



Universität Stuttgart

Evaluation and Control of the Value Provision of Complex IoT Service Systems

Von der Fakultät für Informatik, Elektrotechnik und Informationstechnik
der Universität Stuttgart zur Erlangung der Würde eines Doktors der
Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von
Sina Niedermaier
aus Ellwangen (Jagst)

Hauptberichter: Prof. Dr. Stefan Wagner

Mitberichter: A. Univ.-Prof. Mag. Dr. Reinhold Plösch

Tag der mündlichen Prüfung: 29.03.2022

Institut für Software Engineering

2022

ZUSAMMENFASSUNG

Für viele Unternehmen repräsentiert das Internet der Dinge (englisch: *Internet of Things (IoT)*), eine Möglichkeit durch die Kombination von physischen Produkten mit softwarebasierten Services einen zusätzlichen Nutzen für den Konsumenten zu schaffen. Dabei kann die Qualität des IoT-Services darüber entscheiden, ob dieser auf Dauer genutzt wird und den erwarteten Wert für einen Konsumenten erbringt. Da IoT-Services in der Regel über verteilte Systeme erbracht werden und deren Betrieb zunehmend komplexer und dynamischer wird, ist eine kontinuierliche Überwachung und Kontrolle der zu erbringenden Leistung erforderlich.

Die einzelnen Komponenten der IoT-Service-Systeme werden üblicherweise arbeitsteilig von spezialisierten Teams entwickelt und betrieben. Mit der zunehmenden Spezialisierung, fällt es den Teams schwer, Qualitätsanforderungen basierend auf den Bedürfnissen der Verbraucher abzuleiten. Foglich, beobachten die Teams oft das Verhalten ihrer Komponenten isoliert ohne direkten Bezug zur erbrachten Leistung für einen Konsumenten. Durch unzureichende Überwachung und Kontrolle der Leistungserbringung über die verteilten Komponenten eines IoT-Systems hinweg, kann es zu Qualitätsmängeln und damit einem Verlust an Nutzen für den Konsumenten kommen.

Das Ziel dieser Arbeit ist es, Organisationen mit geeigneten Konstrukten

und Methoden in der arbeitsteiligen Entwicklung und dem Betrieb von IoT-Service-Systemen zu unterstützen um damit die Qualität der für den Konsumenten erbrachten Leistung sicherstellen zu können.

Durch Anwendung von empirischen Methoden haben wir zunächst die in der Industrie bestehenden Schwierigkeiten und verwendeten Praktiken sowie den Stand der Technik analysiert. Ausgehend von den Ergebnissen haben wir bestehende Konstrukte und Methoden weiterentwickelt. Um deren Wirksamkeit und Gebrauchstauglichkeit zu evaluieren, führten wir Aktionsforschungsprojekte in Zusammenarbeit mit Industriepartnern durch.

Auf Basis einer Interviewstudie mit Industrieexperten, haben wir die gegenwärtigen Herausforderungen, Anforderungen und verwendete Lösungen für die Überwachung und den Betrieb von verteilten Systemen genauer analysiert. Die Erkenntnisse dieser Studie bilden die Grundlage für die weiteren Beiträge dieser Arbeit.

Zur Unterstützung und Verbesserung der Kommunikation zwischen den spezialisierten Teams im Umgang mit Qualitätsproblemen, haben wir eine Klassifikation für Systemanomalien erarbeitet. Diese Klassifikation haben wir im Rahmen eines Aktionsforschungsprojekts in der Industrie angewendet und evaluiert. Sie erlaubt es Organisationen, ihr Handeln entsprechend der unterschiedlichen Anomalieklassen zu differenzieren und anzupassen. So können schnelle und wirksame Aktionen zur Sicherstellung der Leistungserbringung oder zur Schadensminimierung getrennt von langfristigen und nachhaltigen Korrekturen des IoT-Systems optimiert werden. Zudem ermöglicht die Klassifikation für Systemanomalien der Organisation, Rückkopplungsschleifen für die Qualitätsverbesserung des Systems, des IoT-Services und der Organisation zu erzeugen.

Um den Nutzen, der durch einen IoT-Service erbracht wird, zu bewerten, zerlegen wir ihn in diskrete Vorgänge, sogenannte IoT-Transaktionen. Unter Anwendung von Distributed Tracing kann in einem weiteren Schritt das dynamische Verhalten einer IoT-Transaktion rekonstruiert und somit „beobachtbar“ gemacht werden. Folglich können das erfolgreiche Abschließen einer IoT-Transaktion und deren Qualität mit Hilfe von Indikatoren festgestellt werden.

Zur systematischen Entwicklung von Qualitätsindikatoren haben wir einen Ansatz entwickelt. Durch Vergleich der im Betrieb ermittelten Ist-Werten mit vorher definierten Zielwerten ist die Organisation in der Lage, Anomalien im zeitlichen Verhalten der Service-Nutzung festzustellen. In Folge, kann die Leistungserbringung mit geeigneten Reaktionen kontrolliert werden. Die Gebrauchstauglichkeit dieses Ansatzes bestätigt sich im Rahmen eines weiteren Aktionsforschungsprojektes mit einem Industriepartner.

Zusammenfassend unterstützt die vorliegende Arbeit Organisationen darin, den erbrachten Nutzen eines IoT-Services quantitativ zu ermitteln und mit wirksamen Aktionen die Leistungserbringung zu kontrollieren. Weiterhin können das Vertrauen des Konsumenten in den angebotenen IoT-Service und in die Organisation erhalten und unter Verwendung von entsprechenden Rückkopplungsschleifen zunehmend gesteigert werden.

ABSTRACT

The Internet of Things (IoT) represents an opportunity for companies to create additional consumer value through merging connected products with software-based services. The quality of the IoT service can determine whether an IoT service is consumed in the long-term and whether it delivers the expected value for a consumer. Since IoT services are usually provided by distributed systems and their operations are becoming increasingly complex and dynamic, continuous monitoring and control of the value provision is necessary.

The individual components of IoT service systems are usually developed and operated by specialized teams in a division of labor. With the increasing specialization of the teams, practitioners struggle to derive quality requirements based on consumer needs. Consequently, the teams often observe the behavior of “their” components isolated without relation to value provision to a consumer. Inadequate monitoring and control of the value provision across the different components of an IoT system can result in quality deficiencies and a loss of value for the consumer.

The goal of this dissertation is to support organizations with concepts and methods in the development and operations of IoT service systems to ensure the quality of the value provision to a consumer.

By applying empirical methods, we first analyzed the challenges and

applied practices in the industry as well as the state of the art. Based on the results, we refined existing concepts and approaches. To evaluate their quality in use, we conducted action research projects in collaboration with industry partners.

Based on an interview study with industry experts, we have analyzed the current challenges, requirements, and applied solutions for the operations and monitoring of distributed systems in more detail. The findings of this study form the basis for further contributions of this thesis.

To support and improve communication between the specialized teams in handling quality deficiencies, we have developed a classification for system anomalies. We have applied and evaluated this classification in an action research project in industry. It allows organizations to differentiate and adapt their actions according to different classes of anomalies. Thus, quick and effective actions to ensure the value provision or minimize the loss of value can be optimized separately from actions in the context of long-term and sustainable correction of the IoT system. Moreover, the classification for system anomalies enables the organization to create feedback loops for quality improvement of the system, the IoT service, and the organization.

To evaluate the delivered value of an IoT service, we decompose it into discrete workflows, so-called IoT transactions. Applying distributed tracing, the dynamic behavior of an IoT transaction can be reconstructed in a further activity and can be made “observable”. Consequently, the successful completion of a transaction and its quality can be determined by applying indicators.

We have developed an approach for the systematic derivation of quality indicators. By comparing actual values determined in operations with previously defined target values, the organization is able to detect anomalies in the temporal behavior of the value provision. As a result, the value provision can be controlled with appropriate actions. The quality in use of the approach is confirmed in another action research project with an industry partner.

In summary, this thesis supports organizations in quantifying the delivered value of an IoT service and controlling the value provision with effective

actions. Furthermore, the trust of a consumer in the IoT service provided by an IoT system and in the organization can be maintained and further increased by applying appropriate feedback loops.

ACKNOWLEDGEMENTS

The work of this dissertation was mainly developed between 2018 and 2020. Without the support of different people and organizations it could not have been realized in this form. I would like to take this opportunity to express my gratitude for all the support I have received in many ways.

First of all, I want to express my gratitude to my supervisor Stefan Wagner, who included me at the Institute for Software Engineering at the University of Stuttgart. I would like to highlight his openness to support my collaboration with the industry as an external PhD student. He gave me room to discover research related to my interests and introduced me into the approaches of empirical software engineering. Moreover, I would like to thank A.Univ.-Prof. Dr. Plösch, who took the role of my co-examiner.

Thanks also to all of the members of the Software Engineering Group of the University of Stuttgart for integrating me and also supporting me with research-related and organizational questions.

As an external Phd student, getting the possibility to work and study in an industrial environment, I would like to also thank Dieter Schmalz and Dr. Frank Bantien. Thank you, for giving me the opportunity to research in collaboration with Bosch and supporting me from an organizational side in realizing conference visits, and trainings. Special thanks to Frank, who supported my work in a broader sense, which was not limited to the work

of his organization and encouraged me to “think big” and grow beyond me.

Special thanks to Dr. Christian Gemmer, as my Bosch supervisor. While he gave me space to pursue my topics, his guidance, his attention to detail, and his extreme accuracy in reading my work were influential. Also, I like to thank him as a friend, for motivating me, especially in the final phase.

Furthermore, I want to express my gratitude to Stefan Heisse. He was a colleague who became a friend through our joint work. The publications we wrote together are essentially influenced by his thoughts, observations, and experience. I highly appreciate his sharp mind and the way he observes and perceives the world. This made our collaboration very intensive, sometimes controversial but it was and further is very inspiring. He influenced my way of thinking and observing the world sustainably.

My thanks goes also to the people, who supported me by reviewing the papers and this manuscript. Thank you: Frank Reichart, Thomas Geoppel, Uwe Gross, Jonas Fritzsich and Daniel Kulesz. Special thanks to Eugen Fuerner for your valuable support and comments, reading the final manuscript.

My gratitude goes also to my family and friends. My family for enabling my education and their loving support. Thanks also to my closest friends who have nothing to do with technical and business work and for reminding me that there are so wonderful and essential things in life other than academic or business stuff.

Paul, our paths crossed at the beginning of my PhD time, and to this day, you continue to accompany my roadless travel. Thank you for being such a loving partner through the ups and downs of this time, reminding me to claim my own space. I am very grateful for our relationship, which is more than the sum of its parts, – and to speak with the words of this dissertation – for me, it is positively emergent.

