of transferable architecture analyses and the subsequent identification and categorization of potentially suitable approaches. Afterwards, a DSL enables the generic description of these. Based on this, the IoT architecture assessment specifications describe requirements of IoT analyses adaptation, present a formalization of IoT architecture analyses and conclude with chosen approaches which are used as a basis for the IoT analyses approaches afterwards. The adapted analyses are arranged in a sequential cycle. The approaches' detailed steps are presented with examples including the provided ArchiAna parts. The chapter finishes with related work.

7.1. Transferable Architecture Analyses

The use of architecture analyses, or more generally analyses applied in the design phase, is not a new development. Almost every area applies approaches for the early phases of development sooner or later. In most cases, these approaches are very similar, follow the same aspects or use very similar techniques. Accordingly, this dissertation uses the experience of already existing approaches. It can be benefited from the fact that these analyses have already been successfully tested and evaluated. Since IoT is not a completely new development, but is the union of many different developments and fields, the related experiences can be transferred. However, it has to be noted that the transfer of analyses, their steps, characteristics and goals can only be applied in a limited way. Only parts of the basic frameworks and the target intentions can be transferred. The degree of transferability depends strongly on the particular analysis. In order to make an analysis adaptable and applicable in IoT, certain conditions must be met (compare section 7.2.1). For example, the new respectively adapted analyses must contain components that are able to capture the sensor-based architecture of IoT systems and to consider the data from these sensors and include them in the assessments. In addition, IoT architectures are likely to be open and more dynamic as architectures in other application fields. An important point for the changes and to enable the application in IoT is the connection of the analysis approaches with the presented IoT(-WD) meta model (compare chapters 3 and 4).

To achieve the main goal of this chapter - identifying suitable architecture analyses which can be adapted to the specific requirements of IoT - all available analyses have to be identified first. In order to limit the selection in a meaningful way, only analyses performed at the architectural level are considered, since the IoT flaw assessment should be performed in the design phase.

The potential of architecture analyses was recognized by many researchers before. However, the major part of architectural approaches is located in other application fields than IoT which leads to the described issues of chapter 1. One of the most strongly represented areas in this field of research is the already mentioned EAM. Since architecture analyses are successfully used in EAM, the analyses are applied in other research fields as well, e.g. generic cyber security approaches [ES09]. [Zim+15] represents one of the approaches which combine EAM principles with IoT. They describe the similarities of both concepts and methods. However, they are not including analysis approaches. Since the similarities have already been examined in this thesis as well (chapter 2) and EAM is one of the most successful application field of architectural analyses, this thesis goes one step further and restricts the identification of architecture analyses to approaches from the EAM area. Details about analyses in the EAM are given in chapter 2 and are discussed in more detail in the following subsections.

To provide a structured overview and subsequent selection, all EAM analyses performed at the architecture level are identified and afterwards categorized. Many of these published analysis approaches have the same type, name or goal. However, the techniques and individual steps and sometimes even the specific goal can differ significantly. Thus, a methodology is needed that allows the respective analyses with broadly the same goal and steps to be described in a generic way to summarize and cover the most important aspects per analysis. Therefore, on the basis of the created categorization a DSL is designed to create a generic analysis language which can be used for analyses description. This way, analyses approaches which are located in the same categories can be summarized and described generically. The categorization and DSL are crucial for the selection of analyses to be transferred into an IoT assessment cycle as they are used to select the characteristics and available techniques. This allows a subsequent matching between analyses already available in EAM and analyses required in IoT.

As mentioned in chapter 1 the content of subsections 7.1.1 and 7.1.2 were already published. Therefore, text passages are based on [RLB16] and [RBL16]. The content belongs mainly to the author of this thesis, as it is based on her previous work.

7.1.1. Identification of EA Analyses

As basis for the following identification a detailed literature research was conducted [Rau13; Rau15].

Analyses are chosen exclusively, which purely analyze EA and are not transferred from other topics. Hereof a pure EA analysis has the focus on collecting data and discovering the current state of an enterprise architecture in a quantitative or functional way to create a summary, alter the state or control different aspects. The goal of this selection is to create an overview of current EAM analyses and to receive approaches utilizable for a categorization. 105 EAM analyses are identified which are roughly grouped into 40 EA analysis types in previous work [Rau13]. An overview of these types and associated approaches and references can be found in appendix A or figure 7.1. Blue colored types have a high level of detail, whereas orange types missing some information and red colored ones are only a rough analysis idea. An analysis type describes analyses, which have the same rough scope and are built independently from the realization method. The goals of the contained analyses can differ significantly. Examples of types are *Quality Analysis* (e.g. [När+08]), *Requirements Analysis* (e.g. [Aie+09]) and *Analysis of Costs* (e.g. [Nie06]). The different types of analyses, which have been discovered in the literature research, can be treated as a first categorization. However, this categorization only makes raw statements about the rough purpose of the con-

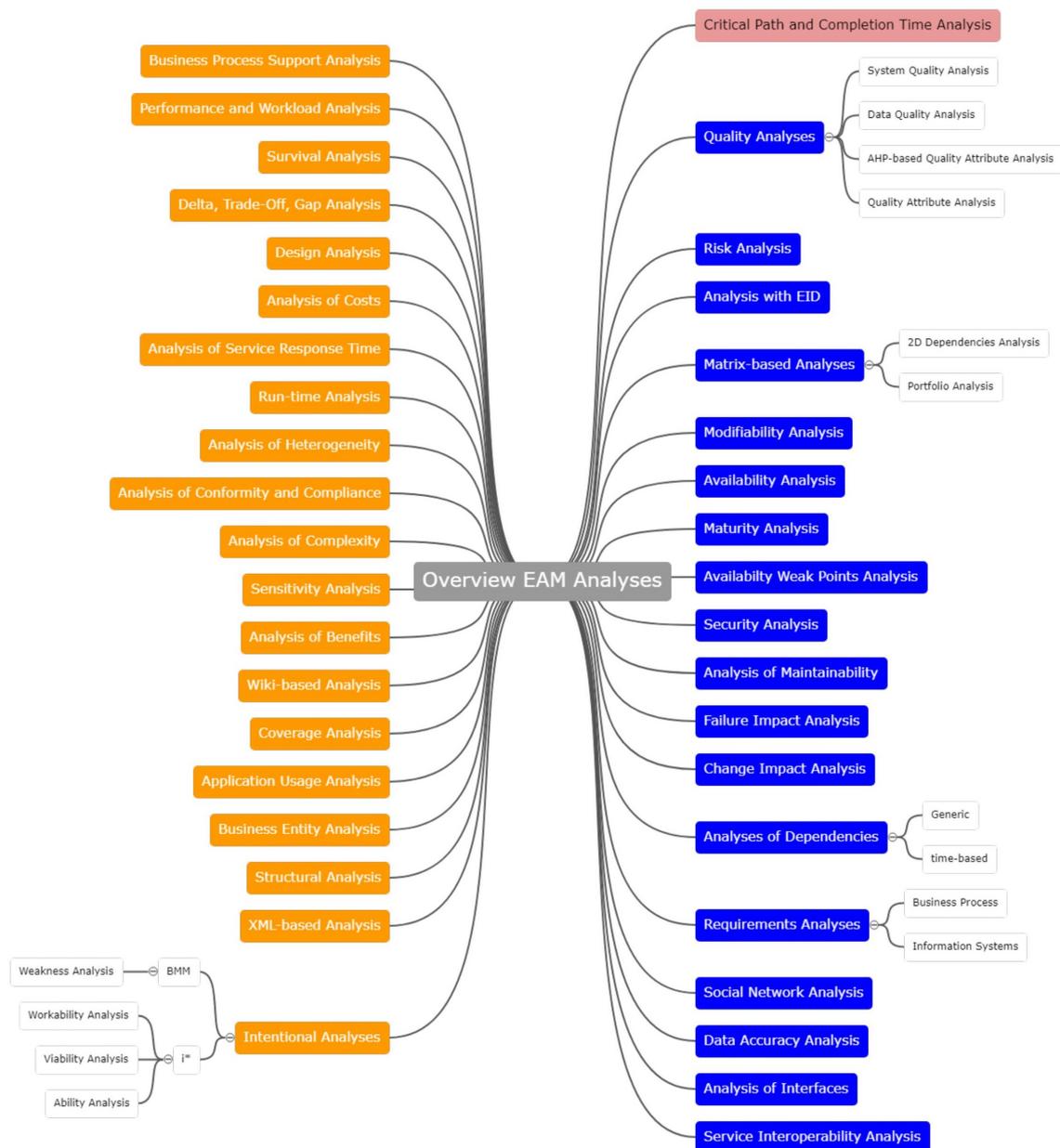


Figure 7.1.: Overview EA analysis types

tained analyses. Although analyses of the same analysis type have the same field of interest, their individual goals and implementations can differ. Thus, the classification in those types is not detailed enough to derive characteristics. Quality analyses, for example, can be conducted in various ways and can target different goals. For instance, this can be the quality of a whole system or maturity quality of a single artifact.

The huge amount of different approaches clarifies importance of EA analyses and coherence of a successful architecture. However, applying existing analyses on established models is expensive, since the corresponding meta models typically

require some adaptations [LSB14]. This makes reuse of existing solutions and research findings hard. Thus, an enterprise architect is appointed as expert. To cover the needs of an architect, the analyses pursue different goals and utilize different techniques. Therefore, in the following the different EA analysis approaches are examined and categorized according to their characteristics and requirements to get an understanding about each analysis. Characteristics of an EA analysis are defined as all necessary steps and components of an analysis to accomplish its goal. The characteristics are a main part of the categorization because they are guaranteeing the accuracy of the conducted analysis and the achievement of the goal. Figure 7.2 gives an overview of the categorization methodology which is used in section 7.1.2. The resulting categories are presented in the following section.

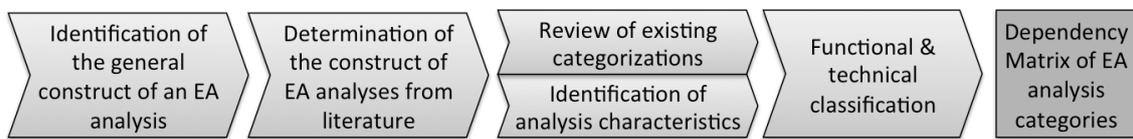


Figure 7.2.: Categorization methodology

7.1.2. Categorization of Analysis Approaches

The identified analyses could all be suitable for investigating safety and security aspects in IoT systems. Accordingly, all identified analyses will be included in the following further investigation. The goal is to identify truly suitable and adaptable analyses through investigation and better understanding.

However, only papers with a high elaboration level are used. Because of missing details it is not possible to analyze rough approaches and to identify their construct and characteristics. For elaborated analyses with less detailed parts, assumptions are necessary. In the case that an EA analysis approach is realized using another non-EA related analysis approach, this non-EA analysis is included too. This proceeding ensures the construction of a data basis with categorizable analyses according to the general construct of intake, processing and outcome.

A target of the categorization is to create a possibility to conduct analyses organized and controlled. Additionally, the categorization enables the creation of new analyses and the selection of the best suited analysis depending on the goal and requested technique to reach the final main goal: Enable the selection of suitable analyses for adaptation to create IoT architecture analyses. Therefore, the single analysis approaches are studied regarding their requirements for execution and their provided results.

Preliminary work for the categorization includes the definition of a general understanding of an analysis to determine a general purpose construct. Three main constructs of an EA analysis could be identified, which can be used as foundation for the categorization and determination of characteristics. These are data intake, processing and outcome. Other parts vary per analysis. Based on these parts the meaning and boundaries of a category are determined: Analyses can be merged to a category if they coincide at least in one of the three parts. As optimal condition all parts are equal, but this is usually not given. Therefore, different analyses are classified to the same category if they have at least the same target or same processing technique. Based on the experiences made while identifying the construct of the EA analysis it is possible to refine the categorization approach.

Considering existing kinds of categorization (e.g. [Lan05; BMS09]) this approach is conducted with two main fields of categories: functional and technical. This decision brought the most advantages in comparison with other approaches because of the division in “How” (technical aspects) and “Why” (functional purpose). The additional distinction in architecture levels is not included because a plain allocation wouldn’t be possible. Most analysis can be conducted in many levels or can only be performed by involving several levels. Through the new and detailed knowledge from the first evaluation of EA analysis constructs, characteristics are introduced to ensure accuracy. Analysis’ properties and steps are used as characteristics in order to retrieve detailed information about the category of each analysis approach. A two-dimensional classification is chosen and not a more detailed fragmentation, because of the high amount of differences within the approaches. Every analysis has special characteristics when sharing the same technique. Therefore, it is not possible to identify categories or classifications on a lower level of abstraction. However, a high abstraction level involves the danger of missing necessary details. To involve all aspects of every analysis in its functional and technical view is observed.

After this step the final categorization of the analyses is received based on the main idea of distinction between functional and technical. The business functions of every analysis are determined based on the concepts *purpose dependent division* - Fundamental, Main and Decision-oriented - and *activity dependent division* - System thinking, Modeling, Measuring, Satisfying, Comparing with requirements and Comparing alternatives [WC12]. These concepts are used to analyze the identified analysis approaches according to their goals and activities. Thereby the functional categories have been determined by using a prepared template of aspects. This template consists of the analysis activities, the intermediate objectives and the main goal. After analyzing all approaches 10 categories are identified from classifying the various analyses goals. Attention should be paid to the fact of multiple classifications. For example, a security analysis is able to analyze dependencies and requirements, and therefore, can be assigned to both functional categories.

After the functional classification is completed, technical categories are conducted. This procedure is more detailed and complex because of the large variety of existing methods in EA analysis. Only analyses with the same method and same steps of goal attainment can form a technical category. This constraint is necessary to enable discovery of shared characteristics. The already mentioned template is altered for creating technical categories. The new focus lies on the constructs, methods, techniques (including single steps) and artifacts. First, rough technical categories have been determined based on the dimensions *quantitative*, *analytic*, *simulation* and *functional* [Lan05]. After this preliminary stage detailed categories are created. Each identified analysis approach passed through this procedure. In contrast to functional categories every analysis is assigned to one specific technical category. As final result 17 technical categories are set.

As only properties and steps can show the components responsible for classification, characteristics to ensure accuracy are introduced. A characteristic as requirement is defined, since an analysis can only be conducted target-aimed with all indispensable artifacts. Requirements support the achievement of goals and are used to identify hidden characteristics [Van01]. Whereas properties can differ significantly, on some spots the most elaborated have to be chosen or create a higher abstraction level. There are two types of characteristics: category specific ones and general characteristics. The second type includes a meta model and scenarios, determined at the beginning of an analysis. Another universal characteristic is the main goal. These three characteristics have to be conducted for all analyses. Together they provide a high level of abstraction. For the specific characteristics five different kinds are distinguished. The conducted kinds of characteristics are important for the identification of properties from technical analyses. Whereas functional categories have rough properties, technical categories have similar structure. Identified characteristics are: *Input*, *Conditions*, *Construct*, *Measurement*, and *Output*. The basis of an analysis is always represented in terms of *Input* data. In every case an architecture or scenario, in form of a model, is needed to conduct the following steps and final measurements. Before the main part of an analysis can be performed, sometimes *Conditions* are needed. For example, the possibility of succeeding must be given. Most of the *Conditions* are analysis independent, and therefore, can be seen as generally valid. The main part and procedure of an analysis is the *Construct*, containing all details of the procedure. It's required to conduct all details successfully to be able to finish the analysis. Examples are detailed steps, mathematical algorithms, relationship types or weighting of artifacts. To prove and measure the results and its calculation, every analysis needs some kind of *Measurement*. This characteristic is responsible to witness the achievement of goals. Mainly a *Measurement* is conducted using scales, KPIs and metrics to control functional and non-functional goals [Dav89]. This characteristic can vary dependent on the analysis and its goals. As last characteristic kind the *Output* is identified. It includes the way of presentation and type of outcome such as percentage, graphics or matrices. This category is crucial because analyses within the same category should have the same *Output*. These characteristic

kinds are used to analyze the approaches again to receive detailed information. Through the new and detailed knowledge the analysis categories had to be rearranged on necessary points. New identified characteristics have been verified on correctness and necessity. After this step the final categorization of the analyses is received. Every analysis has multiple and complex requirements. Therefore, a table including all characteristics of a category is not feasible, because of the amount of details. To provide an insight of this amount of different characteristics short segments of categories and their requirements are presented.

Altogether 96 analysis approaches fulfilled the criteria and have been incorporated. Only nine of them couldn't be classified. These ones are too specific and individual for the mapping to a category. For each analysis exactly one technical category and at least one functional category is identified. To create an overview of all possible combinations of categories three matrices are created. Two matrices represent the combination of the analysis approaches and the categories. These can be found in appendix A (figure A.5 and A.6). The functional matrix has more possible combinations, because analyses can achieve more targets simultaneously. However, the technical matrix has only one combination per analysis. For example, [När+08] is assigned to the functional categories **System Analyses**, **Attribute Analyses** and **Quality Analyses** and to the technical one **Bayesian Networks**. To provide an overview of the functional and technical combinations both matrices have been joint, which resulted in a shared matrix (figure 7.3). Therefore, an overview of the realization techniques of a functional category and also the other way round for which analysis goals a technique is used, can be seen. The numerical values in the table represent the amount of analyses in current literature, which match both categories, the functional and the technical one. However, the sum of the values is more then 96 because of the fact that some analyses have multiple functional categories.

Functional \ Technical	Technical																		
	Bayesian Networks	Business Entities	PRM	Social Network	AHP	Time Evaluation	Tree	KPI	Comparison	Views	Lifecycle	Ontology	EID	Weak Points	Matrices	Design	Structural	Other	
System	1		2										5						
Attribute			7		3		1		2				4	2					
Dependencies	1		2				2		2		1	1			6				
Quality	2	2	8		2	6	1	2					4		6				2
Design				4					4			1			2	1	1		
Effects									3	2		1	4		5				
Requirements										2	1								
Financial								3						2	2				
Data												1							
Business Objects		1		3		2	1		2	1		1		2					
Other										1					2				1

Figure 7.3.: Dependency matrix of the functional and technical categories

Functional Classification

The following are the categories of the functional classification. They are listed combined with an example of an assigned analysis approach. Additionally a short description of the characteristics is given. The complete assignment of all identified analyses to the categories can be found in [Rau15]. Here only a few characteristics and exemplary identified requirements are given in detail.

System Analyses (e.g. [När+08]) check partial or holistic systems and encompass the analysis types *Quality Analysis* and *EID*. Mostly time quality aspects and their optimization are in the main focus. Analyses that are contained in this category are often also part of other functional categories because of possible sub-goals. Examples are an analysis of single quality attributes without considering other parts of a system or an analysis determining a possible impact. Analyses in this functional category have very different realization approaches, thus, various different techniques are utilized. Possible techniques are Probabilistic Relational Model (PRM) or EID.

For instance, specific attributes and their values are analyzed by **Attribute Analyses** (e.g. [RAB11]). Ten analysis types are joint in this category. The observation and management of attributes is the focus of such approaches. For instance, the different states of attributes with changing input can be analyzed or the availability of attributes can be observed. This category contains many approaches, because of the high demand of attributes in EAM. Following, there exist numerous different fields of applications as well as different realization techniques. But the focus lies always on attributes. Through the various fields of application, most of contained analyses are a part of another functional category.

Analyses which prove the relations between the elements are classified as **Dependencies Analyses** (e.g. [Saa10b]). The main goal of these approaches is the identification of dependencies in EAs and relations of single components to receive an understanding of the whole architecture. Therewith critical relations are identified and observed. Additionally, a risk analysis addresses financial aspects beside the relations. The methods range from comparison of scenarios to a weak point analysis of the relations.

Quality Analyses have the main focus on various quality questions regarding attributes, systems, architectures and other components, and target various subjective and measurable goals (e.g. [När+08]). Altogether 13 different analysis approaches target quality issues. This category is based on ISO 9126 standard of software quality metrics and analyzes maintainability, maturity, usability, accuracy, security, efficiency and interoperability. Based on the high variety of contained analyses types, also the possible techniques differ. For instance, PRM and EID can be used to analyze service quality. In most cases this category tries to observe subjective quality through comparisons of alternatives or metrics.

Another category represents the analysis of architecture design (**Design Analyses**), examples are [AGW11; KMP11]. Through receiving an overview of the architecture construct all design variants can be identified. Beside the analysis of holistic or partial design, business entities, procedures and components can be analyzed and used to optimize the architecture. An example for the concentration on a single part is the analysis of interfaces. Without this analysis a holistic overview of the architecture would not be possible. Therefore, it is indispensable for EAM. Analysis in this category utilize specific techniques with rare reuse like social network analysis.

All approaches which control impacts in architectures and actions are joint in **Effect Analyses** (e.g. [Boe+05a]). This includes *Gap Analysis* and *Sensitivity Analysis*. In contrast to dependencies analyses these approaches observe the direct impact and effects of changes in architecture elements. To conduct the effect the change of an artifact is simulated in the model. These artifacts can be data, attributes and quality features. The simulation is done identifying different perspectives and comparisons or using the method of extended influence diagrams.

Requirements Analyses identify the requirements to achieve states or goals (e.g. [Aie+09]). Therefore, the specific conditions have to be determined. Results are either specified values or features and specific business entities, like operations. The main goal of this category is to identify all requirements of an enterprise architecture. Examples are *Security Analysis* and *Survival Analysis*. If requirements are not analyzed, processes can not be aligned optimal and goals are not achievable. It's possible to additionally analyze a lifecycle and its changing requirements.

To identify costs and benefits **Financial Analyses** are used (e.g. [Nie06]). On top financial weak points and possible impacts can be discovered. These analyses present a measurement with mathematical calculations. Therefore, key figures and metrics determine the outcome. However, receiving affected entities is a side-effect of the result. Consequently financial analyses observe too high costs or uncertainty and hence, are an indicator of necessary architecture and procedure changes. While costs and benefits are only calculated in this analyses, weak points and risks can be identified for example with dependencies analyses. Financial analyses evaluate the economical success through assessing the costs and benefits architecture and trigger actions to improve them.

However, **Data Analyses** cover all kinds of data (e.g. [När+09]). The focus lies on quality and accuracy, because data is a critical factor in enterprises. This category has only one technical category, EID. Thereby the data values are analyzed for evaluation purposes. **Data Analyses** target mainly the data quality because the accuracy of data is fundamental for all EA operations.

Finally the category **Business Object Analyses** is identified and approaches like [Del+11] are included. Business objects of every kind, e.g. operations, artifacts and entities, which are part of the architecture are addressed here. This category analyzes single business artifacts and whole operations. Example for measurement procedures are the evaluation of time and therewith an optimal operation or the creation of views. Next to *Business Process Support Analysis* and *Business Entity Analysis* also *Social Network Analysis* belong to this category.

Technical Classification

In the following the 17 technical categories are described. Introduced characteristic kinds with their properties and goals are used. For the description the most important and marked characteristics must be chosen. Again only special chosen characteristics are presented and not all necessary steps for the execution are conducted. Since nearly all technical categories require an architecture model, scenarios and goals as *Input*, it is not always mentioned below. An overview of classified approaches with according categories can be found in appendix A including references.

The first technical category represents analyses conducted with **Bayesian Networks** (e.g. [När+08]). Analyses of this category utilize this technique to analyze the quality of systems and architectures. It is reused in other analyses as part of their procedures, e.g. **PRM** analyses. Requirements of the *Input* are a meta model with entities, attributes and references, different architectural layers and at least two scenarios. These requirements are the most common *Input* prerequisites. The *Condition* are defined attribute states and connection types. They must have discrete areas and are either a causal relation or a definitional relation. In the *Construct*, firstly a model with Bayesian Networks is built, including all nodes and connections of the architecture. A node represents a variable with conditional probability distribution. Therefore, in the next step probabilities of attributes and the whole model can be calculated while creating matrices with discrete ranges, connections and weighted attributes. In conclusion this category has probability values as *Output* and can answer questions about the probability of an attribute's status.

Business Entities is a method to receive artifacts on the one hand and to analyze quality on the other hand (e.g. [Del+11]). Here it can be distinguished between analyzing single entities, combined entities or their quality. As *Input* and *Condition* BMM and UMD diagrams with all relations and processes, the goal type, strategies and quality features have to be determined. The first step of the *Construct* detects advantages, operations and elements of strategies. As a second step, influencer and strategies are combined to observe the goals. Additionally, matching operations and their entities are identified and assigned. The *Measurement* quantifies the goal. Dependent on the chosen goal type, the strategy elements are evaluated. For instance an observation of maturity can be conducted by weight-

ing elements with a scale. Therewith the strategy with the highest efficiency is identified. The *Output* contains valued strategies, quality values and identified operations and entities.

PRM contains 14 analyses and is the most used technique (e.g. [Bus+11]). For instance dependency and quality analyses can be conducted with PRM. Therefore, artifacts and effects are the main focus. An EA model, scenarios, problems and goals are the *Input*. *Conditions* require controllable attributes and determinable goals and criteria for the later determination of metrics. As a pre-step of the *Construct* connections are defined and uncertainties are formalized. Hereafter a specific model is built and the PRM is used to calculate the conditional probability of all scenarios and of the dependencies and attributes. PRM can be seen as template of an architecture model. This model has a set of classes. Every class has attributes, values and references. The connections can be one out of five states. This model is conducted with every scenario of the input data. Therefore, it is possible to calculate the probability of every scenario. Additional, the probability of attributes is determined by using Bayesian Networks. The *Output* contains a probability for attribute values, scenarios and uncertainty values.

Social Network analyses (e.g. [KMP11]) differ deeply from the other categories. For their conduction questionnaires and all available documents, like bills and connections are required. For the *Input* all available nodes (= entities), connections and needed data sources have to be defined. Entities are persons, groups and companies which can have roles. Only if the methods are accepted, the analysis can be successful, because of the needed support of employees. As *Construct* clusters are built and properties can be checked. Additional new entities and connections are found. One method is combining the socio metric data analysis, questionnaires and other data sources to identify new entities and connections and to evaluate them afterwards. For the *Measurement* a matrix with entities and factors is created for quantitative evaluation with factors or for identification of weak points. An overview of the whole architecture model and its entities and connections on a social basis is found in the *Output*.

Analytic Hierarchy Process (AHP) can be used for analyzing attributes and quality aspects and is one of the most elaborated EA analysis technique (e.g. [RAB11]). Therefore, the requirements for execution as well as the procedure of the identified approaches are described in detail. Typical requirements for execution are a model, scenarios and uncertainties values. *Conditions* are expert knowledge, which is used for weighting, as well as quality attributes and their level of success. First in the *Construct* and *Measurement* the quality attributes with their criteria, subcriteria and level of importance are determined. Then the quality attributes are weighted through a pairwise comparison according to the architectural layers by experts. All weightings of the importance level are summarized in a vector and in the next step a prioritized vector of the layers is created. This prioritized list of quality attributes is used for a specific definition of the scenar-

ios. These scenarios are also compared pairwise to each other, which concludes in a table with the priorities of the scenarios according to the quality attributes. Finally the most suitable scenario is selected and the level of suitability and uncertainty of the selection is calculated. The results are described in the *Output* as prioritized lists of quality attributes and scenarios.

The method of **Time Evaluation** (e.g. [Lan05]) observes the quality of business entities and operations through the calculation of time values. Analyses which try to optimize the performance utilize this technical category. For example processes and entities are checked for weak points. As additional information to the common requirements, trigger and arrival times are required as *Input*. Rules are necessary to cut the architecture in views and conclude with five perspectives for single time measurements (*Condition*). In this category *Construct* and *Measurement* are combined and require specific time values and calculation metrics. For each view the specific time values are calculated. Examples are customer view and process view with processing time and response time. First, the workload calculation is conducted with a top-down approach. Afterwards the calculation are applied backwards with a bottom-up approach. Finally, all calculated times are summed up to a total time (*Output*).

Trees are used to analyze and identify dependencies, coherences and quality features. The *Output* of those analyses delivers the probability for the occurrence of a failure or specific quality attribute. All necessary operations, procedures, scenarios and time values are included in the *Input* dependent on the goal type. In the beginning of the *Construct* the goals, entities and relations are defined. Afterwards a fault tree is built using Bayesian Networks, containing all steps or events required for the execution. Thereby all given scenarios have to be conducted. For every component of this tree a conditional probability matrix is created to receive the probability of failures or quality attributes [När+11a]. For the *Measurement* the time values are summed up or probabilities are calculated.

The technique of **KPI** is used in most analyses with quantitative measurement. Because of the high variety of contained analyses, a high level of abstraction is developed. As *Input* a UML meta model with layers is required. The *Condition* is very important for this kind of analysis. A goal has to be defined according to the SMART criteria: It has to be specific, measurable, achievable, realistic and time-bound. The *Construct* starts with the identification of all artifacts that have to be analyzed and with the determination of the matching KPIs. In the *Measurement* the artifacts are evaluated and the determined values are compared. It is possible to measure single artifacts or to summarize them and evaluate the whole system. Another method for measurement is the usage of matrices, where two different dimensions have to be selected. For example, a matrix can present the costs dependent to different organization units. The result in this analysis category represents the goal achievement, unsatisfied quality constraints or the financial situation [Nie06].

Comparison is a simple but powerful method (e.g. [Boe+05a]). Next to whole alternatives, also single scenarios, processes, attributes and dependencies can be compared to each other. It is possible to compare different state in times, i.e. the As-Is with the To-Be, but also different alternatives, i.e. potential future scenarios. Requirements constraint the alternatives and support the achievement of the desirable vision. Additional rules are used to create a consistent model with all requirements and suitable to the end product. First, in the *Construct* viewpoints are chosen and a model is created containing all components which should be analyzed. This model can differ dependent on the analysis object. Afterwards the models are compared with a previous state or another alternative. In addition, the single elements are changed and the impacts observed. On this way all possible states can be observed and the best alternative to achieve the goals is identified. The results of the *Output* show what is required to achieve the To-Be state and the different impacts dependent on the input.

The technique **Views** is used to analyze aspects in detail or to create different perspectives. It is a powerful tool for EAM, nevertheless only a few analyses use this technique explicitly. However, several identified analyses utilize the concept of views in their procedure (e.g. [SK11]). The necessary *Input* and *Conditions* are chosen views, a distinct goal and determinable connections. Criteria and their desirable perspective have to be specified in the *Construct*. Examples are time measures like response time or processing time. After this, the views can be built with all required components. A definite *Measurement* is not contained in this category. However, views can be evaluated with criteria to observe whether the view can achieve its goals, for example focus on the processing time.

A less popular methodology is the observation of **Lifecycle** (e.g. [Saa10b; Aie+09]). These analyses ascertain requirements and dependencies through consideration of different lifecycle phases. Therewith changes are identified and it is possible to determine the state of an artifact at a specific point in time. For the conduction the lifecycle phases (states) of the artifacts have to be given as attribute assignments and time values in the *Input*. In the analysis *Construct* the lifecycle of the artifacts in the respective architecture part is determined. Afterwards, to check the state of an artifact at a specific point in time the life table method is used. Thus, the change cycles are preserved and the validity of dependencies can be determined. In the *Measurement* the probability for an artifact being in a specific state at a specific point in time is calculated. The *Output* is either this probability or the change cycle.

Using **Ontologies** is an uncommon analysis technique in the domain of EAM (e.g. [SKR13]). The contained analyses, *Change Impact Analysis* and *Structural Analysis* analyze dependencies respectively the architectural construct. A special meta model created with ontological rules is required as *Input* and *Condition*. The meta model defines the entities (i.e. artifacts and dependency types) of the EA model whereas the rules define the dependencies between the elements. The *Construct*

analyzes the entities and dependencies in order to determine specific sets of them. Afterwards dependencies, viewpoints and special factors are evaluated for the outcome.

EID is the third most used technique to conduct EA analyses (e.g. [Joh+07b]). Possible results can be statements about maintainability, security and availability. Therefore, systems, attributes, quality aspects, impacts and data are analyzed. The steps of the procedure are independent of the analysis type. The scenarios and alternatives have to be conducted and a goal must be defined. As *Condition* it has to be secured that the contained components can be built with EID. Afterwards all scenarios, goals and entities are represented as EID nodes and connections. For *Measurement* Bayesian Networks are used to calculate the probabilities of the attributes. Thus, it is possible to analyze dependencies by inferring changes and altered values.

For the identification of **Weak Points** and their costs the following requirements can be determined (e.g. [Xie+08]). *Input* data are workflows and resources as well as their availability requirements. In the *Construct* a matrix of the workflows and resources is created, which is used for the availability calculation. Whenever the availability is higher as the availability requirement, the condition is fulfilled. If this is not the case, an enhancement parameter helps to calculate the current level of availability. Afterwards the expected availability for every workflow is calculated. In addition, it is possible to weight resources and receive alternatives with higher availability. The *Output* is the assignment of availabilities to resources.

Another identified technique is the usage of a **Matrices** (e.g. [Szy09]). Application fields are for instance *Coverage Analysis* or *Maturity Analysis*. Matrices can be used in various ways, mostly for the presentation of results. The *Input* is a common architecture model with classes, types and relations. Additionally, the goal and application area is required. In order to build and evaluate the matrix, the dimensions and the kind of measurement have to be determined. Additionally, the elements have to be aligned within the matrices. Dependent on the measurement method a quantitative evaluation or a scale for discrete areas is conducted. The results can vary from quantitative outcomes to weak points, redundant artifacts and functional dependencies.

Analyses joint in the category **Design** are able to observe architecture design in a specific way ([AGW11]). The analysis identifies strengths and weaknesses of the architecture. As *Condition* the considered factors and expert knowledge is required. In the main *Construct* items and data are determined, factors are checked with questionnaires and a cluster analysis is conducted. Similarities and clusters are identified through this way. As *Measurement* a matrix of items and factors is built and evaluated. In the *Output* the results of the matrix evaluation represent the potential of a cluster.

The last technical category contains a **Structural** procedure ([Buc+09b]). This analysis tries to observe design through displaying obstacles of different architecture versions. Therefore, a documentation of the EA is required as *Condition* and the main part of the analysis consists of an observation of changes. The *Output* type is unique and represents potential obstacles caused by different versions.

Domain Specific Language for Classification

The characteristics of the identified 10 functional and 17 technical categories are formalized using a DSL. Through the DSL a generic analysis language is designed to provide a common description for EA analyses for an easy access to their goals and execution requirements. Such a language allows to make the requirements of an analysis visible in a structured way. Therewith the integrity and correctness of the requirements can be elaborated, i.e. if they are sufficient to describe the analyses in an adequate way. The development of this DSL represent the last step of investigating analysis approaches leading to the final goal: Structured and proved identification of architectural analyses suitable for adaptation to develop IoT analysis options.

For the language development Xtext [Xteb] is used again. The DSL is developed according to the meta model development process for abstract syntax development from [BCW17]. This incremental and iterative process consists of three phases: The Modeling Domain Analysis phase, elaborating the purpose and content, the Modeling Language Design phase, defining the meta model, and the Modeling Language Validation phase, verifying the correctness and integrity. For the last step representative EA analyses are selected for each category and formalized them using the modeling language. Difficulties and mistakes during modeling trigger a new iteration of the development process. The specific syntax is developed simultaneously with the abstract syntax due to the nature of Xtext.

The developed DSL is structured in general and in categorization specific parts. Listing 7.1 shows the main rule for the analysis language and the realization of the dimensions. General requirements that occur in all categories are summarized at beginning in the main rule including the name of the analysis, the required meta model and potential scenarios to evaluate.

```

MetaLanguage: 'Analysis Language'
2 //Domain Definition: General Requirements
  '{'
4   'PerformingAnalysis' analysis=STRING
   'Metamodel' model+=UMLModel ('{'
6   'Scenarios:' scenarioName+=NameIdentifier (scenarioModel+=
    UMLModel)*
    (";" scenarioName+=NameIdentifier (scenarioModel+=
    UMLModel)*)*
8   '}'')?
   'Goal' goal=STRING

```

```

10  'Goal Type' ':' goalType+=GoalType ('&' goalType+=GoalType )*
    '}'
12  //Choice of Dimensions
    ('Category functional Dimension' ':' functional+=Functional)
    ?
14  ('Category technical Dimension' ':' technical+=Technical)?
;
16  //-----Functional Categories
    -----//
Functional:
18  //Overview and Choice of func. Categories
    SystemAnalysis | AttributeAnalysis | ... |
        BusinessObjectAnalysis
20 ;
    //-----Technical Categories
    -----//
22 Technical:
    BayesianNetworks | BusinessEntities | ... | Structural
24 ;
    //Choice of a possible technique matching to the chosen func.
    Categ.
26 SystemAnalysis:
    'SystemAnalysis' (':' )?
28 ('Technique' analysisTechnique+=SystemAnalysisTechnique)?
;
30 SystemAnalysisTechnique:
    EID | PRM | BayesianNetworks
32 ;

```

Listing 7.1: DSL for EA analyses - main rule

For description of the meta model and the scenarios a language construct is developed that allows to specify them similar to a UML model. The goal of an analysis is modeled using a string and its type is defined with an enumeration. Possible goal types are: percentage, matrix, probability, dependency, object, effect, scenario, number or boolean. The choice of the analysis dimensions is realized considering the later usage behavior. First, the user can choose either the functional dimension or the technical one. The rule system of the DSL restricts the second dimension to those that are feasible. For example the functional dimension **System Analysis** has realizations with the technical dimensions **EID**, **PRM** and **Bayesian Networks**. The rule *SystemAnalysisTechnique* ensures the integrity of the selection according to the matrix (figure 7.3). If the achievement of a planned goal is most important, the functional dimension is decided first. Thereby decisions about the function and purpose of the analysis have to be made, as in this case the analysis technique is not in the main focus of the user. As second option the user decides first about the utilized technique. This option is used in case only a specific method or technique should be used, e.g. because of a tool restriction or availability issues. After choosing the technical category, it is possible to discover which goals can be achieved with it, i.e. decide about the functional category.

For each technical category a structure is implemented that satisfies the requirements specified before. The structure comprises statements for defining the *Input*,

the *Conditions* and *Construct*, the *Measurement* and the *Output*. Listing 7.2 shows an excerpt of this part of the DSL. Inside the five blocks the specific characteristics of the analysis are defined. In the example the *Input* block of a BBN analysis is shown. The rest of the content blocks are hidden for easier understanding. According to the complexity of the conducted analysis, two blocks can be summarized into one combined block or a block can also be omitted.

```

BayesianNetworks: 'Bayesian Networks'
2 ('Function' analysisFunction+=BayesianNetworksAnalysisFunction
   )?
   'INPUT' '{'
4   //Used Layer
   'Layers:'(layer+=ArchitectureLayer (','?))*
6   //Allocation of architecture
   'Metamodel' metamodel=[Model|STRING] '{'
8   'Scenario' scenarioAlloc1=[NameIdentifier|STRING]
   '}'
10 '}'
   'CONDITIONS' '{' ... '}'
12 'CONSTRUCT' '{' ... '}'
   'MEASUREMENT' '{' ... '}'
14 'OUTPUT' '{' ... '}'
;

enum BayesianNetworkRelationType:
18 Causal = 'Causal Relation' | definitional = 'Definitional
   Relation'
;
20 BayesianNetworksAnalysisFunction:
   SystemAnalysis | DependenciesAnalysis | QualityAnalysis
22 ;

```

Listing 7.2: Excerpt of the description of Bayesian Network analysis using the DSL

To illustrate the application of an analysis definition, listing 7.3 shows an example description of the information security analysis from [Joh+07b]. This analysis evaluates the architecture by calculating the probability of quality attributes for security. Corresponding to the main rule, the description starts with the analysis name followed by a specification of the meta model and two scenarios. The meta model describes the classes, relationships and attributes that are necessary for the analysis. The two scenarios represent different alternatives that should be evaluated. The scenario description is followed by the goal statement and the goal type, in this case *percentage*. Then the functional and technical dimension is defined. The functional dimension is **Attribute Analysis** and the technical one is **EID**. The remaining structure of the analysis specification is specific for analyses of the category **EID**. The *Input* of the analysis is straightforward the defined meta model and both scenarios. The *Construct* part defines the requirements in order to create extended influence diagrams. First the chosen scenario is set and afterwards, the nodes, goals and attributes with their types and values are defined. Finally, the EID specific relations are declared. In the *Measurement* part for each

node in each scenario a matrix with the conditional probability is specified. This is exemplary shown in the listing. The value of the node *User Training Process* is defined with an EID calculation. This calculation determines the probability of the node to be in a specific discrete range, here present, in dependency from further nodes. Finally the result and the goal calculation according to the Bayesian Theorem are declared. Such a calculation can be done for a quality attribute to have a specific value in one scenario. The *Output* contains the scenario with the best values according to the measurement.

```

EAM Analysis Language{ Performing Analysis "Information Security
"
2  Metamodel "Architecture of Information Security"{
   Class "Application"{ ... }
4  ...}
   {Scenarios:
6   "ScenarioModel: Scenario 1"{ ... };
   "ScenarioModel: Scenario 2"{ ... }
8  }
   Goal"Probability of quality attributes for security"
10  Goal Type :Percentage
   }
12  Category functional Dimension:Attribute Analysis:
   Technique Extended Influence Diagram
14  INPUT{ Metamodel "Architecture of Information Security"{
   Scenario 1, Scenario 2}
16  }
   CONSTRUCT{
18   EID MODEL ELEMENTS{Chosen Scenario 1
   //Value assumptions
20   Node: type: Decision Node "Scenario Selection" Value:
   0."90"
   Goal: type: Utility Node "Profit" Value: 0."0"
22   Attributes: type: Chance Node"User Training Process"
   Value: 0."75"
   ...
24   Relations: "Scenario Selection" as Causal Relation to "
   User Training Process",
   ...
26   }
   }
28  MEASUREMENT{
   //Example calculation for one node for one scenario
30  Chance Node:"Incident Management Process"
   Name "Scenario 1"->"Present":
32  Calculation of Section: P("User Training Process")= ...
   Result="0.95"
34  Goal Calculation:  $P(A|B)=P(B|A)P(A)/P(B)$ 
   Result: "Usage of Bayesian network analysis tool"
36  }
   OUTPUT{
38  Results: Best Scenario "Scenario 1"
   }

```

Listing 7.3: Excerpt of the description of EID analysis using the DSL

Application of the categorization on the further course

All identified categories, functional as well as technical, are integrated in the language and it is possible to formalize a representative from each category. Most of the requirements for the technical categories are realized in the language. A few requirements are determined as given and not further mentioned, since these requirements are obvious. Examples are the possibility to raise data, i.e. whether data can be used or be accessible. In addition, requirements which are not verifiable couldn't be included. For instance, it is not verifiable whether the meta model can be used to achieve the goals, whether artifacts can be mapped to EID components or whether used nodes are controllable. Additionally, the acceptance of a used technique or the availability of expert knowledge is not verifiable, and thus, not integrated in the language. Requirements that are defined in a graphical way, for example matrices, are difficult to realize in a textual language.

The lower number of functional categories in contrast to the technical ones can be explained with the focus on one field of interest. Analysis goals are repetitive since the concentration is on pure EA analyses. A problem during categorization is the issue that not all aspects from the analyses are described in detail in the available publications. At this point, it is only possible to identify limited requirements or to make assumptions in order to proceed. An interrelated problem is the fact of the low amount of available descriptions of conducted analyses to evaluate the language. Additionally some analyses use very specific techniques or modeling approaches. Here, it is not possible to consider all details in order to create a sound categorization. In order to define the general requirements for a category some specifics are abstracted. The general valid requirements are received by focusing on the approach with the highest level on elaboration and abstracting from it considering the issues of the other approaches. An example is the technical category **KPI** with a high abstraction level. The contained analyses differ deeply in measuring values with different formulas. Therefore, the mathematical computation cannot be described in full detail in the language.

In these last two subsections, architecture analyses that met certain conditions are identified and categorized. This categorization ended with a structured preparation of analyses' set ups and requirements to fully understand goals of these and their functionalities. The language can be reused for the development of new analyses, since it provides a high abstraction rate and a sound foundation of requirements that have to be included. Therefore, this preliminary and detailed investigation enables steps of the following sections of this chapter. It allows and prepares a justified selection of suitable and transferable analyses for adaptation purposes to develop analyses to assess safety and security aspects of IoT(-WD) models in the design phase.

7.2. IoT Architecture Assessment Specifications

The analyses of the IoT flaw assessment are presented below. They are based on related analysis concepts of the identified EAM analyses from the previous sections. Accordingly, the requirements needed for a concept transfer are presented and subsequently instructions are given on adapting the analyses for IoT purposes and making them usable for this dissertation's approach.

7.2.1. Analysis Adaptation Requirements

The overlap between EAM and IoT has already been described in chapter 2. Based on these facts, the identified EAM analyses can be used as a foundation for IoT analyses. In order for analysis concepts to be used as basis and be adapted to IoT needs, certain requirements must be met.

This is mainly due to the fact that EAM and IoT also have major differences. Different components, layers and stakeholders and their behaviors need to be considered. Also, the main characteristic of an IoT system, the open and dynamic architecture, has strongly different challenges than a closed enterprise system. In particular, IoT devices that are open to the outside world through an Internet connection and communicate automatically pose new challenges for assessment analyses. In addition, this also introduces new safety and security concerns that need to be addressed by analyses. Accordingly, existing analyses can be used in their basic concepts, but must be extended and adapted to cover all IoT specific aspects. Especially in the safety area adjustments have to be made, as EAM is mostly focused on security, as they are mostly only mission-critical systems.

Not all of the EAM analyses identified are appropriate for the evaluation purpose sought here. The selection criteria can be based primarily on the appropriate functional classification, since the objective of the analysis always has priority over the methodology used. In the second step, the associated technical methods are considered. The table A.6 in the appendix can be used as a guide. By preselecting the appropriate functional classification, the choice of methods is already limited. Through considering the objectives, such as automation, further analysis approaches can be excluded. In addition to the classifications, the interoperability of analysis concepts is also considered as a selection criterion, since the goal is an analysis cycle and not stand-alone assessments. Accordingly, analyses whose input cannot be covered by the previous flaw identification, system modeling or a preceding analysis result are eliminated. But also analyses whose conditions are designed for EAM specific conditions or analyses with unusable result types are not included. Based on these requirements, selected analysis concepts are adopted and adapted below.

7.2.2. Formalization of Architecture Analyses

In the following, IoT analyses for design assessment are formally presented before specifically selected and developed analyses and their intended use are described. The formalization builds on chapter 4. IoT architecture assessment analyses are formalized as tuple:

$$AA = (WM, SPT, MT, RUT, GT, RS)$$

where

WM as defined in chapter 4

SPT = {*Specification, Modeling, Mapping, Quantification, Simulation, Calculation, Comparison*} is a set of *step types*

MT is a set of *metric types*

RUT is a set of *rule types*

GT = {*Assessment, DesignDecisionSupport*} is a set of *goal types*

RS = {*Quantitative, Qualitative, Design*} is a set of *result types*

In addition, there is a function:

AF: SPT \rightarrow RUT is an analysis function that returns the rule types the step type is assigned to

An instance of architecture analysis is defined as:

$$AA' = (WM', SP, M, RU, G, RS')$$

where

WM' being an instance of WM as defined in chapter 4

SP is a set of *steps* being an instance of SPT

M is a set of *metrics* being an instance of MT

RU is a set of *rules* being an instance of RUT

G is a *goal* being an instance of GT

RS' is a set of *results* being an instance of RS

In addition, there is a function:

AF': SP \rightarrow RU is an analysis function that returns the rules the step is assigned to

7.2.3. Analysis Selection and Cycle Flow

For selecting analyses, the identified EAM analyses and defined adaptation requirements are used. First, the maturity level of analyses is considered. All red analyses from figure 7.1 are immediately excluded. Orange analyses are only marginally included due to their limited maturity. The remaining analyses are considered as described for their most important aspect: functionality. Accordingly, all analyses of the functional classifications *Business Objects*, *Data* and *Others* are excluded. The remaining functional classifications are suitable for an IoT design use case. Technical categories are considered afterwards. All analyses with unsuitable technologies are excluded. That include *Business Entities*, *Social Networks* and all manual technologies. Another adaptation requirement is the selection based on analyses matching the *Objectives*. Through the *Objectives* it can be derived that an analysis workflow should be created, and therefore, only matching analyses can be selected, furthermore impacts and As-Is or To-Be scenarios are in focus. This means that other inappropriate analyses, such as maintainability or maternity, are not included. Analyses that are too specific and have only one purpose are also excluded, such as analyses on interfaces.

What remains are analyses of failure or change impact, architectural security, dependencies and service interoperability. The remaining change impact approaches aim to plan changes and verify their requirements. Thus, unpredictable changes are not considered, which leads to their elimination. Most of the remaining analysis approaches can be implemented with BBN or with EID, and can thus, form a cycle with related input and output. In addition, a security analysis concept is included that uses KPIs and metrics as technique. Although this analysis cannot be transformed to fit safety, cost monitoring is a critical factor for subsequent countermeasure selection which must be included. Thus, the final analysis approaches could be identified that can be transferred and adapted to IoT. By summarizing and combining the approaches, four IoT architecture analyses can be created: FIA, QIA, CDSA and SIA.

The selected analyses approaches show the most potential for an IoT design assessment. This does not imply that other analyses could not be transferred. For other use cases and other requirements, it is possible to use other architectural analysis approaches.

Table 7.1 summarizes the selected analyses. Regarding the goals, there are two analyses each for flaw assessment and for the subsequent design decision support. They all provide a quantitative result, but some analyses provide additionally other types of results, such as design proposals. All four analyses concern safety and security challenges, except for QIA, since only limited safety aspects can be included in a cost assessment, as the wellbeing of a human being is only indirect quantifiable.

	Goal Type	Result Type	Safety	Security
FIA	Assessment: Impacts of flaws	Quantitative Qualitative	x	x
QIA	Assessment: Costs of impacts	Quantitative	lim.	x
CDSA	Design Decision Support: Countermeasure Alternatives	Quantitative Design	x	x
SIA	Design Decision Support: Service Interoperability	Quantitative Qualitative Design	x	x

Table 7.1.: Analyses characteristics overview

In this dissertation, the concept of assessment division is pursued. While some approaches have only one assessment each for security and safety aspects, a more detailed approach is followed here. A separated, step-by-step examination of impacts and their subsequent necessary consequences has the advantage that different techniques, approaches and objectives can be used or considered. Thus, a heterogeneous and diverse review is possible. In addition, the analysis cycle, presented below, does not necessarily have to be run through completely. This allows short assessment cycles and agile adaptations.

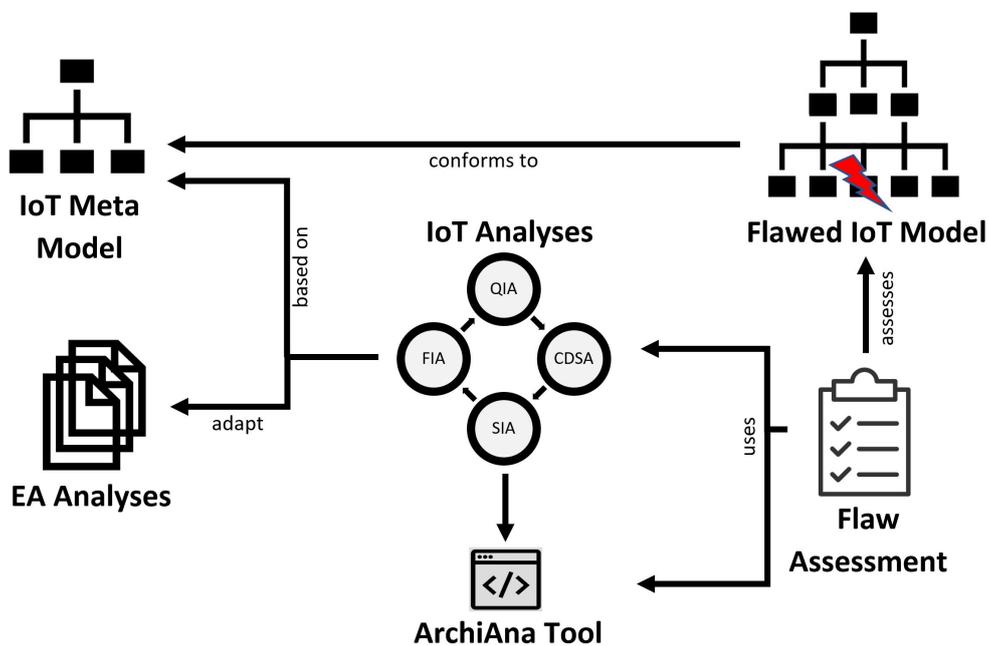


Figure 7.4.: Concept

The approach offers a holistic assessment cycle (figure 7.4) embedded in the design environment of IoT models which have to conform to the IoT(-WD) meta model. After identification of a flawed element, the further assessment included

in the ArchiAna tool comes into use. The cycle of assessment analyses starts with FIA to identify the affected elements and to provide the basis for QIA. QIA reviews the flawed path and estimates the financial impacts. Afterwards the assessment cycle switches to countermeasure decisions. CDSA provides an approach to model possible To-Be alternatives and offers a comparison of these. The assessment comes to a final design decision with SIA which checks interoperability of new services. The four components are based on the used meta model, as they require the used structure and information basis, and have adapted multiple architectural approaches which are realized in ArchiAna.

Thus, the major overall aim of this cycle is to consider the consequences of possible weaknesses in relation to various aspects and subsequently to weigh up possible alternative scenarios and their architectures. This is achieved through the presented assessment chain of IoT specific architecture analyses.

The four analyses of the IoT analysis assessment cycle are described in detail below and applied using the running example. For the analysis description, the structure presented in section 7.1.2 (listing 7.2), which is part of the EA analyses DSL, is used to guarantee a structured and consistent application.

7.3. Failure Impact Analysis

The idea of a transferable FIA is derived from [Hol+09]. The structure of the failure impact management process is transferred. However, the realization of the individual steps require to be adapted to IoT conditions. The ArchiMate version used can not be transferred for the modeling activities, as it does not contain any IoT elements. In addition, a different layer architecture is used as a basis. The most significant difference, however, is the leading question. [Hol+09] use a top-down approach to identify causes of defects, while the analysis cycle presented here requires a bottom-up approach for impact detection of already known flaws. Thus, the structure and logic of BBN is used, but no classical BBN theorem questioning is performed. In addition, comparisons and matrices are used to implement the prediction instead of diagnosis approach. Accordingly, calculations, data origin, tool functionalities, presentation and result interpretation must be adjusted.

After identifying a flaw, the next step is to analyze the impact on other elements in the event of an accident or attack based on the flaw. A FIA is aimed at this, by determining the probability distribution of the effects. By recognizing patterns, causal relations of flaws origins are already known. Accordingly, no top-down approach is needed for the system. A bottom-up approach to monitor effects on elements which are logical dependent on the faulty element has to be applied. This, in turn, provides information about which components could be highly negative affected by a flaw and have to be mitigated by countermeasures in next steps or need further assessments.

Input

In addition to the identified flaw and the affected elements, an IoT model with various probability scenarios is required to execute a FIA. The probabilities are taken from the pattern or anti-pattern definition with causal type and causal relation. The values are based on expert knowledge, hardware specifications, interviews or Monte-Carlo-Simulations. To enable the BBN mapping in the first steps, an assessment attribute is required that corresponds to a requirement type from the PRF. This is provided by the flawed element as input.

Conditions

The existing IoT model must have an assessment attribute assigned to each element. The IoT model must also be modeled with the model editor of the ArchiAna tool. The pattern or anti-pattern used must have fully defined assessment template parts.

Construct

The following meta model is used to build a FIA and its associated diagram. A *FIADiagram* is directly connected to the underlying IoT system to retrieve all stored

information from the system. In addition, each FIA is assigned a metric that is required for mapping. The diagram is divided into *FIALayers* which are oriented to the presented layered architecture. These layers contain the *FIAElements* which always correspond to an IoT model element and are the main component of the analysis. In addition, each element is assigned a *DiffMatrix* and various probability tables that are required for later calculations of the probabilities. The difference matrices are provided with a scale that can be used for the result comparison to estimate how severe the changes are in the event of an attack or accident.

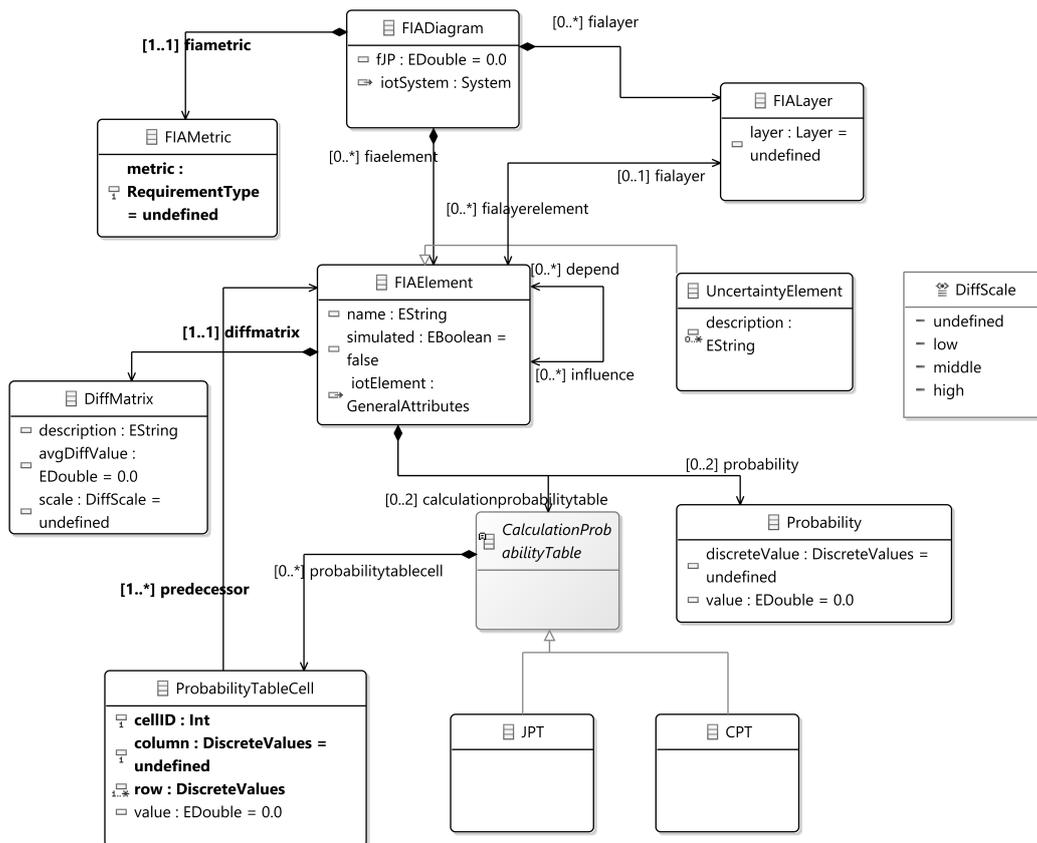


Figure 7.5.: FIA meta model

To perform a FIA, the following steps with their rules are required.

Specification

The specific assessment goal of a FIA is defined by setting an assessment attribute that is also used as a metric. This attribute can be, for example, availability, integrity or other safety or security specific requirements that are discretely quantifiable. Depending on this choice, different impact aspects are examined and different results are obtained by FIA's calculations.

Modeling and Mapping

No manual modeling is required to obtain a FIA diagram. An automated DAG mapping generates the diagram from the corresponding IoT model. Thereby only connections and nodes which could influence the specified assessment attribute are transferred leading to a diminished model with directed edges representing attribute dependencies. Which elements are transmitted can be identified by means of the requirement type stored as an assessment attribute in each IoT element. Connections that could pass on the assessment attribute but do not fulfill the BBN characteristics, such as no cycles, must be removed or adapted. IoT layers are transferred as soon as an element from the corresponding layer can be mapped. Another optional extension is the addition of so-called uncertainty elements. These represent a typical IoT problem. Dynamic IoT systems constantly have new nodes in their networks whose information is not immediately or completely available. These represent a particular safety or security risk. If necessary, these can be included in a FIA to consider possible new or external influences.

Quantification

Each element represents a variable in FIA which has discrete values, as e.g. 'Availability: UP/DOWN'. These values represent the probability that the current element fulfills or does not fulfill the assessment attribute and are stored in probability tables. These are independent probabilities, as these values are determined and quantified without the influence of other elements. In addition, a Joint Probability Table (JPT) must be created for each element and quantified with independent probabilities of the ancestors, as described in chapter 2. These tables indicate the probability that up to this point in the model the assessment attribute is reached or not reached. To check the entire system for the assessment attribute, the final joint probability is determined. This also corresponds to the JP of the elements that have no more descendants. It can be selected whether the best case or worst case should be calculated, depending on which JPT values of the nodes are used for calculation.

Simulation

After the diagram has been fully modeled and quantified, the occurrence of an attack or accident is simulated to calculate the impacts on connected elements. The simulation starts at the element that is determined by the previous flaw identification with the help of a pattern or anti-pattern. The simulation is performed by adjusting the probability table of the flawed element, and thus, setting it down to the new probability that the assessment attribute is still fulfilled. The new value is calculated by the defined causal relation in the associated pattern, which indicates the change in probability in case of a negative event. In addition, from now on conditional probability must be used, as the elements can now be negatively influenced by their ancestors.

Calculation

Since conditional probabilities are required from now on, an additional CPT must be created for each element. The described rules of the BBN basics apply. It should be noted that different cases of node dependencies must be taken into account. Figure 7.6 shows the four different causal types and table 7.2 presents the type 3 associated CPT with example values.

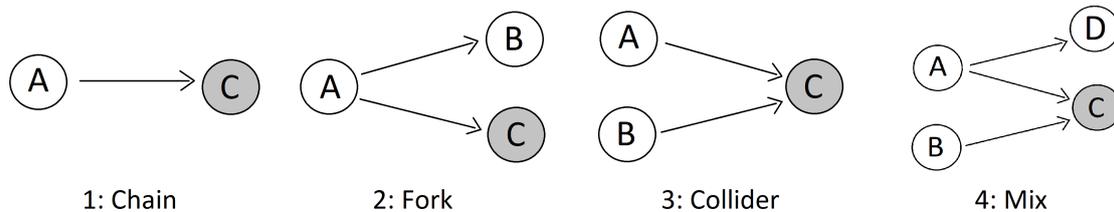


Figure 7.6.: FIA causal node types

Ancestor A	Ancestor B	UP	DOWN
UP	UP	0,9	0,1
DOWN	DOWN	0,01	0,99
UP	DOWN	0,4	0,6
DOWN	UP	0,3	0,7

Table 7.2.: CPT example of node C as collider

The types differ in the number of their ancestor and descendant nodes, and thus, also in terms of the structure and calculation of associated CPTs. A chain represents node connections that have exactly one ancestor and one descendant. The CPT values of node C are calculated by averaging the CPT values of its ancestor (denoted as $\emptyset CP(X)$) and its own probability. The average values are separated by discrete values to fill the individual cells of the CPT.

- $CP = \emptyset CP(A) * P(C)$

A fork connection consists of one ancestor and two descendants. The CPT of a descendant is calculated in the same way as for a chain, but the average probability of the ancestor is adjusted by the number of descendants.

- $CP = (\emptyset CP(A) / \text{number of descendants}) * P(C)$

A collider represents a more complicated composition, since at least two ancestors influence a node. Accordingly, the table shows in the CPT that both ancestors, broken down into their discrete values, have an influence on the probability values of node C. Since only one of the ancestors must inherit a negative event, an OR rule is applied:

- $CP = (\emptyset CP(A) + \emptyset CP(B) - \emptyset CP(A) * \emptyset CP(B)) * P(C)$

The mix type is the most common one and is a mixture of fork and collider. Accordingly, there are several ancestors, which in turn can have several descendants. Accordingly, for this type the calculation is complex and CPTs are extensive.

- $CP = \emptyset CP(A)/\text{descendants} + (\emptyset CP(A) + \emptyset CP(B) - \emptyset CP(A) * \emptyset CP(B)) * P(C)$

After the calculation of all CPTs, the JPTs of all nodes are updated. The rules of conditional probability defined in chapter 2 apply again and use the CPT values for the calculation. Thus, the probability is updated regarding the negative impact on the assessment attribute up to the respective node in case of an attack or accident. Finally, the final joint probability is updated, and thus, recalculated by multiplying all new conditional JPTs.

Comparison

To determine the extent to which the system and the individual components have changed negatively as a result of the simulation, comparisons are made. For this purpose, the difference matrices of the elements are determined, whereby the number of cells is based on the JPT of the element. For this purpose, the values of the old and new JPTs are compared and colored according to a severity scale. In addition, a layered JPT, covering all JPTs of a layer, can be used to observe the change of the assessment attribute of a layer. If further comparisons are necessary, multiple FIAs can be run with different assessment attributes and correspondingly different impacts.

Measurement

In order to perform a measurement, the quantification must be traceable and reconstructable. In the case of 'Availability' as assessment attribute, for example, the formulas of [Ogg01] can be used to determine the availability based on mean time before failure and mean time to repair. To perform measurements, the comparisons are used and the model before the simulation is chosen as a reference value. The final joint probability or the difference matrices can be used in this case. The difference matrices with their scale is divided into red, orange or green. Green matrices, and thus, barely changed JPTs are rarely or not affected by failures of the flawed element. Orange or red matrices are severely impacted and have, for example, a strongly reduced availability after an attack or accident and require mitigation measures.

Output

The result of this analysis is of type quantitative as well as qualitative. The quantitatively determined severity of impacts as well as the identification of design sections requiring countermeasures are provided as output. This represents the starting point for further assessments and needed actions.

The running example is continued in figure 7.7 and shows a complete FIA executed with ArchiAna.

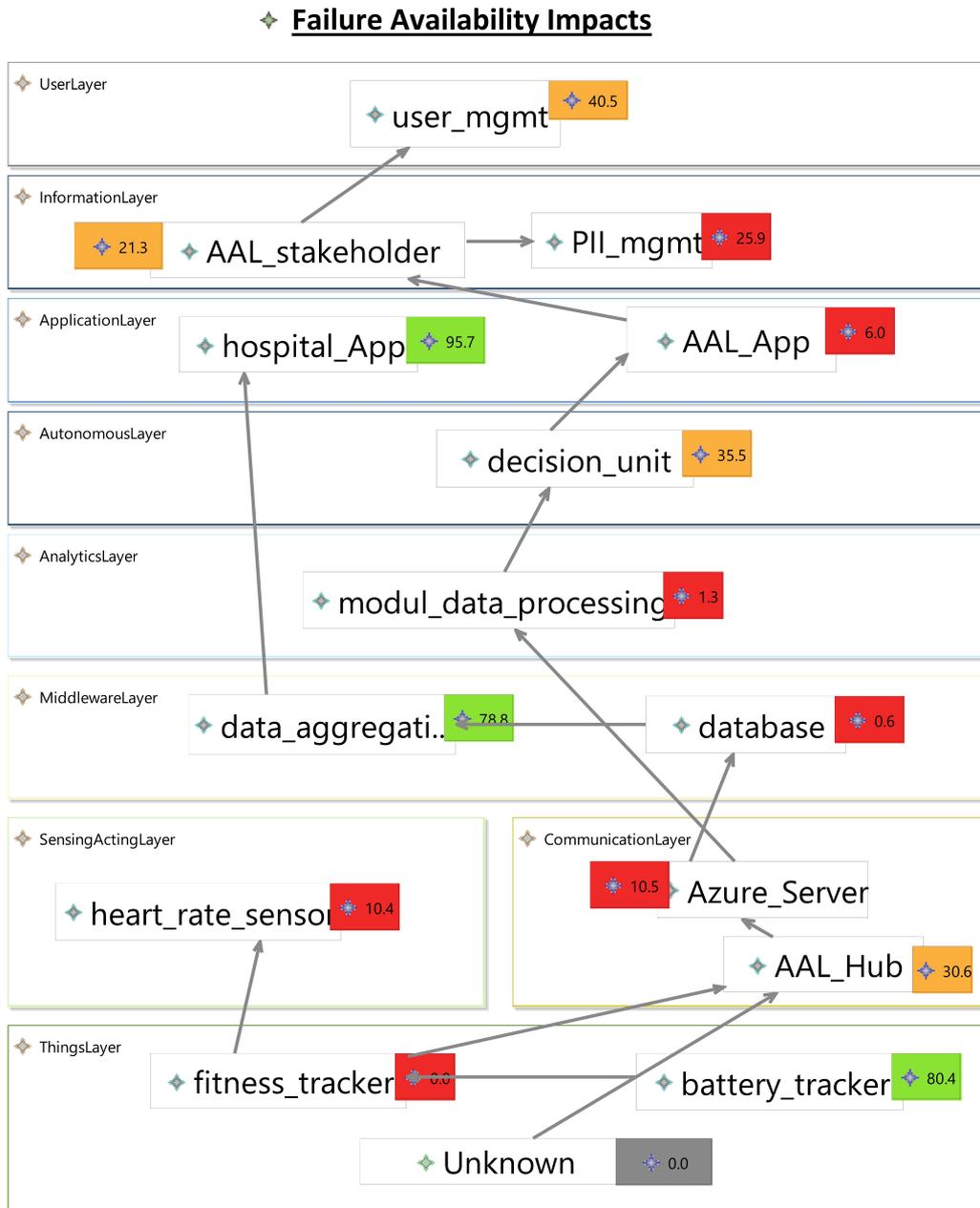


Figure 7.7.: FIA example

The elements of the diagram are based on the results of the example flaw identification from chapter 6. The elements identified as part of a violated pattern are the starting point here. Through the FIA mapping all elements and connections are transferred that are related to this element, if they are related concerning the selected assessment attribute. As assessment attribute Availability is cho-

sen as FIA metric. Accordingly, `digital_twin` and `user` elements, for example, are not included because they did not have set `Availability` as their assessment attribute in the model. In addition, an uncertainty element called `Unknown` is modeled in the things layer, since it is to be expected that new elements could connect to the `AAL_Hub`, and thus, have an influence or be influenced by a failure of the `fitness_tracker`. Subsequently, all elements are assigned their probabilities and JPTs for quantification. After the FIA set up is completed, a tampering attack on the `fitness_tracker` is simulated. As defined in the associated example security pattern, its availability is reduced by more than 30% in case of such an event. Subsequently, the CPTs and JPTS are recalculated and updated. The formulas mentioned above are used for this purpose. For example, the collider formula is used for the `AAL_Hub`, while the fork rule is used to determine the `Azure_Server`'s descendants. Assuming that table 7.2 represents the CPT for the `AAL_Hub`, ancestors A and B are the `fitness_tracker` and the uncertainty element. For example, it is determined that the conditional availability probability of the `AAL_Hub` is 90% in the event that both ancestors can obtain availability (*UP*). However, if the tracker is manipulated by an attack and is no longer available (*DOWN*), the probability that the `AAL_Hub` is not negatively affected is only 30%. After creating a CPT for each element, the difference matrices are created. These colored results can be seen in the example. Each element has a colored result that represents the average scale of the difference matrix. This shows the critical path, which means that these elements will be negatively impacted in the event of an attack on the `fitness_tracker`, and therefore, require countermeasures. For example, the JPT of the `Azure_Server` has an average probability of only 10,5% that it is fully available in case of an event. The uncertainty element has a gray score because no previous reference values are available, and therefore, a negative impact for the element could not be determined. Accordingly, a QIA can be started for the red and orange marked elements to calculate how expensive new countermeasure measures may be.

7.4. Quantitative Impact Analysis

The second analysis in the cycle is QIA whose basic idea is inspired by [BIY08] and [IB06]. The IoT(-WD) meta model and its layers are integrated in order to adapt it to IoT specific needs and to make it integrable into the assessment cycle. Instead of a dependency graph, an impact graph is used, which is generated automatically. All requirements needed for the analysis are not oriented to business objectives, but can be taken from the PRF. These are attached to all nodes with their risks. The weighting of the nodes is also taken from the calculations in the FIA or PRF. Basic concepts for measurement formulas are adopted. However, calculations are adapted to cover the IoT specific conditions.

QIA aims to estimate the cost of impacts, and thus, does not focus on the technical aspects of a flaw. It rather focuses on the upper layers and the financial burden. The background is to estimate the urgency and usefulness of flaw mitigation countermeasures based on the number of flaws and the probability of their vulnerability occurrence. This is done by quantitatively calculating the costs in an impact graph. Thereby QIA is the only analysis of the circuit that is not BBN-based and only include security aspects.

Input

All required data and models are taken from FIA results and patterns defined with PRF. Thereby, the relevant pattern is the one that is responsible for the current flaw identification that is used as input for the associated FIA. Security requirements, severity values, risk and attack goal are taken from a pattern as input. The impact graph is taken from the FIA result. This contains only the elements and edges that are identified as negatively affected by the flaw analyzed in FIA.

Conditions

A fully executed FIA and an existing pattern or anti-pattern must be available. In addition, a possibility of required prices for hardware components or service units are needed.

Construct

The following meta model is used to build a QIA and its associated diagram. The *QIADiagram* contains an assessment of the type Safety or Security. As there is no safety version available at the moment, this is prepared for future work. The security version receives its values from the PRF. In addition, a *QIADiagram* has a direct connection to the IoT model and *FIADiagram* used. Thus, each *QIAElement* can be assigned an element from the FIA, whereby only critical elements are transferred. Furthermore, each diagram has layers in which the elements are inserted. Elements can be divided into transferred leaf elements or calculated elements. Each element has a probability, a risk and costs per element. Enumerations are oriented to PRF enumerations.

as a restriction. In addition, the likelihood value can be taken from the PRF. The nodes from which the identified flaw originated are always used as leaf nodes. For all edges, the relation probability values are taken from the CPT of the FIA, for later calculations of the passing on of risks and costs.

Calculation

Four different types are available for calculating the costs. For this, the previously specified values are required. All non-leaf nodes must calculate their Rate of Occurrence (RO) for this. This is done by multiplying the occurrence value of the ancestor with the probability values of the edges. In addition, before the calculations start, it is necessary to ensure that each element has its Average Single Loss Expectancy (ASLE) set. This is the average cost that would be incurred to replace or repair a component, depending on the chosen requirements.

- Element Loss Expectancy (ELE)

Determines per component possible costs depending on the probability of occurrence. Special case with leaf nodes, as simple occurrence is required.

$$\text{ELE} = \text{RO} * \text{ASLE} \quad (7.1)$$

- Layered Loss Expectancy (LLE)

Layered Rate of Occurrence (LRO) gives additional information about the stability of a layer.

$$\text{LRO} = \sum_{e \in \text{QIAElement}} \text{RO}_e, \text{ if elements are in the same layer} \quad (7.2)$$

Indicative value for the expected costs of an entire layer assuming that all elements contained in the impact graph fail in a layer.

$$\text{LLE} = \text{LRO} * \sum_{e \in \text{QIAElement}} \text{ASLE}_e, \text{ if elements are in the same layer} \quad (7.3)$$

- Total Loss Expectancy (TLE)

Probability that all affected elements in the system will fail or be weakened.

$$\text{TRO} = \sum_{e \in \text{QIAElement}} \text{RO}_e \quad (7.4)$$

Determination of the maximum costs that can be incurred in the event of a total failure of the elements concerned.

$$\text{TLE} = \text{TRO} * \sum_{e \in \text{QIAElement}} \text{ASLE}_e \quad (7.5)$$

- Total Severity Expectancy (TSE)

Determines the magnitude of a total failure based on a severity factor taken from the PRF.

$$\text{TSE} = \text{Severity} * \text{TRO} \quad (7.6)$$

Measurement

The result type, and thus, the measurement is quantifiable in this analysis. The comparison with other scenarios can be used as a metric. So-called best case and worst case scenarios can be used for this purpose. These can be influenced by the probability values taken from the CPTs. For example, for a best case scenario, only the probabilities of the ancestors' values representing the optimistic case would be taken, such as *Fatigue is absent* as explained in chapter 2. Accordingly, one assumes a very unlikely occurrence of flaw, and thus, estimates the probability of occurrence and propagation to be low. These scenarios can be arbitrarily defined and compared with each other.

Output

The quantified output represents the complete cost calculation of an impact graph, and thus, covers the effects of the business side of an IoT system provider. This serves as a financial estimation for the countermeasures in the next analysis cycle step.

The running AAL example and its QIA can be observed in figure 7.9 which is exported from ArchiAna. A QIA diagram with layers can be seen that have affected elements determined in the previous FIA. For example, the *fitness_tracker* and its *heart_rate_sensor* are transferred because they have a red, and therefore, very critical difference matrix. This means that they would be strongly affected by an attack or accident. Since only the severely or moderately affected elements are transmitted, the impact graph of the AAL is shown. *Information Disruption* is selected as the risk of the diagram because of the threat of PII changes as defined in the example pattern. The mapped elements have their partial risk assigned. For example, the weak *AAL_Hub* is vulnerable to brute force attacks, while the analytics layer is at risk of integrity loss. For each element and layer, the ELE and LLE are calculated using the stored ASLEs. For example, an occurrence of 4 is estimated for the *fitness_tracker*. This means that 4 failures can be expected within 12 months. The ASLE for the repair is 13,4€. This results in an ELE of 53,7. Since only one element is in this layer, ELE=LLE in this case. In the information layer, both elements are added to obtain the LLE of 35,5. For example, to obtain the RO of other elements which is necessary for all further calculations, the occurrences from the *fitness_tracker* are multiplied with probability distribution of the relationship to *heart_rate_sensor*. In this example, 90% of the error is passed on to the sensor. This information comes from the CPT of the sensor calculated by FIA. The ASLE for the complex repair of a sensor is set with 8,60€. This

results in: $RO=4*0,9$ and $ELE=RO*8,6$. TRO is obtained by multiplying all ROs and, accordingly, TLE. For the TSE, TRO of 106 can be multiplied by the severity factor. This results from severity level of the diagram which is estimated to be marginal, and thus, calculated with factor 2.

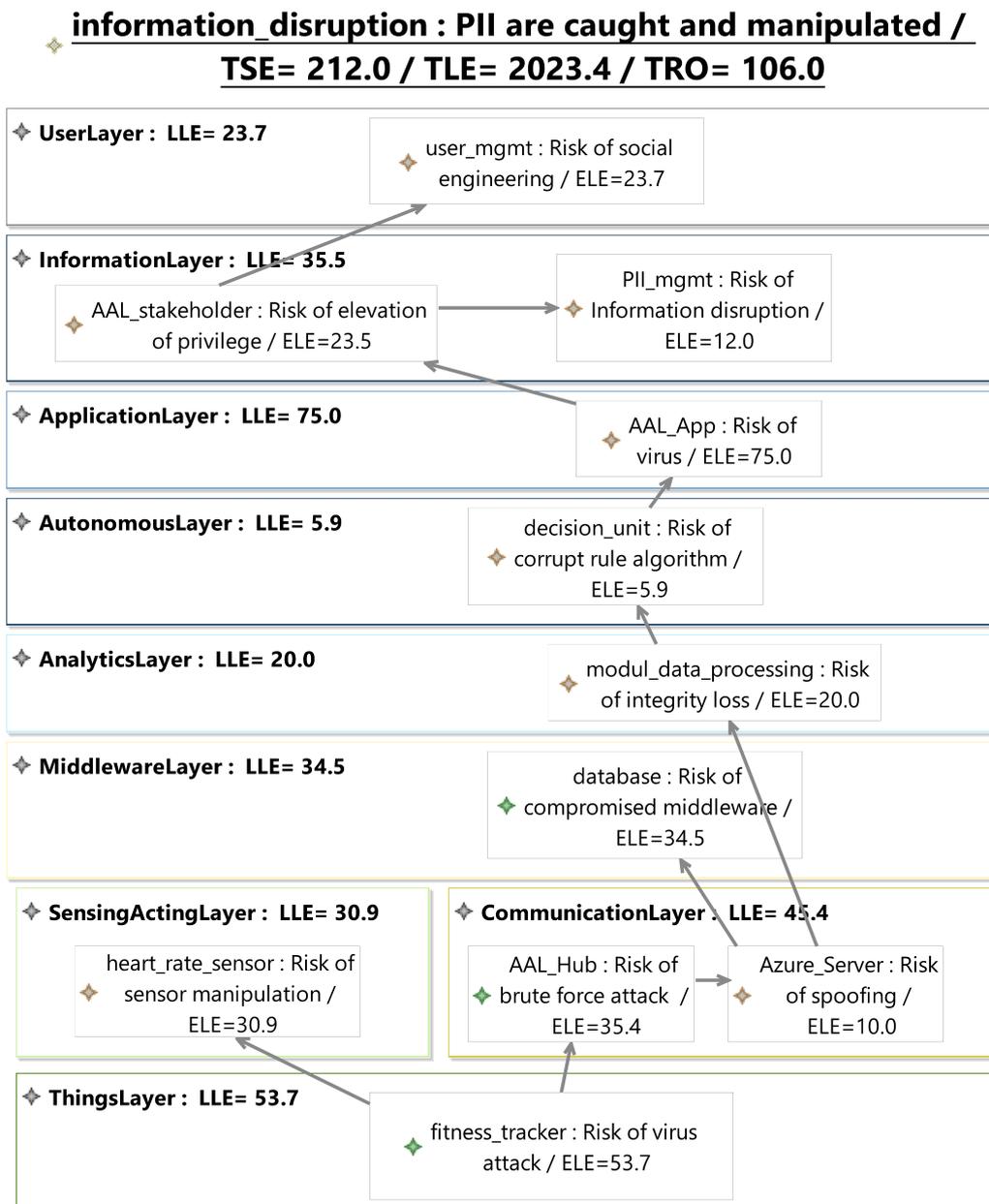


Figure 7.9.: QIA example

7.5. Countermeasure Decision Support Analysis

The CDSA is the first analysis in the cycle to focus on design decision support rather than flaw assessment. As an extension of BBN, EIDs are an ideal analysis method for IoT systems. Through the expandability described in chapter 2, the IoT system can examine a limited view or the entire details with all remote stakeholders and their dependencies. This allows for different levels of assessment and also takes into consideration uncertainties. In addition, EID is designed to analyze different scenarios in order to make a selection. Since the use of EID is considered appropriate, basic ideas from [SEJ08], [ES09] and [Joh+07b] are adopted. While [Joh+07b] focuses on generic EID analyses, [SEJ08] and [ES09] present an example use for defense trees. Here, all possible attacks on a scenario are represented and provided with countermeasures. These are afterwards analyzed for their chances of success. This is a suitable starting point for the IoT analysis cycle for design decision support, but needs to be adapted. For a CDSA, different types of chance nodes are required in addition to the decision and utility nodes. One for requirements of the to be prevented flaws and one for countermeasures to be checked. In addition, each countermeasure can be assigned to a countermeasure type. Furthermore, in a CDSA the countermeasures are divided into layers in order to apply different starting points for them. The approaches of [SEJ08] and [ES09] provide for the analysis of all attacks, whereas in a CDSA only one attack is examined, and thus, only countermeasures against this attack are considered with a strong focus. This prevents countermeasures from being pushed into the background by other attack problems and no longer being taken into account. The division into in-place and out-place transformations allows different types of countermeasure scenarios to be covered. In addition, a safety version is created to cover accidents.