Sina Niedermaier in September, 2021

CONTENTS

1 Introduction	19
1.1 Motivation	19
1.2 Problem Statement	20
1.3 Research Goal and Research Approach	22
1.4 Contributions	24
1.5 List of Publications	25
1.6 Structure of the Dissertation	26
2 Background	29
2.1 Distributed Systems	30
2.1.1 Microservices	31
2.1.2 Complex IoT Systems	33
2.1.3 System Stakeholders and Architectural Views of Distributed Systems	34
2.2 Concepts of Service, Customer and Consumer	34
2.2.1 Concepts of Service	35
2.2.2 Customer and Consumer Value and IoT Service	37
2.3 Quality Models and Operationalization (SQuaRE)	39
2.3.1 Concepts of Quality	39
2.3.2 System and Software Quality Models	41

2.3.3	Operationalization of Quality Characteristics	42
2.3.4	Conceptual Approaches to Quality	44
2.3.5	Quality Requirements and Evaluation	45
2.3.6	Critique of SQuaRE	47
2.4	Service Quality	48
2.4.1	Perform As and when Required – Dependability	51
2.5	Anomaly Classification	52
2.6	Fault Tolerance and Degraded Quality	55
2.7	Observability and Monitoring of Distributed Systems	56
2.7.1	Application Performance Management	58
2.7.2	Concept of Distributed Tracing	58
2.8	Site Reliability Engineering	60
2.8.1	Concepts of SLI and SLO	60
2.8.2	Time-Based Availability vs. Aggregate Availability	61
2.8.3	SRE Adoption in Industry	62
2.9	Situation Management and Postmortem Analysis	62
2.9.1	Situation Management	63
2.9.2	Postmortem Analysis	64
3	State of the Art	67
3.1	Studies on Observability and Monitoring	67
3.2	Classification of Anomalies	71
3.3	Evaluating Quality Requirements	72
3.3.1	Defining Quality Requirements	72
3.3.2	Evaluating Quality Requirements	76
3.3.3	Observability and Monitoring Approaches	77
4	Interview Study on Observability and Monitoring of Distributed Systems	79
4.1	Context and Goals	80
4.2	Research Design	80
4.3	Preparation for Data Collection	84
4.4	Data Collection	85

4.5	Analysis of Collected Data	85
4.6	Results	86
4.6.1	Challenges (RQ1)	86
4.6.2	Requirements and Solutions (RQ2 and RQ3)	93
4.7	Discussion	103
4.8	Threats to Validity	104
4.9	Conclusion	105
5	Correct & Control Complex IoT Systems: A Classification of System Anomalies	107
5.1	Context	108
5.2	Goals	110
5.3	Concepts	111
5.3.1	Concepts of Classification of System Anomalies	112
5.3.2	Fault Tolerance and Situation Management	115
5.3.3	Defect Correction and Quality Management	118
5.3.4	Communicating Anomalies	120
5.4	Research Design	122
5.4.1	Research Objective	122
5.4.2	Research Process	124
5.5	Results	126
5.5.1	IoT System Description	126
5.5.2	Situation	126
5.5.3	Classification Results – Stage 1	127
5.5.4	Discussion – Stage 1 (RQ1)	129
5.5.5	Interview Results – Stage 2 (RQ1, RQ2, RQ3)	130
5.5.6	Discussion – Stage 2	134
5.6	Threats to Validity	136
5.7	Conclusion	137
6	The Concept of IoT Transaction	139
6.1	Context and Goal	139

6.2 Concepts	140
6.2.1 IoT Transaction	140
6.2.2 IoT Transaction Monitoring	142
6.3 Conclusion	145
7 Evaluate & Control Service & Transaction Dependability of Complex IoT Systems	147
7.1 Context and Goals	148
7.2 Research Design	149
7.2.1 Research Objective	149
7.2.2 Research Process	150
7.3 Transaction-Based Approach for Dependability Evaluation - Stage 1	153
7.3.1 Define Requirements for the Design of the Artifact	153
7.3.2 Transaction-Based Approach for Dependability Evaluation (RQ1)	155
7.4 Evaluation of the Transaction-Based Approach for Dependability Evaluation - Stage 2 (RQ2 and RQ3)	158
7.4.1 System Description	158
7.4.2 Application of the Transaction-Based Approach for Dependability Evaluation	159
7.4.3 Focus Group	170
7.5 Discussion	173
7.6 Threats to Validity	176
7.7 Conclusion	178
8 Conclusion	181
8.1 Summary of Contributions	181
8.2 Limitations and Future Work	186
Bibliography	191
List of Figures	209

List of Tables	211
List of Abbreviations	213

CHAPTER



1

INTRODUCTION

This chapter describes the motivation, problem context, research goal, and contributions of this dissertation. Furthermore, it presents the list of publications and the structure of this dissertation.

1.1 Motivation

The internet of things (IoT) represents an opportunity for new ways of providing value to customers and consumers through connected physical products providing services [LWRS12].

The quality of service is deciding whether a service is consumed and whether it provides the expected value for a customer and a consumer. However, the concept of quality is difficult in itself, and there are different approaches to it. How can one evaluate whether a service is of quality and whether it provides value for a customer or a consumer?

This is a challenging task, especially in the context of IoT systems. An IoT system providing services is a distributed complex system of components, integrating hardware, software, and mechanical system elements, and is in interaction with humans. In practice, the components of an IoT system are

usually decentrally developed and operated by specialized teams from different organizational units. Moreover, with emerging paradigms of microservice architectures and cloud computing, components and system elements are implemented as loosely coupled and independently deployable microservices often running on geographically distributed resources. While the development complexity of an individual component is decreased, the system's operational complexity is increased. The growing amount of components and continual changes in the system results in a rise of system complexity and dynamics, which increases the probability of anomalies [ABDP13; HHMO17; NGSR16].

During operations, providers of IoT service systems must control the dynamic system behavior in a permanently changing context of use. This requires continuous observation and control of the value provision.

With the ability to connect products to the internet, organizations have the opportunity to observe service consumption and learn about consumer and system behaviours [OB15]. By observing the value provision, organizations are able to create feedback loops about the quality in use, detect anomalies in the value provision to customers and consumers, and perform actions to control the value provision.

1.2 Problem Statement

As highlighted in research regarding customer feedback techniques and consumer involvement [BB15; Bos12] there is a fundamental challenge for organizations to identify customer or consumer needs. Customers and consumers are not always able to know and express what they want [Ulw02] as their needs can be implied or unaware [ISO11d]. Often they do not know what they want until they experience it [Ulw02]. Therefore, eliciting requirements based on stakeholder needs is a challenging task. Requirements elicitation is particularly challenging for quality requirements, as the *“quality of a system is the degree to which the system satisfies the stated and implied needs of its various stakeholders, and thus provides value”* [ISO11d]. The

evaluation of an entity's quality, whether it satisfies needs and provides value, is based on the operationalization of its quality, the description of how quality is observed and measured. The operationalization defines the concept quality as a set of things that are, or are not part of the concept. A provider of IoT services, in a broader sense, tries to differentiate what is not of quality of service – what represents a loss of value or what constitutes a deficiency in the quality of a service; consequently, the provider derives what an anomaly is. In operations, providers have to detect unwanted system behavior – anomalies – in order to exclude them and control the value provision.

As customer and consumer needs regarding quality can often only be derived inductively from experience, requirements for the concept of quality are just assumptions and need to be treated as hypotheses. However, a common view in industry practice is that requirements are regarded as “*truth*” [OB15]. This usually leads to a normative verification – often on a component level – if the implementation matches the specification, rather than a validation if a customer or a consumer is experiencing anomalies [BBHR16]. However, observation and validation are critical for highly dynamic systems, where anomalies need to be handled during operations to control the value provision to a customer and a consumer. Through the increasing specialization with paradigms like microservices and cloud, individual components or elements of distributed systems are developed, observed, and monitored in isolation, often without context to the value provision to a customer or a consumer [NGSR16]. When monitoring data is available, the data collected often cannot be used for a feedback cycle, as they are unstructured and with too less context. Many organizations struggle to capitalize from the flood of data as they fail to ask the “*right questions*” [OB15]. While there are standards like the SQuaRE series, which aim to provide guidance for the quality requirement elicitation and the process of evaluation, they also tend to support the normative perspective of requirement verification. The SQuaRE series lacks to consider the change of stakeholder needs over time [ABE19] and the operations stage in the system lifecycle.

Our industry study supported the described difficulties regarding require-

ment elicitation and monitoring during operations (see Chapter 4). The study found that on the one hand, there are several approaches and many technologies. On the other hand, we identified a high degree of uncertainty among practitioners about how to derive quality requirements from customer and consumer needs and how to effectively and efficiently observe and control the quality of the value provision. With increasing system complexity and dynamics, practitioners are overwhelmed to process and exchange information across different specialized teams in order to handle anomalies influencing the value provision. We identified that communication and collaboration across different teams and organizations are more challenging than technical aspects. Therefore, concepts and approaches are needed to support organizations to collaboratively develop and operate complex systems and control the value provision to customers and consumers.

1.3 Research Goal and Research Approach

Based on the structure suggested by Wieringa [Wie14] the research goal of this dissertation can be defined as follows:

*Provide organizations developing and operating IoT services
with concepts and approaches
in order to effectively and efficiently
evaluate, correct and control distributed IoT service systems.*

As the research goal focuses on (re-)designing artifacts for the effective and efficient application, we applied interview studies and action research in collaboration with industry, following the design science paradigm. Design science focuses on developing artifacts like concepts, models, methods, processes, and investigations in the context of people, organizations, software, and hardware [HMPR04; MS95; Wie14]. Hevner et al. [HMPR04] originally conceptualized design science for information systems and Wieringa [Wie14] pursued it for software engineering. The paradigm of design science includes problem conceptualization, design of the solution, and validation [RES20]. As design science provides a framework for this dissertation's research, we

started the problem conceptualization with an industry interview study (C1) to analyze specific problem instances in practice and gain insight into the problem context. This study led to challenges, requirements, and solutions in operating, observing and monitoring distributed systems. The study results to explore the problem context and state of the practice served as input for our contributions, which focus on:

- C2: a classification for system anomalies to correct and control complex IoT service systems
- C3: a concept for IoT transactions to evaluate the value provision of IoT service systems
- C4: an approach to evaluate and control IoT transaction dependability of distributed systems.

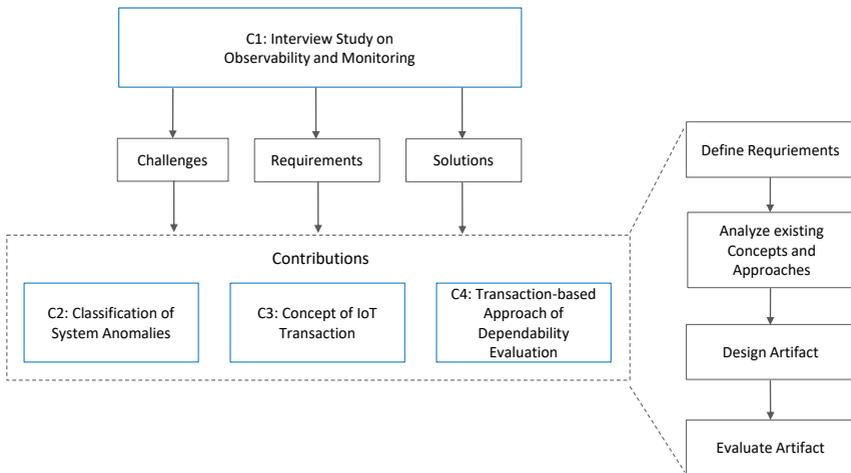


Figure 1.1: Research approach

Our research approach, including our contributions of this dissertation are illustrated in Figure 1.1.

As we address real industry practice, we performed the design and evalu-

ation of the artifacts in collaboration with industry. We followed an iterative approach of action research [Sta19]. In the first stage, we used the results of interview study as input and further explored concrete problem instances at our industry partner, the Bosch Group. Furthermore, we analyzed existing concepts and methods and planned the action procedures. Based on the requirements and analysis results, we designed the artifacts and performed their instantiations [RES20] by implementing them in real-world industry contexts. In the last stage, we evaluated the quality in use of each artifact for the problem instance and performed feedback loops to adjust the artifacts accordingly.

1.4 Contributions

This dissertation focuses on the evaluation of the value provision of complex IoT services and provides the following four contributions:

- **C1: Results of the Interview Study on Observability and Monitoring of Distributed Systems:** We conducted semi-structured interviews to analyze the current state of practice of monitoring and observability approaches in industry. We related challenges, requirements, and applied strategies and solutions from the viewpoints of different stakeholders of monitoring systems and tool providers. This study analyzes the problem context and builds the basis for the further contributions.
- **C2: Correct and Control Complex Distributed Systems: A Classification for System Anomalies:** We extended the IEEE 1044 [IEE09] standard classification for software anomalies with three system views: functional system view, dynamic system view, and system structure view to support inter-team communication across different organizational units. Our adaption allows the organization to differentiate between a quick fault fix and a sustainable defect correction. Thus, quick and effective actions to ensure the value provision or minimize the loss of value can be optimized separately from actions in the context of long-term and sustainable correction of the IoT system. We

applied our adapted classification to a postmortem analysis at the Bosch Group. By observing the application and conducting semi-structured interviews with the stakeholders, we evaluated the quality in use the artifact.

- **C3: The concept of IoT Transaction:** To operationalize the value provision of complex IoT service systems from a customer and a consumer view on value, we present the concept of IoT transaction. We decompose and IoT service into discrete IoT transactions, whose dynamic behavior can be reconstructed and made “observable” through distributed tracing. Thus we are able to indicate IoT transaction completion and transaction dependability.
- **C4: Evaluate and Control Service and Transaction Dependability of Complex IoT Systems:** This contribution applies the IoT transaction concept of the previous contribution. Based on the quality evaluation process of ISO/IEC 25040 [ISO11b] we designed an approach to evaluate the transaction dependability from a consumer view. The approach enables the systematic derivation of quality indicators. By comparing actual values determined in operation with previously defined target values, the organization is able to detect anomalies in the temporal behavior of the value provision. We conducted an action research project, where we applied the designed artifact in the context of an IoT solution at the Bosch Group. In addition, we conducted a focus group discussion to evaluate the approach’s effectiveness, efficiency, and satisfaction.

1.5 List of Publications

In the course of the development of this dissertation, we have already published some parts of it with the following publications.

- S. Niedermaier et al. “On Observability and Monitoring of Distributed Systems – An Industry Interview Study” In: Service-Oriented Comput-

ing, 17th International Conference, ICSOC 2019, Toulouse, France, October 28-31, 2019 Proceedings. Springer, 2019.

- S. Niedermaier et al. “Correct and Control Complex IoT Systems: Evaluation of a Classification for System Anomalies” In: Proceedings of the IEEE 20th Conference on Software Quality, Reliability and Security (QRS 2020). IEEE, 2020, pp. 321-328.
- S. Niedermaier et al. “Evaluate and Control Service and Transaction Dependability of Complex IoT Systems”. In: Software Quality Journal. Springer, 2021.

1.6 Structure of the Dissertation

The dissertation is structured as follows:

- Chapter 2 provides background to understand the context and the contributions of this dissertation.
- Chapter 3 extends the background chapter and discusses the state of the art in the areas that constitutes our contributions.
- Chapter 4 presents our first contribution, which explores the industry problem context in more detail. The interview study analyzes the relations between the challenges, requirements, and solutions stated by software practitioners from different organizations.
- Chapter 5 describes our concept of classification for system anomalies. The artifact was applied and evaluated on a postmortem analysis of an IoT solution at the Bosch Group.
- Chapter 6 presents our concept of IoT Service and IoT transaction and their evaluation of value provision from the perspective of a customer and a consumer.
- Chapter 7 introduces the designed artifact, the transaction-based approach for dependability evaluation. We designed the artifact based on the evaluation process for software product quality of the SQuaRE

series. Furthermore, we applied the concept of IoT transaction, implemented approaches of Site Reliability Engineering and applied distributed tracing as the observation method. We evaluated the artifact at an action research project at the Bosch Group.

- Chapter 8 summarizes and discusses the contributions and limitations of this dissertation and closes with an outlook and future research directions.

CHAPTER



BACKGROUND

This chapter introduces concepts and approaches that are the basis for understanding the contributions to this dissertation. First, we give an overview of the domain of distributed systems (Section 2.1), including the architectural style of microservices and IoT systems, as they represent the target systems of the action research studies of our contributions. In Section 2.2, we discuss different views on the concept of service and present the applied service concept for this dissertation. Section 2.3 introduces the concept quality and presents approaches of quality models and their operationalization based on SQuaRE, a standard for system and software quality requirements and evaluation. In the following, we discuss the concepts of service quality and dependability (Section 2.4). As the basis for quality indication, we present a conceptual model for anomaly classification in Section 2.5 and describe the concept of fault tolerance and quality degradation in Section 2.6. After that, we introduce observability and monitoring approaches in Section 2.7 as a foundation to enable service quality control. In Section 2.8, we present Site Reliability Engineering which includes industry practices for service quality evaluation. We close the background chapter with the handling of anomalies in the context of situation management and postmortem analysis

(Section 2.9).

2.1 Distributed Systems

The spectrum of distributed systems ranges from social networks, web searches to IoT service systems, often implemented as microservice architectures running on heterogeneous infrastructures from serverless to cloud infrastructures. Tanenbaum and van Steen define a distributed system as a “collection of independent computers that appears to its users as a single coherent system” [TV07]. Distributed systems enable parallel computing, where processes can be executed simultaneously. A distributed system contains components that are interconnected and communicate via passing messages [Lam78].

An ideal distributed system is more reliable and scalable as a centralized system. Unfortunately, in practice most systems are not ideal distributed systems. Leslie Lamport one of the pioneers of the concept of distributed computing stated: “A distributed system is one in which the failure of a computer you didn’t even know existed can render your own computer unusable.” [Lam87].

Lamport’s statement indicates that distributed system behavior is often unexpected and difficult to understand and control. Modern distributed systems grow in the number of interconnected and geographically distributed components, which constitutes an increase of system complexity. With the growing complexity of distributed systems, the concept of “emergence” is discussed in research [BSN18; Hud11; Mog06]. By integration of individual components into a new distributed system, system behavior may emerge, which cannot be deduced from the knowledge of the individual components. Emergent behavior is the unexpected behavior of a system that cannot easily be predicted from its individual components’ behavior [Mog06]. The emergent behavior of a system in operation cannot only be controlled in prevention, e.g., through analytical modeling and testing. The operations context of use reveals unknown emergent behavior that has to be handled and controlled during operations. However, often distributed systems lack a

central point of observability and control of workflows spanning across the distributed components. As the prediction of a systems' emergent behavior by examining the system components is not possible, it is challenging to control distributed systems in operations.

A lack of controlling unwanted emergent system behavior can lead to service failures and, consequently, a loss of trust of customers or consumers and related revenue losses for organizations [BJPM16].

In the following sections, we describe the concepts of microservice and IoT systems as examples of implementation of distributed system architectures.

2.1.1 Microservices

The trend towards more flexible and modular distributed systems is characterized by applying microservice architectures. By microservice architectures systems are composed of a set of granular and loosely-coupled software elements. The concept of loosely-coupled software elements is not new and companies like Google, Netflix and Amazon applied the architectural style before calling it microservice architecture. The popularity of microservices is intensified by paradigms of cloud computing [FLR+14] and DevOps [BWZ15]. DevOps is a software practice that integrates development (Dev) and operations (Ops) teams within or across organizations to enable flexible reaction to changing business requirements with more frequent and rapid deployments [KBS14].

Since the concept of microservices is pretty young, there is no uniform description of this architectural style's characteristics. Therefore, we present some of the most common characteristics of the microservice concept by Newman [New15]:

- **Small and focused on doing one thing well:** A microservice represents a business functionality. Concerning the size of a microservice Newman does not present an explicit quantification of small. He rather cites a rule of thumb: “*something that could be rewritten in two weeks*” [New15]. Furthermore, he refers to the sense of developers that they know what is small enough and no smaller.

- **Autonomous:** Microservices are isolated entities that can be changed and deployed independently.
- **Technology heterogeneity:** Each microservice can be built applying the most appropriate technology and tool.
- **Resilience:** Cascades of failing microservice functionalities can be avoided with concepts of reliability patterns like bulkheads. Furthermore, observability and quality degradation of functionalities are important concepts.
- **Independent scaling:** The microservice instances can be scaled individually with regard to traffic spikes.
- **Ease of deployment:** Change releases and deployments are simplified and faster due to the independence of the individual microservices.
- **Organizational alignment:** The concept of microservices allows to align architecture and organization. Therefore, the productivity of teams can be increased as they work on smaller codebases.
- **Composability:** A key idea of microservices is the reuse of functionalities. Functionalities can be consumed by different consumers for different purposes.
- **Optimized for replaceability:** Microservices are the counter-concept to legacy systems. With the individual and small microservice, rewriting and removing them is simpler.

Microservices usually run on cloud-based or containerized systems, where infrastructures change frequently. In practice, different microservice implementations exist that do not follow all of the stated characteristics. Similar to distributed systems in general, the development and operation of microservices in practice are not ideal. With growing modularization and an increasing number of microservices, the interconnections between microservices also increase. Understanding the dependencies of microservices is challenging due to the increasing observation complexity [Blu19; LCZ18].

With the reuse of functionalities in different contexts and for different purposes, microservice systems show unwanted emergent behaviors, as distributed systems in general. Besides, through concepts of continuous integration and delivery, microservices architectures are evolving all the time. Consequently, extensive system tests are often not performed due to the high-frequency releases [HHK+17]. Instead, compensation or and even substitution of quality assurance procedures by monitoring techniques during operation can be observed in practice.

2.1.2 Complex IoT Systems

IoT systems (Internet of Things) are considered distributed systems. In the following, we describe these systems, as they are applied to provide services at our industry partner, where we conducted our action research studies.

The International Telecommunication Union (ITU) defines IoT as “*a global infrastructure for the information society, enabling advanced services by interconnecting physical and virtual things based on existing and evolving interoperable information and communication technologies*” [Uni01].

The idea behind IoT is to embed networked, electronic devices into physical objects, the things. These objects are enabled to interact with their environment and communicate with each other via internet [MLLL+15]. Objects of daily life can be equipped with microchips, sensors and connectivity technologies to enable them to communicate with other things via the internet.

From a business perspective IoT represents an opportunity for new ways of creating value for customers and consumers and therefore new business models can emerge [LWRS12]. With the rise of IoT companies have the ability to release from purely product-based business model and move toward a service-oriented value provision with IoT. Additional customer and consumer value can be generated through merging connected physical products with software-based services.