The goal of a CDSA is to review necessary countermeasures to mitigate or prevent the identified flaw and potential impacts. Since various new design scenarios are possible, these must be weighed against each other and against other countermeasure possibilities.

Input

Information for the requirements nodes can be found in the PRF in the Assessment part. The different countermeasure scenarios can be based on the previous best case and worst case scenarios of the QIA. The specific countermeasures must be specified by experts.

Conditions

In order to implement a CDSA successfully, the entire flaw assessment analyses must be completed in advance. Only at this stage it is possible to assess which countermeasures might be useful and which elements require assistance.

Construct

The following meta model is used to build a CDSA and its associated diagram. A *CDSADiagram*, like the other analyses, is directly connected to the associated IoT system. A diagram is divided into several *Scenarios*, with *ScenarioTypes*, which can be compared with each other. Each scenario has a *Utility* and associated *Requirements* to define the target of the countermeasures. However, the main components are *Countermeasures* with a *CountermeasureType*, which can be divided into *LeafCM* without ancestors, or *ConditionalCM* based on previous countermeasures. Each countermeasure is assigned to one or more requirements whose influence can be calculated by assigning different probabilities.

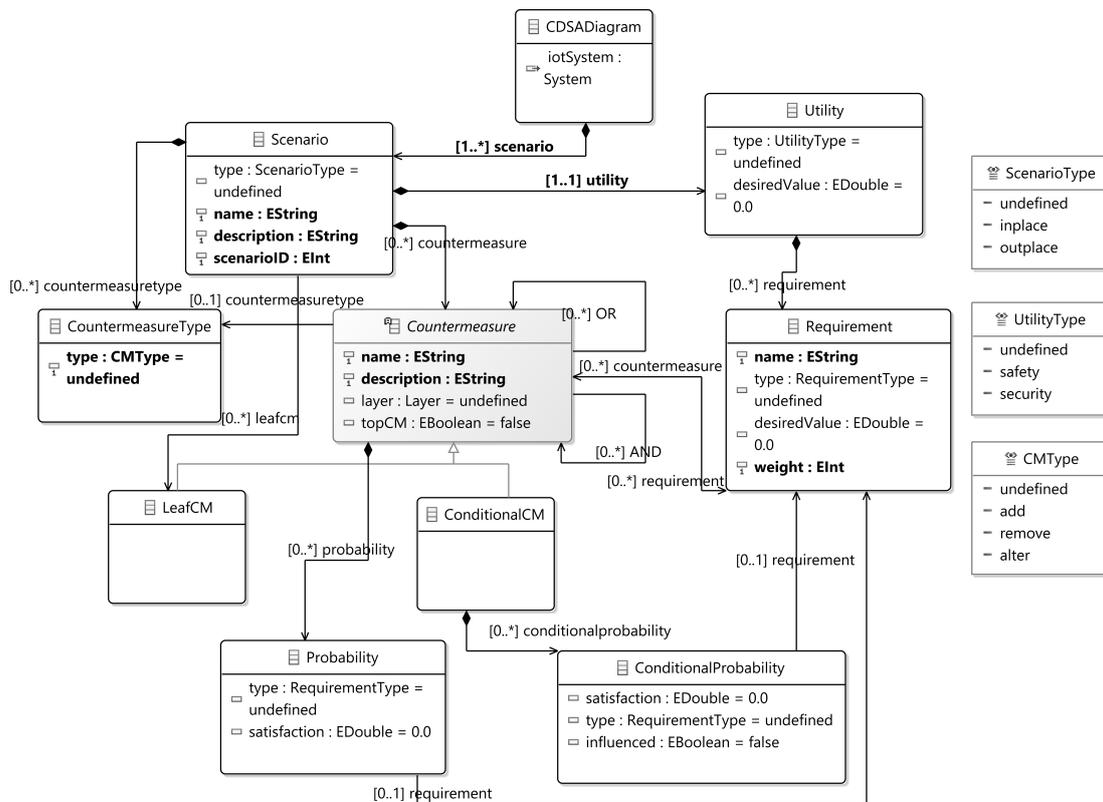


Figure 7.10.: CDSA meta model

To perform a CDSA, the following steps with their rules are required.

Specification

Before modeling, the current design scenario must be defined and the utility node must be specified. There is a choice between safety and security as the target point of the analysis. In addition, it must be specified whether a design transformation in the form of an out-place or in-place change is to be performed. Out-place checks countermeasures that affect the entire model, while in-place only considers countermeasures that affect individual elements or areas.

Modeling

The defined scenario and its utility can then be modeled by using decision and utility nodes. The requirement chance nodes are modeled as gray ovals at the top of the model. They are anchored in the utility node. They represent the functional or non-functional requirements that must be maintained or improved in order to reach the target value of the utility, and thus, prove the usefulness of the countermeasures scenario for flaw prevention. In the center of the diagram, the countermeasures are modeled in rectangles, color-coded depending on the layer in which the countermeasure would be used. On the lowest level are the leaf countermeasures that do not require any pre-steps. All dependent countermeasures are modeled on top of this, and thus, have, in addition to their own probability, a CP per each requirement they affect. The connections are either of type **AND** or **OR**, depending on whether all or only one of the ancestor countermeasures must be used. The top countermeasures are directly linked to their associated requirements. In addition, the countermeasure type is marked as a diamond.

Quantification

All countermeasures have independent probabilities regarding their successful influence on requirements, and thus, successful improvement of the design. This value must be defined with a satisfaction value. In addition, they have CPs, depending on the number of requirements they influence. Leaf nodes have $P=CP$, while non-leaf nodes have calculated CPs. These are calculated by conditional definitions of the edges:

- **AND** CP(A and B): $CP(A)*CP(B) * P(\text{Overlying Countermeasure})$
- **OR** CP(A or B): $(CP(A)+CP(B) - CP(A)*CP(B)) * P(\text{Overlying Countermeasure})$

The conditional definitions can be defined differently depending on expert opinion or use case specifications. In addition, the requirements must be weighted, since not all of them are equally important for the utility to be achieved.

Calculation

For this purpose, bottom-up and top-down approaches are presented. On the one hand, the countermeasures can be used to calculate the probability of compliance with or improvement of the utility or requirements. On the other hand, Bayes' theorem can be used to determine required performance of a countermeasure in order to achieve predefined target values. The first type of calculation determines the utility by the JP per requirement, described as in FIA, and the multiplication of the weights to obtain the total score of the present countermeasure scenario. Calculation types two and three assume a fixed utility that must be achieved and use the Bayes' theorem as described. Thus, on the one hand, the effect of a certain countermeasure on the target utility can be determined and hence the

added value of the countermeasure can be evaluated. On the other hand, it is possible to calculate how CPs of individual countermeasures behave in case an ancestor or descendant countermeasure has a changed probability of success.

- **JP of requirement X** = $\prod_{a \in \text{Ancestor}}^{Descendant} CP_a(X)$
- **Utility** = $\sum_{r \in \text{Requirement}} JP_r * weight_r$

Measurement

The result type of this analysis is a quantifiable decision for a design. The final results, depending on the calculation types, are compared with other modeled and calculated scenarios. Scenarios can contain only a few different countermeasures, have an entirely different set of countermeasures, or differ only in their probability values due to best and worst case estimations. The scenario with the best utility or the fewest countermeasures to achieve the utility target value is selected.

Output

The output shall indicate which countermeasure scenario is the best suited to prevent the attack or accident. This will decide the model needs to be changed and based on this decision, the new services for the subsequent SIA can be determined.

In Figure 7.11, extracted from ArchiAna results, the AAL example is continued after the possible flaw of a missing field gateway and its impacts has been investigated and assessed. For the present countermeasure scenario, an in-place transformation is modeled. Since the start pattern of the analysis cycle is a security pattern, the utility Security is selected here. For requirements, functionality guaranteed, extended privileges and manipulation resistance are selected to determine whether the defined countermeasures have an added value in terms of security. As a leaf countermeasure, it is proposed to remove the old sensor in the things layer and replace it with another sensor, since the currently planned sensor is considered too vulnerable according to FIA results. Based on this, a backup control will be set up in case of a failure. For the physical communication layer and middleware layer, a new Azure field gateway and an improved authentication process are planned. Based on this, further countermeasures are planned, such as the installation of a PUF and digital signatures to reduce the probability of tampering in the event of an attack on the server. A detailed employee briefing is planned in parallel. Each leaf node has defined a probability per requirement. These are not visible in the model, but are stored in the background data. For example, install azure field gateway affects the first two security requirements. For manipulation resistance this countermeasure has a successful impact of 99%. To calculate the CP of integrate PUF the conditional definition for an **AND** relationship must be applied. With a probability of 95% of improved authentication process and 93% as independent probability for

integrated PUF the calculation is: $0,99 \cdot 0,95 \cdot 0,93 = 0,87$ as CP for integrate PUF that all three countermeasures are successful and have a positive effect on the requirement. To determine the JP of the requirement, all CPs of a path must be multiplied, e.g. $0,99 \cdot 0,95 \cdot 0,87 \cdot 0,82 = 0,67$ as JP for manipulation resistance. For the utility calculation, these are again calculated with the weights: $0,97 \cdot 3 + 0,67 \cdot 5 + 0,7 \cdot 1 = 6,96$ utility. Results are prepared and in ArchiAna already compared.

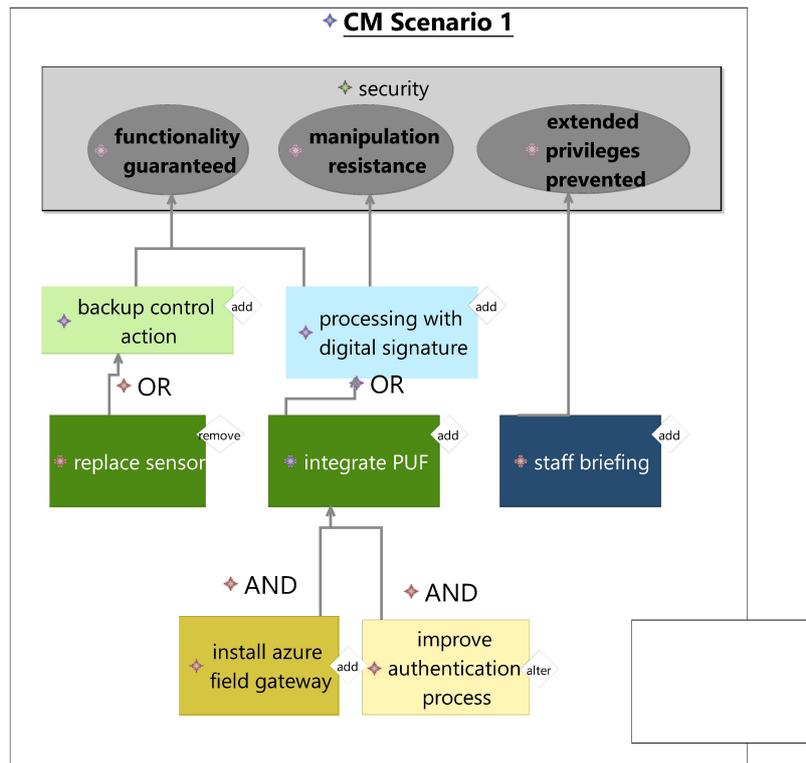


Figure 7.11.: CDSA example

Countermeasure	P (manipul. res.)	CP (manipul. res.)
Install azure field gateway	0,99	0,99
Improve authentication process	0,95	0,95
Integrate PUF	0,93	0,87
Processing with digital signature	0,94	0,82
Requirement	JP	Weight
Functionality guaranteed	0,97	3
Manipulation resistance	0,67	5
Extended privileges prevented	0,7	1

Table 7.3.: Probability values and weights of AAL example - CDSA

7.6. Service Interoperability Analysis

The last analysis in the cycle is based on the basic idea of service interoperability in complex systems by [ULJ08]. This approach divides service interoperability into run time-based and design time-based interoperability and works with ontologies. Thus, only the definitions for service interoperability and service quality can be adopted. It is divided into quality of single services, pairwise comparison of service interoperability and a power set of services that maps the overall quality of services to the system. Therefore, assessment criteria availability (A), correctness (C), verifiability (V) and communication compatibility (CC) are used. However, the approach only uses EID to break down factors to influence interoperability, but not to calculate them. Since in IoT system services are in the center next to things, an approach has to be created that checks the specific new and old services with EID. Consequently, the calculation of these quality types cannot be transferred.

SIA aims to compare an old flawed As-Is scenario with previous services with a new To-Be scenario with new services. This allows to assess whether the newly planned countermeasures would introduce new critical services, and thus, create new problems or prevent a flawless interaction of current services. This is achieved by considering single, pairwise or whole services and evaluating their quality in terms of interoperability, where services are described with WSDL.

Input

The new or modified services needed for SIA are obtained through the planned results of a CDSA. Depending on the decided design changes, new services or needed service changes can be modeled. The design decision also determines which services must be compared in pairs. Required probabilities for calculating the service impact on assessment criteria can be transferred from the CDSA if the CDSA selected requirements match.

Conditions

To perform a SIA, the entire analysis cycle must have been run to this point and flaw mitigation solutions must have been worked out, since this analysis does not propose design changes, but only checks compatibility of planned changes.

Construct

A *SIADiagram*, like the other analyses, is connected to the associated IoT system to be able to transfer the previous services. Multiple scenarios can be modeled in a diagram, with a *Utility* associated with each scenario. The assessment criteria are defined as *Chance* nodes and are subordinate to the utility. Each of these nodes can be assigned a weight depending on the degree of influence on the utility. Each scenario has *ServiceTypes* and the corresponding *SIA Services*. The types are based on the IoT(-WD) meta model. Each service has a *InteroperabilityInterface* for

Comparisons with a rating scale. In addition, each service and service type has the possibility to store probability values to measure the impact on the assessment criteria.

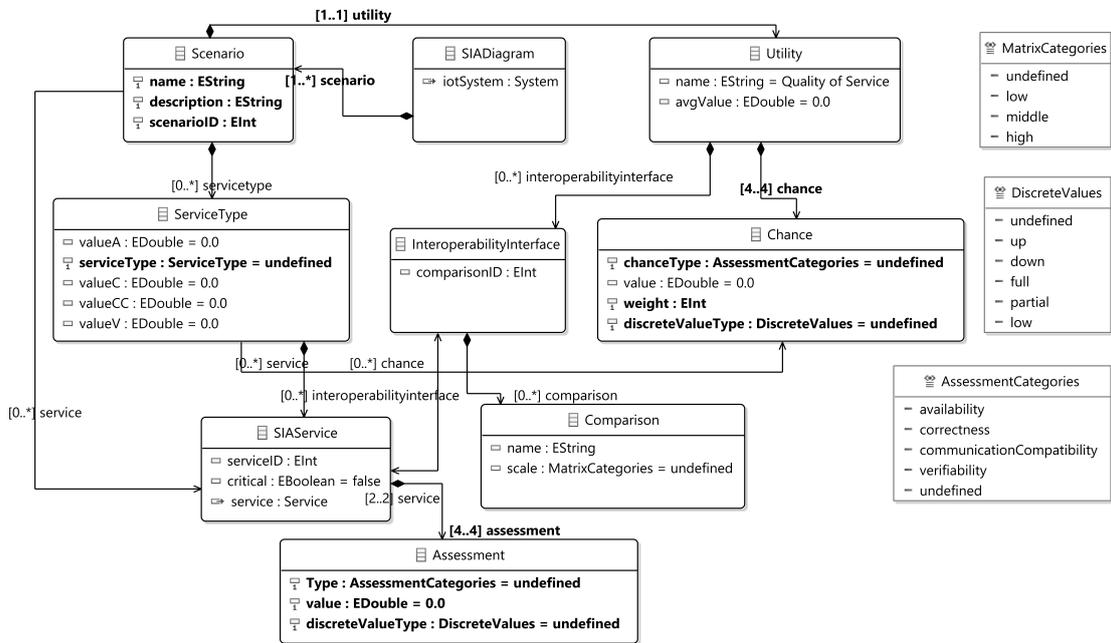


Figure 7.12.: SIA meta model

To perform a SIA, the following steps with their rules are required.

Modeling

At least two scenarios must be modeled per SIA diagram to enable comparison. First, the utility is modeled for each scenario, which represents the quality of service as a power set, and thus, the entirety of the services quality. Below this, the four defined assessment criteria are modeled as ovals. These have connections to all service type containers and services since all services must necessarily consider all criteria. The services are further defined under the service types as rhombus. If there are previous services that need to be retained or adapted, they can be linked to the IoT model service. For each two services, a comparative interface is modeled with which the pairwise interoperability can be determined with a color-coded scale.

Specification and Quantification

According to [ULJ08] services can be considered as independent blocks in this case. Accordingly, untypical for EID no CPTs but PTs must be defined. Therefore, a probability table with discrete values is specified for each service per assessment criteria or chance node. These values are defined to express the probability

with which the services will fulfill the criteria. Past values can be used or assumptions can be made based on CDSA requirements. In flawed As-Is scenarios, these values are typically low. In addition, weights for chance nodes concerning utility impact must be set. Furthermore, the matrices for the pairwise comparison must be specified in order to evaluate whether services can work together currently and in the future. These can be set up with different characteristics like 'Same Protocols', 'Invoked Operations', 'Provided Operations' or 'Service Bus Compatible'. The scale ranges from low to high in a compatibility rating.

Calculation

To calculate the quality of each service type, the average value of the probabilities of the corresponding services is taken for each quality attribute. Thus, each service type has a probability of fulfillment for availability, correctness, verifiability and communication compatibility, depending on its associated services and their quality. These calculated values are used in the next step to calculate the probabilities for the chance nodes to determine the impact of all services of the current scenario on the respective criterion. Finally, the utility, or in this case also called power set of service quality, can be calculated to determine the suitability of the current service scenario to implement the planned countermeasure scenario at a high quality level and to prevent new service-related problems from occurring. Since independent probability is assumed, the following applies:

- $\emptyset P(X)$ of Service Type = $(\sum_{s \in \text{Service}} P(s)) / \text{number of services}$
- JP of chance node X = $\prod_{a \in \text{Ancestor}}^{\text{Descendant}} \emptyset P_a(X)$
- Utility = $\sum_{c \in \text{ChanceNode}} JP_c * \text{weight}_c$

Measurement

Even though some calculations are made in this analysis, the main result type is qualitative to select a suitable design. However, the calculations can be used as an assistant metric to perform the scenario comparisons. Accordingly, service qualities of the newly planned services are compared with the old As-Is scenario. As a reference value, the values in the new To-Be scenario must not be worse. Otherwise, the countermeasure would eliminate some problems but create new interoperability problems. Safety and security are equally covered and measured by the selected assessment criteria. The quality of single services is measured with the independent probabilities of the criteria. Here, no single service should fall below a certain probability, otherwise it is considered a critical service. The quality of the interoperability of two services is measured by comparing the defined characteristics in their matrices. They are color-coded depending on the degree of interoperability. If they match perfectly they are marked green, possible difficulties with orange and if they do not match at all they are marked red. This estimation can be determined by software specifications or experts. The final utility, and thus, the power set of service quality is calculated as described

above and compared with the old utility. In addition, a layered measurement is possible by considering only services of one layer and including them in the calculations.

Output

The final result of the design decision support is a defined new countermeasure To-Be design, which include required services, that is best suited to avoid or mitigate an attack or accident. Thus, the design part found in the flaw identification can be revised.

The AAL example passes the final analysis cycle step by usage of the SIA features of ArchiAna (figure 7.13). The completed execution of a SIA, based on the previous CDSA results, can be seen. This is a minimal example and not a complete representation of all services to be reviewed.

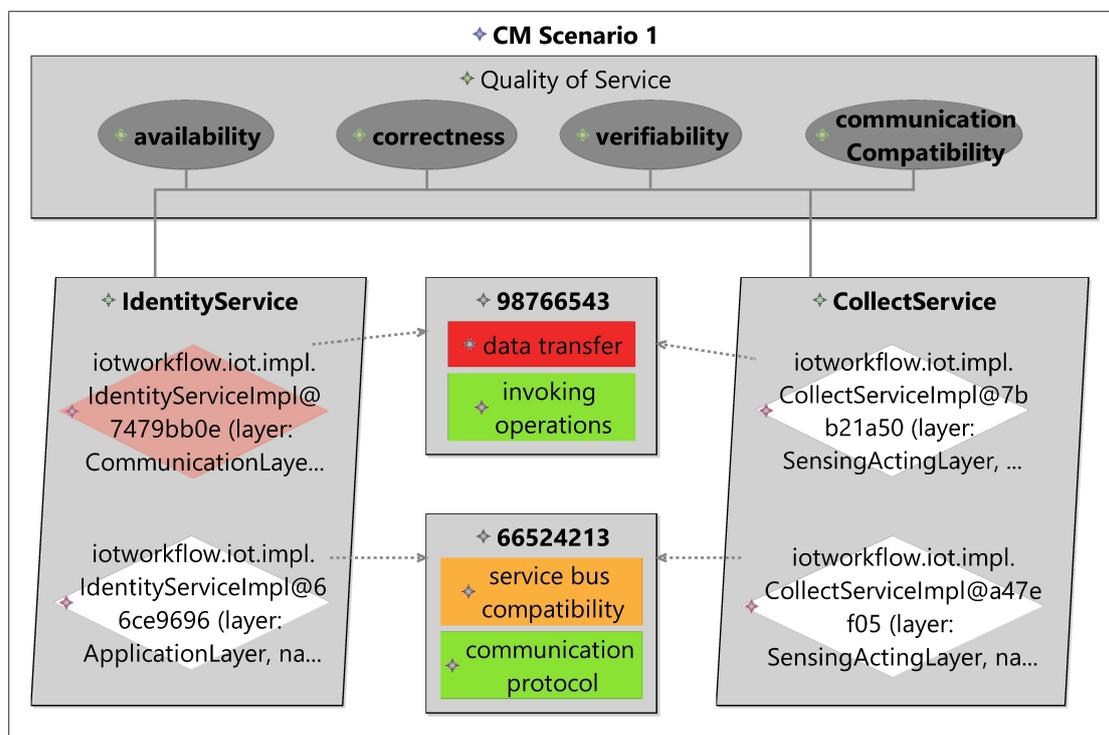


Figure 7.13.: SIA example

Accordingly, four services, divided into IdentityService and CollectService service types, are presented and analyzed in a planned To-Be scenario. These are service communications between the AAL_Hub and the new scheduled field gateway respectively AAL_Hub and the fitness_tracker. Accordingly, there are two new and two old services. The upper service for identity check is new planned due to countermeasures improve authentication process and install azure

field gateway of CDSA results. This ensures that secure and only verified access to the new field gateway is allowed. Connected to this is the upper collect service to enable the field gateway to retrieve data from the AAL_Hub. The lower services are responsible for identification and data collection between tracker and hub and have been slightly adapted due to the countermeasure backup control action. Thus, the collect service is called more often. After modeling, the services are checked. By marking them red, problems with the new services could be recognized, as the pairwise comparison found no matching data transfer methodology. In addition, the new identity service has been marked as critical as its communication compatibility, one of its assessment criteria, undercuts the limit. These are assessments that can be performed independently of the old As-Is scenario. For the comparison with the old design, chance nodes and utility are calculated and values are compared. As described above, the values of the service types are calculated based on the probabilities of the services. For example for IdentityService type for Availability: $(0,99*0,95)/2=0,97$. To obtain the JP for chance node Availability, the service types values are used, e.g.: $0,97*0,93=0,9$. This is to be interpreted as assessment criterion Availability is fulfilled to 90% in the entirety of all services. The utility is calculated in the same way as in CDSA using predefined weights. The utility and chance node values must afterwards be compared with the original scenario to ascertain whether the new To-Be has improved values. However, due to the critical service, a too low value of the criterion Communication Compatibility is to be expected.

Probabilities of assessment criteria	A	C	V	CC
Identity Service 1	0,99	0,95	0,9	0,6
Identity Service 2	0,95	0,87	0,93	0,98
IdentityService Type	0,97	0,91	0,92	0,79
Collect Service 1	0,93	0,88	0,87	0,95
Collect Service 2	0,94	0,91	0,82	0,93
CollectService Type	0,93	0,9	0,85	0,94

Table 7.4.: Probability values of AAL example - SIA

7.7. Related Work

Related work in this chapter is divided into related analysis categorization methods, IoT analyses and other architectural assessment approaches to point out differences to the used categorization and analysis cycle of this dissertation:

EAM Categorization Approaches

There are several other categorization attempts in the literature to group EAM analyses. On the one hand, a distinction can be made between meta model dependent and meta model independent approaches, such as [LSB14] or [Buc+11]. For this work, only meta model dependent approaches are of interest. In addition to this, EA analyses can be distinguished regarding to the point of execution time. Therefore, the analyses are sorted in ex-ante and ex-post to determine whether an analysis is conducted before or after the adoption of an architecture. It is also possible to separate the analyses according to their execution technique: expert-based, rule-based or indicator-based [BMS09]. However, both classifications are not detailed enough to identify characteristics and most of the analyses can't be strictly classified within these divisions. [Lan05] conducted an initial categorization with four dimensions: Quantitative and functional differ at the input and output data. The functional dimension can be further distinguished in static and dynamic. However, this division is not detailed enough to identify the explicit requirements of classified analyses and four categories is a rough classification. Regarding the varieties of containing approaches, the existing categorizations are not sufficient. Accordingly, the existing categorization approaches have not been suitable for this dissertation.

Analysis Approaches in IoT

Even if the focus of IoT analyses is on software related analyses, there are already some approaches trying to perform architectural methods in IoT.

The IIoT project presents its security-related architecture and corresponding methods for assessment. Among others, OWASP and STRIDE are also used for categorization. A security framework is presented that consists of several components. The security model and policy are the basic building blocks. Building on this are endpoint, connectivity and communication protection, security monitoring and analysis, and security configuration. The security monitoring and analysis component is particularly noteworthy. System and traffic information is collected and possible security violations are identified. Behavioral or rule-based analyses can afterwards be performed. These are based on defined security design principles. These are used to support security monitoring in three phases: before incident, during incident and after incident. The aim is to identify problems at an early stage and return to the starting point as quickly as possible in the event of an attack. In parallel, an assessment process runs that rates the current security posture of the system aiming to identify, prioritize and evaluate problems. However,

this approach is strongly geared to industrial use and only considers safety aspects as side-effect. Furthermore, the framework is to be understood as a set of guidelines and does not provide fully defined automated processes for assessment. The main difference, however, is the fact that the analysis measures are design-based, but are only used at run time. [Con+16]

An approach verifying IoT system models from [Oga+17] presents a tool that includes a UML editor for modeling and a verifier for checking the model. Each element is assigned different behaviors. These are defined with rule clauses. This allows their tool to analyze automatically according to these rules and to check the behavior of the modeled IoT system in the design phase. Thus, this is an approach to assess the entire system and its consistency and to adjust it if necessary. No flaw identification is performed, and therefore, no specific areas of the system are considered. Furthermore, their approach does not provide any safety or security specific rules or any further assessment possibilities concerning other aspects.

Architectural Assessment Approaches

In addition to IoT specialized analyses, generic assessment approaches can also be applied.

The FMEA is a widely used and established technique and is applied in different domains like, e.g. automotive, avionics and railway industry. Even the medical area uses FMEA to analyze medical devices and effects in case of failures [Hec+15]. It is purpose of the FMEA to mitigate risks as much as possible. This is done by detecting and preventing failures. For prevention, it is required to indicate and to prevent failures in early phases in the product cycle. The failure detection has four essential goals:

1. Possible fault sources which can cause failures must be detected
2. Causes or consequences must be identified, prevented or avoided
3. Faultless processes during the development cycle must be performed
4. Vulnerabilities must be identified in order that a revision can be applied

To prevent and detect failures it is required to investigate potential risks by means of the FMEA whereas occurrence complies with the probability whether a hazard occurs. Severity corresponds to the severity of hazard. The detection complies with the probability that a hazard is detected. Each of the three factors can range between 1 and 10, i.e. the RPN can range between 1 and 1000. In general, the lower the RPN the better the potential risk. Depending on the RPN, the degree of risk and the necessity of CMs can be identified by means of Table 7.5. [Ber+09]

RPN	Risk of Error	Counter Measure
RPN = 1	none	no CMs required
$2 \leq \text{RPN} \leq 50$	acceptable	additional warning required
$50 < \text{RPN} \leq 250$	medium	additional protective CMs required
$250 < \text{RPN} \leq 1000$	high	constructive CMs absolutely required

Table 7.5.: Interpretation of the RPN [Bun17]

However, even though FMEA is already a widely used approach, it does not allow for the inclusion of IoT specific elements. In addition, FMEA tries to do identification and assessment in one step and is only a guideline and does not provide specific analysis steps. It is, therefore, a very rough cycle that does not allow any subdivision and focusing of the individual assessment steps, and is therefore, unsuitable for the approach presented here.

Another related architectural analysis approach is the use of attack trees for security concerns or fault trees for safety issues. These are two modeling techniques for the representation of events and their evaluation. The fault tree is based on DAG properties and represents events that could trigger an accident. In addition, it is noted how often an event is likely to occur. The individual nodes and edges are connected with boolean gates. The events that are to be prevented are modeled with a top event and the causing basic events. This allows various assessment calculations to be made as to how likely the top event is to occur. However, fault trees do not store CPTs per node. Thus, no BBN questions can be covered and conditional events cannot be considered. Attack trees are very similar in design, but cover security critical events, with the root node representing the exploit, and the nodes below being driven by an attacker to reach the target. A distinction is made between leaf nodes for basic actions by attackers and intermediate nodes that can lead to intermediate targets of the exploit by performing or combining leaf actions. In addition, there are approaches for an attack-defense tree. Here, possible countermeasures that could prevent these actions are defined directly under the attacker actions. However, this approach is not suitable for the present dissertation, since the exploit is known in attack trees, whereas for the IoT assessment cycle only a vulnerability is known and it must first be determined in which manner the attacker can exploit this. [NFW17]

Another well-known method for assessing vulnerabilities or flaws is the Common Vulnerability Scoring System. Each vulnerability is assigned a numerical value between 0 and 10. Each assessment can have several assessment attributes with one value. The different attributes try to cover different assessment aspects. Therefore, attributes can be divided into base metrics, temporal metrics and environmental metrics. The values are summed up and the vulnerabilities are prioritized according to the severity or urgency of the prevention. Most architectural approaches start with the consideration of threats. However, this approach also focuses on vulnerabilities and the origin of threat opportunities. Nevertheless,

the quality of the scoring is strongly dependent on which attributes are rated, as most of the attributes are optional. Therefore, while it provides a structured approach, it does not control this structure. Thus, it cannot be guaranteed that the same aspects are always checked and that the results of the analysis are always comparable. As the results, therefore, always differ, this procedure cannot be reliably integrated into an analysis cycle. [MSR+07]

After reviewing the related literature approaches, no similar approach could be identified that simultaneously considers safety and security, provides a structured semi-automated analysis cycle with individual focus, or transfers IoT specific elements into existing architectural approaches.

Part IV.

CASE STUDIES AND
EVALUATION

8

Evaluation

After all parts of the concept have been presented, and thus, the *Objectives* addressed, various methods need to be applied to verify the applicability and correctness of the approach. For this purpose, this chapter is structured as follows: A related work-based evaluation considers related approaches and points out whether the design approach of this dissertation covers and extends the related approaches. For a case study-based evaluation, two use cases are presented and different aspects of the approach are tested. As the last part of the evaluation, several scenarios are defined and discussed for the design approach and the architecture optimization. This three-part evaluation is intended to allow different aspects of the *Contributions* to be covered.

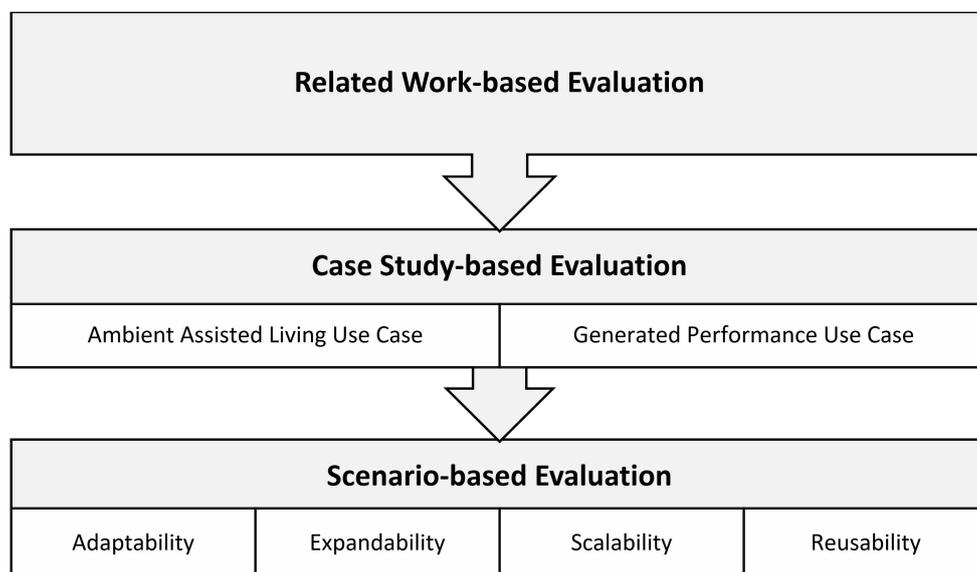


Figure 8.1.: Evaluation concept

8.1. Related Work-based Evaluation

The first part of the evaluation examines whether the presented IoT-WD meta model, which is the basic building block for the entire approach, includes all necessary components. For this purpose, related work approaches are used to check whether their contained components can be covered by the meta model of this dissertation. Since the IoT-WD meta model is intended to be a generic meta model, and thus, is use case independent, it should be proven that it includes all components of related work, and therefore, makes the other approaches obsolete or transferable. In addition, it should be determined which areas the IoT-WD meta model contains that related work approaches do not consider.

In the following, seven related work approaches are considered one after the other and compared with the IoT-WD meta model. The approaches used are those included in chapter 4 or presented as related approaches. Approaches that were strongly use case focused were excluded as no comparison is possible. For each comparison, the elements or areas of the related approach are listed in a table and evaluated with the categories *Match*, *Coverage* or *No Match/Coverage*. *Match* means that the element occurs identically in both approaches. *Coverage* implies that the IoT-WD meta model contains components that are able to cover the element from the related work approach. In the case of *No Match/Coverage*, the IoT-WD meta model does not offer any fitting component.

Microsoft Azure [Mic18a] ↔ IoT-WD Meta Model

The first comparison (table 8.1) is between Azure's reference architecture and the IoT-WD meta model.

	Match	Coverage	No M/C
IoT Device	X		
IoT Edge Device		X	
Cloud Gateway		X	
UI & Reporting Tool		X	
Stream Processing	X		
Storage	X		
Warm Path Store		X	
Cold Path Store		X	
Business Integration		X	
User Management	X		
IoT Device Application	X		
Networking		X	
Machine Learning		X	

Table 8.1.: Comparison of approach with related work of [Mic18a]

The first column lists the areas that Azure considers as basic building blocks of IoT systems. It should be noted that they do not present a classical meta model but only broad areas to be included. As can be seen, all areas could be found in the IoT-WD meta model, either completely identical or covered by other elements. In particular, IoT devices, their applications, stream processing, its storage and user management have a clear match in the IoT-WD meta model. The other areas have no obvious counterpart in the meta model, but can be modeled by other elements. For example, a reporting tool can be represented by an IoT application, warm and cold path storages can be defined as storage types, networking is covered by network/communication types and components, whereas machine learning components can be modeled with different elements of the analytics layer. Business integration is covered by the ability of the IoT-WD meta model to model business systems.

Accordingly, the approach of [Mic18a] is considered to be fully covered, and thus, use cases based on Azure can use the approach of this dissertation.

IoT-A Project [Bau+13] \leftrightarrow IoT-WD Meta Model

The next comparison considers the meta model of IoT-A, illustrated in table 8.2. Their meta model contains some basic elements that are part of every IoT model. For example, services, physical and virtual entities, and devices with their sub-components such as sensors or actuators. These elements are identical in the IoT-WD meta model.

	Match	Coverage	No M/C
Service	X		
User	X		
Digital Artifact		X	
Augmented Entity			X
Virtual Entity	X		
Physical Entity	X		
Device	X		
Actuator	X		
Tag	X		
Sensor	X		
Network Resource		X	
On-Device Resource		X	

Table 8.2.: Comparison of approach with related work of [Bau+13]

Three elements could not find a direct match. In IoT-A, a digital artifact is divided into an active and a passive digital artifact. Active digital artifacts can be covered as digital users in the IoT-WD meta model. Passive digital artifacts, on the other hand, are only a subspecies of virtual entities, which in turn have a

match. The network and on-device resources are software components in IoT-A. Accordingly, network resources in the IoT-WD meta model can be modeled as the various IoT services responsible for the various activities in the network. The on-device resources are covered with different attributes of the IoT devices. In this comparison, there is also an element that finds neither a match nor a coverage in IoT-WD meta model. Augmented entities represent the connection of physical entities and their digital twins. Since this has already been solved by links in the IoT-WD meta model, this element can be omitted.

Thus, this comparison can also be considered as complete satisfaction by the IoT-WD meta model.

IoT-RA [ISO16b] ↔ IoT-WD Meta Model

The meta model of the IoT-RA standard is compared in table 8.3. It is a frequently used or cited meta model, and is therefore, very generically oriented. It contains only basic building blocks of an IoT system and does not contain any attributes or other details that could be used for later analysis. Since only basic elements can be found here, such as IoT gateways, devices or users, this is a complete match. Each element of the IoT-RA approach can also be found identically in the IoT-WD meta model.

	Match	Coverage	No M/C
IoT User	X		
Service	X		
Application	X		
Network	X		
IoT Gateway	X		
Data Store	X		
Virtual Entity	X		
PhysicalEntity	X		
IoT Device	X		
Tag	X		
Sensor	X		
Actuator	X		

Table 8.3.: Comparison of approach with related work of [ISO16b]

Mission-Critical IoT Systems [Cic+17] ↔ IoT-WD Meta Model

Another related meta model focusing on mission-critical IoT systems originates from [Cic+17] and is compared in table 8.4. This meta model also contains some basic IoT components such as connected physical entities or communication protocols. Resources have been grouped together here for clarity, but consist of sensors, actuators and storage options. According to the other comparisons, again several matches are found. The remaining five elements can be covered by the

IoT-WD meta model with other options. A goal is always assigned to a stakeholder, and can therefore, be specified. User interfaces are modeled via IoT applications, smart objects as different IoT devices and computational services are covered via the six different service types of the IoT-WD meta model. Since a description can be assigned to each IoT device, a context can be modeled.