Providers of cloud-based IoT solutions compile distributed complex systems of components, integrating hardware, software, and mechanical system

elements. Components are part of the system, providing functionalities. By compiling individual components to an IoT system, the system is able to provide more functionality and performance than the sum of the individual components. At the center of an IoT system is the human who is consuming the services provided by an IoT system. Therefore, we consider humans as components of the IoT systems.

To ensure the value provision from a customer or a consumer view, the provider of IoT service systems must control the dynamic system behavior in a permanently changing context of use.

2.1.3 System Stakeholders and Architectural Views of Distributed Systems

With the increasing size and complexity of the distributed systems, conceptual models are required to design and operate them.

System architecture descriptions are applied to enable communication and collaboration of system stakeholders about behavior, structure, and evolution of a system [ISO07a]. The system stakeholders can be individuals or organization having a concern, “*an interest in a system*” [ISO07a]. These concerns can be expressed in an architecture view from an architecture viewpoint, specifying the conventions for construction and application of the view. A view represents a system and the relation of its entities according to specific system concerns and addresses a set of stakeholders [ISO07a].

The views to a system will be an essential part of our contributions to this dissertation, as they are the basis on how to organize and guide development and operation in a setting of diverse stakeholders with different concerns.

2.2 Concepts of Service, Customer and Consumer

Outsourcing non-core activities as services has become a key strategy of the twenty-first century global economy [LROR07]. As the evaluation and control of services are central parts of this dissertation, we present and discuss relevant concepts of services. Moreover, we investigate the value provision

in relation to a customer and a consumer view of value.

2.2.1 Concepts of Service

Companies are increasingly moving from providing products to providing services or products combined with services. But how is the concept of service defined? In the following, we present and discuss different concepts of service.

As presented in Section 2.1.1 regarding microservice architectures, the concept service represents a provided functionality of a software system. Avizienis et al. defined the concept of service as follows: *“The service delivered by a system (in its role as a provider) is its behavior as it is perceived by its user(s); a user is another system that receives service from the provider.”* [ALRL04]. This definition refers to the perceived behavior of an interactions between a provider and a user.

As stated by ISO/IEC/IEEE 24765:2017 systems and software engineering vocabulary [ISO17b], a service can be expressed by *“means of delivering value for the customer by facilitating results the customer wants to achieve”* or *“performance of activities, work, or duties”*. Besides, a service is characterized as intangible [ISO17b]. This consideration of the concept of service is more from the viewpoint of a business activity of providing value to a customer or a consumer. The service concept by ISO/IEC/IEEE 24765:2017 is almost equal to the service concept provided by ISO/IEC 20000-1:2018 [ISO18b], an international standard for IT service management. Instead of *“facilitating results”* [ISO17b], ISO/IEC 20000-1:2018 [ISO18b] applied the concept of *“facilitating outcomes”*.

The different notions of the concepts of service have in common that a service is an activity. Since services are intangible, the consumer is part of the activity in which the service is produced and consumed at the same time [WKH98]. From a business perspective, organizations or individuals outsource tasks in order to concentrate on their core competencies, to rationalize their business, to increase flexibility, or to acquire know-how.

The increasing specialization and the outsourcing of activities are means to reduce complexity.

By accessing a service, a customer or a consumer trusts the provider to receive the outsourced activity. The interactional context of providing services is based on a trust-relationship between the entities [CEC90]. The interpersonal dyad of trust-relationships are addressed by social psychology [JS16; Sim07]. The customer or consumer (trustor) is willing to rely on the actions of the provider (trustee). The trustor places trust in the trustee in handing over responsibility and control of the actions (service) performed by the trustee. Providing a service the trustee delivers value – by facilitating outcomes as and when required – to keep the trustor’s trust. We can further transfer the logic of the dyad to the concept of service. A provider acts as a server, providing a service to a servee (customer or consumer) as the recipient of a service [Cha91].

Compared to selling products, being served with a service “*does not result in the ownership of anything*” [Kot88]. From a provider view, the provision of services is about “owning”, i.e. controlling the customer and consumer relationship to keep the servee’s trust. The service relationship goes beyond a one-time provision of a product and requires continuous management of the entire set of outsourced tasks of the servee. From a system-theoretical perspective provided by Luhmann [Luh14], trust is a concept, that can only be placed in the present and must be continuously preserved. As the environment and the service system with which we provide services are dynamic, continuous service observation, evaluation and control are needed.

As stated above, services are intangible and have a continuous nature. Nevertheless, we want to make them evaluable; therefore, we propose to apply the concept “*outcome*” for our service definition applied in this dissertation. ISO/IEC 24774 [IEE12], an international standard, providing guidelines for process description of system and software engineering defines an outcome as “*observable result of the successful achievement of the process purpose. Outcomes are measurable, tangible, technical, or business results that are achieved by a process [...]. Outcomes are observable and assessable*” [IEE12].

2.2.2 Customer and Consumer Value and IoT Service

In this section, we present our concept of service for this dissertation. Before, we introduce two different stakeholder concerns; the customer concern regarding the value and the consumer concern regarding the value a service delivers.

We start with the concept of service as described in the technical specification regarding service quality models ISO/IEC TS 25011:2017 [ISO17a] that refers to a service in relation to a user as “*means of delivering value for the user by facilitating results the user wants to achieve*”. The concept of ISO/IEC TS 25011:2017 is based on the concept of service of ISO/IEC 20000-1:2011 [ISO11a], but changed the concept of “*customer*” to the concept of “*user*”. ISO/IEC TS 25011:2017 [ISO17a] differentiates a customer from a user by referring to ISO/IEC 20000-1:2011 [ISO11a], defining a customer as an “*organization or part of an organization that receives a service or services*” and a user as a “*person or an organization that uses an IT service*” [ISO17a].

This description contains two different stakeholder concerns: a customer receives a service, a user or consumer consumes a service. In research and practice, the concepts of *customer*, *user* and *consumer* are used interchangeably for addressing concerns of a service. As the purpose of a service is to deliver value, we investigate the different stakeholder concerns in relation to their intended value.

In the domain of marketing research, we find a clear distinction between the concept of *customer* and *consumer*. A **customer** is an individual or organization that purchases a product or service via financial transactions. Hence, a customer has a financial relationship. The customer value focuses on a **normative perspective** of the evaluation at the point in time of purchase [Lai95]. If we consider services rather than one-time product purchases, this also includes performance-based billing, e.g., per transaction performed. The feedback mechanism of a customer is normative, following a dual logic based on a value proposition or terms and conditions of the contract, which allows, e.g. withholding of transaction-related payments.

A **consumer** is an individual who consumes the product or service. The

consumer value focuses on the evaluation during consumption. This view of consumer value highlights the **dynamic perspective** of the consumption in a specific consumption context [Lai95]. The consumer is interested in the **dependability**, “*the ability to perform as and when required*” [IEC15] when consuming the service. In contrast to a customer, a consumer (if not simultaneously being the customer with a contractual relationship) does not have a direct feedback mechanism to communicate quality deficiencies and improve quality. Therefore, it is necessary to observe service consumption in the context of use to indicate the value provision and detect anomalies that impact service provision’s dependability.

Since the concept *user* can include an individual or organizations and is used interchangeably for the concepts customer and consumer, we avoid applying the concept *user* for our contributions.

Our resulting concept of an IoT service is based on ISO/IEC/IEEE 24765:2017 [ISO17b] and integrates the concept of outcome. Furthermore, it differentiates the service’s view of value with respect to customer and consumer concerns.

IoT Service:

performance of activities delivering

- a) value to the **customer**
- b) value to the **consumer**

by facilitating

- a) the outcomes the **customer**
- b) the outcomes the **consumer**

wants to achieve.

The differences between customer and the consumer value for evaluating the value provision is part of our contribution in Chapter 6.

2.3 Quality Models and Operationalization (SQuaRE)

This section gives an overview of models applied to abstract and evaluate the quality of software, systems, and services. We describe relevant parts of the SQuaRE series, including the system and software quality models and their operationalization [ISO11d] as they are the foundation for the contributions of this work.

2.3.1 Concepts of Quality

The concept of quality is often used in daily life and is perceived as self-explanatory and obvious [Bev95]. However, there are multifaceted notions of the concept of quality from different views [Gar84]. According to DIN EN ISO 9000, the European standard for quality management system, quality is the “*degree to which a set of inherent characteristics of an object fulfil[]s requirements*” [DIN15]. Objects include entities like products, services, software, organizations, or processes. ISO/IEC 25010, the international standard for system and software quality models describes the quality of a system as “*the degree to which the system satisfies the stated and implied needs of its various stakeholders, and thus provides value*” [ISO11d]. Both concepts appear similar as the determination of an entity’s quality depends on the set of selected characteristics and requirements used for the evaluation.

On closer examination, the definition of DIN EN ISO 9000 [DIN15] indicates that it is possible to deductively define the requirements of a product completely. Thus, according to this notion, quality means conformity to its specification. This perception corresponds to a manufacturing emphasis of prevention “*doing it right the first time*”; because poor quality is associated with rework [Gar84] from a process-based approach.

However, the concept of quality of ISO/IEC 25010, with the statement: “*satisfies stated and implied needs [...] and thus provides value*”, includes a broader and value-based approach in contrast to a “normative” conformity to the specification. Implied needs include: “*needs that may not have been stated but are actual needs*” [ISO11d]. Further ISO/IEC 25010 explicates that

some “*implied needs only become evident when the software product is used in particular conditions.*” [ISO11d]. On the one hand, this statement indicates that some quality properties of a system or software can only be assigned in relation to its stakeholders, context of use, and the value it is providing [Bev95]. On the other hand, ISO/IEC 25010 notes an example, in which “*Implied needs include needs not stated but implied by other stated needs and needs not stated because they are considered to be evident or obvious.*” [ISO11d]. The description of implied needs creates, similar to DIN EN ISO 9000 [DIN15], the impression that all needs can be deductively derived into requirements. However, some of the implied needs only become obvious through empirical knowledge and i.e. became common practice in retrospective. Moreover, ISO/IEC 25010 [ISO11d] further explicates that stakeholder needs and expectations can be stated, implied or unaware. Unaware expectations and needs can only unveil in the individual context of use of a stakeholder through experience [Bev95; Ulw02]. The concept of context is any information which is applied to characterize the situation of the interaction of entities [Dey01]. Therefore, the *context of use* is an open concept and includes “*users, tasks, equipment (hardware, software and materials), and the physical and social environments in which a system, product or service is used.*” [ISO11c]. Since a quality model is based on derived requirements and assumptions of user needs, an entities’ quality cannot be better than the quality of its specification [Bev95]. In turn, the specification can only be as good as there is sufficient knowledge of the context of use, which must be explored and validated iteratively.

With the increasing complexity of the constantly changing IoT systems, normative verification of specifications is not sufficient. Consequently, continuous observation of operations and context of use to control anomalies in the value provision to a customer or a consumer are needed. Moreover, quality assumptions, i.e., the quality model, have to be validated with empirical knowledge from operations.

2.3.2 System and Software Quality Models

A quality model is an conceptual construct to describe the quality of an entity as a decomposition into characteristics and sub-characteristics. Moreover, it enables to “*assess and/or predict [the] quality*” of an entity by the operationalization of its quality characteristics [Wag13].

In research and industry, several quality models exist. The leading quality model for systems and software is the ISO/IEC 25010 [ISO11d], it influenced different models of research and industry. It is part of the systems and software quality requirements and evaluation (SQuaRE) ISO/IEC 25000 series. This series of standards is a framework to specify and evaluate system and software quality. ISO/IEC 25010 evolved from ISO/IEC 9126 which is considered a standard decomposition of quality characteristics and suggests a small number of measures for measuring them [ISO01]. By dividing system and software quality into two models: the quality in use model (QIU) and the product quality model (PQ), ISO/IEC 25010 replaced its predecessor ISO/IEC 9126 [ISO01].

The QIU model of ISO/IEC 25010 [ISO11d] “*relate[s] to the outcome of interaction when a product is used in a particular context of use.*” It takes the perspective of a consumer and is composed of five characteristics: the effectiveness, efficiency, satisfaction, freedom from risk and context coverage with corresponding sub-characteristics (see Figure 2.1).



Figure 2.1: Quality in use model (based on[ISO11d])

The system and software product quality (PQ) model “*relate[s] to static*

properties of software and dynamic properties of the computer system.” It represents a normative perspective on a target entity and is usually considered from the viewpoint of developers. The PQ model decomposes into eight characteristics: functional suitability, performance efficiency, compatibility, usability, reliability, security, maintainability, and portability with corresponding sub-characteristics (see Figure 2.2) [ISO11d].

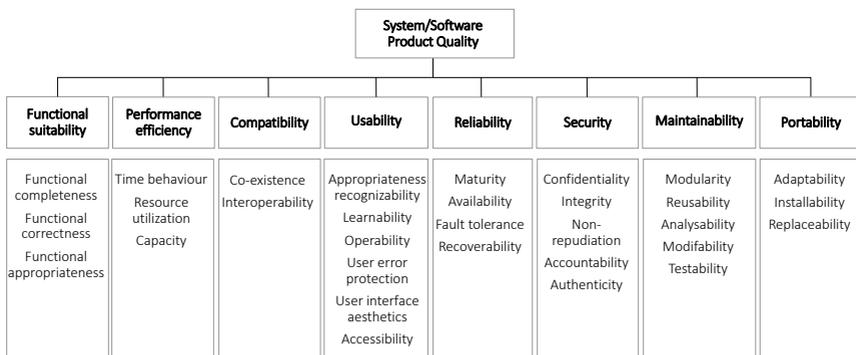


Figure 2.2: Product quality model (based on[ISO11d])

2.3.3 Operationalization of Quality Characteristics

Quality characteristics and sub-characteristics of a quality model need to be measurable, either directly or indirectly, with a set of associated measurable properties of a target entity. The conceptual relation between a characteristic and a measurable property is called operationalization.

Figure 2.3 illustrates the relationship between quality (sub-) characteristics of a conceptual quality model and their indication through quality measures. Measures are “variables to which a value is assigned as the result of measurement” [ISO07b]. Quality measures exist of quality measurement elements (QMEs) that are connected through a measurement function. QMEs themselves are common measures that are generated through a measurement method to quantify a property of a target entity [ISO11c; ISO11d]. A target entity is a “fundamental thing of relevance to the user, about which

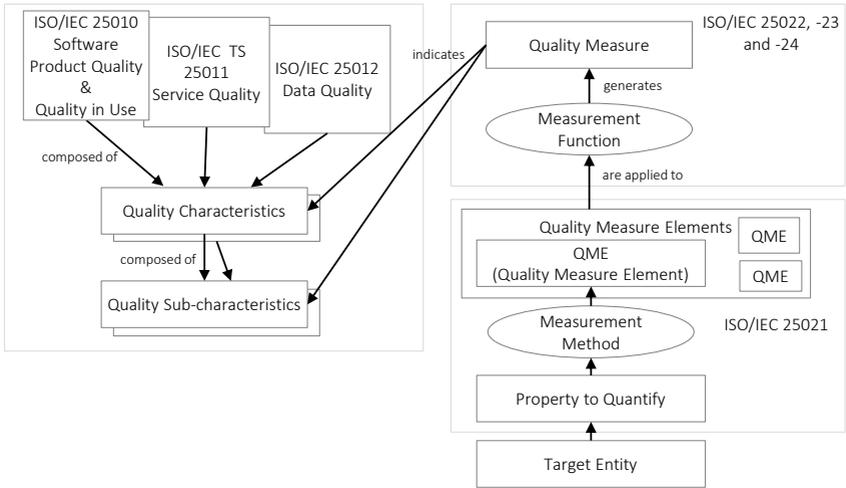


Figure 2.3: Relationship among quality characteristic, measure and target entity (adapted from [ISO12])

information is kept, and need to be measured”. A target entity can include “a work product or behavior of a system, a software or stakeholders” [ISO12].

The SQuARE series further contains a data quality model in ISO/IEC 25012 [ISO08a] and a service quality model in ISO/IEC TS 25011 [ISO17a] (see Figure 2.3). Both conform to the logic structure of the quality models of ISO 25010. The service quality model is the latest, published in 2017. It is still a technical specification and not a standard. We can map most of the quality characteristics and sub-characteristics of the service quality model to the PQ model. We will not consider the service quality model in detail for this dissertation, because of its symmetry to the PQ model, the fact that it is still not released as a standard and does not propose quality measures but refers to the product quality model.

The SQuARE series proposes a basic set of measures, including measurement functions for the different quality models. ISO/IEC 25022 [ISO11c] deals with quality measures related to the quality in use. Measures for the indication of (sub-) characteristics of the product quality can be found in

ISO/IEC 25023 [ISO16] and measures for data quality in ISO/IEC 25024 [ISO15]. A guide to quantify properties which describe the consistent application of QMEs is included in ISO/IEC 25021 [ISO12].

2.3.4 Conceptual Approaches to Quality

The approaches to quality and the related models of ISO/IEC 25010 have a conceptual relation. Software product quality is evaluated by quantifying internal properties. These internal properties, typically measured by static measures, influence external properties of a software product. In turn external properties, which are typically measured by quantifying the behavior of code execution, influence the quality in use. The quality in use is represented by the effect of a software product which is dependent on the contexts of use.

All these properties influence each other and the resulting quality, as abstracted in Figure 2.4

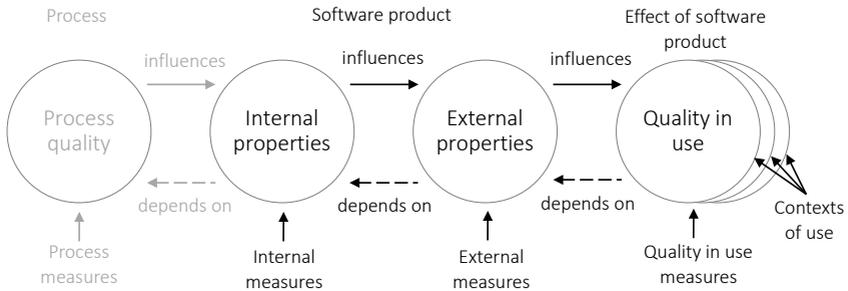


Figure 2.4: Conceptual approaches to quality (based on [ISO12])

Figure 2.4 further includes the process-based approach to quality with process quality, which corresponds to the manufacturing quality. Process conformance is evaluated, using process outcomes as evidence [ISO08b]. The concept of process quality originates from the manufacturing quality, described in Section 2.3.1, and assumes that processes with high quality will also produce high-quality products that contribute to the system quality in use. This assumption can lead to process assessments independent of the

product quality [KP96] and the quality in use. Furthermore, the relationship between process quality and product quality is complex as it is influenced by various factors [Wag13]. Therefore, we focus on the product quality and quality in use for this thesis.

Still, as the quality models are conceptualizations, where an entity's quality is abstracted in a model, it needs to be validated through feedback from customers and consumers, which we will further address with our contributions.

2.3.5 Quality Requirements and Evaluation

The quality models and the quality measures are the necessary resources for the elicitation of quality requirements and quality evaluation. Based on stakeholder needs related to a target entity, quality requirements are specified. Stakeholder needs can be stated, implied or unaware [ISO11d]. With the development and the implementation of quality requirements, the target entity must be evaluated whether it meets the specified requirements.

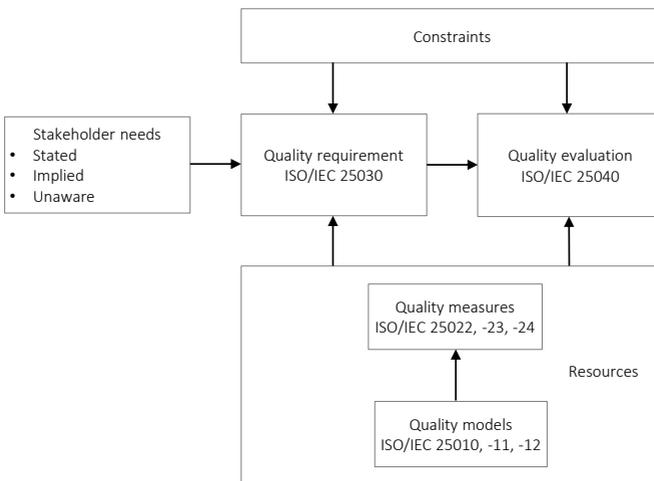


Figure 2.5: Relationship among quality requirements and evaluation (adapted from [Esa13])

As presented in Figure 2.5, ISO/IEC 25030 of the SQuaRE series provides a framework for the elicitation and definition of quality requirements [ISO19b]. For the selected quality (sub-) characteristics of a quality model, quality requirements can be specified by applying quality measures in the system or software design phase. Based on the defined requirements, evaluation of the system and/or software quality can be conducted applying ISO/IEC 25040 [ISO11b]. This standard provides an abstract process for evaluating the requirements specified by applying ISO/IEC 25030. Constraints like the specific customer and consumer needs, costs, or tools and methodologies can influence the quality requirements definition and the evaluation and need to be considered [ISO11b; ISO19b].

ISO/IEC 25040 describes the quality evaluation process as a sequence of five activities (see Figure 2.6). This process is generic and applicable to acquired software/systems and intermediate and final software/systems. In the following, we describe the activities of it:

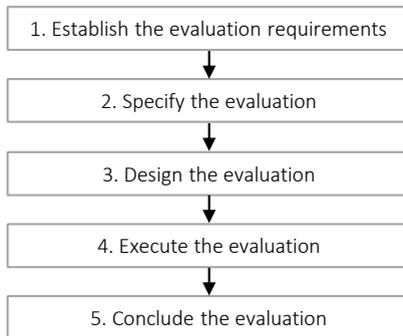


Figure 2.6: Software product quality evaluation process (adapted from [ISO11b])

1. **Establish the evaluation requirements:** In this activity, the purpose of the evaluation is determined. This may differ for assuring, assessing, and predicting the quality of software/system. The specification of quality requirements serves as an input for this activity. According

to the purpose and the stage of the lifecycle, the product parts for evaluation are identified. Finally, the stringency of evaluation in relation to the set of characteristics and sub-characteristics is determined. The evaluation depth depends on the evaluation techniques applied to evaluate the quality (sub-) characteristics.