Since there are no unexplained *No Matches/Coverages* here either, the comparison is considered to be satisfied by the IoT-WD meta model.

	Match	Coverage	No M/C
User	X		
Goal		X	
IoT Application	X		
User Interface		X	
Network	X		
Smart Object		X	
Connected Physical Entity	X		
Computational Service		X	
Context		X	
Communication Protocol	X		
Resources (Supercategory)	X		

Table 8.4.: Comparison of approach with related work of [Cic+17]

Wipro Reference Architecture [Dig17] ↔ IoT-WD Meta Model

Table 8.5 represents the comparison of IoT-WD meta model with Wipro's reference architecture. This reference architecture is focused on the use of IoT in enterprises. Accordingly, business-related components are included in addition to the classic IoT elements.

	Match	Coverage	No M/C
IoT Device	X		
IoT Gateway	X		
Device Management		X	
IoT Services	X		
Big Data / BI		X	
BPM Platform			X
Government Application		X	

Table 8.5.: Comparison of approach with related work of [Dig17]

The classic IoT elements are matched by the IoT-WD meta model. Device management is covered by IoT hubs and the identity registry. Big data or BI approaches are covered by components of the analytics layer, while government

applications are generally covered by stakeholders and their applications. The integration of BPM platforms can neither be matched nor covered in the IoT-WD meta model. This is due to the fact that the reference architecture is strongly use case specific and the meta model of this dissertation ends with the connection to business systems. Further consideration or modeling is not necessary due to the technical focus of this approach.

After excluding the non-relevant parts, the reference architecture was fully satisfied by the IoT-WD meta model.

IoT Service Classification [ME15] \leftrightarrow IoT-WD Meta Model

The next comparison relates to an IoT service classification and not to an entire IoT meta model. Accordingly, only the services of the IoT-WD meta model are used as comparative values. The approach of [ME15] divides IoT services into four areas. One of the most important services is the identity-related service, which has a unique match with the IoT-WD services. Three other services are covered by the approach of this dissertation, but are divided and structured differently. The information-aggregation service is divided into a collect and a processing service in the IoT-WD meta model, while the collaborative-aware service can be modeled as a collect and acting service. The clearer division in the IoT-WD meta model is due to the assumption that each action requires many small services to cover the full functional scope of the various IoT operations. For the very generic ubiquitous service, the access and control service can be used in the IoT-WD meta model.

The scope of IoT services is, thus, fully covered by the IoT-WD meta model.

	Match	Coverage	No M/C
Identity-Related Services	X		
Information-Aggregation Services		X	
Collaborative-Aware Services		X	
Ubiquitous Services		X	

Table 8.6.: Comparison of approach with related work of [ME15]

SABSA with ArchiMate [SAB20] \leftrightarrow IoT-WD Meta Model

The last comparison differs from the others. No meta model or reference architecture is considered. Instead, the abstract structure of the IoT-WD meta model is examined. ArchiMate is used for this purpose, since the latest versions are also suitable for modeling IoT models. Thus, it shall be proven that the presented meta model is structurally capable of representing semantical equivalents, independent of its specific elements.

	Match	Coverage	No M/C
Active Structure Elements	X		
External Active Structure Elements		X	
Internal Active Structure Elements	X		
Behaviour Elements	X		
External Behaviour Elements	X		
Internal Behaviour Elements			X
Passive Structure Elements		X	
Structural Relationships	X		
Dependency Relationships		X	
Dynamic Relationships		X	

Table 8.7.: Comparison of approach with related work of [SAB20]

Active structure elements that perform behavior like human or technical actors can be represented with the IoT-WD meta model. However, ArchiMate distinguishes between external and internal active structure elements. Externals, such as interfaces, can only be represented indirectly in the IoT-WD meta model via connections from services to their actors and assigned to the associated right management. Internals, e.g. roles, applications or devices, have a direct match in the IoT-WD meta model. Behavior elements, the units of activities, can be partially represented. External behavior elements represent units of functionality and can be represented in the IoT-WD meta model with services. Internal behavior elements, on the other hand, are intended to represent detailed implementation details. This is not covered by the IoT-WD meta model, as it concentrates on the design phase in which no implementation details are yet available. Passive structure elements are intended to represent components with which behavior elements operate and are covered by the meta model by modeling various data measurements, data records and storage systems. In addition to elements, ArchiMate includes various relationship types. Structural relations, like aggregations or compositions, have a direct match in the IoT-WD meta model. Behavioral relations such as dependency and dynamic relations to represent influences or data flows are modeled indirectly through attributes, like requirement assignments, or component mappings such as connections to data records.

Thus, all proposed elements for a meta model, designed for the design phase, can be modeled.

Percentage Congruence

To conclude the related work-based evaluation, the percentage to which the meta model approach of this dissertation covers the compared related works is considered. In addition, it is reviewed whether a related approach could replace and fully cover the IoT-WD meta model. For this purpose *Match* and *Coverage* are combined and defined as successfully covered. Next, the set of covered elements

is divided by the total set to calculate results. Figure 8.2 shows the total overview of the percentage comparison. Blue bars depict whether the IoT-WD approach covers the related approach. It can be clearly seen that four of seven related approaches are covered 100%. The remaining approaches could only be covered between 85.7% and 91.7%. The exact reasons are already discussed above. In comparison, gray bars show the percentage of related works that cover the IoT-WD meta model. The approaches of [ISO16b] and [Cic+17] are clearly below 50%, which is due to their very limited meta models, and therefore, do not offer all the possibilities of the IoT-WD meta model, such as detailed cloud connections. The approaches of [Mic18a] and [Bau+13] cover over 50%. However, e.g. they do not offer integration of safety/security aspects, right management or measurement artifacts, and thus, still differ significantly. The reference architecture from Wipro only covers 45% of the required elements, as their focus is on the connection to enterprise systems, and therefore, do not include some technical aspects. The IoT services approach has a fairly high match that is due to the fact that only service matches are considered in this comparison. Only the ArchiMate approach offers 100% coverage of the IoT-WD meta model. However, this only refers to the meta model structure.

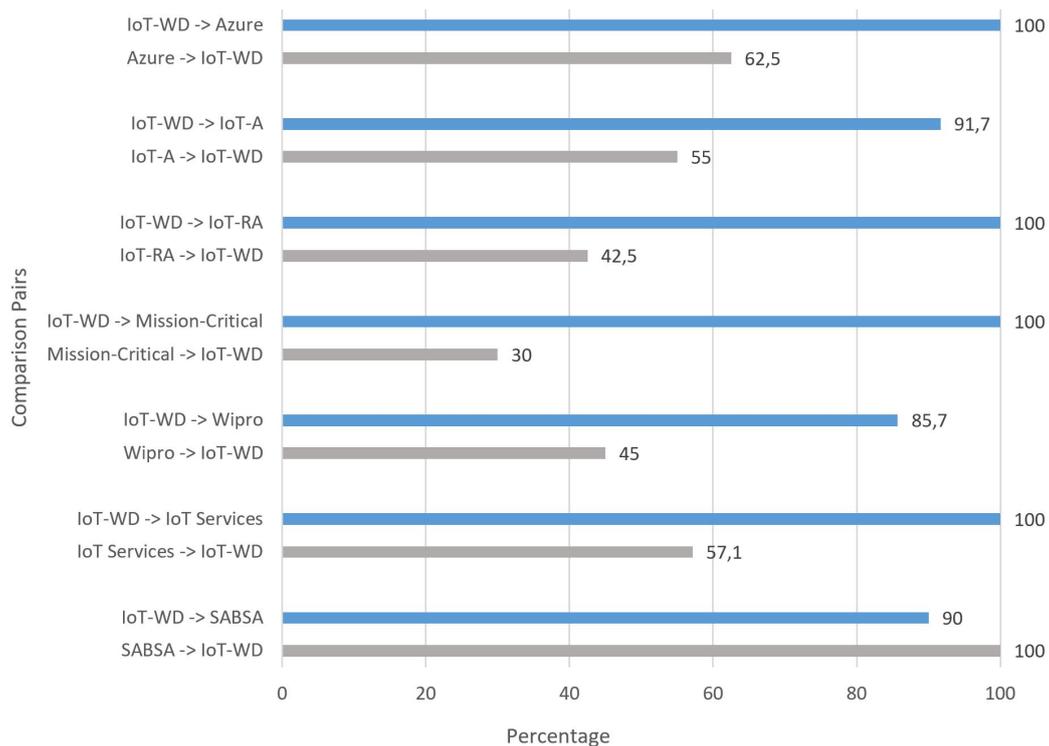


Figure 8.2.: Pairwise percentage comparison of congruence

The comparison clearly shows that the presented IoT-WD meta model covers all required elements or areas and that it could not be replaced by any existing approach. The reason for this is that the meta model is designed generically, but also contains extra components to include safety/security and special wellbeing elements if required, which enables the further assessment steps.

8.2. Case Study-based Evaluation

The case study-based evaluation is divided into two IoT-WD use cases and is intended to prove the applicability of the approach by using the developed Archi-Ana tool. On the one hand, the individual steps and functionalities are applied, and on the other hand, performance tests are executed to prove that the automated flaw identification and assessment approach is faster than a manual investigation, especially in large complex systems.

8.2.1. Ambient Assisted Living Use Case

The first use case is an extension of the AAL running example, and thus, is based on a smart home in which elderly live independently and can be supported and monitored if necessary. In addition, some medical devices are included. Figure 8.3 shows a small illustration of the use case.

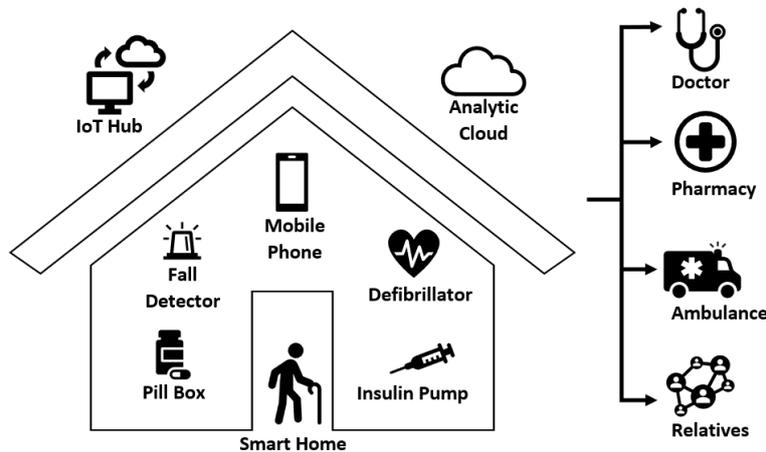


Figure 8.3.: Concept of AAL functions

In the smart home an elderly resident is monitored through multiple devices with different locations and goals throughout the house. For instance, the defibrillator is a fully implanted device to monitor the heart, whereas the insulin pump is a kind of wearable to measure the current insulin level. The other devices are located in diverse places, as the fall detector is positioned in every room and the mobile phone and pillbox vary. To collect and process the data, two kinds of cloud respectively gateway are included. In addition, the smart home is connected with diverse stakeholders with several goals and rights, like ambulance for critical situations or relatives who may be informed about the resident. This use case was inspired by [IoT13], [Pir+18] and [Doh+10]. The network details of this use case were presented in [LRB19].

8.2.1.1. System Model

In order to be able to check the safety and security aspects of the smart home, first the architecture of the system is modeled. Figure 8.4 shows the system model. It is arranged in layers from top to bottom and contains all layers of the layered architecture. The IoT and wellbeing devices described above are modeled in the bottom layer. In addition, other physical components of a smart home are included, as these can also be influenced or have an impact. In addition, the sensors, actuators and other according components were modeled for all IoT things. In the sensing/acting layer, the corresponding metrics, measurements and related services were included. In order to forward the data records, the elements required for communication, such as physical connections, hub or server, are modeled in the communication layer. In the upper layers, all elements for data processing such as events and rules are modeled accordingly. The results can be displayed with the help of the virtual entities and applications. The management of sensitive data and stakeholders or users described above, are modeled in the top layers. In between, technical or functional services or relations are included in the model.

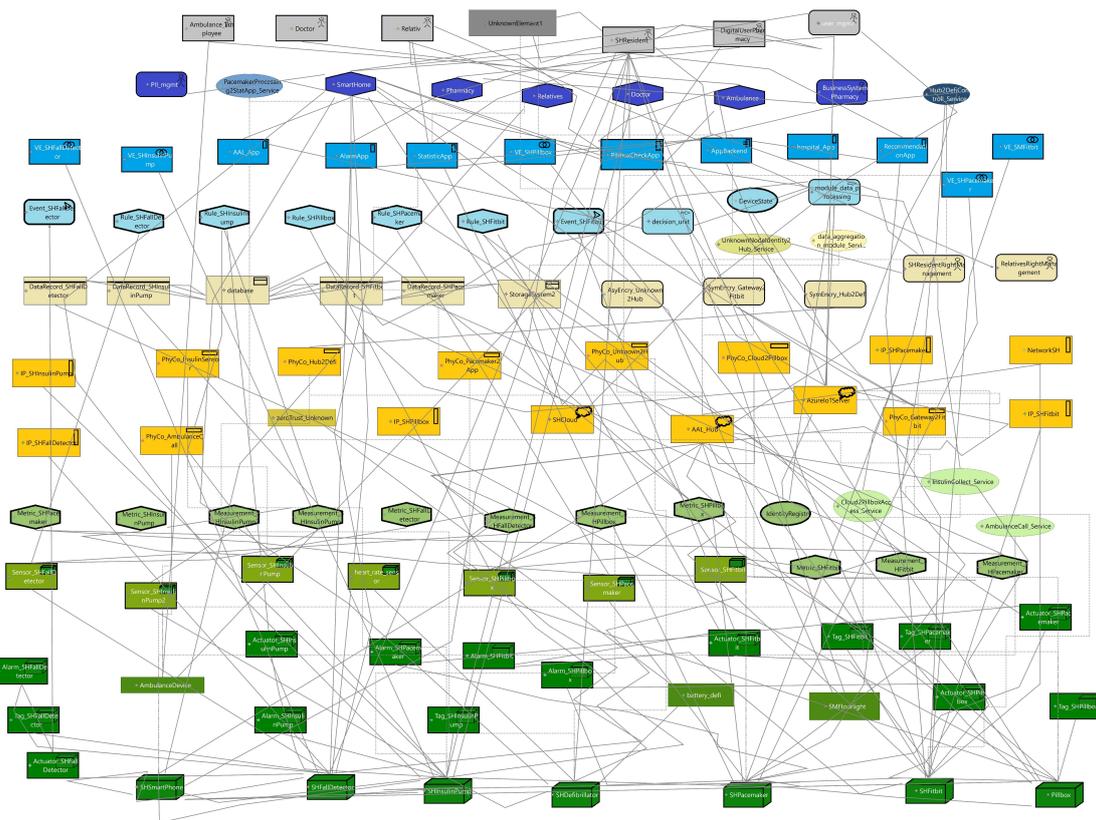


Figure 8.4.: AAL case study system model - IoT diagram overview

In addition, filters can be applied with the editor to focus on specific aspects. In figure 8.5 a filter was applied to show only the elements within the things layer. In addition, the relations were reduced and limited to technical connections. Thus, influences on higher levels are missing in the view. Therefore, this filter can only be used for limited estimations.

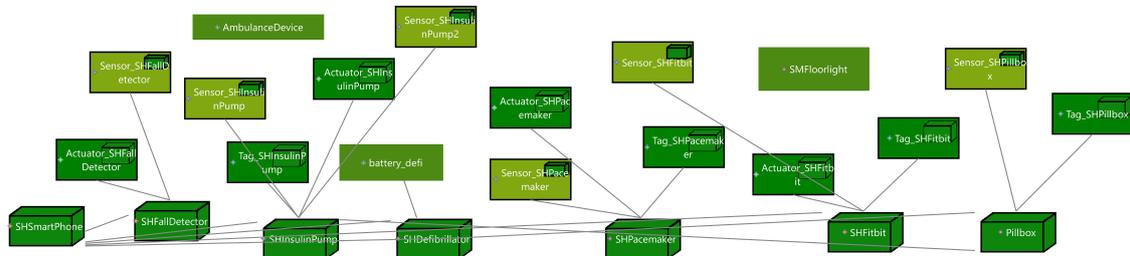


Figure 8.5.: AAL case study system model - things layer filter

In figure 8.6 the system model of the use case is shown in the IoT container diagram provided by the model editor of the ArchiAna tool. This allows a different view and estimation of the architecture. However, it only affects manual estimation and not automated checks. This is discussed in more detail in the second use case. On the right, container element creation functions can be seen in the figure. In addition, the pattern recognition services of the next step that are called from the pattern database can also be seen, as they can be applied on the IoT diagram and the IoT container diagram.

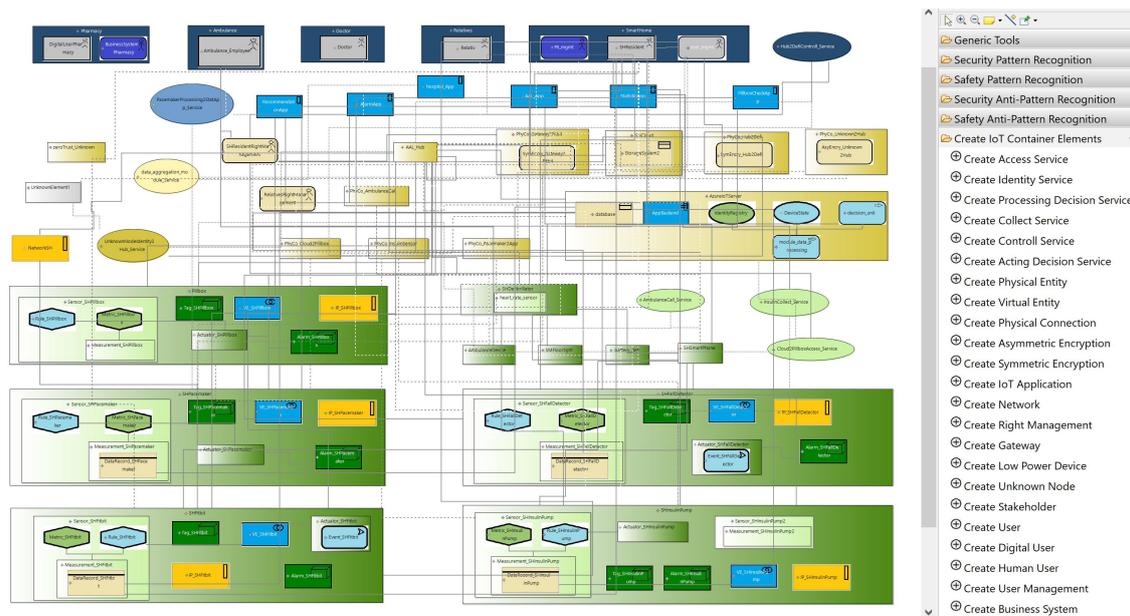


Figure 8.6.: AAL case study system model - IoT container diagram

The container view puts the focus on affiliations and its use of containers. The representation of relations is significantly reduced by this, which positively affects the readability of the smart home model. For example, the SHInsulinPump or the SHFallDetector are two highly complex nodes with subcomponents such as sensor containers with rules, metrics and measurements, whereas the smart SMFloorlight does not need to model any other relevant components in itself. The containers represent not only physical relationships, but also functional or business relationships, for instance, the Pharmacy and its DigitalUserPharmacy and BusinessSystemPharmacy to distribute the drugs. Thus, the AzureIoTServer can be represented as a container with its functions for data storage or processing. Elements that cannot be clearly assigned to components are represented without containers. An example is the AmbulanceCall_Service, which can be assigned to the SHInsulinPump but also to the SHSmartPhone.

8.2.1.2. Pattern and Anti-Pattern Specifications

The use case is checked for safety and security aspects with the help of a defined SAP and a SEAP. The SAP is defined in table 8.8 and refers to smart insulin pumps with special continuous glucose monitoring sensors that measure the blood glucose level of the patient and deliver insulin accordingly. For communication, a ZigBee connection is used to transmit the data measured with a collect service to an external device. The SAP has zero tolerance, and thus, a high degree of urgency. It checks the risk that a measurement is influenced by a failure or external spontaneous problems, and thus, dispenses an insulin quantity that is too high. This can happen, for example, due to wear and tear or faulty environmental conditions. Accordingly, reliability is considered as assessment attribute for this SAP, since an occasional fault can cause reliability to decrease and lead to serious health problems. The most important requirements of this SAP is recovery time and failure resistance to return to the correct measurement as soon as possible. A desirable design is that every insulin pump needs two different sensors which conduct measures at the same time and needs a connection to a mobile phone. Thus, values can be collected several times and manually checked on a device.

Table 8.9 contains the defined SEAP. It checks if a vulnerable design exists for a device with a low trust level with a direct connection to the Internet. This can lead, for example, to unauthorized access to a pillbox, manipulation of the medication, and erroneous or incorrect quantities of medication being ordered. Accordingly, this anti-pattern addresses access services that are performed over Wifi connections, and can therefore, come locally as well as from the cloud. It is a moderately severe anti-pattern as it is temporarily tolerable and a short-term denial of service is bearable. However, it is a critical issue that can occur frequently and that must be prevented from compromising privacy and integrity. Therefore integrity is chosen as a later assessment attribute.

ID	654322
Name	Insulin Pump injects wrong dose
Component	Supercategory: Element to Protect
Element Type	WellbeingIoTDevice (Insulin Pump)
User Group	Patient
Element Category	Leaf of Devices
Hardware	ZigBee Connection
Software	Collect Service
Layer	SensingActing Layer
Location	Local
Disruption Tolerance	Zero tolerant
Fault	Measurement was affected external
Hazard	Insulin dose to high
Classification	Failure, External Problem
Fault Class	Calculation
Assessment Feature	Reliability
Probable Direct Impact	Supercategory: Estimated Consequences
Causal Types	Collider
Causal Relation	Reliability decreases: $(P(B)*(a-2d))^x$
Indirect Impact	Skin friction
Severity	Death
Likelihood of Occurrence	Occasional
Safety Requirements	Recovery, Failure Resistance
Protective Design	Pumps need two sensors and mobile connection
Implementation	Supercategory: Artifact Combination
Filter Conditions	Subcategory: Concerned Elements
Node2Node	WellbeingIoTDevice
NodeAttribute	deviceType:InsulinPump
Pattern Requirements	Subcategory: Recognition Details
Node2Node	Sensor/ Measurement/ IoTDevice
Node2Relation	sensor/measurement/physicalconnection:Pump
NodeAttribute	sensor&×tamp:!equal/deviceType:Mobile
Flaw	sensor1==sensor2 / deviceType != Mobile

Table 8.8.: AAL case study Safety Pattern

ID	372865
Name	Pillbox manipulation
Component	Supercategory: Element to Protect
Element Type	IoT Device
User Group	Patient
Element Category	Leaf of Devices, Intermediate Devices
Hardware	Wifi
Software	Access Service
Layer	Application Layer
Location	Local or Cloud
Disruption Tolerance	Temporary tolerant
Intent	Manipulate pillbox to change medication
Risk	Patient orders wrong or no pills
Classification	Tampering, Denial of service
Attack Goal	Manipulation
Assessment Feature	Integrity
Probable Direct Impact	Supercategory: Estimated Consequences
Causal Types	Fork
Causal Relation	Integrity decreased $(P(B)*((a-d)*x))/a$
Severity	Critical
Likelihood of Occurrence	Frequent
Security Requirements	Integrity, Manipulation Resistance, Privacy
Vulnerable/Hazardous Design	Low trust device has Internet connection
Implementation	Supercategory: Artifact Combination
Element Filter Conditions	Subcategory: Concerned Elements
Node2Node	WellbeingIoTDevice/Service/Cloud
Node2Relation	service/ peereddevice
NodeAttribute	trustLevel:low/service:Access/type:analytic
Anti-Pattern Requirements	Subcategory: Recognition Details
Node2Node	PhysicalConnection
Node2Relation	physicalconnection:cloud
NodeAttribute	protocol:Wifi
Flaw	trustLevel=low / protocol=Wifi

Table 8.9.: AAL case study Security Anti-Pattern

8.2.1.3. Sirius Service Application

After defining the pattern and the anti-pattern, they are transferred into the DSL and automatically converted into executable Java code for the Sirius pattern services. Finally, they can be applied to the system model in the ArchiAna tool.

Figure 8.7 shows on the left a segment of the Xtext file of the defined SAP, while on the right a segment of the SEAP is displayed. The pattern contains the solution or desirable design that should be found so that there is no flaw. Thus, only the insulin pumps are selected from all model elements and additionally only those with the special continuous glucose monitoring sensors are filtered. These are checked for different measurements and mobile connections. The anti-pattern, on the other hand, shows the vulnerable design of the SEAP that must not be present in order to have a flaw-free design. For this purpose, systems are browsed for low trust pillboxes that allow external access services, and thus, their information can be accessed from outside. If an insecure low trust Wifi connection is present, the design, and thus, the model is at risk.

```

49@ ImplementationPatternPart{
50  solution "Every insulin pump needs a backup sensor which conduct
51  measurements at the same time if a prone to failure sensor was
52  used and needs a smart phone to receive error messages."
53@  PatternRule SAP654322_1{
54    algorithmElement "WrongInsulinDose.SAP654322_ElementToProtect"
55    filterCondition FC1{
56      if
57@      WellbeingIoTDevice WellbeingIoTDeviceNameSAP654322{
58        deviceType InsulinPump} }
59    filterCondition FC2{
60      and
61@      WellbeingIoTDevice WellbeingIoTDeviceNameSAP654322{
62        sensor{
63@          Sensor "Sensor1SAP654322"{
64            sensorType Type1}}}}
65    Requirement PR1{
66      and
67@      WellbeingIoTDevice WellbeingIoTDeviceNameSAP654322{
68        sensor{
69@          Sensor "Sensor2SAP654322"{
70            sensorType Type2
71            measurement{
72@              Measurement MeasurementSAP654322{
73                timestamp 123}}}}}}
74    Requirement PR2{
75      and
76@      IoTDevice IoTDeviceNameSAP654322{
77        deviceType Smartphone
78        connectedPhysicaldevice (
79          "WellbeingIoTDeviceNameSAP654322"}}}

```

```

49@ ImplementationSecurityAntiPatternPart{
50  vulnerableDesign "Low trust device has a direct internet connection"
51@  PatternRule SEAP372865_1{
52    algorithmElement "PillboxManipulation.SEAP372865_ElementToProtect"
53    filterCondition FC1{
54      if
55@      IoTDevice IoTDeviceNameSEAP372865{
56        trustLevel low
57        deviceType Pillbox
58        service(
59          "ServiceNameSEAP372865")
60        physicalconnection(
61          "PhysicalConnectionNameSEAP372865") )} }
62    filterCondition FC2{
63      and
64@      Service ServiceNameSEAP372865{
65        type AccessService
66        superelement(
67          "IoTDeviceNameSEAP372865"}}}
68    filterCondition FC3{
69      and
70@      Cloud CloudNameSEAP372865{
71        cloudType analytic
72        physicalconnection(
73          "PhysicalConnectionNameSEAP372865"}}}
74    Requirement APR1{
75      then
76@      PhysicalConnection PhysicalConnectionNameSEAP372865{
77        protocol Wifi
78        element (
79          "IoTDeviceNameSEAP372865", "CloudNameSEAP372865"}}}

```

Figure 8.7.: AAL case study pattern and anti-pattern DSL

With the help of the presented code generation rules from chapter 6 the pattern services in Java are generated with Xtend which can be seen partly in figure 8.8. On the left can be seen the first requirement query of the SAP and corresponds to lines 65-73 of the DSL from figure 8.7. The filtered wellbeing IoT devices are taken from a previously created list and their sensors are checked if they correspond to a specific sensor type, and therefore, differ from the first sensor's type. In addition, the measurement of this second sensor is checked for its timestamp. All insulin pumps that match the pattern are removed and only pumps with a flawed design remain in the list and are passed on to the second requirement query. The same is done for the SEAP on the right side of the picture. However, the SEAP contains

only one requirement query, and therefore, the figure shows the final sorting. The filtered IoT devices are checked for a Wifi connection and hits remain in the list to be marked as flawed. This corresponds to lines 74-49 of the DSL.

```

218 public void requirementCheck1() {
219     List<WellbeingIoTDevice> removeList = new ArrayList<>();
220     //uses current flawed list
221     //updates list if there are elements to protect
222     if(flawedList!=null && (!flawedList.isEmpty())){
223         for (WellbeingIoTDevice element : flawedList) {
224             boolean check = false;
225             for (Sensor sensor : element.getSensor()) {
226                 if(sensor.getSensorType()==
227                     SensorType.SENSOR_TYPE2) {
228                     for (Measurement measurement :
229                         sensor.getMeasurement()) {
230                         //if: dont MATCH
231                         if(measurement.getTimestamp()!=123.0) {
232                             measurement.setFlaw(true);
233                             sensor.setFlaw(true);
234                             check=true;}}}}
235             if(check) {
236                 element.setFlaw(true);}
237             //not concerning element -> remove from list
238             else {
239                 removeList.add(element);} }
240     flawedList.removeAll(removeList);}
241     else {
242         return;}}

```

```

238 public void requirementCheck1() {
239     List<IoTDevice> removeList = new ArrayList<>();
240     //uses current flawed list
241     //updates list if there are elements to protect
242     if(flawedList!=null && (!flawedList.isEmpty())){
243         for (IoTDevice element : flawedList) {
244             boolean check=false;
245             for (PhysicalConnection connection :
246                 element.getPhysicalconnection()) {
247                 //if: does MATCH
248                 if(connection.getProtocol()==
249                     CommunicationProtocolType.WIFI) {
250                     connection.setFlaw(true);
251                     check=true;}}
252             if(check) {
253                 element.setFlaw(true);}
254             //not concerning element -> remove from list
255             else {
256                 removeList.add(element);}
257             flawedList.removeAll(removeList);}
258         else {
259             return;}}

```

Figure 8.8.: AAL case study pattern and anti-pattern Services

After the pattern services have been fully implemented, they can be applied to any model at any time, even if they are extended or modified and need to be rechecked. In Figure 8.9, the SAP with ID 654322 was taken from the pattern database and triggered with the ArchiAna tool. Marked red elements and relations can be seen. These were marked red by the pattern query because they do not correspond to the SAP654322 desired design and represent a dangerous composition of components and relations on the architecture. The flaw results from the not obvious, hidden fact that the SHInsulinPump has not included two different sensor types and measurement timestamps. Thus, the modeled SHInsulinPump in the use case must be altered. The effects of the flaw that need to be considered in the design change are determined in the assessment steps. The pattern recognition result window provides information about which element is the main cause of the flaw and which design should actually be present.

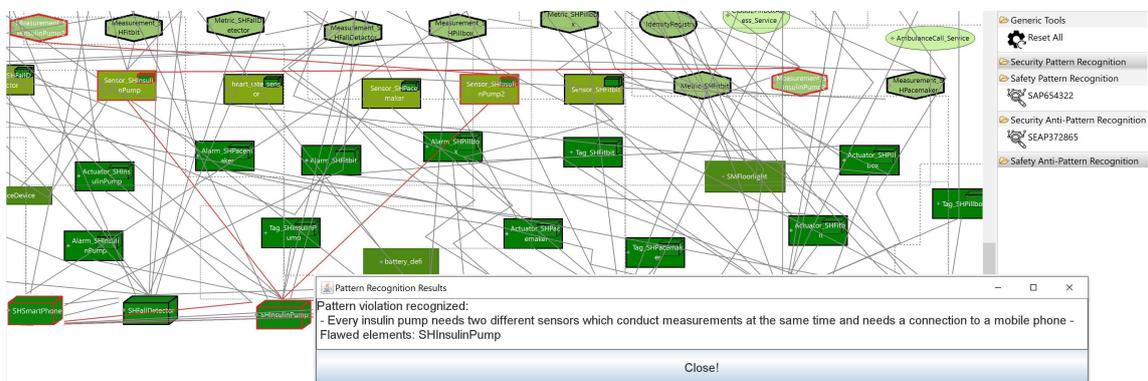


Figure 8.9.: AAL case study flaw identification - SAP

The anti-pattern with ID 372865 is queried in figure 8.10 on the AAL use case. The defined design could be found in the model, which in case of an anti-pattern means the presence of a flaw. The modeled Pillbox, its physical connection PhyCo_Cloud2Pillbox and Cloud2PillboxAccess_Service are marked as flawed because it has a connection to the AzureIoTServer cloud and no security mechanism was planned in between, as the AAL_Hub is only a data collection point and does not provide encryption or access protection. If there were two pillboxes in the use case, both would have been checked. In the current case, however, only one endangered pillbox was found.

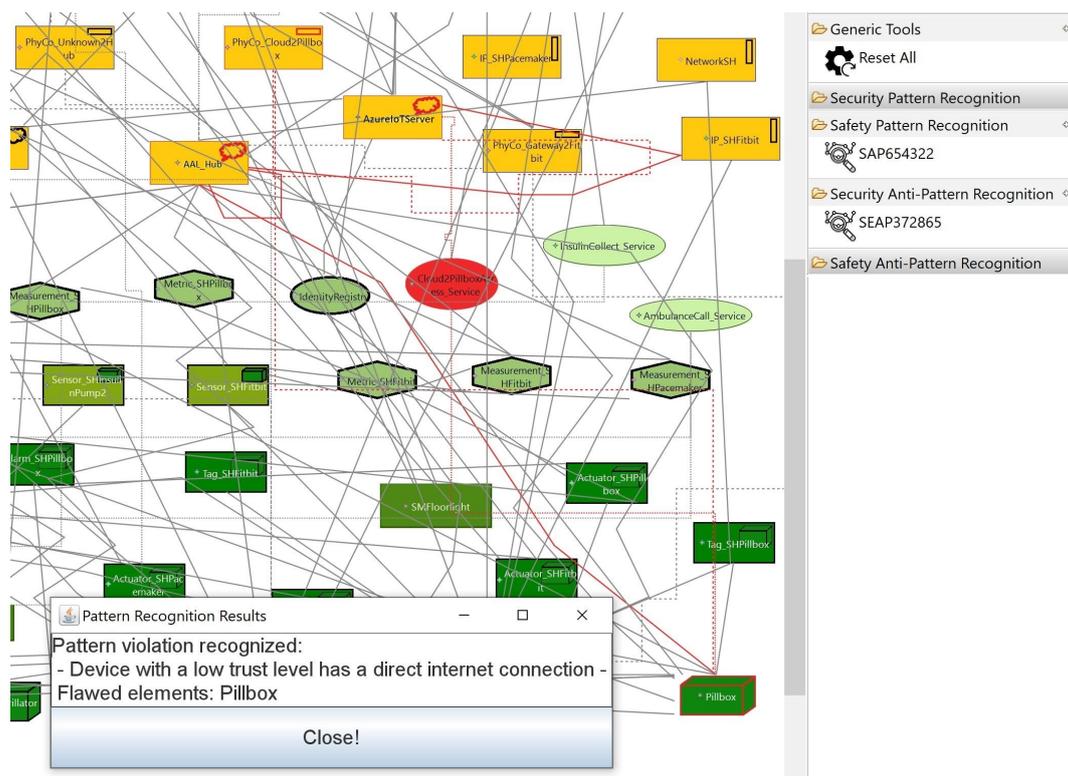


Figure 8.10.: AAL case study flaw identification - SEAP

8.2.1.4. Assessment Cycle Application

After the flaws have been identified, they must be assessed one by one. For the further course of this case study, only the identified SAP violation, and thus, the risk of a flawed insulin pump is further assessed and mitigated. The assessment of the identified SEAP violation would be performed in a compliant manner.

Failure Impact Analysis

An insulin pump has been discovered that does not have two different types of sensors as recommended, and therefore, cannot compare its measurement results. Inconsistencies in the glucose measurement are not detected and incorrect data may be passed on for insulin calculation. This has an impact on the reliability of the system. Accordingly, FIA metric reliability is selected, and thus, a Failure Reliability Impact Analysis is performed.

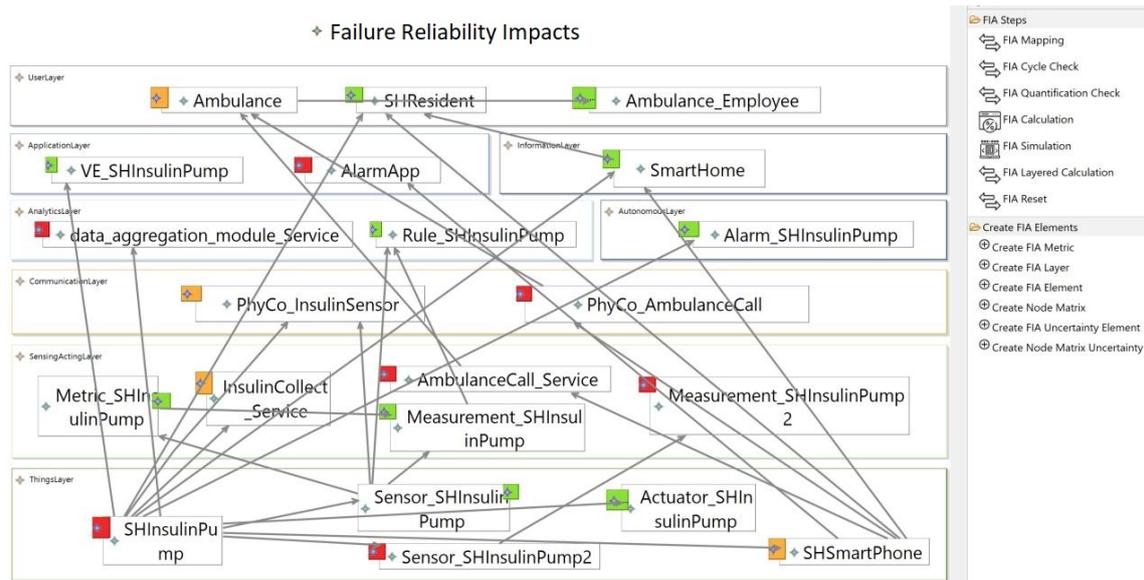


Figure 8.11.: AAL case study flow assessment - FIA

Figure 8.11 shows the fully executed FIA including analysis step functions offered by ArchiAna. Using the FIA mapping, the marked elements were transferred from figure 8.9. Additionally, all elements were mapped that are connected to flawed elements and also have the assessment attribute reliability. These are represented in almost all layers, and thus, corresponding FIA layers are created. Afterwards the cycle check is performed to not violate DAG requirements. Cycles were found between *Measurement_SHInsulinPump* - *Metric_SHInsulinPump* - *SHInsulinPump* - *SHSmartPhone* and accordingly must be deleted.

Subsequently, each FIA element is assigned discrete probability values relating to reliability in order to quantify them. The discrete states *Full* and *Low* are selected for this purpose. Table 8.10 shows the probability values of the elements of the things layer. In addition, the JPT is calculated for each element via its ancestors. For JPT, the values of the *Full* state are chosen, since no accident has yet been simulated, and therefore, the best case is to be assumed. For all other layers it is continued conform to this. For example, for *Rule_SHInsulinPump*, JPT is calculated via its ancestors *Sensor_SHInsulinPump*, *Measurement_SHInsulinPump*, and *Metric_SHInsulinPump*: $0.837 \cdot 0.99 \cdot 0.98$.

Single Probabilities - Reliability	FULL	LOW	JPT
SHInsulinPump	0,9	0,1	0,9
Actuator_SHInsulinPump	0,95	0,05	0,855
Sensor_SHInsulinPump1	0,93	0,07	0,837
Sensor_SHInsulinPump2	0,93	0,07	0,837
SHSmartPhone	0,91	0,09	0,819

Table 8.10.: FIA single probabilities - things layer

Thus, accident simulation and final FIA assessment is ready. SHInsulinPump is selected as starting point for an external hazard (Sensor_SHInsulinPump2 attrition) and the beginning of the resulting reliability reduction. Accordingly, an initial CPT must be created for this element which is calculated with the formula from the SAP654322: $(P(B) \cdot (a-2d))^x$. The probability value is taken from the table and the remaining values are created from the ancestors and descendants and the severity: $(0,9 \cdot (0-22))^4 = -79,2$. This means, that the reliability probability (state=*Full*) of SHInsulinPump is reduced by 79,2% in case of an accident. This value is the starting point for further CPT calculations of all FIA elements. For the element Rule_SHInsulinPump the calculated CPT can be seen in table 8.11 and was determined through the collider rule of chapter 7. Since its ancestors are not directly affected by the weakened sensor, the condition probabilities are still high and the reliability is not greatly affected.