2. **Specify the evaluation:** With this activity, the quality measures and the target values for the quality measures are determined. The evaluator selects the quality measures to cover the selected quality (sub-) characteristics. The measurement procedure needs to operationalize the quality (sub-) characteristics.
3. **Design the evaluation:** A plan to schedule the evaluation activities, the methods, the tools, and the resources is developed. The evaluation can be performed at different lifecycle stages. Therefore, the evaluation plan must be revised and adjusted iteratively with evolving evaluation activities.
4. **Execute the evaluation:** In this activity, measurements are performed according to the selected quality measures. The measured values are compared against the target values to assess the degree to which the software product/system meets the quality requirements.
5. **Conclude the evaluation:** In this activity, the evaluation results are reviewed, and an evaluation report is created. In addition to the evaluation results, the validity of the evaluation process and the measures applied are discussed and documented. Therefore, limitations, deficiencies, workarounds, and errors related to measurements and human errors are reported. Finally, the evaluation report is provided for feedback to the organization.

2.3.6 Critique of SQuaRE

The international standard series of SQuaRE is a framework for determining quality requirements and evaluating an entity's quality. This framework

includes the relationships among quality models, quality requirements and the evaluation process in detail.

Quality evaluation of an entity depends on the ability to model an entity's characteristics with a quality model, to observe and to quantify properties or indicate them indirectly. A primary deficiency of the SQuaRE series is its abstract nature and therefore requiring high interpretation effort from a user. This may lead to misleading assessment results and i.e., a threat of the validity of results [WGH+15b]. Besides, the SQuaRE series does not consider the change of stakeholder needs over time [ABE19] and neglect considering the operations stage of the software/system lifecycle. Moreover, as ISO/IEC 25010 [ISO11d] explicates, that stakeholder needs and expectations can be stated, implied or unaware. At the same time the standard gives the impression that stakeholder requirements can be deductively defined; here we see a contradiction. In practice, requirements are incomplete. Particularly unaware expectations and needs can only be unveiled by the individual context of use of a stakeholder [Bev95; Ulw02]. We identify, that the SQuaRE series tends to focus on normative verification against requirements but not on validation against a subjective context of use and the detection of anomalies in operations. This can lead to systems being optimized to whether the implementation conforms to the requirement specification, rather than whether customers and consumers are experiencing anomalies. With a lack of an examination of the operations stage with empirical data and validation of the operationalization of quality characteristics, we identify a deficiency in a continuous and iterative adaption of the quality model which we aim to address with our contribution of the dependability evaluation approach in Chapter 7.

2.4 Service Quality

Primary attention is paid to the development of functionalities of services. However, the quality of service, especially to perform as and when required has an increasing impact on customer and consumer satisfaction and is a

critical success factor for organizations [JG03; LV16].

Similar to the general definition of quality in Section 2.3.1, IT service quality is the “*degree to which an IT service satisfied stated and implied needs when used under specified conditions.*” The concept of quality of service (QoS) in the domain of software systems considers dynamic behavior, i.e., the non-functional runtime characteristics such as reliability, availability, and performance of a system to satisfy consumer needs.

One of the most common concepts for indicating and communicating about service quality are the concepts of availability [TMD20]. There are several concepts and measures in research and practice for the term availability. According to Avizienis et al. [ALRL04] “**availability** is a system’s readiness for correct service.” ISO/IEC 25010 [ISO11d] refers to availability as the: “*degree to which a system, product or component is operational and accessible when required for use*”. The central questions in defining availability are:

- Availability of what?
 - The availability of the performing entity or
 - the availability of a provided value and the quality of it?
- Furthermore, how to indicate availability?

According to ISO/IEC 25023 [ISO16], system availability can be indicated by the proportion of the system operation time actually provided and the system operation time specified in the operation schedule. This indication for availability is usually applied in Service Level Agreements (SLAs).

With this concept, a system considered from dual logic, is either available or unavailable. This dual concept of availability indicates product quality but does not provide any information about the provided value and the experience of service with its quality from the viewpoint of a servee. It does not give information about the experience: “*to perform as and when required*” [IEC15]. This means:

- **to perform** (a value provision)
- **as** (optimized according to a quality model)

- **and when** (the context as a function of time)
- **required.**

For example, it may occur that during a short downtime, many attempts of a consumer have failed. A time-based measure for system availability does not provide information about how many attempts have failed to receive value. Usually, a consumer does not care whether a system is up or down but he cares about consuming value “*as and when required*”. In other words, important for a consumer is the performance of required functionality at a required time in a required quality, providing value in the consumer’s context of use. The required quality may also include the time behavior of a value provision. In 2017 Google published a study that shows that if the page load times extend from one to five seconds, the probability of mobile site visitors bouncing increases by 90% [Goo17]. The study indicates that the quality of time behavior of a service, such as the turnaround time of a transaction, which consumers experience and implicitly demand and address with the concept of availability, is of increasing importance.

According to Google’s Site Reliability Engineering (further explained in Section 2.8) an availability concept with system uptime and downtime is not suitable in the context of distributed and fault-tolerant systems. With the approach of isolating faults, Google states to be “*partially up at all times*” [BJPM16]. Therefore, they define availability by counting a request success rate. In Rosenberg et al. [RPD06], we find a similar approach with the concept of dependability and the sub-characteristic accuracy, where the accuracy of a service is defined as the “*success rate produced*” by the service.

For this dissertation, we differentiate ourselves from the concept of system availability. Instead, we are interested in a system’s dynamic behavior providing service to satisfy customer and consumer needs; in other words, to evaluate and control if its service “*performs as and when required*” [IEC15] with the concept of **dependability**. With this concept, we do not consider the components in isolation but observe and quantify the performance of the functionalities of the components in the context of an IoT transaction to provide value to a customer or a consumer. We present and explain the

concept of an IoT transaction in the contribution in Chapter 6.2.1.

2.4.1 Perform As and when Required – Dependability

In the following, we discuss different concepts of dependability and reliability as they contain dynamic characteristics of system behavior on which the concept “*perform as and when required*” [IEC15] can be mapped.

According to IEC [IEC15] **dependability** “*is the ability to perform as and when required and is time dependent in application*”. It is an aggregated concept for the time-related quality characteristics of an entity.

The IEEE standard glossary [IEE90b] defines the concept of **reliability** as the “*ability of a system or component to perform its required functions under stated conditions for a specified period of time.*”

Sommerville [Som04] takes a user perspective and defines a concept of **user-oriented reliability**. “*Informally, the reliability of a system is the probability, over a given period of time, that the system will correctly deliver services as expected by the user.*” Sommerville presents no explicit description what is considered as functioning. However, he applied the definition “*correct*” in relation to user expectations.

This aspect of user centricity, which corresponds to consumer centricity in this dissertation, can be also found at Cheung’s [Che80] concept of **reliability**. “*From the users point of view, the reliability of the system can be measured as the probability that when a user demands a service from a system, it will perform to the satisfaction of the user.*” For concepts applying user satisfaction as a measure for reliability, characteristics such as time behavior, can be included.

In Avizienis et al. [ALRL04] we find two different concepts for **dependability**. First, they define dependability as “*the ability to deliver service that can justifiably be trusted.*” Avizienis et al. further provide an alternate concept in which “*the dependability of a system is the ability to avoid service failures that are more frequent and more severe than is acceptable.*”

The definitions show that the concepts **reliability** and **dependability** are closely related, as both refer to the dynamic behavior of a system, com-

ponent or service. As we are especially interested in a service to “*perform as and when required*” we will apply the concept of dependability for this dissertation. As dependability is “*a collective term for the time-related quality characteristics of an*” entity [IEC15] it focuses on consumer experience, the performance of activities to deliver value, not on a dual approach of component or system uptime. The consideration of dependability enables to define context-dependent quality characteristics for IoT transactions. Therefore, for fault-tolerant systems, which allow degraded quality (further explained in Section 2.6), this can be quantified through measures for different sub-characteristics.

2.5 Anomaly Classification

The indication and control of quality is based on the observation and detection of anomalies. According to Wagner [Wag08], there exist several anomaly and defect classifications with different purposes, such as enhancing defect detection and training developers to enable precise communication. One of them is the IEEE standard 1044-2009 [IEE09]. The standard provides an approach to classify software anomalies and models the relation between software anomalies and maintenance activities [IEE09]. The IEEE 1044-2009 introduces different concepts for anomalies among them, problem, failure, fault, and defect.

An anomaly according to IEEE 1028-2008 [DD90], standard for software reviews and audits, is “*any condition that deviates from expectations based on requirements specifications, design documents, user documents, standards, etc. or from someone’s perceptions or experiences. Anomalies may be found during, but not limited to, the review, test, analysis, compilation, or use of software products or applicable documentation.*” The term anomaly is generic and can be applied for terms such as error, fault, failure, incident, flaw, problem, glitch, defect, gripe or bug. People may associate different meanings with the same term for a concept and apply different terms referring to the same concept [IEE09]. In direct person-to-person communication, ambiguities re-

garding an anomaly concept can be mitigated by providing context. However, in large organizations, misunderstandings can occur when geographically distributed people with different cultural and language backgrounds, or coming for specialized domains, need to communicate and exchange information effectively. Therefore, IEEE 1044-2009 [IEE09] provides a common logical model defining each concept of anomaly and model the relationships between the concepts, focusing on failure, fault, and defect. IEEE 1044-2009 [IEE09] provides the following definitions for these concepts of anomalies.

- **Problem:** Difficulty or uncertainty experienced by one or more persons, resulting from an unsatisfactory encounter with a system in use. (adapted from ISO/IEC 24765:2009 [ISO17b]).
- **Failure:**
 - An event in which a system or system component does not perform a required function within specified limits. (adapted from ISO/IEC 24765:2009 [ISO17b]).
 - Termination of the ability of a product to perform a required function or its inability to perform within previously specified limits. (adapted from ISO/IEC 25000:2005 [ISO05]).
- **Fault:** A manifestation of an error in software. (adapted from ISO/IEC 24765:2009 [ISO17b]).
- **Defect:** An imperfection or deficiency in a work product where that work product does not meet its requirements or specifications and needs to be either repaired or replaced. (adapted from the Project Management Institute [Ins08]).

Figure 2.7 models the relationships of anomalies and maintenance activities in an entity relationship diagram. Software change request (SCR), software release, and adaptive and perfective maintenance are included in the entity relationship diagram to clarify the scope but are not further addressed in the standard.