Measurement_SHInsulinPump	Sensor_SHInsulinPump	FULL	LOW
FULL	FULL	0,95	0,05
LOW	LOW	0,8	0,2
FULL	LOW	0,91	0,09
LOW	FULL	0,92	0,08

Table 8.11.: CPT of Rule_SHInsulinPump

The probabilities are taken from the CPTs and the new JPTs are determined as described. Based on these, the color-coded evaluations are output in the FIA by comparing the old and new JPTs. Red or orange boxes indicate that the JP of reliability of the corresponding element has significantly decreased in case of an accident. As an example, element Sensor_SHInsulinPump2 now only has a JPT of under 1% through its CP(*Full*) 0,465 and its ancestor's CP(*Full*) of $0,9 - 0,792 = 0,108$. 11 elements have been identified whose reliability has been severely weakened and must be assessed further to observe financial possible loss.

Quantitative Impact Analysis

QIA is used to calculate the cost of a failure. However, as described, this is only applicable to a limited extent for the value of human wellbeing. Accordingly, indirect costs are considered in this step. In FIA, it was determined what happens when a sensor attrition causes a wrong measurement, and thus, the wrong insulin dose is dispensed. Accordingly, it can now be evaluated how expensive such a failure can be in terms of costs for an ambulance call. In a QIA, several risks and cases are considered. On the one hand, the insulin pump may trigger a medical emergency due to technical problems and justifiably call for help. On the other hand, only a faulty data transmission without affecting the insulin dispense may lead to an unnecessary ambulance call. These different cases change the causes and frequency but not the reliability and costs incurred, and are therefore, considered in one QIA. Figure 8.12 displays the fully conducted QIA, its results and QIA step functions of ArchiAna.

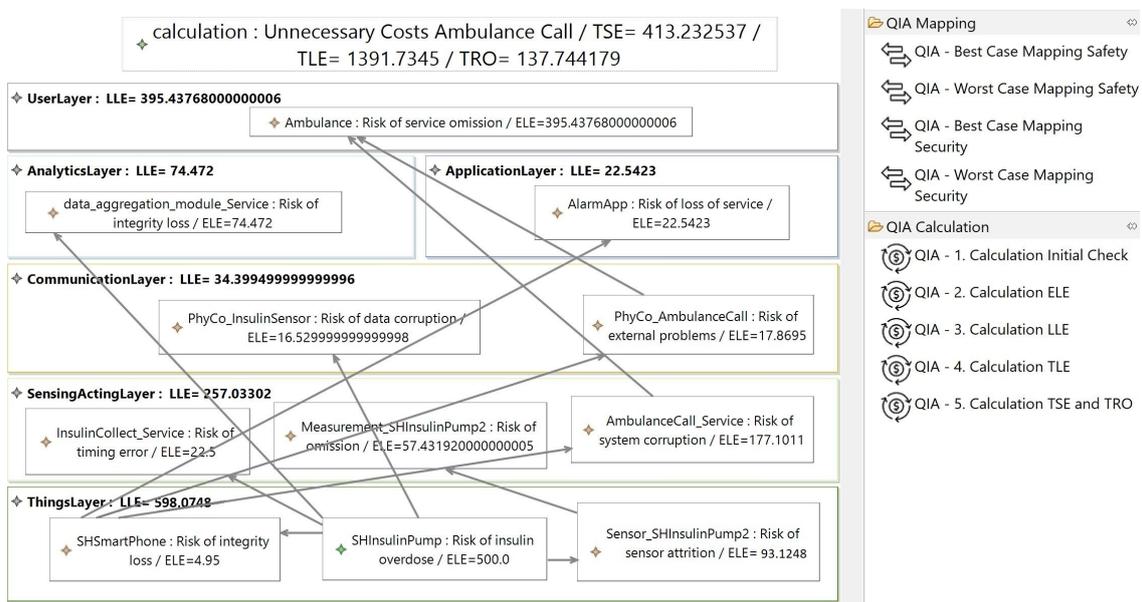


Figure 8.12.: AAL case study flaw assessment - QIA

The QIA mapping is performed to transfer red marked elements from FIA because they have a significant effect in case of accidents. The worst case mapping for safety concerns is selected, because a safety pattern is the initial starting point. The selection of worst case causes that the *Low* values from the FIA reliability CPTs are taken when transferring the probabilities of the element relations. For example, the relation between SHInsulinPump and Sensor_SHInsulinPump2 corresponds as calculated above: $1-0,108=0,892$. Subsequently, a risk is assigned to each QIA element. These range from risk of loss of services, to risk of omission, to major risk of system corruption problems.

Subsequently, ASLE are defined for all elements and the initial rate of occurrence

for the leaf nodes. For example, `Sensor_SHInsulinPump2` has an ASLE of 52,5€ to replace a sensor. The `data_aggregation_module_Service` is estimated to have ASLE of 42,8€ for the support units to recover and `Ambulance` is estimated to have ASLE of 120€ for personnel costs. For the leaf node `SHInsulinPump`, an occurrence of 2 is estimated based on the information in SAP654322. This provides all the information for QIA calculations, the results of which can be seen in figure 8.12. For instance, `Sensor_SHInsulinPump2` has an ELE of 93,1248 through $(0,892*2)*52,5$ with reference to formulas of chapter 7. The results of the LLE show that things and user layer have to expect the highest costs. The results of the whole model consideration show that almost 138 accidents are possible. If all accidents occur at the same time, a damage of about 1391€ can be expected. Due to the moderate severity level of 2 (based on SAP654322) and the high level of possible accidents, a high TSE is obtained. This means that countermeasures are urgently needed.

Countermeasure Decision Support Analysis

After critical points and their effects have been considered, various countermeasure scenarios are now compared with each other in order to counteract the hazardous system design. Two scenarios were planned for the current use case (figure 8.13). Scenario 1 shows an alternative outplace scenario and plans different countermeasures distributed over several areas of the flawed design. Scenario 2, on the other hand, is inplace and plans its countermeasures focused on one area. Both scenarios have recovery and failure resistance as requirements to be met by the planned countermeasures. This is derived from SAP654322.

Scenario 1 proposes to change the ambulance call workflow to include a control call to check if help is really needed. In addition, the weak sensor should be replaced and the insulin pump should only dispense insulin when the recommendation is confirmed in the app on the mobile phone. On a functional level, manual nutrient tracking is to be added to the app as well. With additional machine learning units on an Azure server, the correct insulin quantity can be suggested and compared with the measurements before it comes to dispensing. Thus, the scenario covers the faulty sensor, but also plans more control mechanisms for false recommendations and false emergency medical services. Accordingly, the countermeasures are distributed over several layers.

Scenario 2, on the other hand, concentrates mainly on the lower layers and focuses only on the problem of incorrect insulin levels. First and foremost, the faulty sensor should be replaced and exchanged with a sensor that can withstand more stress. In order to obtain even more safety, a third sensor is to be installed in each insulin pump, which will also increase the measurement interval. Furthermore, each insulin pump will be equipped with a smart actuator with more calculation capacity to detect inconsistencies. As a final check, there will also be a change in the insulin dispense workflow in this scenario in which a button must be pressed on the pump.

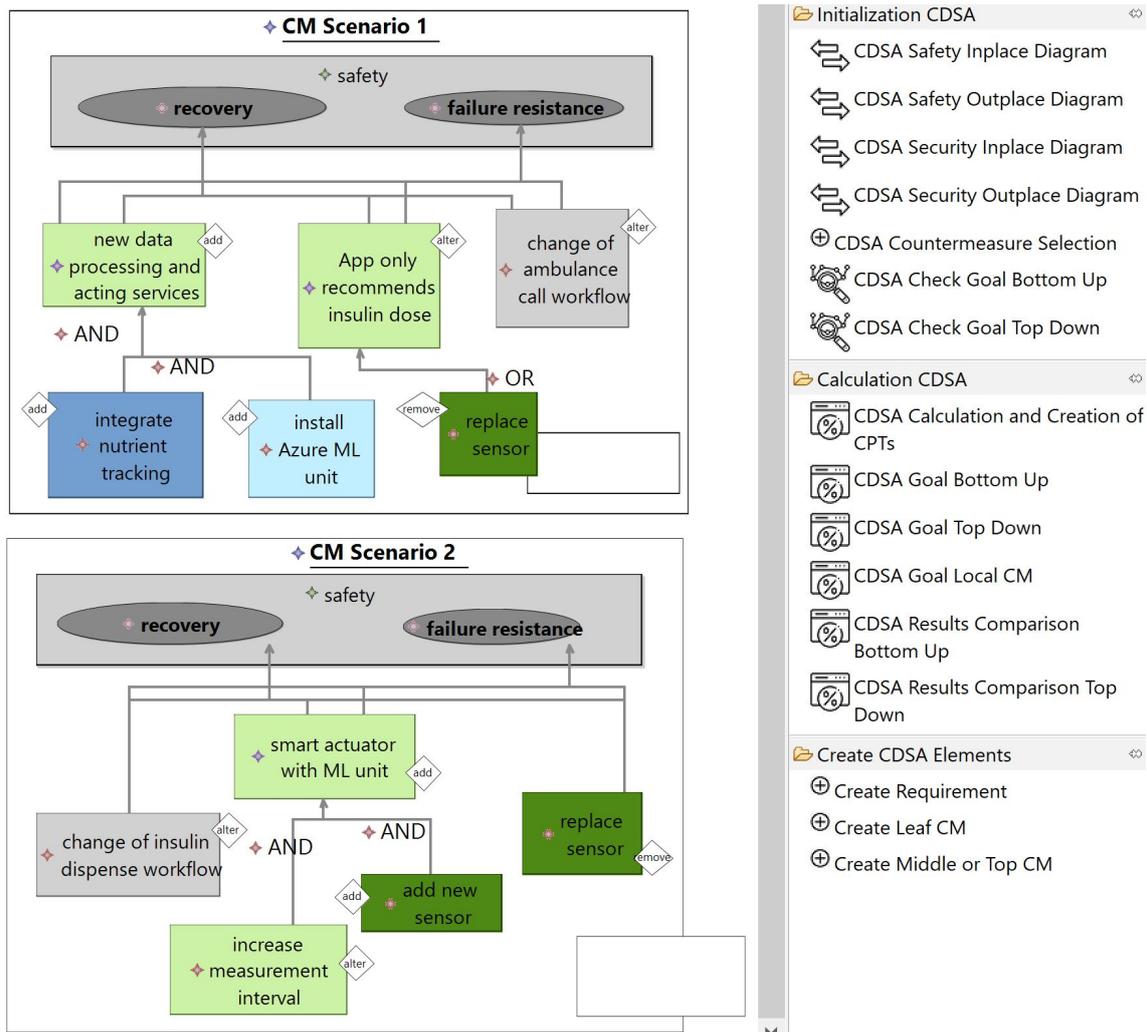


Figure 8.13.: AAL case study flow assessment - CDSA

For each planned countermeasure, the probability whether it has a positive effect on the requirements is determined. For countermeasures that are not leaf nodes and depend on other countermeasures, additional CP values are calculated. Table 8.12 shows the overview of all probability values for scenario 2. Each countermeasure has a value for each requirement. Since only smart actuator with ML unit is a dependent top countermeasure only its CP has to be calculated. The remaining CP are equal to P. According to chapter 7, the AND rule applies for the calculation and thus: $0,98 \cdot 0,89 \cdot 0,91 = 0,79$. Therefore, the JPs can also be determined and a utility of 5,11 ($0,62 \cdot 5 + 05 \cdot 4$) is calculated using the defined weights. This is a low utility and an indication that this countermeasure scenario is not sufficient. Scenario 1, on the other hand, has a value of 8,5 and is therefore, more suitable from a probability point of view to prevent or mitigate possible accidents. However, no countermeasure scenario can be finally selected until it has been verified that new services that come with countermeasures can be integrated into the previous system model.

Countermeasure	P (r.)	CP (r.)	P (f.r.)	CP (f.r.)
change of insulin workflow	0,99	0,99	0,95	0,95
smart actuator with ML unit	0,91	0,79	0,99	0,62
increase measurement interval	0,89	0,89	0,93	0,93
add new sensor	0,98	0,98	0,95	0,95
replace sensor	0,92	0,92	0,95	0,95
Requirement	JP	Weight		
recovery (r.)	0,62	5		
failure resistance (f.r.)	0,5	4		

Table 8.12.: Probability values and weights of CM Scenario 2

Service Interoperability Analysis

For each CDSA scenario a corresponding SIA scenario is created which contains new and previous services that are relevant for the comparison. Figure 8.14 shows the completed comparison of these scenarios. SIA scenario 1 contains the old `InsulinCollect_Service` in `CollectService` type and `AmbulanceCall_Service` in `ActingDecisionService` type. In addition, new services have been created for nutrient tracking, ambulance control, ML processing and recommendations. SIA scenario 2 contains the previous `data_aggregation_module_service` in `ProcessingService` type and `AmbulanceCall_Service` in `ActingDecisionService` type. Both scenarios contain the same chance node requirements as this is mandatory in the SIA to determine service quality.

Probabilities of chance nodes	A	C	V	CC
previous data processing service	0,92	0,93	0,9	0,85
new ML analytic service	0,95	0,87	0,94	0,95
ProcessingService Type	0,87	0,81	0,85	0,81
previous ambulance call service	0,95	0,98	0,97	0,95
new acting dispense service	0,94	0,92	0,76	0,96
ActingDecisionService Type	0,89	0,9	0,73	0,91
new additional glucose collect service	0,95	0,92	0,92	0,86
CollectService Type	0,95	0,92	0,92	0,86

Table 8.13.: Probability values of SIA Scenario 2

After setting up the SIA, service probabilities are defined for affecting service quality requirements (= chance nodes). Table 8.13 shows the values for scenario 2. Service type values are calculated using the JP of their associated services. For example, `ProcessingService` type P(correctness): $0.93 \cdot 0.87 = 0.81$. This means that all services of this type will be 81% completely correct in the system. This value is low in this scenario because the ML service will operate on a weak actuator, and therefore, have little computing capacity. Services that need to act directly with each other form service pairs. In these SIA scenarios they are checked

for the possibility to exchange data. In scenario 2, two critical pairs were discovered. The new service of the insulin dispense workflow and the previous AmbulanceCall_Service have issues exchanging data because the previous service could automatically retrieve data and now relies on manual confirmation. The old service data_aggregation_module_service, on the other hand, can only exchange data with the new service for insulin dispense with restrictions because the old service is still adapted to the old measurement time interval.

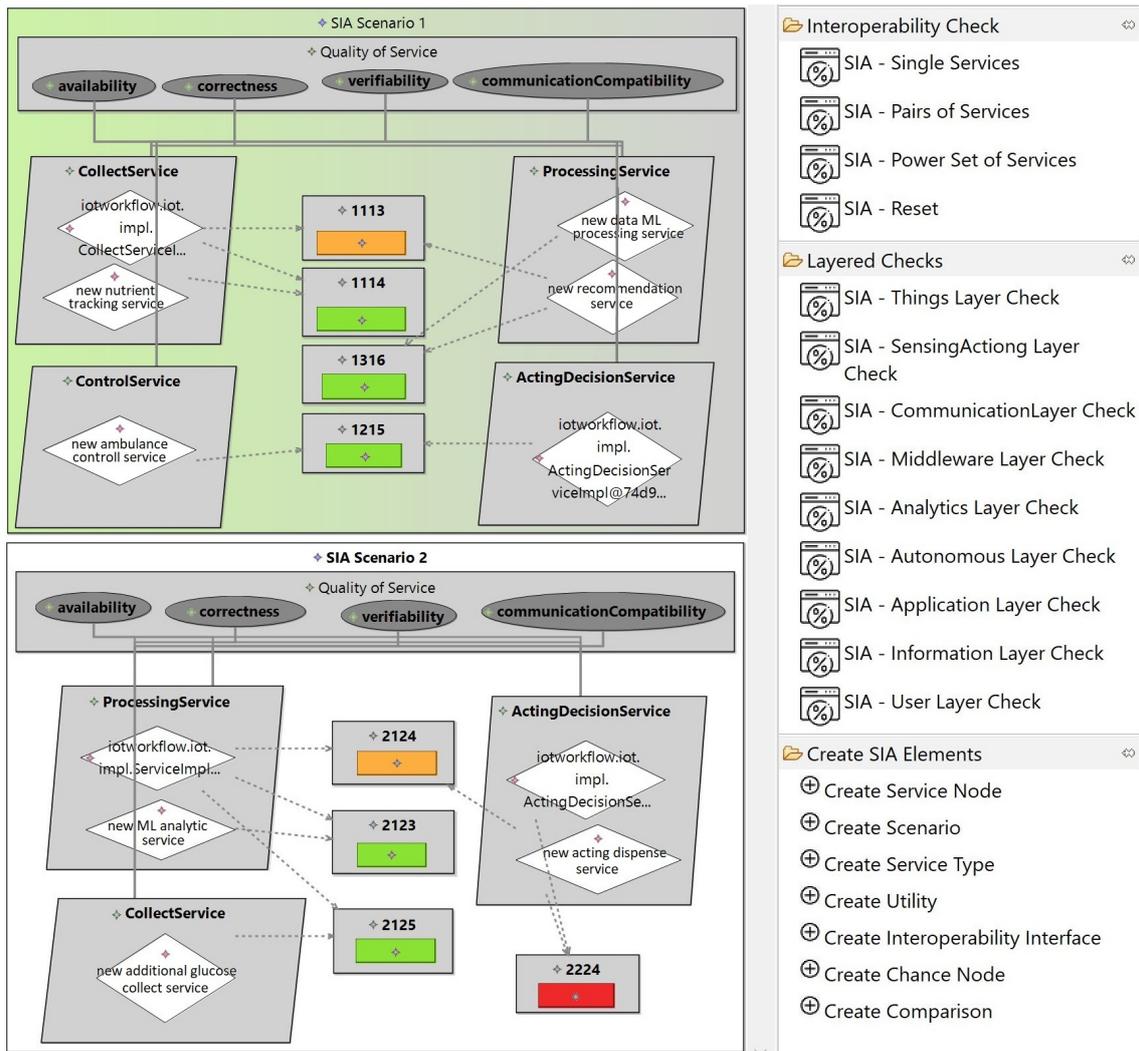


Figure 8.14.: AAL case study flaw assessment - SIA

Finally, the utility of scenarios is calculated and compared. For instance, scenario 2 utility: $(0,87*0,89*0,95)*2+(0,81*0,9*0,92)*3+(0,85*0,73*0,92)*2+(0,81*0,91*0,86)*4 = 7,16$, with the chance node weights 2/3/2/4. Since scenario 1 has a higher utility calculated, it is colored green and is recommended as the new design, since the service quality is estimated to be higher and CDSA scenario 1 also has a higher utility, and thus, seems to be better suited to mitigate the SAP654322 flaw.

8.2.2. Generated Performance Use Case

While in the first use case proper functionalities and calculations of the approach were checked, the performance of the approach is to be monitored next. Thus, it shall be checked if the approach can be executed in a reasonable time to justify the automation effort. For this purpose, different sized models are tested which are generated using a model generator to consider differences in the performance duration. The measurements were performed on the following set up:

- Processor: Intel(R) Core(TM) i7-6700HQ CPU @ 2.60 GHz
- Installed RAM: 16,0 GB
- System: 64-Bit-OS, Windows 10 Pro

The use case is based on the idea that IoT systems can become very large. Accordingly, a smart city is considered and oriented to environmental quality monitoring and threat detection systems of [CPS]. Accordingly, the IoT model and its things are spread across an entire city. Smart sensors can measure air pollution locally and accurately, improving the wellbeing of city residents and analyzing it for underlying factors. For the threat detection of the city, an acoustic method is used in which sensors are positioned on the streets, and thus, immediately notice abnormal situations. This includes car accident sounds, screams or gunshots. Thus, appropriate help can be sent directly to the affected location. Accordingly, smart cars, hospitals including bed management, ambulances with tables and other stakeholders are also involved. This leads to a huge, cluttered model.

8.2.2.1. Model Editor Complexity Test

Figure 8.15 shows an example of a smart city IoT model. Thereby, not only all required physical entities like sensors for air pollution and acoustic signals are modeled, but also their physical connections, services, processing units as well as the different stakeholders with their business systems and users.

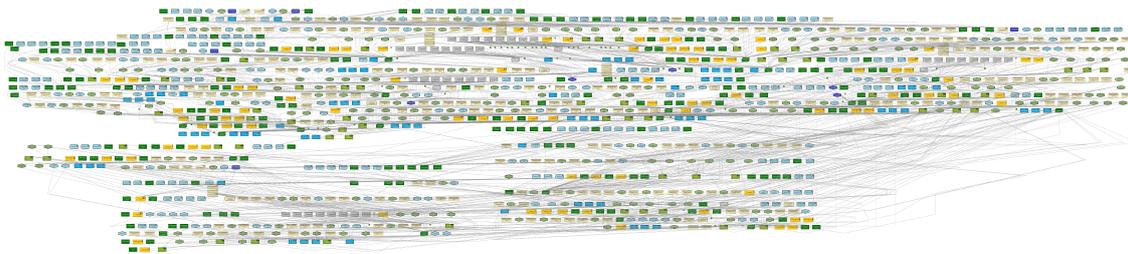


Figure 8.15.: Performance use case system model

Smart cities in particular can be viewed as a whole or only in sub-areas such as individual streets or neighborhoods. Accordingly, the model sizes can also vary in a use case. The performance test of the model editor considers smart city model sizes with 10, 100, 1000 and 10000 nodes. The above depicted model contains about 2000 elements. Thus, it can be seen that even very large models can be displayed with the model editor. The exact performance data depending on the model size can be seen in figure 8.16. The results have as axes the number of nodes and the time needed to generate and display the model from a XMI file. The performance test was split into the model view and the container view. It can be seen that in the low node range the views have almost the same times and can be generated very quickly. This is mainly due to the fact that a low number of nodes often results in a low complexity of the connections. If there are 1000 or more nodes, it is noticeable that the container view is created more quickly. This is due to the fact that fewer connections have to be displayed because of the container nesting. The sharp increase in time is due to the disproportionate increase in the number of connections to the nodes. However, all measured times are still in reasonable time for models with 10000 nodes and could be reduced by a better technical test set up. Also, models larger than 10000 nodes are uncommon, since the model editor is used manually during the design phase and most of the time cannot be filled by architecture mining functions.

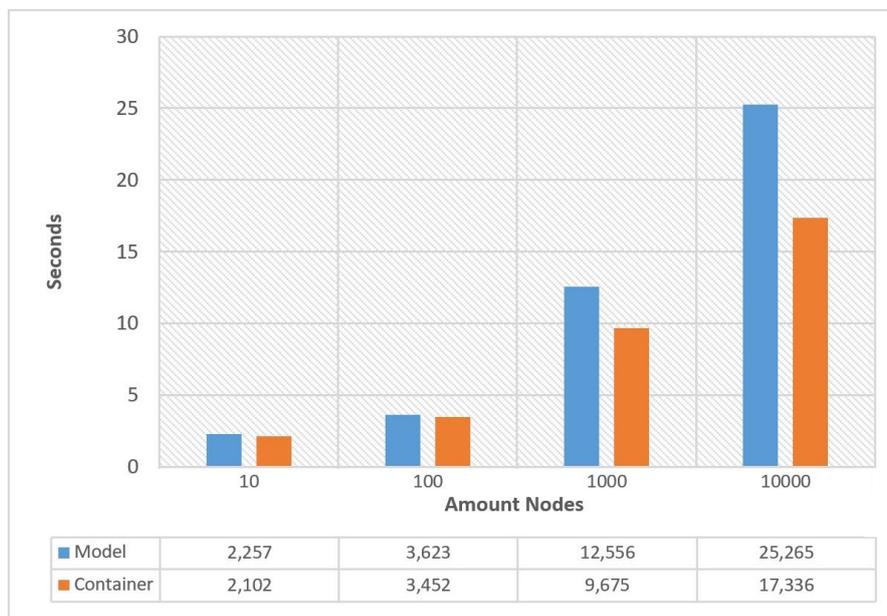


Figure 8.16.: Performance test of model editor

8.2.2.2. Flaw Identification Performance

To measure performance of flaw identification, all types of PRF templates should be used. Thus, a SAAP for smart cities is defined.

ID	927392
Name	Authentication in urgent situations
Component	Supercategory: Element to Protect
Element Type	Service
Element Category	Service, Facilities
Hardware	3G connection
Software	Acting Service
Layer	SensingActing Layer
Location	Local
Disruption Tolerance	Temporary tolerant
Fault	Complicated authentication method
Hazard	Help cannot be contacted fast enough
Classification	Loss of Service
Fault Class	Configuration Management, Initialization
Assessment Feature	Confidentiality
Probable Direct Impact	Supercategory: Estimated Consequences
Causal Types	Chain
Causal Relation	Confidentiality decreased: $(P(B)*((a-d)/x))/a$
Indirect Impact	Misuse of ambulance
Severity	Death, Serious Injury
Likelihood of Occurrence	Probable
Safety Requirements	Availability
Vulnerable/Hazardous Design	2-way authentication is implemented in a service to call an ambulance
Implementation	Supercategory: Artifact Combination
Element Filter Conditions	Subcategory: Concerned Elements
Node2Node	Stakeholder/ IoTDevice/ Service
Node2Relation	physicalentity/ service
NodeAttribute	type:Ambulance/ software:CallFunction/ serviceType:ActingService
Anti-Pattern Requirements	Subcategory: Recognition Details
Node2Node	Service
NodeAttribute	authentication:2Way
Flaw	authentication==2Way

Table 8.14.: AAL Safety Anti-Pattern

The SAAP addresses the exploitation of emergency services, some of which are triggered anonymously in cities. However, authentication of an ambulance call is only useful and secure under certain conditions. The present SAAP, therefore, verifies that no complicated authentication methods have been implemented, which would result in a possible blocking of emergency calls. The acting services are checked by ambulance stakeholders to prevent loss of service and to ensure proper configuration management.

As described, the performance tests should be performed with all PRF template types. Besides the presented SAAP, defined SEP, SAP and SEAP of the previous chapters and sections are used. Again, different sized smart city models were utilized for testing. Thus, all four patterns or anti-patterns were run on the different model sizes and the time to complete verification was measured. The results are comparable, but the actual numbers depend on the selected use case, since the amount of elements that are filtered affects the time. Accordingly, the results could vary with a different test set up, but the ratios would remain the same. It can be clearly seen that all patterns and anti-patterns are close to each other in time and the ratios to each other are independent of the model size. In addition, it can be seen that in small models with few elements with fewer attributes and relations, a check is very fast. Only from 10000 nodes onwards there is a clear increase, as the links become more complex as well. Since this is only 300 milliseconds, this can still be considered a reasonable time as it is a proportional increase and is significantly faster than a manual check.

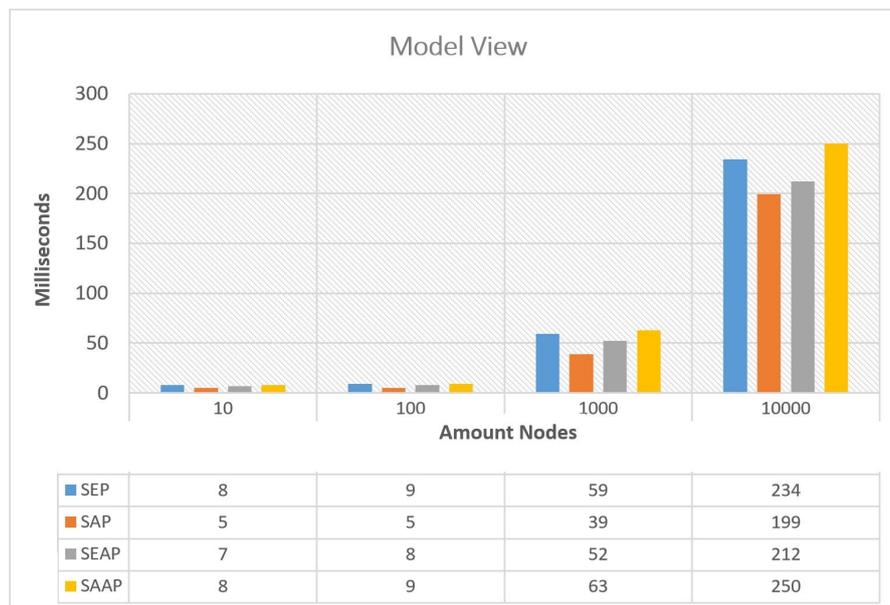


Figure 8.17.: Performance test of flaw identification - model view

The tests were performed on the model view and the container view. However, in figures 8.17 and 8.18, it can be seen clearly that there is no significant difference in

time, since the flaw identification is performed on the underlying XMI files, and thus, the complexity of the representation is irrelevant. Minimal fluctuations are based on technical reasons.

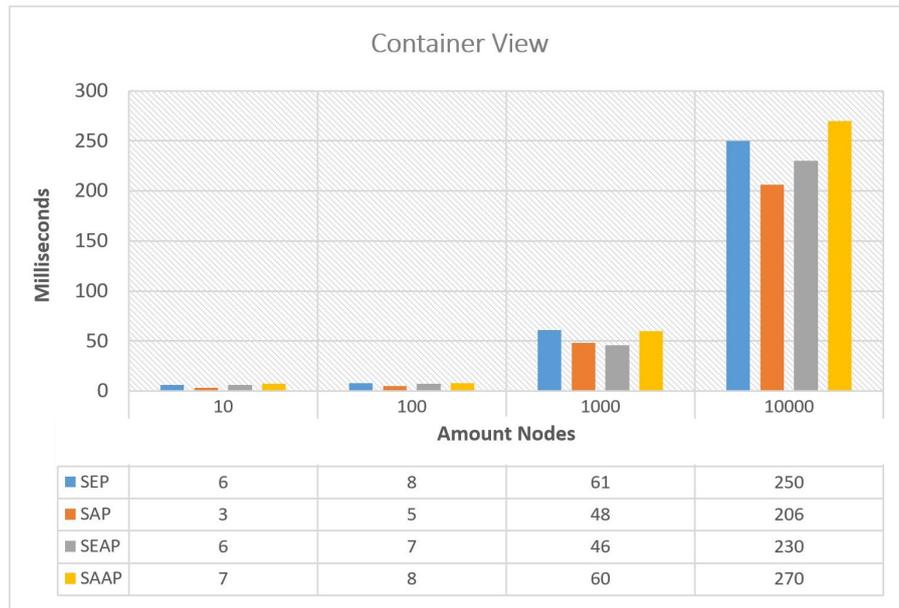


Figure 8.18.: Performance test of flaw identification - container view

8.2.2.3. Analyses Performance

Finally, performance of the assessment cycle is checked. Again smart city models are used, but up to a maximum of 1000 nodes, since it is not realistic for a single pattern to mark more than 10% of a model and only flaws and related elements are included in the assessment. Each analysis is divided into its most performance relevant analysis steps as each step can be performed individually.

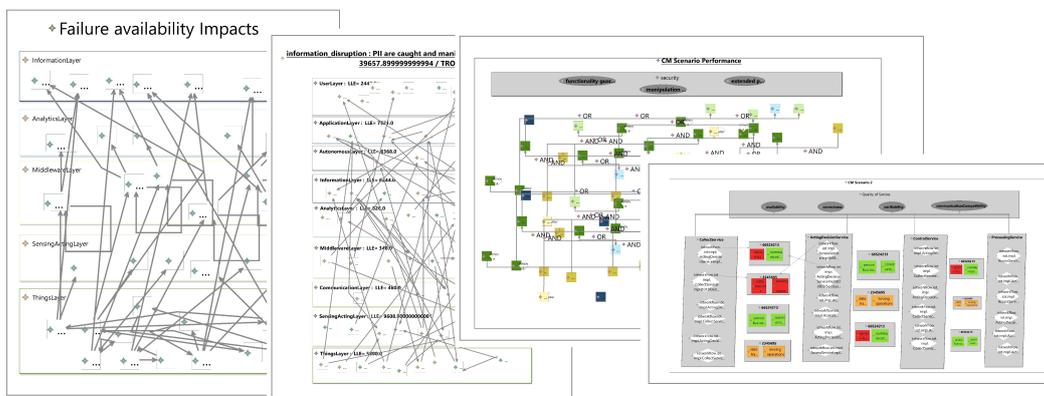


Figure 8.19.: Performance use case - analyses cycle

Several steps can be executed within a FIA. Six of them are computation intensive and were checked in figure 8.20 for their performance duration depending on the model size. The fastest steps are the cycle check, quantification check and layered calculation, since no BBN calculations are used here and only limited elements have to be considered. Accordingly, they perform very well in any model size. In the quantification and simulation steps, the times are significantly higher with increasing model size, since BBN calculations are used here which requires relationships of ancestors and descendants whereas these relationships are more complex in large models. However, the most critical performance step is the initial mapping from the IoT model to the FIA model, since not only the previous much larger model must be completely gone through, but also the graphical FIA representation must be created including the element classification in the individual layers. Critical filters in the flaw identification could reduce the number of elements to be transferred and improve performance. Since the FIA node quantity can be regarded as realistic, measured times are to be understood as reasonable.

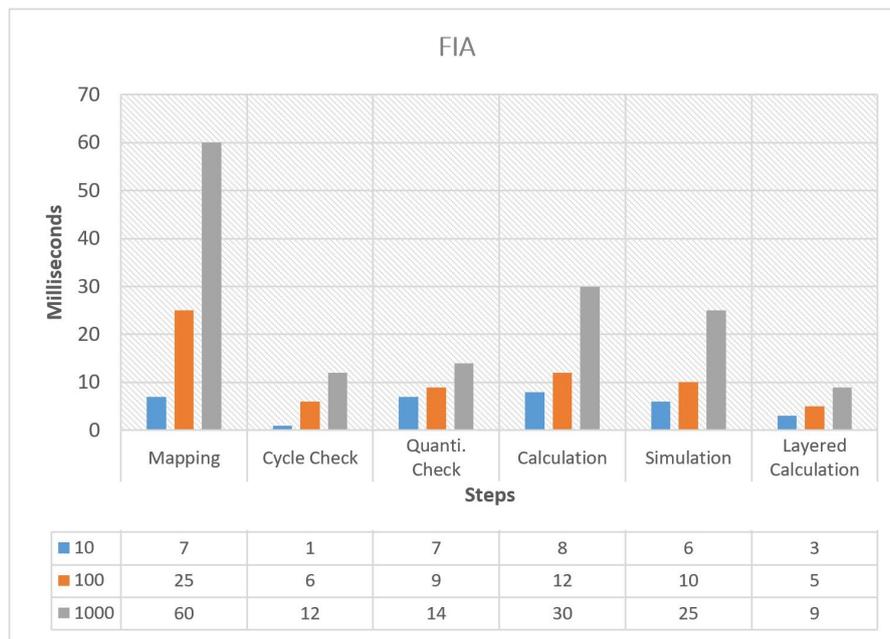


Figure 8.20.: Performance comparison of FIA

The performance results of QIA can be seen in figure 8.21. The performance times of all steps are significantly lower than in the FIA steps. This has two main reasons. Firstly, only a reduced selection of FIA diagram elements are mapped in a QIA and secondly, no complex BBN calculations are included. For reasons of comparability, the same model sizes are tested anyway. All four calculation analysis steps are very close to each other in their execution time, since the calculations are all based on approximately the same calculation types. The calculations for ELE and TSE&TRO are slightly higher because they consider each individual element and perform several calculations at the same time. Only in the mapping

a difference can be seen in very small and very large models. Thus, with 1000 nodes, the mapping is the highest time while the other models have the peak at ELE. This has the same reasons as the FIA mapping.

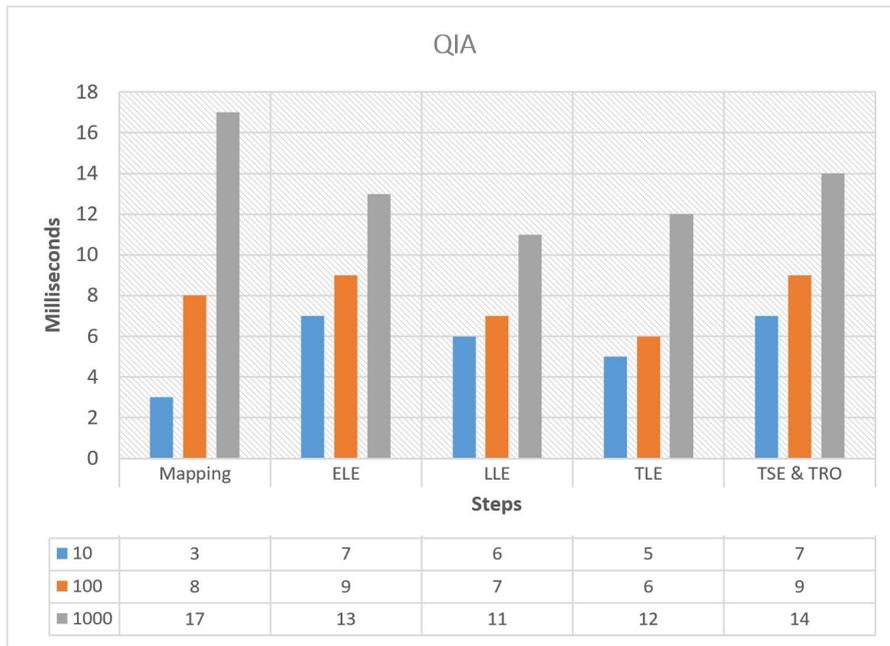


Figure 8.21.: Performance comparison of QIA

Next figure 8.22 shows the performance results of most important CDSA steps.

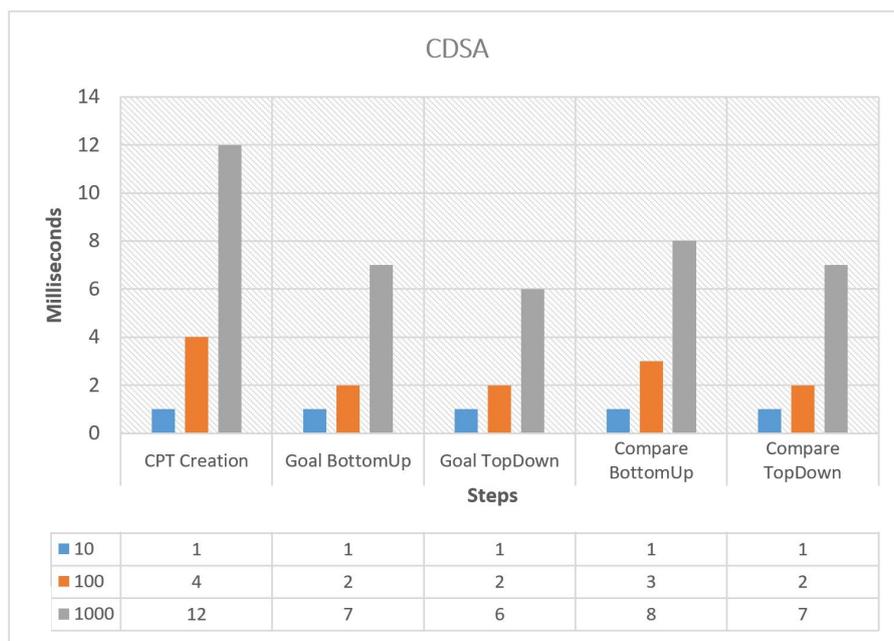


Figure 8.22.: Performance comparison of CDSA

The measurements for a CDSA are also significantly lower than for a FIA. Although BBN calculations are again included and performed here, a CDSA involves a few selected relations between nodes and not a large number of branched model components. Accordingly, the model sizes are no longer examples for smart cities, but for planned countermeasures. Accordingly, the measurements for 1000 planned countermeasures are rather unrealistic, but were nevertheless carried out for comparability. For small models, all steps take about the same amount of time. For medium and large models, however, it can be seen that the CPT creation is the most time-consuming, since BBN calculations are used here. After the initial preparation, the remaining steps are not time consuming regardless of the nodes.

The last performance measurements are performed for SIAs in different numbers of services under consideration. These times are once again higher because they contain more relations that were transferred from the original IoT model in order to be able to perform the comparisons of old with new services and their connections and interoperability. The steps increase in their performance gradually. While in the first step only individual services are considered and calculated, in the second step complex calculations of all pairs have to be made and some services have to be observed several times. In the power set, pairs, single services and their joint probability must be determined, and thus, are computation intensive. In the comparison step, only individual values are evaluated. Even if the performance increases significantly with the size of the models, the increase is proportional, and thus, are reasonable times.

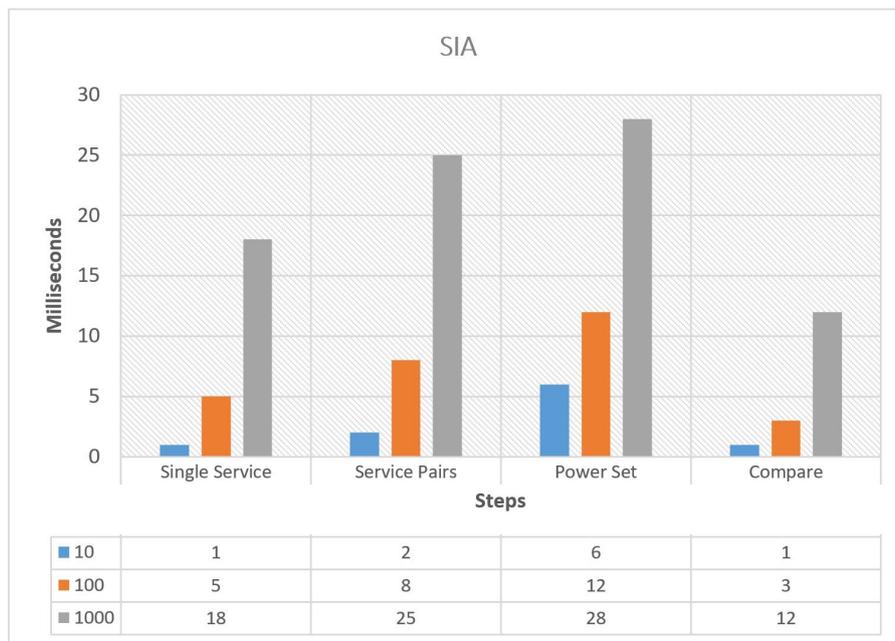


Figure 8.23.: Performance comparison of SIA

8.3. Scenario-based Evaluation

In order to cover further aspects, a qualitative scenario-based evaluation is also carried out. For this purpose, various scenarios are discussed. These will be used to test the quality of the approach under different conditions. For this purpose, several quality attributes are defined and evaluated: *adaptability*, *expandability*, *scalability* and *reusability*.

For each quality attribute or defined scenario, the areas of system modeling and architecture optimization are discussed.

- **Design Approach:** The meta model with its elements and relations as well as the layered architecture and the model editor should withstand the quality attributes since only a functioning interaction of the three components allows the use of the approach part.
- **Flaw Identification:** In this approach part, template categories, DSL as well as Xtend service generation functions shall be checked, since they represent a self-contained process. If only one link in the chain cannot fulfill the attribute, the entire approach part is not satisfied in this quality area.
- **Flaw Assessment:** The analysis cycle shall be checked in its generic and its analysis specific characteristics in order to fulfill the individual quality attributes. The functions implemented in the tool are also checked.

8.3.1. Adaptability

First, it is evaluated whether the approach parts are adaptable when the conditions of a use case change. Based on [SC01] and [Kur05] *adaptability* is defined as

the ability of software, systems, models or other artifacts to adjust to changing environmental conditions while preserving previous functionalities.

In the context of this evaluation, this definition is used to examine the three scopes mentioned above, to verify whether use case specific or generic changes have an impact on the processes, components or tools and which adjustments are possible and necessary. This ensures that the entire approach can continue to be applied in the future.

The following scenario is defined for the evaluation of the quality attribute *adapt-*

ability: The current approach is focused on use cases in the wellbeing sector, but can also be applied in other areas, and thus, be subject to requirement or guideline changes. These are usually influenced by new stakeholders or different physical environments. However, even minor use case changes may need to be adapted to new challenges as soon as new business targets are defined. For example, a medical IoT use case implemented in a hospital with critical equipment like a smart defibrillator or connected surgical tools underlays different and stricter laws to allow the connection of remote devices with this equipment. To include an operating room with surgeons into a complex IoT network the safety and security guidelines must be adapted (compare chapter 2).

Design Approach

The presented meta model is designed to cover several use cases in the IoT area. Accordingly, mostly no adaptation is necessary in the event of a use case change. However, if use case-related adaptations are necessary, the meta model can adapt its attributes in order to be able to include information appropriately in the model. Enumeration values can also be modified, for example to incorporate modified scales for SIL. In this way, important service or physical connection specifications can be adapted to new conditions as well. The underlying layered architecture is likewise designed to fit generically for all IoT use cases. However, element-layer mappings can be customized to adapt to possible new policies affecting layered protection. The corresponding model editor is adaptable to changed conditions because it is always equivalent to the meta model. If, in extreme cases, the entire meta model needs to be replaced, far-reaching changes to the flaw identification process will be necessary, but are possible.

Flaw Identification

If new regulations are introduced, this also includes the query of new possible flaws that are not allowed to be present anymore. The PRF template categories can have their values readjusted as mentioned in chapters 2.1.4 and 6. For example, the category disruption tolerance can be adapted if critical medical devices require a new rating scale. If adjustments have been made in the meta model, this must also be adapted in the PRF. This applies to the pattern language and Xtend generation methods in order to be able to query modified meta model attributes. It must be noted that predefined patterns can subsequently no longer be used or must be adjusted afterwards. However, if due to the changed conditions, previously defined patterns can no longer be used, the automatic pattern service generation can still take place in order to make the modified patterns directly usable again.

Flaw Assessment

The analysis cycle of the assessment cycle is based on approach adaptation, and is therefore, ideally suited for necessary adjustments. If changes are made to the meta model, this has no effect on the previous analyses, since the use case elements are transferred directly to analysis diagram elements independently of the

meta model. Thus, the functionality of the analyses is still guaranteed. However, if changes in guidelines, environmental influences or simply changes in the business goal affect the assessment details, its cycle can be adjusted. However, care must be taken to ensure that outputs and inputs continue to match in order to guarantee an uninterrupted cycle. By making adjustments in the PRF, several values in the analyses are automatically adapted, such as the defined causal relation used in FIA for calculating the conditional probability. In this way, the results or the calculation path of the analyses can also be influenced and adapted. The specification of the individual analyses can also be adapted to new conditions, such as the weighting for CDSA and SIA. All values that are first defined in the specification can be adjusted while the automation of the analyses and their functionality continues to operate without restriction. This is made possible by the dynamic implementation of the analyses.

Based on these characteristics of the approach, the quality attribute defined above is considered to be satisfied.

8.3.2. Expandability

As an extension to *adaptability*, *expandability* is checked as a further quality attribute. Inspired by [Ber+95], *expandability* is defined as

the ability to integrate new functionalities or parts into an approach to enable new features while preserving existing capabilities.

This quality attribute is tested with a scenario to prove the possibilities of an extension of the design approach and the architecture optimization. The aim is not only to check that new components can be included, but also whether previous functions still operate, and thus, whether previously defined use cases still perform. This quality attribute, thus, ensures that the approach can be enlarged if necessary.

The following scenario is defined for the evaluation of the quality attribute *expandability*: Due to the constant development of technology, new use cases are always possible, which may want to incorporate artifacts into existing use cases in the future. In addition, further developments entail new challenges or new requirements from stakeholders. This leads to extensions of the whole approach. For example, an existing smart home adds another stakeholder to order directly groceries based on the contents of the refrigerator. The new set up includes a camera in the refrigerator which may lead to privacy concerns. This and the new requirements added by the new business stakeholder cause the smart home provider to extend the assessment.

Design Approach

Since IoT systems are dynamic and constantly changing, extensions to the existing meta model may be necessary if further development reveals new areas of IoT networks that were not previously explored. These can range from physical elements to virtual product management elements, e.g. of a food delivery service. The present meta model can be expanded by new elements via new extended connections with the help of the present EMF functions. The layered architecture is also able to accommodate new elements. By extending the layer enumeration, an expansion of the architecture is also possible, although all possible meta model extensions are covered by the present layered architecture. Meta model changes always entail modeling editor expansion to ensure modeling of the new elements. If previous elements are only expanded by additional attributes, the model editor is automatically updated.

Flaw Identification

As described, an expansion scenario may also introduce new challenges, such as privacy violations. In order to identify corresponding design flaws in newly modeled extended use cases, an extension of the PRF is necessary. On the one hand, the values of the categories can be extended, e.g. the enumeration of attack goals. On the other hand, entire new template parts or categories can be included. In the case of privacy-related patterns, adding a privacy challenge part to the templates is a reasonable option. The templates are not a closed approach and can be extended. If the templates are only extended and old parts are not changed, previously defined patterns and anti-patterns can still be used in contrast to adaptability scenarios. As soon as the new extensions have been included in the DSL and in the service generation methods, extended patterns can also be queried automatically and do not influence the functionality of the flaw identification. An additional possible extension is the connection to other already existing vulnerability databases. The necessary integration factors have already been discussed in chapter 5.3.

Flaw Assessment

The addition of new stakeholders also expands new assessment requirements. For example, a new business stakeholder may be interested in an assessment of data accuracy after a discovered flaw or wants to determine the change impact on targeted business processes. Accordingly, the analysis cycle must either be extended, or parallel analyses must be built in. Therefore, new analyses can be added to the cycle or detached to interact and enable a second analysis cycle, depending on the assessment goals. In chapter 5.3 necessary factors, such as the interaction of the analysis steps, have already been discussed. For example, the connection components of the analysis meta models must be extended to allow connections to new analyses and to enable the input/output transfer. However, existing analyses can also be expanded. For example, new BBN formulas can be added to expand the selection of leading questions, or new cost functions can be added to enable further financial quantitative estimations. The existing tool

functions remain unchanged, but may have to be expanded to include additional functions.

Based on these characteristics of the approach, the quality attribute defined above is considered to be satisfied.

8.3.3. Scalability

One of the most significant characteristic of IoT networks is their size and the number of nodes they contain. Therefore, all approaches that are to be applied on IoT systems shall be checked for *scalability*. According to [Tec17] and [Kol+13] *scalability* is defined as

the ability to resize while covering increased needs and ensuring all functionalities. Scalability includes the capability to build and store large models and their connected DSLs for subsequent model querying, regardless of their size.

In the context of this evaluation, the quality attribute *scalability* is used to check whether the model editor can handle a large number of nodes and examine the ability of the functionalities of the flaw identification and assessment to adapt to the changed demand. *Scalability* is intended to test the stability and competitiveness of the approach parts.

The following scenario is defined for the evaluation of the quality attribute *scalability*: Even though use cases can vary greatly in size, this does not necessarily mean any change in their basic conditions and challenges, but the effort required to monitor such a system increases immensely. Accordingly, it is necessary to enable automated and time-efficient monitoring and evaluation, especially in such large use cases. The scenario is based on the second use case of case study-based evaluation in which performance tests have already been carried out. Accordingly, an IoT use case is considered distributed over an entire city with a large number of sensors that directly notify help in the event of an accident. As this is a safety-critical use case, it must be ensured that the system does not contain any malfunctions. Therefore, the PRF must be able to define and interrogate complex patterns, while the assessment must subsequently be able to consider all the connections and reliably evaluate the impacts.

Design Approach

As defined, the scalability of the approach starts with the capability to represent large models. The meta model and the associated layered architecture are not affected by this, since this is independent of the size of model instances. However, the capability of the model editor must be considered. As shown in the examples

from the case study-based evaluation, the model editor performs in a reasonable time regardless of the use case size. The layout can become cluttered depending on the number of edges. However, since a container model representation has been implemented, a better element structure of large models can be ensured. Using the views and filters provided, additional areas, elements or edges can be hidden to allow manual verification.

Flaw Identification

However, especially in large models, a manual check is not effective. Accordingly, automated model queries must continue to be possible as the system grows. PRF templates are independent of the size of the systems to which patterns or anti-patterns are to be applied. It is possible to define small simple patterns, but also large complex patterns if corresponding use cases require them. Accordingly, the DSL and the associated service generation methods are not negatively affected by demand growth. Effects can be seen in the execution of the flaw identification in ArchiAna. As shown in the performance tests, the functionality is guaranteed in a reasonable time, since the implementation is designed in order to query the edges and nodes for their interaction as effectively as possible. The described filter conditions are used for this purpose. They allow direct narrowing down of the elements and edges to be checked. Finally, only a significantly reduced number of components must be checked for the pattern requirements.

Flaw Assessment

The analysis cycle is not affected by the size of a model. However, the functionality of the individual assessment steps must be checked. Since FIA starts the cycle, the first analysis step is FIA mapping. Thus, the model is reduced again in components. As described, attributes are only transferred if the assessment attribute matches and additionally only elements that are related to the identified flawed elements are mapped. Thus, FIA has to handle only a reduced number, and thus, demand. The subsequent QIA reduces this quantity once again by concentrating on critical difference matrices. However, costs must be available for all components for the calculation. This is a manual challenge, but does not affect the technical scalability. The two decision design analyses CDSA and SIA cannot make use of the reduced models since they concern other aspects. Through the flaw identification, one pattern after the other is queried. Accordingly, in the CDSA, countermeasures are searched for only one problem at a time. Thus, the size of the CDSA does not depend on the size of the system, but on the complexity of the planned countermeasures and their requirements, and thus, calculations scales for large use cases. SIA calculations are always based on the same four requirements, and therefore, have a boundary on the calculation effort from the beginning. In addition, only the service elements of the old and new models are considered, thus, excluding all other elements of the large model.

Based on these characteristics of the approach, the quality attribute defined above is considered to be satisfied.

8.3.4. Reusability

As a last quality attribute, *reusability* is checked to ensure the added value of the automatization, since a one-time applicability would not justify the automatization effort. Based on [SR90] *reusability* is defined as

a two-part attribute, whereas reusability represents firstly the reuse of artifacts in different process phases and secondly the reuse of artifacts in context of another application.

Reusability is to be evaluated with a scenario for both meanings. It is checked whether the individual approach parts are connected with each other in that artifacts are passed on and reused in the course of the process. Furthermore, it is evaluated if the approach is generic and can be reused in other use cases, and is therefore, use case independent. This is examined for the generic approach as well as for completed artifacts, e.g. patterns or anti-patterns.

The following scenario is defined for the evaluation of quality attribute *reusability*: Given that IoT use cases usually have similar basic characteristics, the presented approach structure can be applied to different use cases and results can be reused in new use cases and benefit from them. But mainly it is important that within one use case everything is consistent and results can be reused. This allows the most benefit from results, but also ensures that previous decisions are not neglected or be in conflict with new decisions. In this scenario, a security expert leaves an IoT network provider. If new elements are added to the network in the future, it must be possible to check how security has been tested in the past and how previous problems have been assessed and addressed in order to plan further consistently. For instance, flaws should not suddenly be evaluated differently if nothing changed technically but a new employee weights them differently. Since the approach represents a cycle, the expert knowledge must be included in all steps.

Design Approach

According to the *Contributions* of this approach, the IoT(-WD) meta model and layered architecture was built based on use case independent safety and security challenges and existing IoT reference architectures with the goal to design a meta model which is suitable for reuse for IoT use cases of any kind. Since the model editor is based on the same principles, it is also reusable. Because safety and security aspects can be modeled with the editor, knowledge is not lost when an employee leaves and the previous models can continue to be used. The meta model components are also reused in the further approach process. They are needed in every step of the flaw identification and in the flaw assessment only the connection and reuse of the structure ensures the automated analysis start.

Flaw Identification

By definition from 2.1.4, a fundamental property of patterns is reusability. Accordingly, the PRF was designed to be suitable for expert knowledge preservation and to reuse the knowledge of the defined patterns in the further course of the assessment. On the one hand, a defined pattern can be used in other use cases if the pattern is generic enough. But the main task of the defined patterns and anti-patterns is the reuse in the same use case. Thus, former decisions can be made traceable and problem solutions from the past can be reused. However, this only applies under the assumption that no requirements have changed. On the other hand, the PRF structure is designed in such a way that the individual parts can be reused. While the challenge part is used to classify threats or hazards, implementation parts are reused in the DSL and in service generation methods. The assessment parts are transferred to the analysis cycle, for instance the severity level as weights for analysis calculations. In addition, the PRF is based on the reuse of meta model structure to be able to design pattern combinations of the components to fit the model.

Flaw Assessment

The flaw assessment can also be reused in two ways: Deployment of the cycle in other use cases and dependency of the analyses based on previous approach parts. The cycle was designed independently of the use case and relates to the basic assessment questions, and can therefore, also be reused in other areas. Furthermore, knowledge about analysis specifications can be transferred when no expert is available, such as the ASLE of a QIA for known and common IoT things. But the main reuse takes place through the described transfer of the PRF assessment part values. Thus, the assessment attribute defined by experts, already known impacts and further specification relevant details are reused, such as requirements for the CDSA set up. In addition, the entire cycle is based on reusability, as the outputs of the previous analysis are used as input for the next steps.

Based on these characteristics of the approach, the quality attribute defined above is considered to be satisfied. Thus, all quality attributes to be tested could be considered as fulfilled by the approach, and thus, the operational capability is proven.

Part V.

CONCLUSION AND OUTLOOK

9

Conclusion

In the last chapters several approaches have been presented to enable detection and assessment of safety and security vulnerabilities already in the design phase. These approaches were evaluated in different ways to prove the correct implementation, applicability and benefit of the approach. In this chapter the approach and results of this dissertation are summarized. *Objectives* presented in chapter 1 are taken up to be verified and to evaluate the accomplishments of this dissertation. For this purpose, the parts of IoT system modeling, flaw identification and flaw assessment are addressed separately.

9.1. Design Approach

This thesis deals with the concept of ever growing IoT networks and their problematic use in safety- and security-critical systems. The main focus is on the management possibilities of these complex systems and the early detection and prevention of flaws. A three-part approach was developed for this purpose. The first part is the design approach to provide the basis for flaw detection in the design phase.

The design approach is intended to model IoT systems with all their intertwined connections while also specifying, and subsequently considering, safety and security aspects. This aims in having IoT networks under control at an early stage and being aware of weak points that can already be incorporated in the design phase, and thus, directly can be prevented. Accordingly, a modeling basis is needed. For this purpose, a meta model was developed that is capable of modeling IoT networks, especially in the field of wellbeing. In order to obtain architectural requirements for this meta model and to make vulnerability relevant aspects depictable, a IoT safety and security challenge review was carried out.

The review considered safety hazards, security threats and specific challenges in the wellbeing area, as this is a highly critical area. For safety hazards, the fault sources were examined to explore in which areas the origins of unintentional failures lie. For this purpose, different fault classes were investigated, such as software changes, logic errors or data corruptions. Accordingly, the aspects that already have their causes in the architecture were extracted to create architectural requirements for modeling activities. For security threats, classic violations of security policy or recommendations and their resulting attacks were considered. The attacks were divided into physical object attacks, software attacks, protocol attacks and attacks on data. Aspects that have architectural reasons were again identified to consider these aspects in the design phase. Finally, a challenge categorization was prepared for the review in order to be able to transfer the identified architectural requirements to the other parts.

As a result, a meta model was presented that is based on a layered architecture. This layered architecture consists of nine layers and considers all elements of an IoT network from small physical things to the executing actors. Based on further defined meta model requirements, the design approach was specified. On the one hand, the required hardware and software aspects were included to be able to map the structure, processes and functions of an IoT system. On the other hand, qualitative requirements were taken into account. For example, the meta model was designed to be context-aware, modular, scalable and various management elements as well as stakeholder requirements were included. Finally, the model editor of the developed ArchiAna tool was presented to make the design approach applicable.

Verification of Objectives and Contributions

The design approach addresses *Objective 1*, and thus, aims at creating "a unified and reusable IoT system modeling method" that has a "possibility to analyze safety and security required aspects". This *Objective* was satisfied by identifying architectural requirements through the challenge review and building on those requirements to develop a layered meta model that is not only suitable for IoT modeling but also includes vulnerability assessment aspects and additionally allows the connection to further flaw identification methods. For this purpose, the requirements from the chapters 3 and 4 were successfully implemented.

9.2. Architecture Optimization

The second and third part of the approach of this dissertation are united in the architecture optimization approach. This is based on the design approach and mainly deals with the challenges of early flaw detection, subsequent flaw assessment and final mitigation by design. The detailed connection of these parts has been described in chapter 5.

9.2.1. IoT Flaw Identification

Especially in large and complex IoT systems, vulnerability detection is a major problem. Combined with frequent flaw detection after deployment, problems are detected too late or can only be eliminated at high cost. Accordingly, the flaw identification approach of this dissertation is designed to automate and structure the analysis of early modeled IoT systems and to detect weaknesses caused by design flaws. For this purpose, a recognition framework based on patterns and anti-patterns has been developed which, in addition to automated recognition, ensures knowledge preservation in order to make comprehensible decisions. Patterns represent desirable design decisions that must not be violated. Anti-patterns on the other hand represent problematic architectural relationships that should not be present.

The PRF is divided into four parts. In order to structure patterns or anti-patterns, templates were first developed. Depending on whether safety or security patterns are to be considered, different templates were presented. These templates consist of several sections for challenge, assessment and implementation details to provide all necessary information for flaw detection and subsequent assessment. These include details on the identification and categorization of the patterns, but also information on the interrelationships of affected elements. The implementation details are used to enable automation and scalability of the recognition process. All templates focus on covering IoT and wellbeing aspects in their structure.

In the second step of the PRF, a DSL was presented to specify the patterns in a machine-readable way. For this purpose, several rules were set up to define patterns or anti-patterns, based on the presented and connected meta model, and to make them applicable to IoT models. In the third PRF part, a pattern service code generation method was developed that generates executable patterns to make them automatically queryable. This method is based on the implementation details of the PRF and depends on the meta model structure to check all safety and security aspects of a model.

Ready-to-run patterns or anti-patterns can be stored in a pattern database and used in the final step of the PRF, the flaw identification process. For this purpose, functions have been implemented in the ArchiAna tool to perform this automated identification by calling the pattern services and checking the entire model for the presence of a pattern or anti-pattern. If a pattern is violated or an anti-pattern is found, a flaw is reported.

Verification of Objectives and Contributions

The first part of the architecture optimization addresses the *Objective 2*, and thus, aims to develop "an IoT specific flaw identification process to mitigate impacts of possible threats or hazards at an early architectural level". This *Objective* was satisfied by the development and deployment of the PRF and its ability to leverage and preserve expert knowledge by structurally defining and querying patterns or anti-patterns with known design desires or problems. The architectural requirements from chapters 3 and 4 were considered and defined optimization requirements from chapter 5 were fulfilled.

9.2.2. IoT Flaw Assessment

Once a flaw has been identified and a weak point in the design of a planned IoT system is known, further steps must be considered. These include, on the one hand, analyses of consequences and, on the other hand, the elimination of the flaw. For this purpose, the second half of the architecture optimization approach provides a flaw assessment methodology. Therefore, a consecutive assessment cycle was developed, which takes up the optimization requirements from chapter 5. The impacts of a possible flaw are to be evaluated and afterwards different countermeasure designs are to be compared in order to find the appropriate solution for a flaw. In order to have a solid basis for these assessments, existing architecture analyses were examined, selected and transferred.

The consideration of existing architecture analyses was limited to analyses from the EAM area, as these analyses focus on the design level and allow a transfer due to their generic structure. 40 analysis types were identified on the basis of a literature review. These were reviewed for maturity, functional goals and technical methods. Thus, it was possible to sort out which analyses can be used for the purpose of IoT flaw assessment. Since the cycle was to be designed to build on each other, it was necessary to select technically compatible analyses in order to be able to continue using inputs and outputs. Finally, four analyses were identified. In addition, an analysis structure was developed to structure the IoT flaw assessment methods.

This resulted in the cycle consisting of FIA, QIA, CDSA and SIA. The first two analyses are responsible for the assessment of impacts. FIA receives as input the

results of the flaw identification and uses a BBN methodology to determine the impacts on associated elements based on a specified assessment attribute. Conditional probability is used to determine how the status of the associated elements will deteriorate if the identified flaw leads to a hazard or threat. The QIA, as second analysis, takes these results and determines, on the critical path of impacts, the potential costs incurred if an IoT system suffers limitations due to attacks or accidents based on the identified flaw. Once the flaw assessment was completed, two analyses were included in the cycle for countermeasure planning. For the CDSA, a BBN-based approach was developed that analyzes different countermeasures at different layers for multiple safety or security requirements. Thus, it is calculated which countermeasures have a positive impact on the design and are able to mitigate the flaw. As the last analysis in the cycle, SIA was presented to assess services that are newly added by new countermeasures in IoT systems. This is done by calculating the probabilities that new and old services are working properly with each other, and thus, not pose any new problems in the architecture. For all four analyses, functions have been implemented in the ArchiAna tool to enable automated assessment of large IoT models.

Verification of Objectives and Contributions

The final approach part addresses *Objectives* 3 and 4, which aim to "identify requirements to transfer existing architecture analysis approaches into IoT safety and security management" and to develop "a structured and consecutive architecture analyses workflow to use flaw assessment methods to conduct a holistic IoT design optimization". These *Objectives* were satisfied by identifying and categorizing analysis approaches through a literature review, and thus, determining which analysis goals and techniques fit for an IoT flaw assessment. Based on the selection of analyses with matching techniques, the consecutive cycle could be developed that can perform the automated assessment of As-Is scenarios based on potential hazards or threats and includes the subsequent planning of flawless To-Be scenarios. As with flaw identification, the optimization requirements from chapter 5 were complied.

10

Limitations and Future Work

Even though this work offers a holistic approach some areas are out of scope of this dissertation. Accordingly, in the following additional possibilities for further development are pointed out. This includes extensions, modifications or necessary adaptations to apply it in other areas. The capability of the approach for these future works was demonstrated in chapters 5 and 8. In order to have a structured vision of the possible future work, it is necessary to consider first the limitations of this thesis.

The meta model presented here has been optimized for IoT systems in the medical and wellbeing sector. However, if the approach is to be applied in other areas, the modeling of these systems with all necessary attributes is only possible to a limited extent. Accordingly, a further development could be that the meta model is designed more generically, or further meta models are developed and connected. One example of this is the use of IoT in industry or Cyber Physical Systems. These are also security-critical systems which could benefit from this approach. However, the meta model requires adjustments, e.g. in the area of coordinates or job workflows. An example of an adapted meta model can be seen in project deliverables mentioned in chapter 1.

At the current time, this work still contains a few manual steps. One of these steps is the modeling of systems. If new systems that do not yet exist are to be investigated, this manual step is unavoidable. However, if an existing system is to be investigated, various architecture mining options could be applied in the future. This automated capture of systems, including their nodes and connections, can be performed using external methods, e.g. with the tool developed by qbilon [qbi]. In this way, elements could be determined automatically and transferred to the meta model and displayed. This prevents important information from being lost. Although the use of architectural analysis is most useful before the system has been implemented, an after-the-fact analysis of the design is useful too to identify hidden causes of weaknesses or to detect unnoticed errors before they occur. Software agents can be helpful in this respect which already are used by tools like dynatrace [Dyn] or OpenTelemetry [Opea].

The presented PRF is intended to be used independently of programming knowledge. However, the creation of patterns and anti-patterns requires the use of the DSL, and thus, a development environment. This requires a certain computer affinity. In the future, a more convenient use could be made by using a tool such as Office Excel to select the PRF categories. This could be automatically transferred to the DSL and generated in the next step in Sirius Services.

In addition, the selection of PRF categories is limited. The current approach only provides for the identification of safety or security patterns and their assessment criteria in the wellbeing area. This could be improved by the future extension of criteria. The technical conditions for this option are given as long as the filter and requirement conditions of the implementation part are still included. Thus, the identification would not have to be limited to safety and security flaws, but could also cover other areas such as privacy or green IoT guidelines. Additionally, the assessment and challenge parts can be extended by adding use-case specific criteria.

The last limitation of the PRF and further developments for it is the pattern database. When the approach is newly introduced, the database is empty and shall be filled by experts. This method is only possible in companies that have such expert knowledge available. Accordingly, the PRF could be connected to existing databases in an automated way and transferred to the DSL and accordingly to Sirius Services. Examples of such databases are: NVD [NVD], Exploit Database of Rapid7 [Rap] and CVE [Mit]. These known vulnerabilities could be used to populate the pattern database and keep it up-to-date.

The analysis cycle is also subject to limitations as not all identified available architecture analyses were able to adapt to the defined IoT requirements, e.g. heterogeneity analysis or interface analysis. In chapter 7 only four analyses have been designed and are included in the assessment process. However, as already mentioned in chapter 5 further analyses can be connected to related analyses. Generic analysis or assessment approaches that have already been successfully used in use case independent applications can be used for this purpose. For example, the mentioned FMEA is a well-known tool for risk management in unrestricted purposes. As many guidelines require the usage of this, a connection to the designed IoT architecture analyses can be built. As FMEA is used to determine severity and probability of threats or hazards, this possible future connection can be used to provide data for FIA and QIA calculations.

Another point for future work can be the further automation of the assessment cycle. Accordingly, automated countermeasures could be proposed in the CDSA depending on the identified pattern. Alternatively, a countermeasures database could be created that could make various suggestions and be adapted based on the associated architecture layer.

The last point in the outlook is subsequent linkage to code analyses. After the system has been secured on the architectural level, it is possible to go one step further and additionally examine or plan code based on the results of the architectural analyses. Examples for this kind of developments are JOANA which examines information flow in Java code for security leaks [Sne] or the code-based analyses which are developed during the CPS4EU project. [CPS]

Part VI.

Annex

Bibliography

- [AAA13] Nawaf Alharbe, Anthony S. Atkins, and Akbar Sheikh Akbari. "Application of ZigBee and RFID Technologies in Healthcare in Conjunction with the Internet of Things". In: *Proceedings of International Conference on Advances in Mobile Computing & Multimedia*. MoMM '13. Vienna, Austria: Association for Computing Machinery, 2013, pp. 191–195.
- [Age16] TechTarget IoT Agenda. *Internet of Things in the Enterprise - Your expert guide to getting started with IoT*. accessed on 12.08.21. 2016. URL: https://cdn.ttgtmedia.com/digitalguide/images/Misc/EA-Marketing/Eguides/IoT_in_the_Enterprise.pdf.
- [AGW11] Stephan Aier, Bettina Gleichauf, and Robert Winter. "Understanding Enterprise Architecture Management Design-An Empirical Analysis". In: *Proceedings of 10th Conference on Wirtschaftsinformatik*. 2011.
- [AHA13] Mervat Abu-Elkheir, Mohammad Hayajneh, and Najah A. Ali. "Data management for the internet of things: Design primitives and solution". In: *Sensors* 13.11 (2013), pp. 15582–15612.
- [Ahl+12] Frederik Ahlemann et al. *Strategic enterprise architecture management: challenges, best practices, and future developments*. Springer Science & Business Media, 2012.
- [Aie+09] Stephan Aier et al. "A Survival Analysis of Application Life Spans based on Enterprise Architecture Models". In: *3rd Workshop on EMISA*. 2009, pp. 141–154.
- [AKM18] Hezam Akram Abdul-Ghani, Dimitri Konstantas, and Mohammed Mahyoub. "A comprehensive IoT attacks survey based on a building-blocked reference model". In: *International Journal of Advanced Computer Science and Applications* 9.3 (2018), pp. 355–373.
- [Alb19] Mohamed Albanna. "Effects of Electromagnetic Field (EMF) On Implantable Medical Devices (IMD)". In: *International Journal of General Science and Engineering Research (IJGSER)*. Vol. 4. 2019, pp. 47–57.
- [Ale77] Christopher Alexander. *A pattern language: towns, buildings, construction*. Oxford university press, 1977.
- [All20] Cloud Security Alliance. "Managing the Risk for Medical Devices Connected to the Cloud". In: (2020). accessed on 12.08.21, pp. 1–19. URL: cloudsecurityalliance.org/artifacts/managing-the-risk-for-medical-devices-connected-to-the-cloud.
- [Ana+18] Muhammad Rizwan Anawar et al. "Fog computing: An overview of big IoT data analytics". In: *Wireless Communications and Mobile Computing* 2018 (2018).

- [Ant21] FinancesOnline - James Anthony. *10 IoT Trends for 2021/2022: Latest Predictions According To Experts*. accessed on 21.04.21. 2021. URL: <https://financesonline.com/iot-trends/>.
- [Ant96] Annie I Anton. "Goal-based requirements analysis". In: *Requirements Engineering, 1996., Proceedings of the Second International Conference on*. IEEE. 1996, pp. 136–144.
- [AR10] Terje Aven and Ortwin Renn. "Risk management". In: *Risk Management and Governance*. Springer, 2010, pp. 121–158.
- [Arm10] Ashraf Armoush. "Design patterns for safety-critical embedded systems." PhD thesis. RWTH Aachen University, 2010.
- [AV13] Shashank Agrawal and Dario Vieira. "A survey on Internet of Things". In: *Abakós 1.2* (2013).
- [Bau+13] Martin Bauer et al. "Internet of Things – Architecture IoT-A Deliverable D1.5 – Final architectural reference model for the IoT v3.0". In: *EU Project IoT-A* (July 2013).
- [BCW17] Marco Brambilla, Jordi Cabot, and Manuel Wimmer. "Model-driven software engineering in practice". In: *Synthesis lectures on software engineering 3.1* (2017), pp. 1–207.
- [Bec+17] Sven Beckmann et al. "Generic sensor framework enabling personalized healthcare". In: *2017 IEEE Life Sciences Conference, LSC 2017* (2017), pp. 83–86.
- [Ber+09] Bernd Bertsche et al. *Zuverlässigkeit mechatronischer Systeme, Grundlagen und Bewertungen in frühen Entwicklungsphasen*. Springer-Verlag Berlin-Heidelberg, 2009.
- [Ber+95] Brian N. Bershad et al. "Extensibility safety and performance in the SPIN operating system". In: *Proceedings of the fifteenth ACM symposium on Operating systems principles*. 1995, pp. 267–283.
- [BHS07] Frank Buschmann, Kevlin Henney, and Douglas C Schmidt. *Pattern-oriented software architecture, on patterns and pattern languages*. Vol. 5. John Wiley & Sons, 2007.
- [BIT11] BITKOM. *Enterprise Architecture Management – neue Disziplin für die ganzheitliche Unternehmensentwicklung*. Berlin-Mitte. Bundesverband Informationswirtschaft, Telekommunikation und neue Medien, 2011.
- [BIY08] Ruth Breu, Frank Innerhofer-Oberperfler, and Artsiom Yautsiukhin. "Quantitative Assessment of Enterprise Security System". In: *Third International Conference on Availability, Reliability and Security April* (2008), pp. 921–928.
- [BJ17] Radia Belkeziz and Zahi Jarir. "A survey on Internet of Things coordination". In: *Proceedings - 2016 3rd International Conference on Systems of Collaboration, SysCo 2016* (2017), p.1–6.

-
- [BM12] S Balaji and M Sundararajan Murugaiyan. "Waterfall vs. V-Model vs. Agile: A comparative study on SDLC". In: *International Journal of Information Technology and Business Management* 2.1 (2012), pp. 26–30.
- [BMR04] Bernhard Bauer, Jörg P Müller, and Stephan Roser. "A model-driven approach to designing cross-enterprise business processes". In: *OTM Confederated International Conferences "On the Move to Meaningful Internet Systems"*. Springer. 2004, pp. 544–555.
- [BMS09] Sabine Buckl, Florian Matthes, and Christian M Schweda. "Classifying enterprise architecture analysis approaches". In: *Enterprise Interoperability*. Springer, 2009, pp. 66–79.
- [BMS10] Sabine Buckl, Florian Matthes, and Christian M Schweda. "Towards a method framework for enterprise architecture management—a literature analysis from a viable system perspective". In: *5th International Workshop on Business/IT Alignment and Interoperability (BUSITAL 2010)*. 2010, pp. 46–60.
- [Boe+05a] Frank S de Boer et al. "Change impact analysis of enterprise architectures". In: *IEEE International Conf. on Information Reuse and Integration*. 2005, pp. 177–181.
- [Boe+05b] Frank S de Boer et al. "Enterprise architecture analysis with XML". In: *System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on*. IEEE. 2005.
- [BSI12] BSI. "Leitfaden Informationssicherheit". In: *Bundesamt für Sicherheit in der Informationstechnik* (2012), pp. 1–90.
- [BT10] James M Blum and Kevin K Tremper. "Alarms in the intensive care unit: too much of a good thing is dangerous: is it time to add some intelligence to alarms?" In: *Critical care medicine* 38.2 (2010), pp. 702–703.
- [BTM15] Rajendra Billure, Varan M. Tayur, and V. Mahesh. "Internet of Things - A study on the security challenges". In: *Souvenir of the 2015 IEEE International Advance Computing Conference, IACC 2015* (2015), pp. 247–252.
- [Buc+06] Tobias Bucher et al. "Analysis and application scenarios of enterprise architecture: An exploratory study". In: *2006 10th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW'06)*. IEEE. 2006, pp. 28–28.
- [Buc+09a] Sabine Buckl et al. "A pattern-based approach to quantitative enterprise architecture analysis". In: *AMCIS 2009 Proceedings* (2009), p. 318.
- [Buc+09b] Sabine Buckl et al. "A wiki-based approach to enterprise architecture documentation and analysis". In: *ECIS 2009 Proceedings*. 2009, pp. 1476–1487.
-

- [Buc+11] Sabine Buckl et al. "A meta-language for Enterprise Architecture analysis". In: *Enterprise, Business-Process and Information Systems Modeling*. Springer, 2011, pp. 511–525.
- [Bun17] Bundesverwaltungsamt. "Fehlermöglichkeits- und Einflussanalyse (FMEA)". In: *Organisationshandbuch* (2017).
- [Bun21] Bundesrat. *Zweites Gesetz zur Erhöhung der Sicherheit informationstechnischer Systeme*. BGBl I Nr. 25. 2021. URL: <https://www.bundesrat.de/bv.html?id=0324-21>.
- [Bus+11] Markus Buschle et al. "A tool for enterprise architecture analysis using the PRM formalism". In: *Lecture Notes in Business Information Processing* 72 LNBIP (2011), pp. 108–121.
- [Bus+96] Frank Buschmann et al. *Pattern-Oriented Software Architecture: Patterns for Resource Management*. John Wiley & Sons, 1996.
- [CC13] Michael J Covington and Rush Carskadden. "Threat implications of the internet of things". In: *2013 5th International Conference on Cyber Conflict (CYCON 2013)*. IEEE. 2013, pp. 1–12.
- [CD13] Ann Cavoukian and Mark Dixon. *Privacy and Security by Design: An Enterprise Architecture Approach*. Information and Privacy Commissioner, Canada. 2013.
- [CHX17] B. Cai, L. Huang, and M. Xie. "Bayesian Networks in Fault Diagnosis". In: *IEEE Transactions on Industrial Informatics* 13.5 (Oct. 2017), pp. 2227–2240.
- [Cic+17] Federico Ciccozzi et al. "Model-driven engineering for mission-critical iot systems". In: *IEEE software* 34.1 (2017), pp. 46–53.
- [CIS] CISS. *Safety- and security-critical systems*. Center for embedded software systems. accessed on 19.05.21. URL: <https://www.ciss.dk/safety-and-security-critical-systems/>.
- [CMR14] Abiy Biru Chebudie, Roberto Minerva, and Domenico Rotondi. "Towards a definition of the Internet of Things (IoT)". PhD thesis. 2014.
- [Com] Foster & Company. *Internet of Things (IoT)*. accessed on 21.04.21. URL: fostec.com/de/kompetenzen/digitalisierungsstrategie/internet-of-things/.
- [Com00] Software Engineering Standards Committee. *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*. IEEE-SA Standards Board. 2000.
- [Com14] Michael Koster - Arm Community. *Design Patterns for an Internet of Things*. accessed on 22.07.21. 2014. URL: <https://community.arm.com/iot/b/blog/posts/design-patterns-for-an-internet-of-things>.