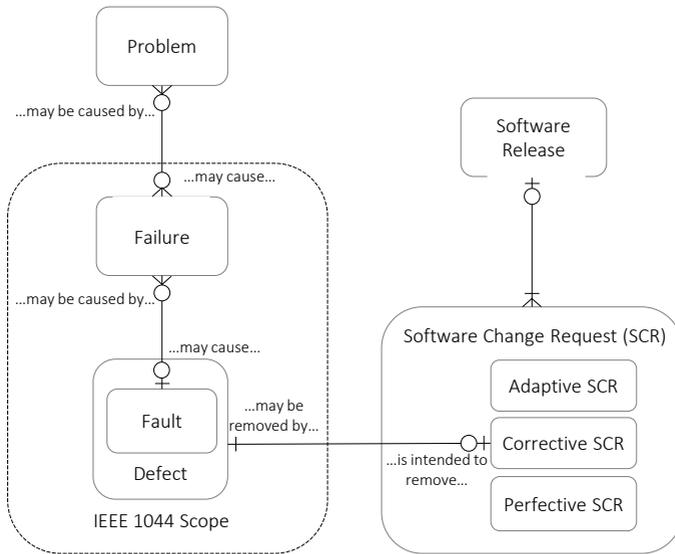


Figure 2.7: Relationships of concepts of anomalies modeled as an entity relationship diagram (based on [IEE09])

The relationships of the concepts of anomaly are explained in Table 2.1.

Table 2.1: Relationships of concepts of anomalies of IEEE Std 1044 (based on [IEE09])

Class/entity pair	Relationships
Problem – Failure	A problem may be caused by one or more failures. A failure may cause one or more problems.
Failure – Fault	A failure may be caused by (and thus indicate the presence of) a fault. A fault may cause one or more failures.
Fault – Defect	A fault is a subtype of the supertype defect. Every fault is a defect, but not every defect is a fault. A defect is a fault if it is encountered during software execution (thus causing a failure). A defect is not a fault if it is detected by inspection or static analysis and removed prior to executing the software.
Defect – Change Request	A defect may be removed via completion of a corrective change request. A corrective change request is intended to remove a defect. (A change request may also be initiated to perform adaptive or perfective maintenance.)

In summary, the IEEE 1044-2009 standard provides a uniform model for the classification of software anomalies. It targets to enable precise

communication and information exchange about the different classes of anomalies across teams, and organizations. The logical concept of IEEE 1044-2009 builds the basis for our classification of system anomalies, presented in Chapter 5.

2.6 Fault Tolerance and Degraded Quality

With the increasing complexity of distributed systems, the probability of faults is increased. To avoid service failure in the presence of faults, these systems need to be designed fault tolerant [ALRL04]. Especially in the domain of functional safety of electrical and electronic systems the concept of fault tolerance is essential. The international functional safety standard ISO 26262-1:2018 [ISO18a] defines fault tolerance as the “*ability to deliver a specified functionality in the presence of one or more specified faults*”. The standard models the following time intervals for handling faults which are illustrated in Figure 2.8.

- **Fault Detection Time Interval:** time-span from the occurrence of a fault to its detection.
- **Fault Reaction Time Interval:** time-span from the detection of a fault to reaching a safe state or reaching emergency operation.
- **Fault Tolerant Time Interval:** minimum time-span from the occurrence of a fault in an item to a possible occurrence of a hazardous event, if the safety mechanisms are not activated.

If no safety mechanism is implemented, a fault can propagate to a malfunctioning behavior, which corresponds to a failure and in turn, can result in a hazardous event. If a system has a safety mechanism implemented and detects a fault, the system is transferred with a fault reaction to a safe state. The safe state is an operation mode, where the system is no longer dangerous in case of a failure. This safe state depends on the type of overall system [ISO18a]. The concept of fault tolerance includes maintaining the

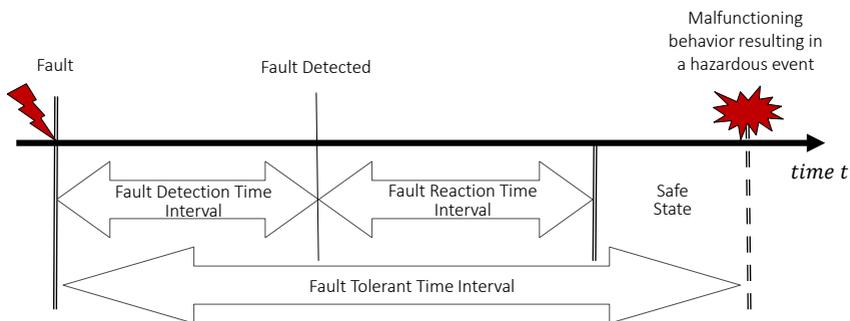


Figure 2.8: Safety relevant time intervals (adapted from [ISO18a])

full functionality of a product or service, e.g. by activating redundancies but also includes the operation in an degraded mode.

In the domain of distributed systems, the concept of degraded quality is used to express that a system still operates at a reduced level of performance in case of faults or provides the basic functionality to achieve a subset of outcomes [ALRL04; DHWC08]. Serving stale data is one common strategy to provide degraded quality [Clo18]. A recent example from several video stream providers in February 2020 was to degrade the resolution of the streamed contents. This was a reaction to the COVID-19 pandemic situation, during which an increasing streaming demand of people being full-time at home have occurred [Swe20].

For providing degraded quality, it is necessary to define a system state as operational, other than providing complete functionality [DGG+08]. Therefore, an organization needs to define a minimal functionality derived from customer or consumer needs.

2.7 Observability and Monitoring of Distributed Systems

Due to the high frequency in changes, the dynamics and the complexity of distributed systems, there is a shift of quality assurance from extensive system test to observation and monitoring in operation [HHK+17]. In the fol-

lowing sections, we give an overview about the concepts of observability and monitoring. Furthermore, we present application performance management as a discipline of IT operations and give an overview about the approach of distributed tracing, to approximate causal relationships of workflows along distributed architectures.

The concept observability originates from control theory and measures the degree to which a system's internal state can be determined from its output [Gop93]. In the domain of distributed systems, observability indicates to what degree infrastructure and applications and their interactions can be observed [PPM+18].

IEEE defines monitoring as the supervising, recording, analyzing, or verifying the operation of a system or component [IEE90a]. According to Beyer et al. [BJPM16], monitoring is the process of “*collecting, processing, aggregating and displaying real-time quantitative data about a system.*”

According to different stakeholder concerns, there are different purposes for monitoring. Below, we state some of the most frequently described purposes:

- situation management (often referred to as troubleshooting) contains the detection and control of anomalies in operations which may have impact on the value provision to a customer or a consumer [ABDP13]
- postmortem analysis, to detect defects and correction options [BJPM16]
- capacity management, to handle different workloads [ABDP13; FEH+14]
- billing, to charge consumption-based fees [ABDP13; FEH+14]
- product improvement [ABDP13; BJPM16].

As monitoring increasingly replaces upfront extensive system test, in industry often a new version is firstly released to a subset of consumers to detect anomalies, before rolling it out to the whole consumer base. This strategy is called canary release [Dan14]. For example, Netflix is using automated canary analysis with real-world traffic of consumers to detect unexpected system behaviors [BBHR16].

2.7.1 Application Performance Management

Application Performance Management (APM) is a discipline of IT operations to collect and analyze application behavior at runtime. Its purpose is to continuously monitor and control the performance of software applications [ABC+16]. Agents are applied to collect dynamic data from heterogeneous systems implemented with diverse technologies [HHMO17]. The expected behavior of applications can be indicated with baselines. However, it is difficult for distributed systems architectures to determine normal behavior due to the constant changes in systems and their dynamic behavior [HHK+17]. The domain of APM represents a fast-growing market with commercial as well as open-source solutions [HHMO17]. According to Gartner, the most mature and leading solutions are AppDynamics, Dynatrace, and New Relic [CS20], which we applied for our contribution in Chapter 7.

2.7.2 Concept of Distributed Tracing

In the past decade, distributed tracing was developed as a method to analyze distributed system behaviors. Distributed tracing, also known as end-to-end tracing, is a method to approximate a workflow path in a distributed system. One of the first tracing systems is the industry project Dapper [SBB+10]. Further scientific projects such as Pinpoint [CKF+02], Pip [RKW+06], Magpie [BIMN03], and Pivot Tracing [MRF18], reflect the scientific interest in the topic. Organizations like Google, Uber, and Facebook invest in developing tracing tools, especially in the last decade, due to the increasing application of large and complex distributed systems [Mac18].

In the following, we describe the concept of distributed tracing based on “Dapper” [SBB+10], which was published by Google in 2010 and the data model of OpenTracing, which is an open-source tracing community, launched in 2016, representing a vendor-neutral standard for instrumentation. Dapper is designed for productive usage in large distributed systems and serves as a template for many tracing frameworks, such as open-source implementation like Zipkin [Zip20] and Jaeger [Jae20], that we applied for the contribution

in Chapter 7.

Fundamental for distributed tracing is the concept of a span. A span approximates an operation as part of a workflow which is handled by multiple components. Every span has a name, describing the operation and a start and end time, to determine a span duration. Every span propagates span context, which enables the reconstruction of an instance of a workflow, represented as a trace. The span context includes different elements. It contains the parent span ID, which is the context relating a child span to its parent, and the trace ID, which defines the trace, a span is related to. Each span creates an individual ID and propagates its ID as the parent span ID and the trace ID within the span context to a child span. Spans without a parent are called root spans [SBB+10]. A trace can be represented as causal relationship with a directed acyclical graph (DAG) (see Figure 2.9) and as temporal relationship with a time bar (see Figure 2.10).

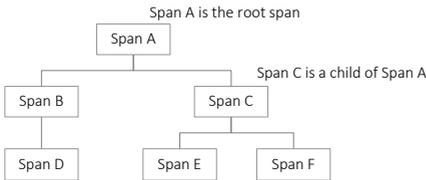


Figure 2.9: Causal relationship as DAG (adapted from [Ope20])

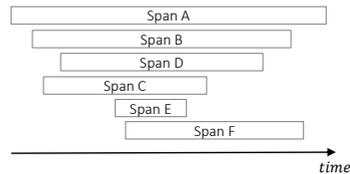


Figure 2.10: Temporal relationship as time bar (adapted from [Ope20])

2.7.2.1 Tracing Overhead and Sampling

Concerning trace generation and trace collection, distributed tracing comes with a performance degradation of the system. However, low performance overhead is a key design goal of modern distributed tracing systems. Strategies of sampling can control the overhead generated by tracing. With sampling, a fraction of all possible traces are recorded. There are different strategies of sampling: based on a fixed sampling probability or adaptive sampling. Adaptive sampling defines a targeted rate of sampled traces per

unit time. For example, in a workload with less traffic, the sampling rate is automatically increased and for high traffic the sampling rate is decreased to control the overhead [SBB+10].

2.8 Site Reliability Engineering

Site Reliability Engineering (SRE) [BJPM16] is a discipline that originates from Google. It applies software engineering methods and procedures to operation and infrastructure problems with a focus on customers and consumer experience. SRE constantly evolves with its growing application in different organizations from various industries and sizes [MF20]. Famous adopters are Netflix, Spotify, and Microsoft, but also in organizations from other industries like retail, financial services, and manufacturing, SRE is an emerging practice. [MF20].

2.8.1 Concepts of SLI and SLO

A core principle is the derivation and evaluation of system behavior, most important from a consumer or customer view [MF20]. This is done by applying the concepts of service level indicators (SLI), which are measures indicating quality characteristics and service level objectives (SLOs), which represent target values of service level indicators and are verified against the measures at operation stage [BJPM16].