- [Con+16] Industrial Internet Consortium et al. "Industrial Internet of Things Volume G4: Security Framework, 2016". In: *Object Management Group: online* (2016).
- [Cor21] MITRE Corporation. *MITRE ATTACK*. accessed on 13.07.2021. 2021. URL: <https://attack.mitre.org/>.
- [CPS] CPS4EU. *Cyber Physical Systems for EU*. Grant Agreement Number 826276. URL: <https://cps4eu.eu/>.
- [DA09] M. R. Davoudi and F. S. Aliee. "A New AHP-based Approach towards Enterprise Architecture Quality Attribute Analysis". In: *Third International Conference on Research Challenges in Information Science, RCIS 2009*. 22-24 April 2009, pp. 333–342.
- [Dav89] Fred D Davis. "Perceived usefulness, perceived ease of use, and user acceptance of information technology". In: *MIS quarterly* 13.3 (1989), pp. 319–340.
- [DD18] Medical Device Directive and Medical Devices. "Cyber Security Requirements for Network Connected Medical BSI Publications on Cyber Security". In: (2018), pp. 1–9.
- [Dee00] Mary Jo Deering. "Federal Issues and Approaches". In: *The Internet and Health Communication: Experiences and Expectations* (2000), p. 355.
- [Del+11] Matteo Della Bordella et al. "Towards a method for realizing sustained competitive advantage through business entity analysis". In: *Enterprise, Business-Process and Information Systems Modeling*. Springer, 2011, pp. 216–230.
- [Dep12] Department of Defense USA. "Department of Defense Standard Practice System Safety Amsc". In: *Mctechsystems.Com* (2012). accessed on 25.05.21, pp. 1–98. URL: <http://www.mctechsystems.com/resources/MIL-STD-882E.pdf>.
- [Dep20] Statista Research Department. *Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025*. accessed on 25.05.2021. 2020. URL: <https://bit.ly/38TUYg0> (visited on 02/25/2020).
- [Dev12] Medical Devices. "Essential Principles of Safety and Performance of MD and IVD MD". In: *Strategic Analysis* 6.1-2 (2012), pp. 101–135.
- [Dig17] Wipro Digital. "Reference Architecture : A Primer on IoT-Based System: The 7 high-level architecture requirements for an IoT-based system The 3 tiers of IoT logical reference architecture". In: (2017). accessed on 26.08.21, pp. 1–8. URL: <https://wiprodigital.com/2017/11/21/enterprise-reference-architecture-a-primer-on-iiot-based-systems/>.
- [Dim16] Dimiter V Dimitrov. "Medical internet of things and big data in healthcare". In: *Healthcare informatics research* 22.3 (2016), pp. 156–163.

- [DIN13] Deutsche Kommission Elektrotechnik Elektronik Informationstechnik in DIN und VDE. *Medical electrical equipment - Part 1: General requirements for basic safety and essential performance (IEC 60601-1:2005 + Cor. :2006 + Cor. :2007 + A1:2012)*. 2013.
- [DM13] Jim DelGrosso and Gary McGraw. *Opinion: Software [in]security – software flaws in application architecture*. TechTarget. accessed on 05.05.21. 2013. URL: <https://searchsecurity.techtarget.com/opinion/Opinion-Software-insecurity-software-flaws-in-application-architecture>.
- [DN02] Liliana Dobrica and Eila Niemela. “A survey on software architecture analysis methods”. In: *IEEE Transactions on Software Engineering* 28.7 (2002), pp. 638–653.
- [Doh+10] Angelika Dohr et al. “The internet of things for ambient assisted living”. In: *2010 seventh international conference on information technology: new generations*. Ieee. 2010, pp. 804–809.
- [Dou12] Charalampos Doukas. *Arduino, Sensors, and the Cloud*. APress, 2012.
- [DP11] Ken Decreus and Geert Poels. “A goal-oriented requirements engineering method for business processes”. In: *Information Systems Evolution*. Springer, 2011, pp. 29–43.
- [DS12] M. Razavi Davoudi and K. Sheikhvand. “An approach towards enterprise architecture analysis using AHP and fuzzy AHP”. In: *International Journal of Machine Learning and Computing* 2.1 (2012), pp. 46–51.
- [DS98] Mark T. Dishaw and Diane M. Strong. “Supporting software maintenance with software engineering tools: A computed task–technology fit analysis”. In: *Journal of Systems and Software* 44.2 (1998), pp. 107–120.
- [Duf14] Jim Duffy. *8 Internet things that are not IoT*. accessed on 11.05.21. 2014. URL: <https://www.networkworld.com/article/2378581/8-internet-things-that-are-not-iot.html>.
- [DV17] Jyoti Deogirikar and Amarsinh Vidhate. “Security attacks in IoT: A survey”. In: *Proceedings of the International Conference on IoT in Social, Mobile, Analytics and Cloud, I-SMAC 2017* (2017), pp. 32–37.
- [Dyn] Dynatrace. *Dynatrace*. URL: <https://www.dynatrace.de/>.
- [Eks+09] Mathias Ekstedt et al. “A tool for enterprise architecture analysis of maintainability”. In: *European Conference on Software Maintenance and Reengineering*. IEEE. 2009, pp. 327–328.
- [EMB96] 1073-1996 EMB IEEE. “Standard for Medical Device Communications — Overview and Framework”. In: (1996).
- [EMF] EMF. *Eclipse EMF*. accessed on 24.04.21. URL: <https://bit.ly/3bUNnjU>.

- [Eng18] Ralf S. Engelschall. *Architecture Fundamentals*. 2018.
- [Ent] Hewlett Packard Enterprise. *Reference Architectur Definition*. accessed on 04.05.21. URL: <https://bit.ly/3nLI90g>.
- [ER19] Nampuraja Enose and S. Ramachandran. *Die Industrie 4.0 ist mehr eine Evolution als eine Revolution*. Digitale Welt. accessed on 24.04.21. 2019. URL: <https://digitaleweltmagazin.de/2019/08/09/die-industrie-4-0-ist-mehr-eine-evolution-als-eine-revolution/>.
- [ES09] Mathias Ekstedt and Teodor Sommestad. "Enterprise architecture models for cyber security analysis". In: *2009 IEEE/PES Power Systems Conference and Exposition (2009)*, pp. 1–6.
- [Exp18] Exponent. *Battery-Powered Medical Devices: Their Failure Modes and Mitigation Strategies*. accessed on 28.05.21. 2018. URL: <https://www.exponent.com/knowledge/alerts/2018/11/battery-powered-medical-devices/?pageSize=NaN&pageNum=0&loadAllByPageSize=true>.
- [Far+10] Matthias Farwick et al. "Towards living landscape models: Automated integration of infrastructure cloud in enterprise architecture management". In: *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*. IEEE. 2010, pp. 35–42.
- [FDA17] FDA. *Cybersecurity Vulnerabilities Identified in St. Jude Medical's Implantable Cardiac Devices and Merlin@home Transmitter: FDA Safety Communication*. accessed on 18.02.21. 2017. URL: <https://bit.ly/2qqkgiA>.
- [FFJ09a] Ulrik Franke, Waldo Rocha Flores, and Pontus Johnson. "Enterprise architecture dependency analysis using fault trees and bayesian networks". In: *Proceedings of the 2009 Spring Simulation Multiconference*. Society for Computer Simulation International. 2009, p. 55.
- [FFJ09b] Ulrik Franke, Waldo Rocha Flores, and Pontus Johnson. "Enterprise Architecture Dependency Analysis using Fault Trees and MODAF". In: *Proc. 42nd Annual Simulation Symposium ({ANSS}) (2009)*, pp. 209–216.
- [FHK09] Ulrich Frank, David Heise, and Heiko Kattenstroth. "Use of a domain specific modeling language for realizing versatile dashboards". In: *Proceedings of the 9th OOPSLA workshop on domain-specific modeling (DSM)*. Citeseer. 2009.
- [FJW97] Henry M. Franken, Henk Jonkers, and Mark K. de Weger. "Structural and quantitative perspectives on business process modelling and analysis". In: *Proceedings of the 11th European simulation multiconference, Istanbul, Turkey*. Citeseer. 1997, pp. 595–599.
- [Fla18] Data Flair. *5 IoT Applications in Healthcare Field You Must Know*. accessed on 07.05.21. 2018. URL: <https://data-flair.training/blogs/iot-applications-in-healthcare/>.

- [Fou16] OWASP Foundation. *Security by Design*. Website. accessed on 21.04.21. 2016. URL: https://wiki.owasp.org/index.php/Security_by_Design_Principles.
- [Fow18] Martin Fowler. "Refactoring: improving the design of existing code". In: *Addison-Wesley Professional* (2018).
- [Fre12] William Freeman. "Designing for Patient Safety: Developing Methods to Integrate Patient Safety Concerns in the Design Process". In: *The Center for Health Design* (2012).
- [Fre15] Paul Fremantle. "A reference architecture for the internet of things". In: *WSO2 White paper* (2015), pp. 02–04.
- [Fu+17] Kevin Fu et al. "Safety , Security , and Privacy Threats Posed by Accelerating Trends in the Internet of Things". In: *CRA Report* (2017), pp. 1–9. URL: <http://cra.org/ccp/wp-content/uploads/sites/2/2017/02/Safety-Security-and-Privacy-Threats-in-IoT.pdf>.
- [Gam95] Erich Gamma. *Design patterns: elements of reusable object-oriented software*. Pearson Education India, 1995.
- [GB20] Johannes Geismann and Eric Bodden. "A systematic literature review of model-driven security engineering for cyber–physical systems". In: *Journal of Systems and Software* 169 (2020), p. 110697.
- [Gen+17] Dimitris Geneiatakis et al. "Security and privacy issues for an IoT based smart home". In: *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. 2017, pp. 1292–1297.
- [Gla+03] C. Glaros et al. "A wearable intelligent system for monitoring health condition and rehabilitation of running athletes". In: *4th International IEEE EMBS Special Topic Conference on Information Technology Applications in Biomedicine, 2003*. IEEE. 2003, pp. 276–279.
- [Gla+08] John Glaser et al. "Advancing personalized health care through health information technology: an update from the American Health Information Community's Personalized Health Care Workgroup". In: *American Medical Informatics Association* 15.4 (2008), pp. 391–396.
- [GLJ11] Gang Gan, Zeyong Lu, and Jun Jiang. "Internet of things security analysis". In: (2011), pp. 1–4.
- [Groa] The Open Group. *ArchiMate® 3.0.1 Specification*. accessed on 29.04.21. URL: <https://pubs.opengroup.org/architecture/archimate301-doc/>.
- [Grob] The Open Group. *TOGAF*. accessed on 27.04.21. URL: <https://pubs.opengroup.org/architecture/togaf8-doc/arch/>.

-
- [Har19] Chris Harvey. *Lithium-ion Batteries and Recall Challenges for Medical Devices*. accessed on 28.05.21. 2019. URL: <https://www.mddionline.com/components/lithium-ion-batteries-and-recall-challenges-medical-devices>.
- [Has+19] Vikas Hassija et al. "A Survey on IoT Security: Application Areas, Security Threats, and Solution Architectures". In: *IEEE Access* 7 (2019), pp. 82721–82743.
- [Hat+12] John Hatcliff et al. "Rationale and architecture principles for medical application platforms". In: *Proceedings - 2012 IEEE/ACM 3rd International Conference on Cyber-Physical Systems* (2012), pp. 3–12.
- [HC10] Mike Hinchey and Lorcan Coyle. "Evolving critical systems: A research agenda for computer-based systems". In: *2010 17th IEEE International Conference and Workshops on Engineering of Computer Based Systems*. IEEE. 2010, pp. 430–435.
- [Hec+15] Myron Hecht et al. "Automated generation of failure modes and effects analysis for a medical device". In: (2015), pp. 29–32.
- [Hol+09] Oliver Holschke et al. "Using enterprise architecture models and bayesian belief networks for failure impact analysis". In: *ICSOC Workshops*. Springer. 2009, pp. 339–350.
- [Hou16] White House. *Presidential Policy Directive on United States Cyber Incident Coordination*. accessed on 05.06.21. 2016. URL: obamawhitehouse.archives.gov/the-press-office/2016/07/26/fact-sheet-presidential-policy-directive-united-states-cyber-incident-1.
- [HWM95] C. Haythornthwaite, B. Wellman, and M. Mantei. "Work relationships and media use: A social network analysis". In: *Group Decision and Negotiation* 4.3 (1995), pp. 193–211.
- [IB06] Frank Innerhofer-Oberperfler and Ruth Brey. "Using an Enterprise Architecture for IT Risk Management." In: *ISSA*. Citeseer. 2006, pp. 1–12.
- [IEC06] IEC. *Standard 62304, Medical device software — Software life cycle processes*. 2006.
- [IEC10] 61508 IEC. *Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems*. ICE, 2010.
- [IEC11] 80001-1:2010 IEC. *Application of risk management for IT-networks incorporating medical devices*. 2011.
- [IJ06] Maria-Eugenia Iacob and Henk Jonkers. "Quantitative analysis of enterprise architectures". In: *Interoperability of Enterprise Software and Applications*. Springer, 2006, pp. 239–252.
-

- [Inf] Department of Information Technology Maryland. *System Development Life Cycle (SDLC)*. accessed on 05.05.21. URL: <https://bit.ly/2Rve0pN>.
- [Ins] Johner Institut. *Software and IEC 62304, Medizingeräte-Software, Software-Lebenszyklus-Prozesse*. accessed on 08.05.21. URL: <https://www.johner-institut.de/blog/category/iec-62304-medizinische-software/>.
- [Ins18] Johner Institut. *V-Modell vs. Wasserfallmodell für Hardware- und Softwareentwicklung*. accessed on 05.05.21. 2018. URL: <https://www.johner-institut.de/blog/iec-62304-medizinische-software/v-modell/>.
- [Int] European Research Cluster on Internet of Things (IERC). *Internet of Things*. accessed on 13.05.21. URL: http://www.internet-of-things-research.eu/about_iot.htm.
- [IoT13] IoT-A. *Internet of Things Architecture*. 2013. URL: <https://cordis.europa.eu/project/id/257521/de>.
- [Isc17] Iscoop. "Internet of Things – the complete online guide to the IoT". In: (2017). accessed on 21.05.21. URL: <https://www.i-scoop.eu/internet-of-things-guide/>.
- [Isl+15] SM Riazul Islam et al. "The internet of things for health care: a comprehensive survey". In: *IEEE access* 3 (2015), pp. 678–708.
- [ISO] ISO. *ISO/IEC 29115:2013 Information technology — Security techniques — Entity authentication assurance framework*.
- [ISO09] ISO. *ISO 31000:2009(en) Risk management — Principles and guidelines*. 2009.
- [ISO11a] ISO. *ITU-T Recommendation X.1254 | International Standard ISO / IEC DIS 29115 Information technology — Security techniques — Entity authentication assurance framework*. 2011.
- [ISO11b] ISO/IEC/IEEE. *Systems and software engineering - architecture description*. 42010:2011 International Standard V1, 2011.
- [ISO15] ISO/IEC/IEEE. *15288:2015 Systems and software engineering — System life cycle processes*. 2015.
- [ISO16a] ISO. *13485: Medizinprodukte – Qualitätsmanagementsysteme – Anforderungen für regulatorische Zwecke*. 2016.
- [ISO16b] 30141:20160910 ISO/IEC. "Information technology – Internet of Things Reference Architecture (IoT RA)". In: (2016).
- [ISO20] ISO. *DIN EN ISO 14971:2020-07 - Medizinprodukte - Anwendung des Risikomanagements auf Medizinprodukte*. Beuth Publishing DIM. 2020.
- [ISO99] 51:1999 ISO/IEC. *Safety aspects — Guidelines for their inclusion in standards*. 1999.

-
- [IT-18] Kastl & Rieter IT-Service. *IT-Sicherheit – lästig oder unerlässlich?* accessed on 22.04.21. Feb. 2018. URL: <https://www.kastl-rieter.de/it-sicherheit-laestig-oder-unerlaesslich/>.
- [JI08] Henk Jonkers and Maria-Eugenia Iacob. “Performance and cost analysis of service-oriented enterprise architectures”. In: *Global implications of modern enterprise information systems: Technologies and applications* (2008).
- [JJM18] Maia Jacobs, Jeremy Johnson, and Elizabeth D. Mynatt. “MyPath: Investigating breast cancer patients’ use of personalized health information”. In: *Proceedings of the ACM on Human-Computer Interaction* 2.CSCW (2018), pp. 1–21.
- [JNL07] Pontus Johnson, Lars Nordström, and Robert Lagerström. “Formalizing Analysis of Enterprise Architecture”. In: *Enterprise Interoperability*. Springer London, 2007, pp. 35–44.
- [Joh+06] Pontus Johnson et al. “Extended influence diagrams for enterprise architecture analysis”. In: *Proceedings - IEEE International Enterprise Distributed Object Computing Workshop, EDOC* (2006), pp. 3–12.
- [Joh+07a] Pontus Johnson et al. “A tool for enterprise architecture analysis”. In: *11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007)*. IEEE. 2007, pp. 142–142.
- [Joh+07b] Pontus Johnson et al. “Enterprise architecture analysis with extended influence diagrams”. In: *Information Systems Frontiers* 9.2-3 (2007), pp. 163–180.
- [Joh05] Erik Johansson. “Assessment of enterprise information security: How to make it credible and efficient”. In: *KTH* (2005).
- [Jon+99] Henk Jonkers et al. *Completion time and critical path analysis for the optimisation of business process models*. 1999.
- [Jov+00] E. Jovanov et al. “Wireless personal area networks in telemedical environment”. In: *Proceedings 2000 IEEE EMBS International Conference on Information Technology Applications in Biomedicine*. 2000, pp. 22–27.
- [JS+99] Henk Jonkers, Marco van Swelm, et al. “Queueing analysis to support distributed system design”. In: *Proc. 1999 Symposium on Performance Evaluation of Computer and Telecommunication Systems*. Citeseer. 1999.
- [KA09] Stephan Kurpjuweit and Stephan Aier. “Ein allgemeiner Ansatz zur Ableitung von Abhängigkeitsanalysen auf Unternehmensarchitekturmodellen.” In: *Wirtschaftsinformatik* (1). 2009, pp. 129–138.
- [Kat+11] C.D. Katsis et al. “A wearable system for the affective monitoring of car racing drivers during simulated conditions”. In: *Transportation research part C: emerging technologies* 19.3 (2011), pp. 541–551.
-

- [Kaz+98] Rick Kazman et al. "The architecture tradeoff analysis method". In: *Engineering of Complex Computer Systems, 1998*. IEEE, 1998, pp. 68–78.
- [Kel07] W. Keller. "IT-Unternehmensarchitektur: Von der Geschäftsstrategie zur optimalen IT-Unterstützung. dpunkt". In: *Verl., Heidelberg 1* (2007).
- [Kha12] Rafiullah Khan. "Future Internet : The Internet of Things Architecture , Possible Applications and Key Challenges". In: (2012), pp. 257–260.
- [KMP11] Przemyslaw Kazienko, Radosław Michalski, and Sebastian Palus. "Social network analysis as a tool for improving enterprise architecture". In: *Agent and Multi-Agent Systems: Technologies and Applications*. Springer, 2011, pp. 651–660.
- [KN11] Timo Koski and John Noble. *Bayesian networks: an introduction*. Vol. 924. John Wiley & Sons, 2011.
- [Ko+10] Jeong Gil Ko et al. "Wireless sensor networks for healthcare". In: *Proceedings of the IEEE 98* 98.11 (2010), pp. 1947–1960.
- [Kol+13] Dimitrios S Kolovos et al. "A research roadmap towards achieving scalability in model driven engineering". In: *Proceedings of the Workshop on Scalability in Model Driven Engineering*. 2013, pp. 1–10.
- [KR09] David C Klonoff and Juliet S Reyes. *Insulin pump safety meeting: summary report*. 2009.
- [Krc15] Helmut Krcmar. *Informationsmanagement*. Gabler, 2015.
- [Kri+09] Bernd Krieg-Brückner et al. "Technik für Senioren in spe im Bremen Ambient Assisted Living Lab". In: *Ambient Assisted Living, 2. Deutscher AAL-Kongress 2009. Deutscher AAL-Kongress (AAL), January 27-28, Berlin, Germany* (2009).
- [Kur05] Ivan Kurtev. *Adaptability of model transformations*. Citeseer, 2005.
- [Lag+09] Robert Lagerström et al. "A method for creating enterprise architecture metamodels: applied to systems modifiability". In: *International Journal of Computer Science and Applications* 6.5 (2009), pp. 89–120.
- [Lag07] Robert Lagerström. "Analyzing system maintainability using enterprise architecture models". In: *Workshop on Trends in Enterprise Architecture Research*. 2007, pp. 31–39.
- [Lai+13] Xiaochen Lai et al. "A survey of body sensor networks". In: *Sensors* 13.5 (2013), pp. 5406–5447.
- [Lan05] Marc Lankhorst. *Enterprise Architecture at Work: Modelling, Communication and Analysis*. Springer, 2005.
- [Lan13] Marc Lankhorst. "A Language for Enterprise Modelling". In: *Enterprise Architecture at Work*. Springer, 2013, pp. 75–114.

- [Lan17] Einar Landre. "Safety & Security in Mission Critical IoT Systems". In: *13th Software Engineering Institute (SEI) Architecture Technology User Network (SATURN) Conference*. 2017.
- [LB09] Ying Tat Leung and Jesse Bockstedt. "Structural analysis of a business enterprise". In: *Service Science* 1.3 (2009), pp. 169–188.
- [Lee+16] Myeonggeon Lee et al. "Security threat on wearable services: Empirical study using a commercial smartband". In: *2016 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*. 2016, pp. 1–5.
- [Leg17] Legifrance. *Code of Public Health - Article L5211-4*. 2017. URL: <https://www.cabinetbarbey.com/blog-en/summary-of-characteristics-of-a-medical-device>.
- [LH94] Jaynarayan H. Lala and Richard E. Harper. "Architectural Principles for Safety-Critical Real-Time Applications". In: *Proceedings of the IEEE* 82.1 (1994), pp. 25–40.
- [LHJ08] Antti Lahtela, Marko Hassinen, and Virpi Jylha. "RFID and NFC in healthcare: Safety of hospitals medication care". In: *2008 Second International Conference on Pervasive Computing Technologies for Healthcare*. 2008, pp. 241–244.
- [Li17] Shancang Li. "IoT Node Authentication". In: *Securing the Internet of Things* (2017), pp. 69–95.
- [Liu17] Christina Liu. *Cyber Attacks and Cyber Security*. bookboon, 2017.
- [LJ08] Robert Lagerström and Pontus Johnson. "Using architectural models to predict the maintainability of enterprise systems". In: *Software Maintenance and Reengineering, 2008. CSMR 2008. 12th European Conference on*. IEEE. 2008, pp. 248–252.
- [LJE10] Robert Lagerström, Pontus Johnson, and Mathias Ekstedt. "Architecture analysis of enterprise systems modifiability: a metamodel for software change cost estimation". In: *Software quality journal* 18.4 (2010), pp. 437–468.
- [LJH10] Robert Lagerström, Pontus Johnson, and David Höök. "Architecture analysis of enterprise systems modifiability—models, analysis, and validation". In: *Journal of Systems and Software* 83.8 (2010), pp. 1387–1403.
- [LJN07] Robert Lagerström, Pontus Johnson, and Per Närman. "Extended influence diagram generation". In: *Enterprise Interoperability II*. Springer, 2007, pp. 599–602.
- [LL17] Wen-Tin Lee and Po-Jen Law. "A Case Study in Applying Security Design Patterns for IoT Software System". In: (2017), pp. 978–1.
- [LM14] Dan Ledger and Daniel McCaffrey. "Inside Wearables, How the Science of Human Behaviour Change Offers the Secret to Long-Term Engagement". In: *Endeavour Partners LLC* (2014), p. 10.

- [Loh20] Philipp Lohmüller. “Multi-Concerns Engineering for Safety-Critical Software Systems: Multi-Criteria Decision Making, Change Management and Variability”. In: (2020).
- [Lop+16] Antonio Lopez-Villegas et al. “A systematic review of economic evaluations of pacemaker telemonitoring systems”. In: *Revista Espanola de Cardiologia (English Edition)* 69.2 (2016), pp. 125–133.
- [Lop00] Marta Lopez. *An evaluation theory perspective of the Architecture Trade-off Analysis Method (ATAM)*. Tech. rep. DTIC Document, 2000.
- [LRB19] Philipp Lohmüller, Julia Rauscher, and Bernhard Bauer. “Failure and change impact analysis for safety-critical systems”. In: (2019), pp. 47–63.
- [LRJ11] Chunxiao Li, Anand Raghunathan, and Niraj K Jha. “Hijacking an insulin pump: Security attacks and defenses for a diabetes therapy system”. In: *2011 IEEE 13th International Conference on e-Health Networking, Applications and Services*. IEEE. 2011, pp. 150–156.
- [LSB14] Melanie Langermeier, Christian Saad, and Bernhard Bauer. “A unified framework for enterprise architecture analysis”. In: *18th IEEE International EDOC Conference Workshops*. 2014, pp. 227–236.
- [Luo+09] Jing Luo et al. “A methodology for analyzing availability weak points in SOA deployment frameworks”. In: *Network and Service Management, IEEE Transactions on* 6.1 (2009), pp. 31–44.
- [MA18] Daa Salama Abdul Minaam and Mohamed Abd-Elfattah. “Smart drugs: Improving healthcare using smart pill box for medicine reminder and monitoring system”. In: *Future Computing and Informatics Journal* 3.2 (2018), pp. 443–456.
- [Mar17] Niamh Marriott. *Why the Internet of Medical Things is the future of healthcare*. accessed on 07.05.21. 2017. URL: <https://bit.ly/2WZVRdf>.
- [Mar21] Peter Marwedel. *Embedded system design: embedded systems foundations of cyber-physical systems, and the internet of things*. Springer Nature, 2021.
- [Mat+12] Florian Matthes et al. “EAM KPI catalog v 1.0”. In: *Technical University Munich, Tech. Rep* (2012).
- [McG06] Gary R. McGraw. *Software Security - Building Security In*. Addison-Wesley Professional, 2006.
- [ME15] Farah Hussein Mohammed and Roslan Esmail. “Survey on IoT Services: Classifications and Applications”. In: *International Journal of Science and Research* 4.1 (2015), pp. 2124–2127.
- [Mem+14] Mukhtiar Memon et al. “Security modeling for service-oriented systems using security pattern refinement approach”. In: *Software and Systems Modeling* 13.2 (2014), pp. 549–572.

- [MHO11] Martin Meyer, Markus Helfert, and Conor O'Brien. "An analysis of enterprise architecture maturity frameworks". In: *Perspectives in Business Informatics Research*. Springer, 2011, pp. 167–177.
- [Mic07] Microsoft. *Microsoft STRIDE*. 03.05.21. 2007. URL: <https://bit.ly/37SKuWE>.
- [Mic16] Microsoft. *Microsoft Azure IoT Reference Architecture - VVersion 1.0*. 2016.
- [Mic18a] Microsoft. *Azure IoT Reference Architecture - Version 2.0*. 2018.
- [Mic18b] Microsoft. *Internet der Dinge – Sicherheitsarchitektur*. 2018. URL: <https://docs.microsoft.com/de-de/azure/iot-fundamentals/iot-security-architecture>.
- [Mit] CVE Mitre. *Common Vulnerabilities and Exposures Database*. accessed on 23.07.21. URL: <https://cve.mitre.org/>.
- [Mit+18] Jarernsri Mitranont et al. "I-WISH: Integrated Well-Being IoT System for Healthiness". In: *15th International Joint Conference on Computer Science and Software Engineering (JCSSE)*. 2018, pp. 1–6.
- [MJ16] Arsalan Mohsen Nia and Niraj K. Jha. "A Comprehensive Study of Security of Internet-of-Things". In: *IEEE Transactions on Emerging Topics in Computing* 5.4 (2016), pp. 1–1.
- [MLY05] Suvda Myagmar, Adam J Lee, and William Yurcik. "I-WISH:Threat modeling as a basis for security requirements". In: *Symposium on requirements engineering for information security (SREIS)*. Vol. 2005. Cite-seer. 2005, pp. 1–8.
- [MM20] Markets and Markets. *IoT in Healthcare Market by Component (Medical Device, Systems & Software, Services, and Connectivity Technology), Application (Telemedicine, Connected Imaging, and Inpatient Monitoring), End User, and Region - Global Forecast to 2025*. accessed on 21.04.21. June 2020. URL: <https://www.marketsandmarkets.com/Market-Reports/iot-healthcare-market-160082804.html>.
- [Mos17] Moschip. *How IoT-MD has the Potential to Improve Healthcare*. accessed on 23.04.21. 2017. URL: <https://moschip.com/blog/medical/how-iot-md-has-the-potential-to-improve-healthcare/>.
- [MSR+07] Peter Mell, Karen Scarfone, Sasha Romanosky, et al. "A complete guide to the common vulnerability scoring system version 2.0". In: *Published by FIRST-forum of incident response and security teams*. Vol. 1. 2007, p. 23.
- [MT10] Nenad Medvidovic and Richard N Taylor. "Software architecture: foundations, theory, and practice". In: *2010 ACM/IEEE 32nd International Conference on Software Engineering*. Vol. 2. IEEE. 2010, pp. 471–472.

- [Mur15] Sean Murphy. "Is cybersecurity possible in healthcare?" In: *National Cybersecurity Institute Journal* 1.3 (2015), pp. 49–63.
- [När+08] Per Närman et al. "Using enterprise architecture models for system quality analysis". In: *12th IEEE International EDOC Conference*. 2008, pp. 14–23.
- [När+09] Per Närman et al. "Enterprise architecture analysis for data accuracy assessments". In: *13th IEEE International EDOC Conference*. 2009, pp. 24–33.
- [När+10] Per Närman et al. "Hybrid Probabilistic Relational Models for System Quality Analysis". In: *IEEE International Enterprise Distributed Object Computing Conference (EDOC)*. 25-29 Oktober 2010, pp. 57–66.
- [När+11a] P Närman et al. "Enterprise architecture availability analysis using fault trees and stakeholder interviews". In: *Enterprise Information Systems* 8.1 (2011), pp. 1–25.
- [När+11b] Per Närman et al. "Data accuracy assessment using enterprise architecture". In: *Enterprise Information Systems* 5.1 (2011), pp. 37–58.
- [När+13] Per Närman et al. "Using enterprise architecture analysis and interview data to estimate service response time". In: *The Journal of Strategic Information Systems* 22.1 (2013), pp. 70–85.
- [NBE12] Per Närman, Markus Buschle, and Mathias Ekstedt. "An enterprise architecture framework for multi-attribute information systems analysis". In: *Software & Systems Modeling* 13.3 (2012), pp. 1085–1116.
- [Nea+04] Richard E Neapolitan et al. *Learning bayesian networks*. Vol. 38. Pearson Prentice Hall Upper Saddle River, NJ, 2004.
- [NFW17] Vidhyashree Nagaraju, Lance Fiondella, and Thierry Wandji. "A survey of fault and attack tree modeling and analysis for cyber risk management". In: *2017 IEEE International Symposium on Technologies for Homeland Security (HST)*. IEEE. 2017, pp. 1–6.
- [Nie06] Klaus D Niemann. *From enterprise architecture to IT governance*. Springer, 2006.
- [NIS12] NIST. "Information Security". In: *NIST SP 800-30 - NIST Technical Series Publications*. Ed. by U.S. Department of Commerce. 2012.
- [NJN07] P. Närman, P. Johnson, and L. Nordström. "Enterprise Architecture: A Framework Supporting System Quality Analysis". In: *11th IEEE International EDOC Conference*. 15-19 Oktober 2007, pp. 130–142.
- [NVD] NVD. *National Vulnerability Database*. accessed on 02.06.21. URL: <https://nvd.nist.gov/vuln>.
- [Oga+17] Shinpei Ogata et al. "A Tool to Edit and Verify IoT System Architecture Model". In: *MODELS (Satellite Events)*. 2017, pp. 571–575.
- [Ogg01] Chris Oggerino. *High availability network fundamentals*. Cisco Press, 2001.

-
- [Opea] OpenTelemetry. *OpenTelemetry*. accessed on 02.06.21. URL: <https://opentelemetry.io/>.
- [Opeb] Openxcell. *SDLC Design Phase*. accessed on 05.05.21. URL: <https://www.openxcell.com/blog/design-phase-in-sdlc/>.
- [OWA18] OWASP. *Internet of Things Top Ten 2018*. 2018.
- [Pal20] Paloalto Networks. *IoT Threat Report*. Accessed on 25.08.2020. 2020. URL: <https://unit42.paloaltonetworks.com/iot-threat-report-2020/>.
- [Par] Visual Paradigm. *What is a Software Process Model?* accessed on 04.05.21. URL: visual-paradigm.com/guide/software-development-process/what-is-a-software-process-model/.
- [Par+16] Youngseok Park et al. "This ain't your dose: Sensor spoofing attack on medical infusion pump". In: (2016).
- [Par20] European Parliament. *Medical Device Regulation MDR – Medizinprodukteverordnung (2017/745)*. 2020.
- [Pir+18] Gabriel Pires et al. "VITASENIOR-MT: a telehealth solution for the elderly focused on the interaction with TV". In: (2018), pp. 1–6.
- [Piw+16] Lukasz Piwek et al. "The rise of consumer health wearables: promises and barriers". In: *PLoS medicine* 13.2 (2016).
- [Pla16] (4.0) Platform Industrie. *Referenzarchitekturmodell Industrie 4.0 (RAMI 4.0)*. 2016.
- [PM18] Judea Pearl and Dana Mackenzie. *The book of why: the new science of cause and effect*. Basic books, 2018.
- [PR13] Roberto Pietrantuono and Stefano Russo. "Introduction to Safety Critical Systems". In: *Innovative Technologies for Dependable OTS-Based Critical Systems*. Springer, 2013, pp. 17–27.
- [Pro15] IoT-A EU Project. "IoT-A - final project report". In: - September 2012 (2015). <https://cordis.europa.eu/project/id/257521/de>.
- [PSP12] Henk Plessius, Raymond Slot, and Leo Pruijt. "On the categorization and measurability of enterprise architecture benefits with the enterprise architecture value framework". In: *Trends in Enterprise Architecture Research and Practice-Driven Research on Enterprise Transformation*. Springer, 2012, pp. 79–92.
- [PV08] Vladimir Popović and Branko Vasić. "Review of hazard analysis methods and their basic characteristics". In: *FME Transactions* 36.4 (2008), pp. 181–187.
- [qbi] qbilon. *qbilon GmbH*. URL: <https://www.qbilon.io/>.
- [RAB11] Mahsa Razavi, Fereidoon Shams Aliee, and Kambiz Badie. "An AHP-based approach toward enterprise architecture analysis based on enterprise architecture quality attributes". In: *Knowledge and information systems* 28.2 (2011), pp. 449–472.
-

- [RAC17] Mashfiqui Rabbi, Min Hane Aung, and Tanzeem Choudhury. "Towards health recommendation systems: an approach for providing automated personalized health feedback from mobile data". In: *Mobile Health*. Springer, 2017, pp. 519–542.
- [Rap] Rapid7. *Vulnerability & Exploit Database Rapid*. URL: <https://www.rapid7.com/de/db/>.
- [Rat+19] Annanda Rath et al. "Security pattern for cloud SaaS: From system and data security to privacy case study in AWS and azure". In: *Computers* 8.2 (2019).
- [Rau13] Julia Rauscher. "Analysen in Unternehmensarchitekturen - Ziele, Techniken, Anwendungsbereiche". In: *Bachelor Thesis, University Augsburg* (2013).
- [Rau15] Julia Rauscher. "Anforderungen an und Definition von einer Analysensprache für das Enterprise Architecture Management". In: *Master Thesis, University Augsburg* (2015).
- [RB17] Julia Rauscher and Bernhard Bauer. "Smart Data Integration within a Wellbeing Application Platform". In: *4th IEEE International Conference on Biomedical and Health Informatics* 4953 (2017), p. 4953.
- [RB18] Julia Rauscher and Bernhard Bauer. "Safety and security architecture analyses framework for the internet of things of medical devices". In: *2018 IEEE 20th International Conference on e-Health Networking, Applications and Services, Healthcom 2018* (2018), pp. 3–5.
- [RB20] Julia Rauscher and Bernhard Bauer. *Design optimization of IoT models: structured safety and security flaw identification*. 2020, pp. 84–102.
- [RB21] Julia Rauscher and Bernhard Bauer. "Adaptation of Architecture Analyses : An IoT Safety and Security Flaw Assessment Approach". In: *HealthINF* (2021).
- [RBL16] Julia Rauscher, Bernhard Bauer, and Melanie Langermeier. "Characteristics of Enterprise Architecture Analyses". In: *Proceedings of the Sixth International Symposium on Business Modeling and Software Design* (2016), pp. 104–113.
- [Res] Connected Futures Cisco Research. *IoT Value: Challenges, Breakthroughs, and Best Practices*. accessed on 23.04.21. URL: <https://www.slideshare.net/CiscoBusinessInsights/journey-to-iot-value-76163389>.
- [RLB16] Julia Rauscher, Melanie Langermeier, and Bernhard Bauer. "Classification and Definition of an Enterprise Architecture Analyses Language". In: *International Symposium on Business Modeling and Software Design* (2016), pp. 1–20.
- [RNE09] Jakob Raderius, Per Närman, and Mathias Ekstedt. "Assessing system availability using an enterprise architecture analysis approach". In: *Service-Oriented Computing–ICSOC 2008 Workshops*. Springer. 2009, pp. 351–362.

-
- [Roo94] M.A. Rood. "Enterprise architecture: definition, content, and utility". In: *Proceedings of 3rd IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*. 1994, pp. 106–111.
- [RV16] Brian Russell and Drew Van Duren. *Practical internet of things security*. Packt Publishing Ltd, 2016.
- [SA14] Alberto M. C. Souza and José R. A. Amazonas. "A Novel IoT Architecture with Pattern Recognition Mechanism and Big Data". In: *Journal of Machine to Machine Communications* 1.3 (2014), pp. 245–272.
- [Saa10a] Jan Saat. *Planung der Unternehmensarchitektur: Vorgehen - Gestaltungsgegenstand - Alternativenbewertung*. Logos Verlag Berlin GmbH, 2010.
- [Saa10b] Jan Saat. "Zeitbezogene Abhängigkeitsanalysen der Unternehmensarchitektur". In: *MKWI*. 2010, pp. 119–130.
- [SAB20] SABSA. *T100 – Modelling SABSA with ArchiMate*. accessed on 16.08.21. 2020. URL: <https://sabsa.org/white-paper-requests/>.
- [Sar16] S. Sarkar. "Internet of Things—Robustness and reliability". In: *Internet of Things*. Elsevier, 2016, pp. 201–218.
- [SB15] Mark Stanislav and Tod Beardsley. "Hacking IoT : A Case Study on Baby Monitor Exposures and Vulnerabilities". In: *IOTSec* (2015). accessed on 03.06.21. URL: rapid7.com/docs/Hacking-IoT-A-Case-Study-on-Baby-Monitor-Exposures-and-Vulnerabilities.pdf.
- [SC01] Nary Subramanian and Lawrence Chung. "Metrics for software adaptability". In: *Proc. Software Quality Management (SQM 2001)* 158 (2001).
- [SC16] Nigel Stanley and Mark Coderre. "An introduction to medical device cyber security - A European Perspective". In: *Healthcare Information and Management Systems Society* October (2016).
- [Sch+13] Markus Schumacher et al. *Security Patterns: Integrating security and systems engineering*. John Wiley & Sons, 2013.
- [Sch96] Helmut Schneider. *Failure Mode and Effect Analysis: FMEA From Theory to Execution*. 1996.
- [Sec20] SecurityToday. *The IoT Rundown For 2020: Stats, Risks, and Solutions*. accessed on 01.03.21. 2020. URL: <https://bit.ly/3BTF6c1>.
- [SEJ08] Teodor Sommestad, Mathias Ekstedt, and Pontus Johnson. "Combining defense graphs and enterprise architecture models for security analysis". In: *Proceedings - 12th IEEE International Enterprise Distributed Object Computing Conference, EDOC 2008* (2008), pp. 349–355.
- [Sen21] Rebecca Sentance. *7 examples of how the internet of things is facilitating healthcare*. Econsultancy. accessed on 07.05.21. 2021. URL: <https://econsultancy.com/internet-of-things-healthcare/>.
- [Ses15] Sesamo. *SESAMO Project Homepage*. <http://sesamo-project.eu/>. accessed 24.07.2021. 2015.
-

- [Shi+08] Bingu Shim et al. "A design quality model for service-oriented architecture". In: *2008 15th Asia-Pacific Software Engineering Conference*. IEEE. 2008, pp. 403–410.
- [Sil+17] Jonathan De Carvalho Silva et al. "IoT Network Management: Content and Analysis". In: *IoT Network Management (2017)*, pp. 821–825.
- [Sim21] SimpliLearn. *TOGAF® and the Internet of Things*. accessed on 29.04.21, 2021. URL: <https://www.simplilearn.com/togaf-applications-in-internet-of-things-iot-article>.
- [Sir] Eclipse Sirius. *Sirius*. accessed on 24.04.21. URL: <https://www.eclipse.org/sirius/>.
- [SK11] Ana Sasa and Marjan Krisper. "Enterprise architecture patterns for business process support analysis". In: *Journal of Systems and Software* 84.9 (2011), pp. 1480–1506.
- [SK16] Devon Scott and Mohammed Ketel. "Internet of Things: A useful innovation or security nightmare?" In: *SoutheastCon 2016*. IEEE. 2016, pp. 1–6.
- [SKR13] Sagar Sunkle, Vinay Kulkarni, and Suman Roychoudhury. "Analyzable Enterprise Models Using Ontology." In: *CAiSE Forum*. Vol. 998. 2013, pp. 33–40.
- [SMP18] Saleh Soltan, Prateek Mittal, and H Vincent Poor. "BlackIoT: IoT Botnet of High Wattage Devices Can Disrupt the Power Grid". In: *27th {USENIX} Security Symposium ({USENIX} Security 18)* (2018), pp. 15–32.
- [Sne] Gregor Snelting. *JOANA (Java Object-sensitive ANALysis) - Information Flow Control Framework for Java*. accessed on 23.07.21. URL: <https://pp.ipd.kit.edu/projects/joana/>.
- [Som11] Ian Sommerville. *Software engineering 9th Edition*. Addison-Wesley, 2011, p. 18.
- [Son] SonarSource. *SonarQube*. <https://www.sonarqube.org/>. accessed on 24.07.2021.
- [Soo+07] Sanjay Sood et al. "What is telemedicine? A collection of 104 peer-reviewed perspectives and theoretical underpinnings". In: *Telemedicine and e-Health* 13.5 (2007), pp. 573–590.
- [Spa11] John Spacey. *Risk vs. Return for IT Investments*. accessed on 03.07.21. Apr. 2011. URL: <http://simplicable.com/new/risk-vs-return-for-IT-investments>.
- [Spa17] John Spacey. *IoT vs Embedded Systems*. accessed on 11.05.21. 2017. URL: <https://simplicable.com/new/iot-vs-embedded-systems>.
- [SR90] Dorothy E. Setliff and Rob A. Rutenbar. "Software Reusability". In: *Automatic Programming Applied to VLSI CAD Software: A Case Study*. Springer, 1990, pp. 33–42.

-
- [SR95] Alexander K Schömig and Harald Rau. *A petri net approach for the performance analysis of business processes*. 1995.
- [SS17] Pallavi Sethi and Smruti R Sarangi. "Internet of things: architectures, protocols, and applications". In: *Journal of Electrical and Computer Engineering* 2017 (2017).
- [SSH+15] Jinsoo Shin, Hanseong Son, Gyunyoung Heo, et al. "Development of a cyber security risk model using Bayesian networks". In: *Reliability Engineering & System Safety* 134 (2015), pp. 208–217.
- [Sta08] John A. Stankovic. "Wireless Sensor Networks". In: *Computer* 41.10 (2008), pp. 92–95.
- [Ste+10] Marlies van Steenbergen et al. "The dynamic architecture maturity matrix: Instrument analysis and refinement". In: *Service-Oriented Computing. ICSOC/ServiceWave 2009 Workshops*. Springer. 2010, pp. 48–61.
- [Szy09] B. Szyszka. "Analysis and classification of maturity models in enterprise architecture management". In: *Bachelor Thesis, Technical University Munich* (2009).
- [Szy17] Ted H. Szymanski. "Security and privacy for a green internet of things". In: *IT Professional* 19.5 (2017), pp. 34–41.
- [Tao+19] Fei Tao et al. "Digital Twins and Cyber-Physical Systems toward Smart Manufacturing and Industry 4.0: Correlation and Comparison". In: *Engineering* 5.4 (2019), pp. 653–661.
- [Tas02] Gregory Tasey. *The Economic Impacts of Inadequate Infrastructure for Software Testing*. National Institute of Standards and Technology Acquisition and Assistance Division. 2002.
- [Tec17] Techopedia. *What Does Scalability Mean?* Accessed on 11.08.21, 2017. URL: <https://www.techopedia.com/definition/9269/scalability>.
- [Thi+18] Montbel Thibaud et al. "Internet of Things (IoT) in high-risk Environment, Health and Safety (EHS) industries: A comprehensive review". In: *Decision Support Systems* 108 (2018), pp. 79–95.
- [Thi19] IoT Architecture - Internet of Things (IoT) Architecture. "Standard for an architectural framework for the Internet of Things (IoT) - IEEE P2413". In: *Proceedings of the International Instrumentation Symposium* (2019), pp. 67–75.
- [Tok+17] C Arcadius Tokognon et al. "Structural health monitoring framework based on Internet of Things: A survey". In: *IEEE Internet of Things Journal* 4.3 (2017), pp. 619–635.
- [TTF79] Noel M. Tichy, Michael L. Tushman, and Charles Fombrun. "Social network analysis for organizations". In: *Academy of management review* 4.4 (1979), pp. 507–519.
-

- [TW14] Gorica Tapandjieva and Alain Wegmann. "Specification and implementation of a meta-model for information systems cartography". In: *Forum at the 26th International Conference on Advanced Information Systems Engineering (CAiSE)*. 2014, pp. 113–120.
- [Ula17] David Ulander. *Software Architectural Metrics for the Scania Internet of Things Platform From a Microservice Perspective of Things Platform - From a Microservice Perspective*. 2017.
- [ULJ08] Johan Ullberg, Robert Lagerström, and Pontus Johnson. "A framework for service interoperability analysis using enterprise architecture models". In: *Proceedings - 2008 IEEE International Conference on Services Computing, SCC 2008 2* (2008), pp. 99–107.
- [Ull+08] Sana Ullah et al. "A study of implanted and wearable body sensor networks". In: *Proceedings of the 2Nd KES International Conference on Agent and Multi-agent Systems: Technologies and Applications* (2008), pp. 464–473. arXiv: 0911.1546.
- [Ull+10] Johan Ullberg et al. "A tool for interoperability analysis of enterprise architecture models using pi-OCL". In: *Enterprise Interoperability IV*. Springer, 2010, pp. 81–90.
- [US 07a] U.S. Department of Health and Human Services. "HIPAA Security series: 1 Security 101 for Covered Entities". In: *Centers for Medicare & Medicaid Services 2* (2007), pp. 1–11.
- [US 07b] U.S. Department of Health and Human Services. "HIPAA Security series: 6 Risk Assessment". In: *Centers for Medicare & Medicaid Services 2* (2007).
- [Van01] Axel Van Lamsweerde. "Goal-oriented requirements engineering: A guided tour". In: *5th IEEE International Symp. on Requirements Engineering*. 2001, pp. 249–262.
- [Var] Vario. *Industrie 4.0*. accessed on 21.04.21. URL: <https://www.vario-software.de/lexikon/industrie-4-0/>.
- [Var+17] Pal Varga et al. "Security threats and issues in automation IoT". In: *IEEE International Workshop on Factory Communication Systems - Proceedings, WFCS* (2017), pp. 1–6.
- [VCS06] Marcos A.M. Vieira, Adriano B. da Cunha, and Diógenes C. da Silva. "Designing wireless sensor nodes". In: *International Workshop on Embedded Computer Systems*. Springer. 2006, pp. 99–108.
- [VM01] John Viega and Gary R McGraw. *Building secure software: How to avoid security problems the right way*. Pearson Education, 2001.
- [Voc12] Martha Vockley. "Safe and secure? Healthcare in the Cyberworld". In: *Biomedical instrumentation & technology* 46.3 (2012), pp. 164–173.

- [VS00] Ariën J Van Der Wal and Ming Shao. "Sensor fusion: the application of soft computing in monitoring and control for railroad maintenance". In: *Proceedings of the 3rd World Congress on Intelligent Control and Automation (Cat. No. 00EX393)*. Vol. 1. IEEE. 2000, pp. 341–346.
- [VSP08] Jose Luis de la Vara, Juan Sanchez, and Oscar Pastor. "Business process modelling and purpose analysis for requirements analysis of information systems". In: *Advanced Information Systems Engineering*. Springer. 2008, pp. 213–227.
- [WAF17] Daniel Wood, Noah Apthorpe, and Nick Feamster. "Cleartext data transmissions in consumer iot medical devices". In: (2017), pp. 7–12.
- [WC12] Haining Wan and Sven Carlsson. "Towards an Understanding of Enterprise Architecture Analysis Activities". In: *Proceedings of 6th EC-IME*. 2012, p. 334.
- [WF06] Robert Winter and Ronny Fischer. "Essential layers, artifacts, and dependencies of enterprise architecture". In: *10th IEEE International EDOC Conference Workshops*. 2006, pp. 30–33.
- [Wil19] Marc Wilczek. *Cyber-Angriffe haben sich mehr als verdoppelt*. Computerwoche. accessed on 22.04.21, 2019. URL: [computerwoche.de/a/cyber-angriffe-haben-sich-mehr-als-verdoppelt](https://www.computerwoche.de/a/cyber-angriffe-haben-sich-mehr-als-verdoppelt), 3546370.
- [Win+09] R. Winter et al. "Patterns in der Wirtschaftsinformatik". In: *Wirtschaftsinformatik* 51.6 (2009), pp. 535–542.
- [Wir] Gabler Wirtschaftslexikon. *Definition Ziel*. accessed on 27.04.21. URL: <https://wirtschaftslexikon.gabler.de/definition/ziel-49980/version-182026>.
- [WK01] Dolores R. Wallace and Richard D. Kuhn. "Failure Modes in Medical Device Software: an Analysis of 15 Years of Recall Data". In: *International Journal of Reliability, Quality and Safety Engineering* 08.04 (2001), pp. 351–371.
- [Wu+10] Miao Wu et al. "Research on the architecture of Internet of Things". In: 5 (2010), pp. V5–484.
- [Xie+08] Lei Xie et al. "Availability weak point analysis over an SOA deployment framework". In: *IEEE Network Operations and Management Symposium*. 2008, pp. 473–480.
- [Xtea] Eclipse Xtend. *Xtend/Xtext*. accessed on 12.06,21. URL: <https://bit.ly/2HQcaIc>.
- [Xteb] Eclipse Xtext. *Xtext*. accessed on 12.06,21. URL: <https://www.eclipse.org/Xtext/>.
- [Yen09] Vincent C. Yen. "An integrated model for business process measurement". In: *Business Process Management Journal* 15.6 (2009), pp. 865–875.

- [YM95] Eric Yu and John Mylopoulos. "From ER To "AR"—Modelling Strategic Actor Relationships for Business Process Reengineering". In: *International Journal of Cooperative Information Systems* (1995), pp. 125–144.
- [YSD06] Eric Yu, Markus Strohmaier, and Xiaoxue Deng. "Exploring intentional modeling and analysis for enterprise architecture". In: *Enterprise Distributed Object Computing Conference Workshops*. IEEE. 2006, pp. 32–32.
- [Yu11] Eric Yu. "Modeling Strategic Relationships for Process Reengineering." In: *Social Modeling for Requirements Engineering* 11.2011 (2011), pp. 66–87.
- [ZAA11] Mehmooda Jabeen Zia, Farooque Azam, and Maria Allauddin. "A Survey of Enterprise Architecture Analysis Using Multi Criteria Decision Making Models". In: *Intelligent Computing and Information Science*. Springer, 2011, pp. 631–637.
- [Zac00] Michael H Zack. "Researching organizational systems using social network analysis". In: *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*. IEEE. 2000, p. 7.
- [Zal19] Janusz Zalewski. "IoT Safety: State of the Art". In: *IT Professional* 21.1 (2019), pp. 16–20.
- [Zan+14] Marco Zanoni et al. "Pattern detection for conceptual schema recovery in data-intensive systems". In: *Journal of Software: Evolution and Process* 26.12 (2014), pp. 1172–1192.
- [ZG97] Jianying Zhou and D. Gollmann. "An efficient non-repudiation protocol". In: *Proceedings 10th Computer Security Foundations Workshop*. 1997, pp. 126–132.
- [ZHN11] M. Zhang, T. Hall, and Baddoo N. "Code Bad Smells: a review of current knowledge". In: *J. Softw. Maint. Evol. Res. Pract.* 23.3, pp. 179–202 (2011).
- [Zhu+05] Liming Zhu et al. "Tradeoff and sensitivity analysis in software architecture evaluation using analytic hierarchy process". In: *Software Quality Journal* 13.4 (2005), pp. 357–375.
- [Zim+15] Alfred Zimmermann et al. "Enterprise Architecture Management for the Internet of Things". In: *Digital Enterprise Computing2015, Lecture Notes in Informatics (LNI), Gesellschaft für Informatik* (2015), pp. 139–150.
- [Zim+18] Alfred Zimmermann et al. "Evolution of enterprise architecture for digital transformation". In: (2018), pp. 87–96.

List of Acronyms

AAL	Ambient Assisted Living
AHP	Analytic Hierarchy Process
ASLE	Average Single Loss Expectancy
BBN	Bayesian Belief Network
BSI	Bundesamt für Sicherheit in der Informationstechnik
BSN	Body Sensor Networks
CDSA	Countermeasure Decision Support Analysis
CP	Conditional Probability
CPT	Conditional Probability Table
DOS	Denial of Service
DSL	Domain Specific Language
EA	Enterprise Architecture
EAM	Enterprise Architecture Management
ELE	Element Loss Expectancy
FIA	Failure Impact Analysis
EID	Extended Influence Diagram
IoT	Internet of Things
IoT-WD	Internet of Things of Wellbeing Devices
JPT	Joint Probability Table
JP	Joint Probability
KPI	Key Performance Indicators
LLE	Layered Loss Expectancy
LRO	Layered Rate of Occurrence
MAP	Medical Application Platform
NFC	Near Field Communication
PAN	Personal Area Network
PII	Personally Identifiable Information
PRF	Pattern Recognition Framework
PRM	Probabilistic Relational Model
PUF	Physical Unclonable Function
QIA	Quantitative Impact Analysis
RFID	Radio Frequency Identification
RO	Rate of Occurrence
SAP	Safety Pattern
SAAP	Safety Anti-Pattern
SEP	Security Pattern
SEAP	Security Anti-Pattern
SIA	Service Interoperability Analysis
SIL	Safety Integrity Level
SLA	Service Level Agreement
TLE	Total Loss Expectancy

TSE	Total Severity Expectancy
UICC	Universal Integrated Circuit Card
WAP	Wellbeing Application Platform
WSN	Wireless Sensor Network

List of Figures

1.1.	Historic development of industry ([ER19])	4
1.2.	Number of installed IoT devices around the world [Ant21]	5
1.3.	IoT management lifecycle	8
1.4.	Challenges in IoT management lifecycle	9
1.5.	Objectives in IoT management lifecycle	12
1.6.	Approach stack	16
1.7.	Outline of this thesis	18
2.1.	Context of architecture description [ISO11b]	29
2.2.	Waterfall model of project activities [Par]	31
2.3.	V-model of development process [Par]	32
2.4.	IoT reference architecture [Fre15]	38
2.5.	IoT reference architecture [SS17]	39
2.6.	IoT system architecture with WSN [Tok+17]	41
2.7.	IoT data lifecycle and management [AHA13]	44
2.8.	JDCF components [Bec+17]	52
2.9.	Different types of uncertainty [Buc+11]	62
2.10.	Four enterprise architecture layer	62
2.11.	Example of Bayesian Belief Network [Nea+04]	66
2.12.	Generic influence diagram notation [Joh+07b]	67
2.13.	CPT of two nodes of an EID [Joh+07b]	67
2.14.	Node collapse [Joh+06]	68
4.1.	Layered architecture for IoT Systems	88
4.2.	Example of a formalization conform meta model (MM)	94
4.3.	Example of a formalization conform IoT-WD model (WM)	95
4.4.	Meta model part general elements	96
4.5.	Meta model part services and physical connections	97
4.6.	Meta model part physical entities	99
4.7.	Meta model part measurements and storages	101
4.8.	Meta model part gateways and clouds	102
4.9.	Meta model part stakeholders and users	103
4.10.	Exemplary usage of the ArchiAna model editor to model an AAL example	105
4.11.	Exemplary service-physical connection depiction with ArchiAna	106
4.12.	Extract of list of available filter in IoT-WD systems	106
4.13.	Exemplary usage of the ArchiAna model editor to model an AAL example with container view	107
5.1.	Security management process of [IB06]	116
5.2.	Optimization process workflow	118

6.1.	Concept and workflow of flaw identification with PRF	123
6.2.	Example of formalization conform generic and instance specific pattern/anti-pattern structure	126
6.3.	Identified flaw in AAL running example	147
7.1.	Overview EA analysis types	156
7.2.	Categorization methodology	157
7.3.	Dependency matrix of the functional and technical categories . . .	160
7.4.	Concept	176
7.5.	FIA meta model	179
7.6.	FIA causal node types	181
7.7.	FIA example	183
7.8.	QIA meta model	186
7.9.	QIA example	189
7.10.	CDSA meta model	191
7.11.	CDSA example	194
7.12.	SIA meta model	196
7.13.	SIA example	198
8.1.	Evaluation concept	207
8.2.	Pairwise percentage comparison of congruence	214
8.3.	Concept of AAL functions	215
8.4.	AAL case study system model - IoT diagram overview	216
8.5.	AAL case study system model - things layer filter	217
8.6.	AAL case study system model - IoT container diagram	217
8.7.	AAL case study pattern and anti-pattern DSL	221
8.8.	AAL case study pattern and anti-pattern Services	222
8.9.	AAL case study flaw identification - SAP	222
8.10.	AAL case study flaw identification - SEAP	223
8.11.	AAL case study flaw assessment - FIA	224
8.12.	AAL case study flaw assessment - QIA	226
8.13.	AAL case study flaw assessment - CDSA	228
8.14.	AAL case study flaw assessment - SIA	230
8.15.	Performance use case system model	231
8.16.	Performance test of model editor	232
8.17.	Performance test of flaw identification - model view	234
8.18.	Performance test of flaw identification - container view	235
8.19.	Performance use case - analyses cycle	235
8.20.	Performance comparison of FIA	236
8.21.	Performance comparison of QIA	237
8.22.	Performance comparison of CDSA	237
8.23.	Performance comparison of SIA	238
A.1.	All enumerations of meta model part 1	296
A.2.	All enumerations of meta model part 2	297
A.3.	Complete meta model	298

A.4. Overview of meta model elements illustration 299
A.5. Overview categorization of analyses types into functional categories 301
A.6. Overview categorization of analyses types into technical categories 303

List of Tables

3.1.	Generic architectural requirements	83
3.2.	Component architectural requirements	83
3.3.	Software, service and data architectural requirements	84
3.4.	User and authentication architectural requirements	85
3.5.	Communication and encryption architectural requirements	85
3.6.	Assessment architectural requirements	86
4.1.	Architectural requirements for IoT(-WD) meta model part 1	92
4.2.	Architectural requirements for IoT(-WD) meta model part 2	92
4.3.	Meta model part general elements	97
4.4.	Meta model part services and physical connections part 1	98
4.5.	Meta model part services and physical connections part 2	98
4.6.	Meta model part physical entities part 1	99
4.7.	Meta model part physical entities part 2	100
4.8.	Meta model part physical entities part 3	100
4.9.	Meta model part measurement and storage	101
4.10.	Meta model part gateways and clouds part 1	102
4.11.	Meta model part gateways and clouds part 2	102
4.12.	Meta model part stakeholders and users	103
6.1.	Composition of PRF templates structure	127
6.2.	PRF Generic Part	128
6.3.	AAL example Generic Part	129
6.4.	PRF Security Challenge Part	130
6.5.	PRF Safety Challenge Part	130
6.6.	AAL example Security Challenge Part	131
6.7.	PRF Security Assessment Information Part	132
6.8.	PRF Safety Assessment Information Part	133
6.9.	AAL example Security Assessment Information Part	133
6.10.	PRF Pattern Implementation Part	134
6.11.	PRF Anti-Pattern Implementation Part	135
6.12.	AAL example Pattern Implementation Part	136
7.1.	Analyses characteristics overview	176
7.2.	CPT example of node C as collider	181
7.3.	Probability values and weights of AAL example - CDSA	194
7.4.	Probability values of AAL example - SIA	199
7.5.	Interpretation of the RPN [Bun17]	202
8.1.	Comparison of approach with related work of [Mic18a]	208
8.2.	Comparison of approach with related work of [Bau+13]	209
8.3.	Comparison of approach with related work of [ISO16b]	210

8.4. Comparison of approach with related work of [Cic+17]	211
8.5. Comparison of approach with related work of [Dig17]	211
8.6. Comparison of approach with related work of [ME15]	212
8.7. Comparison of approach with related work of [SAB20]	213
8.8. AAL case study Safety Pattern	219
8.9. AAL case study Security Anti-Pattern	220
8.10. FIA single probabilities - things layer	225
8.11. CPT of Rule_SHInsulinPump	225
8.12. Probability values and weights of CM Scenario 2	229
8.13. Probability values of SIA Scenario 2	229
8.14. AAL Safety Anti-Pattern	233

Listings

6.1.	PRF DSL generic and components rule	137
6.2.	PRF DSL implementation pattern rules	138
6.3.	PRF DSL element rules	138
6.4.	Running example excerpt of Xtext	139
6.5.	Algorithm flaw identification process	141
6.6.	Xtend excerpt of code generation - filter check	143
6.7.	Xtend excerpt of code generation - filter check middle part	144
6.8.	Xtend excerpt of code generation - filter check attributes	145
6.9.	Sirius service Java code excerpt - filter	145
6.10.	Sirius service Java code excerpt - requirement	146
7.1.	DSL for EA analyses - main rule	168
7.2.	Excerpt of the description of Bayesian Network analysis using the DSL	170
7.3.	Excerpt of the description of EID analysis using the DSL	171

A

Appendix

A.1. Meta Model and Enumerations

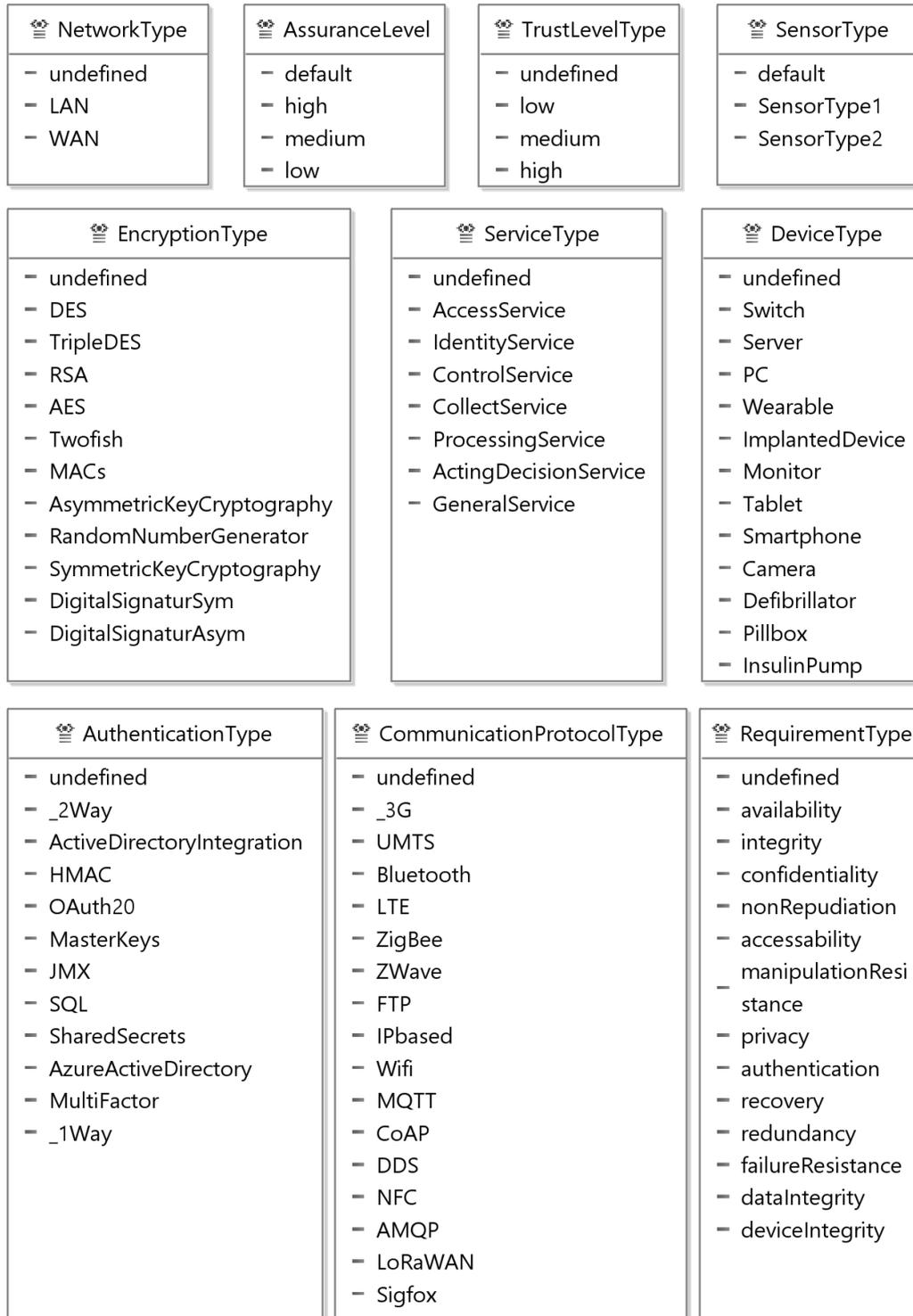


Figure A.1.: All enumerations of meta model part 1

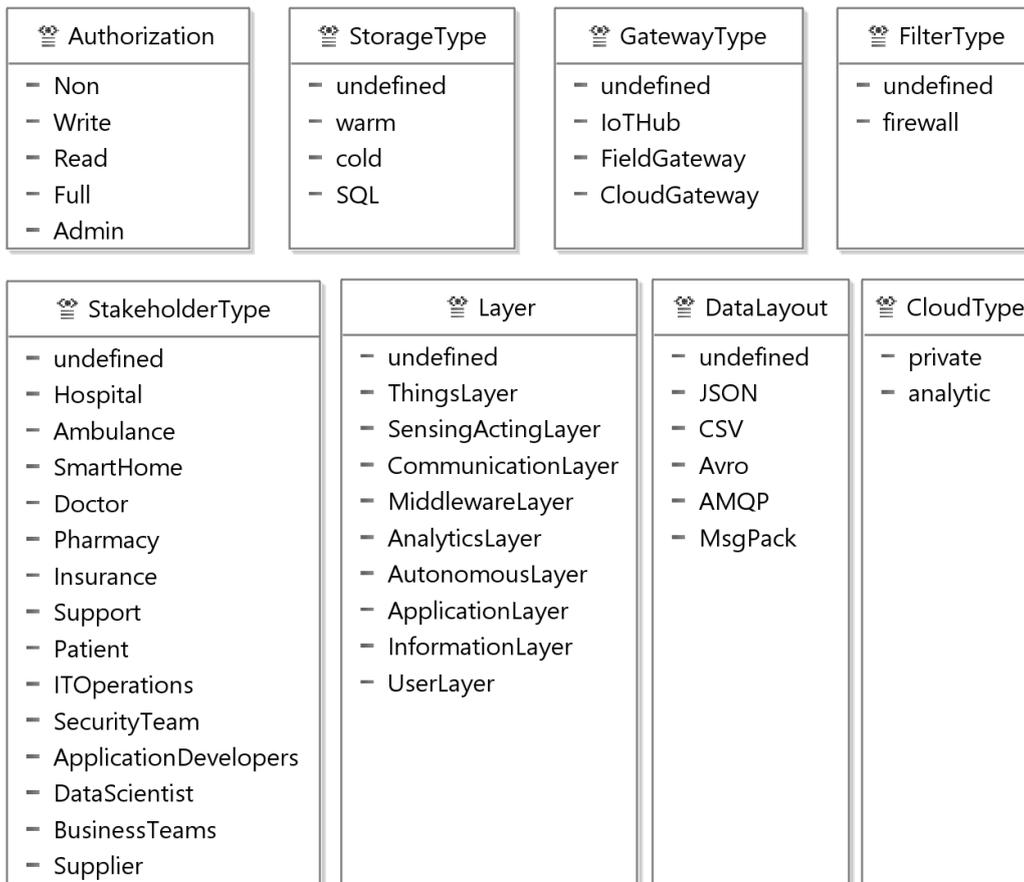


Figure A.2.: All enumerations of meta model part 2

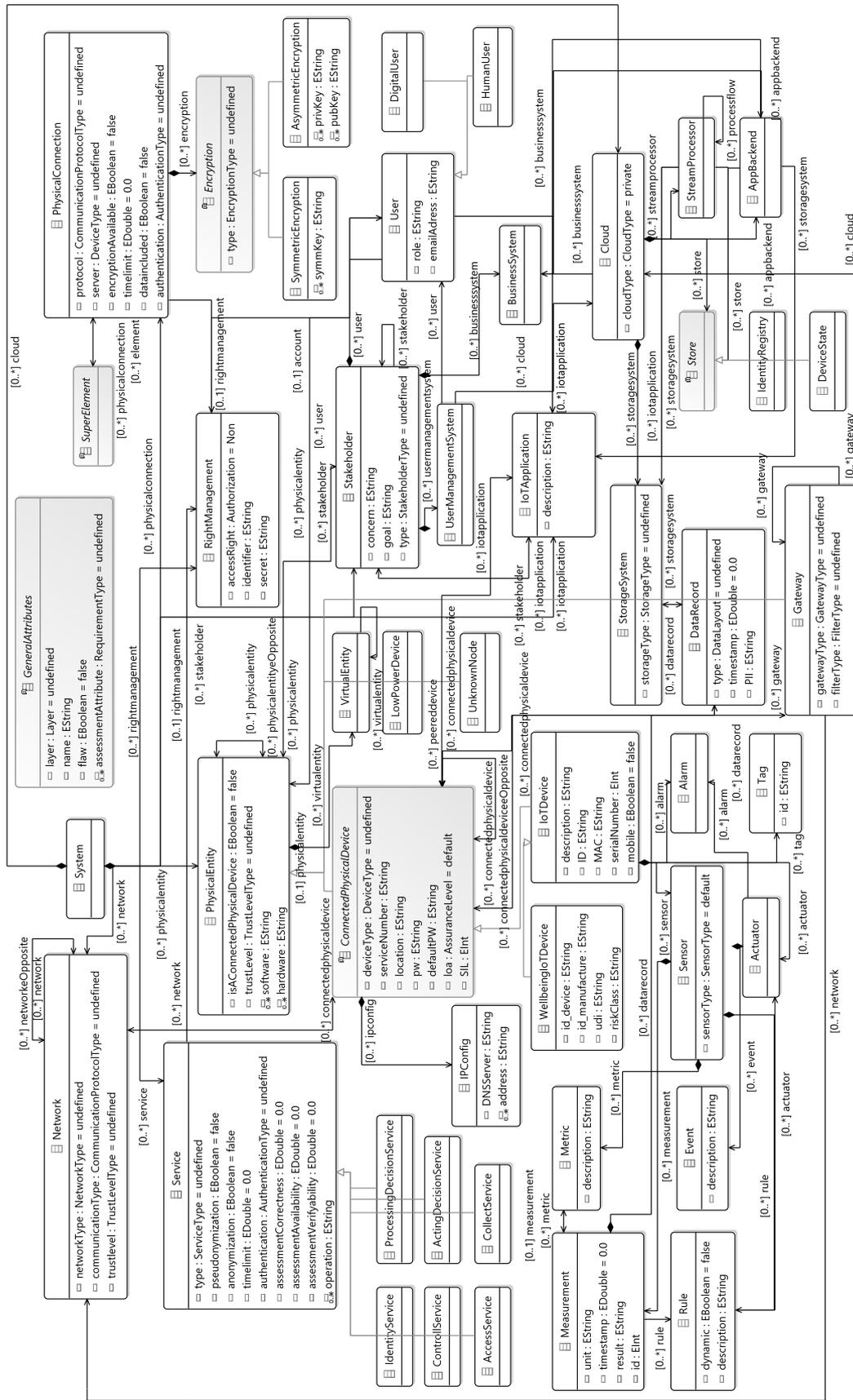


Figure A.3.: Complete meta model

A.2. Model Editor Elements

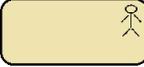
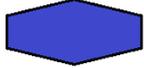
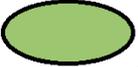
Meta Model Element	Illustration	Meta Model Element	Illustration
Actuator		Network	
Alarm		PhysicalConnection	
AppBackend		PhysicalEntity	
AsymmetricEncryption		RightManagement	
BusinessSystem		Rule	
Cloud		Sensor	
DataRecord		Service (all types)	
DigitalUser		Stakeholder	
Event		StorageSystem	
Gateway		StreamProcessor	
HumanUser		SymmetricEncryption	
IdentityRegistry		System	
IoTApplication		Tag	
IoTDevice		UnknownNode	
IPConfig		User	
LowPowerDevice		UserManagementSystem	
Measurement		VirtualEntity	
Metric		WellbeingIoTDevice	

Figure A.4.: Overview of meta model elements illustration

A.3. Architecture Analyses

Overview EAM analysis types with references ([Rau13; Rau15]):

Analysis of Complexity: [Nie06]

Analysis of Conformity and Compliance: [Nie06]

Analysis of Dependencies: [Lag+09], [FFJ09a], [FFJ09b], [Nie06], [Saa10b], [KA09]

Analysis of Heterogeneity: [Buc+06], [Nie06]

Analysis of Maintainability: [LJ08], [Eks+09], [Lag07]

Analysis of Service Response Time: [NBE12], [När+13], [IJ06]

Analysis of the Benefits: [Nie06], [PSP12]

Analysis of the Costs: [Nie06]

Analysis of the Interfaces: [Nie06]

Analysis with XML: [Boe+05b]

Application Usage Analysis: [NBE12], [DS98]

Availability Analysis : [NBE12], [När+11a], [RNE09]

Availability Weak Point Analysis: [Xie+08], [Luo+09]

Business Entity Analysis: [Del+11]

Business Process Support Analysis: [SK11]

Change Impact Analysis: [Boe+05a], [SKR13]

Coverage Analysis: [Nie06]

Critical Path and Completion Time Analysis: [JS+99], [Yen09], [Jon+99]

Data Accuracy Analysis: [NBE12], [När+11b], [När+09]

Delta, Trade-off and Gap Analysis: [Zhu+05], [WC12], [Kaz+98], [Lop00]

Design Analysis: [AGW11]

Extended Influence Diagrams Analysis: [Joh+06], [Joh+07b], [LJN07], [NJN07]

Failure Impact Analysis: [Hol+09]

Intentional Analysis: [YSD06], [Yu11], [YM95]

Matrix Analysis: [Nie06], [Kel07], [KA09]

Maturity Analysis: [MHO11], [Szy09], [Ste+10]

Modifiability Analysis: [Lag+09], [LJH10], [LJE10]

Performance and Workload Analysis: [JI08], [Lan05], [SR95]

Quality Analysis: [När+08], [När+10], [DA09], [NJN07]

Requirements Analysis: [VSP08], [Ant96], [FJW97], [DP11]

Risk Analysis: [IB06], [Spa11]

Run time Analysis: [Far+10], [FHK09]

Security Analysis : [Bus+11], [IB06], [SEJ08], [ES09], [Joh05]

Sensitivity Analysis: [Zhu+05], [WC12], [RAB11]

Service Interoperability Analysis: [Ull+10], [ULJ08]

Social Network Analysis: [Zac00], [KMP11], [HWM95], [TTF79]

Structural Analysis: [LB09]

Survival Analysis: [Aie+09]

Wiki-based Analysis: [Buc+09b]

Others: [Buc+09a], [DS12], [Mat+12]

Functional	Systeme	Attribute	Dependencies	Quality	Design	Effects	Requirements	Financial	Data	Business Objects	Other
	Analysis of Complexity				x	x					
Analysis of Conformity and Compliance		x		x							
Analysis of Dependencies			x								
Analysis of Heterogeneity											x
Analysis of Interface					x						x
Analysis of Maintainability				x							
Analysis of Service Response Time		x		x							
Analysis of the Benefits								x			
Analysis of the Costs								x			
Analysis with XML											x
Application Usage Analysis		x									
Availability Analysis		x									
Availability Weak Point Analysis		x						x		x	
Business Entity Analysis										x	
Business Process Support Analysis										x	
Change Impact Analysis			x			x					
Coverage Analysis				x		x					
Critical Path and Completion Time Analysis				x						x	
Data Accuracy Analysis		x		x					x		
Delta/Tradeoff/Gap Analysis		x				x					
Design Analysis					x						
EID	x					x					
Failure Impact Analysis			x								
Intentional Analysis					x						
Matrix Analysis			x			x					
Maturity Analysis				x							
Modifiability Analysis		x									
Performance and Workload Analysis				x							
Quality Analysis	x	x		x					x		
Requirements Analysis							x			x	
Risk Analysis			x					x			
Run-time Analysis				x							
Security Analysis			x	x			x				
Sensitivity Analysis		x				x					
Service Interoperability Analysis				x							
Social Network Analysis					x						x
Structural Analysis					x						x
Survival Analysis							x				
Wiki-based Analysis					x						

Figure A.5.: Overview categorization of analyses types into functional categories

*Overview EAM analysis categorized into technical categories with references
([Rau13; Rau15]):*

Bayesian Networks: [LJ08], [Hol+09], [När+08]

Business Entities: [Del+11], [MHO11], [Far+10]

PRM: [Lag+09], [Eks+09], [NBE12], [När+13], [LJH10], [LJE10], [När+10], [Bus+11], [Ull+10], [Buc+09a]

Social Network: [Zac00], [KMP11], [HWM95], [TTF79]

AHP: [Zhu+05], [DA09], [RAB11], [DS12]

Time-Evaluation: [IJ06], [JS+99], [Yen09], [JI08], [Lan05], [FHK09]

Tree: [FFJ09b], [FFJ09a], [När+11a], [Jon+99]

KPI and Metrics: [Nie06], [IB06], [Spa11], [BIY08] [Mat+12]

Comparison: [Nie06], [Boe+05a], [WC12], [YSD06], [Yu11], [YM95], [VSP08], [DP11]

Views: [Buc+06], [SK11], [Kaz+98], [Lop00], [Ant96], [FJW97]

Lifecycle: [Saa10b], [Aie+09]

Ontology: [SKR13], [LB09]

EID: [Lag07], [RNE09], [När+11b], [När+09], [Joh+06], [Joh+07b], [LJN07], [NJN07], [SEJ08], [ES09], [ULJ08]

Weak Points: [Xie+08], [Luo+09]

Matrix: [Nie06], [KA09], [Nie06], [PSP12], [Kel07], [Szy09], [Ste+10], [Joh05]

Design: [AGW11]

Structural: [Buc+09b]

Others: [Boe+05b], [DS98], [SR95]

Technical	Bayesian Networks	Business Entities	PRM	Social Network	AHP	Time-Evaluation	Tree	KPI and Metrics	Comparison	Views	Lifecycle	Ontology	EID	Weak Points	Matrix	Design	Structural	Others
	Analysis of Complexity																	
Analysis of Conformity and Compliance																		x
Analysis of Dependencies			x				x		x		x							x
Analysis of Heterogeneity										x								x
Analysis of Maintainability	x		x										x					
Analysis of Service Response Time			x			x												
Analysis of the Benefits								x										x
Analysis of the Costs								x										
Analysis of the Interfaces																		x
Analysis with XML																		x
Application Usage Analysis			x															x
Availability Analysis			x				x						x					
Availability Weak Point Analysis														x				
Business Entity Analysis		x																
Business Process Support Analysis										x								
Change Impact Analysis									x			x						
Coverage Analysis																		x
Critical Path and Completion Time Analysis						x	x											
Data Accuracy Analysis			x											x				
Delta, Trade-off and Gap Analysis					x				x	x								x
Design Analysis																		
EID														x				
Failure Impact Analysis	x																	
Intentional Analysis									x									
Matrix Analysis																		x
Maturity Analysis		x																x
Modifiability Analysis			x															
Performance and Workload Analysis							x											x
Quality Analysis	x	x		x										x				
Requirements Analysis									x	x								
Risk Analysis								x										
Run-time Analysis		x				x												
Security Analysis			x					x					x					x
Sensitivity Analysis					x				x									
Service Interoperability Analysis			x											x				
Social Network Analysis				x														
Structural Analysis													x					
Survival Analysis											x							
Wiki-based Analysis																		x

Figure A.6.: Overview categorization of analyses types into technical categories