In the center of SRE is the concept of reliability. With the SLOs, a target level for the reliability of a service is determined. SLIs are only determined for the most important characteristics from the customer or the consumer view. SRE states four key SLIs, the “golden signals of monitoring” *namely latency, traffic, error, and saturation*. *Latency* represents a measure of the time it takes to serve a request. With the signal *traffic*, the number of requests are measured. The rate of failed requests is measured by the rate of *errors*. Lastly, the signal *saturation* is a characteristic with specific measures to indicate resource utilization.

With the focus on key SLIs and the definition and evaluation of target values with SLOs, SRE tries to balance the development of functionalities versus the investment in service reliability. SLOs represent shared reliability goals for development, operations, and product teams. McCoy et al. [MF20] state, that without SLIs and SLOs “*you cannot effectively decide between investing in functionality that will win and retain customers, and investing in reliability and scalability that will keep customers happy.*” Therefore, SLIs and SLOs represent means for prioritizing operation efforts and engineering work [BMR+18].

2.8.2 Time-Based Availability vs. Aggregate Availability

SRE presents two measures for their concept of availability, applied to define SLOs; *time-based* and *aggregate* availability. The following measurement function represents the time-based availability:

$$\text{time-based availability} = \frac{\text{uptime}}{\text{uptime} + \text{downtime}}$$

This corresponds to the traditional view of system uptime that we described in Section 2.4. For example, a system’s acceptable downtime can be expressed with an SLO of $\geq 99.99\%$ time-based availability, meaning 52.56 minutes of annual downtime (in a year of 365 days), can be tolerated not to violate the SLO.

Google describes that this approach of a time-based measure is not meaningful regarding their distributed service system. If faults occur, they are able to isolate the faults, and therefore, they can serve “*at least a subset of traffic for a given service somewhere in the world at any given time (i.e., we are at least partially up at all times).*” Instead of applying a measure for uptime, they indicate availability in terms of a request success rate and call this measure (SLI) aggregate availability.

$$\text{aggregate availability} = \frac{\text{successful requests}}{\text{total requests}}$$

For example, considering a system serving 2.5 million requests per day with an SLO of 99.99% of aggregate availability, meaning up to 250 requests can fail daily not to violate the SLO [BJPM16].

This measure of aggregate availability inspired our consideration of the concept of dependability, as described in Section 2.4.1 and the derivation of measures in our contribution in Chapter 7.2.

2.8.3 SRE Adoption in Industry

In 2020 Google published a survey [MF20] with 572 industry professionals representing the state of practice of applying SRE principles, focusing on SLOs. The respondents were coming from development and operations of heterogeneous industries dominated by technology. Industries surveyed include financial services, e-commerce, healthcare, industrial & manufacturing, telecommunication, and media & entertainment. The participants mainly came from North America (42%) and Europe (35%).

The study indicates that applying SLOs is a relatively young practice. 54% of the participants do not currently use SLOs; however, half of the 54% plan to implement it in the future. Of the remaining 46% who implemented SLOs in their organizations, 44% started applying SLOs in 2019. 66% started implementing it within the last three years. The data indicate that larger organizations have more experience in applying SLOs, as around 35% of companies with more than 10,000 employees have SLOs in place for five years or longer [MF20].

2.9 Situation Management and Postmortem Analysis

The following two sections introduce concepts related to the handling of anomalies. We introduce situation management for controlling the system and situation and postmortem analysis for the subsequent analysis of defects.

2.9.1 Situation Management

Research on situations and management of situations is performed by different scientific domains such as psychology, cybersecurity management, artificial intelligence, defense, and military [JBL07]. Situation management comprises sensing and collecting information, selecting what is important to perceive situations, and taking actions performed in a control loop to control a situation [JBL07; JLM+05]. Situation management is increasingly applied in the context of managing complex dynamic operations and systems [JBL04; JLM+05]. In our case of complex IoT service systems, situation management aims to keep the customer's and consumer's trust.

Approaches for observing, reasoning, and controlling situations appear in various forms in literature, e.g., with the OODA-loop. The OODA (observation-orientation-decision-action)-loop is an information strategy concept assigned to situation management and was developed by military strategist John Boyd in the 1950s [Boy96]. The concept includes the observation and orientation in a situation and the subsequent decision and performance of tactical options to control the situation. With changes in the environment, an organization continuously runs through the decision loop and iteratively adapts and improves its behavior while being able to improve the capability for decision making [ZZ17]. The approach is applied beyond military strategies and finds implementation in cybersecurity [ZZ17] and business intelligence [Mid07]. In the domain of software systems, the practice of managing situations of service failures is also referred to as troubleshooting, firefighting, or incident response.

In situation management, the organization acts in a kind of “task force” mode. The organization needs to handle anomalies of the dynamic system behavior. The organization primarily performs the activities from a tactical perspective with a goal of keeping the customer's and consumer's trust. While the organization needs to be “effective before efficient”, often corrective actions from a strategic viewpoint are not taken.

2.9.2 Postmortem Analysis

After a system and situation are under control, with the concept of postmortem analysis, service failures are analyzed in retrospective [BJPM16]. A postmortem analysis focuses on investigating anomalies in the effect chain and examining how well the organization reacted during situation management. Postmortems are an essential part of Google’s SRE discipline [BJPM16].

After a situation is under control, people tend to search for isolated individual causes with the approach of root cause analysis [Coo98]. According to Dekker et al. [DHWC08], the conceptualization of a linear chain of events, with a root cause, one trigger at the start of the chain, similar to a domino model, is an oversimplification. Such a linear chain of events does not examine the complexity and dynamics in cause effect chains [Coo98; DHWC08; Kah11]. Humans tend to overestimate the predictability of a situation and failure in retrospective, called “hindsight bias” [Coo98]. Therefore, root cause analysis can easily be a subject of human fallacies. As stated by Cook [Coo98] for distributed complex systems, there is no isolated root cause. In contrast, there are dynamic interactions among system components. There are multiple faults and contributors, where each fault is necessary, but they are only jointly sufficient for failure [Coo98]. Since complex systems are socio-technical systems that contain not only technology but also interactions with people, their relationship and dynamic interactions need to be examined. Thus isolating a single *root cause* is not possible; rather, it is a reflection of a “*social, cultural need to blame specific, localized forces or events for outcomes.*” [Coo98]. As a contrast to a “blame culture”, the concepts of an “blameless” or “just culture” were developed in the context of healthcare and aviation [Dek12; FLD06]. A “blameless culture” and “blameless postmortems” [BJPM16] are based on systems thinking with the goal of continuous improvement of the organization. Hence, for this dissertation, we differentiate ourselves from the concept of root cause analysis and assigning blame. In contrast, we investigate in dynamic effect chain and assign defects with options for defect correction, as described in our contribution in Chap-

ter 5. Compared to situation management, postmortem analysis and defect corrections are performed by organizations from a strategic perspective, with the goal of profitability by being efficient.

CHAPTER
3

STATE OF THE ART

Based on the previous chapter presenting the background for this dissertation, we describe in this chapter further studies representing the state of the art in the areas that we address with our contributions. First, we analyze studies that are related to our interview study about monitoring and observability of distributed systems. Afterward, we discuss different classifications of anomalies. Following that, we summarize and discuss approaches for the definition of quality requirements. Moreover, we present a study evaluating quality requirements and studies of observability and monitoring methods. This chapter includes contributions that are similar to our studies or that influenced our work. We describe and discuss limitations for every related work, which we tried to address with our contributions. Where we considered it necessary, we discussed the differences and similarities of related studies in more detail in the related chapters of our contributions.

3.1 Studies on Observability and Monitoring

In the following, we describe preceding surveys in the context of monitoring and observability of distributed systems.

In 2015, an empirical multi-case study by Olsson and Bosch [OB15] with six global software-intensive companies revealed several difficulties in validating customer needs. One of them is the “*open loop*” problem, which represents a situation where product or service decisions are made based on a “*gut feeling*” instead of customer feedback or customer observation. Hence, product management decisions are not aligned with actual customer needs. The study identifies that due to a lack of customer feedback, an increasing number of developed functionalities are not proven valuable from a customer viewpoint. According to Olsson and Bosch [OB15], a common view in industry practice is that requirements are regarded as “*truth*”. However, requirements have to be treated as hypotheses that need continuous validation. Hence, requirement specification is one of the most challenging tasks and is mission-critical, as projects may fail, because they do not meet customer requirements. Although the studied companies described having a considerable amount of monitoring data, they cannot sufficiently use it. The available data is not applied for feedback cycles, as the companies struggle to ask the “*right*” questions [OB15]. Similar to Olsson and Bosch [OB15], our interview study identified that practitioners struggle with the specification of requirements and the validation of the assumptive requirements through operation. Furthermore, the study of Olsson and Bosch [OB15] supports our finding of a lack of goal-oriented observation and monitoring, leading to a plethora of unused monitoring data.

In 2013, Aceto et al. [ABDP13] performed a literature survey with a focus on cloud monitoring. They stated motivations, “then-current” tool support, and challenges. The survey correctly predicted the rise in complexity and dynamics of cloud architectures and proposed actions to handle these, such as diagnosis, filtering/summarizing data, and cross-layer/cross-platform monitoring.

Similarly, another early survey (2014) of cloud monitoring tools by Fatema et al. [FEH+14] identifies different challenges regarding cloud monitoring tool capabilities such as scalability, robustness, interoperability, and customizability. They state that most cloud-specific monitoring tools are developed by commercial cloud providers, which make them platform-dependent and

proprietary, and that tools from academia are not production-ready. The lack of these capabilities indicates the speed of innovation in that field of distributed cloud-based systems.

Natu et al. [NGSR16] discuss monitoring challenges of cloud applications in the context of enterprise applications. They identified the scale and complexity of applications as the main challenge. Their work confirms our observation of the segmented monitoring of the distributed systems. Different teams observe and monitor the individual components and system elements with different tools and cannot create a unified view of the system behavior regarding the quality of service and fault detection. Natu et al. [NGSR16] identified needs such as a comprehensive system-wide solution overcoming the traditional isolated monitoring of individual components. Furthermore, adaptive monitoring solutions, noise reduction by providing contextual information, and a shift from reactivity to proactivity of observation are needed, which we can confirm with the findings of our interview study.

Heger et al. [HHMO17], in 2017, provide an overview of the state of the art of Application Performance Management (APM). They describe typical activities, capabilities, and APM solutions. They found several challenges, such as frequent releases causing a challenge to define normal system behaviors with baselines and the challenge of fault detection and diagnosis in distributed system architectures, which are constantly in change. Similar to the findings of our interview study, Heger et al. [HHMO17] state that monitoring is not a purely technical activity but needs to be aligned with business concerns and vice versa. Furthermore, they highlight a need for technology evaluation with real-world APM data.

In 2018, Sfondrini et al. [SML18] conducted a survey with more than 60 multinational companies on public cloud adoption and assessed technical and business aspects. While the application of public cloud infrastructure is on the rise, critical aspects like security, regulatory compliance, and monitoring remain. Regarding monitoring, the survey has shown that half of the companies rely solely on their cloud providers' monitoring dashboard and are not able to extract valuable information from the monitoring data.

Participants stated a need for monitoring of the service quality, which has to be integrated into their monitoring tool.

In 2019, Knoche and Hasselbring [KH19] conducted a survey of German experts on microservice adoption. Barriers to adoption are mainly operational. Running distributed microservice-based systems are prone to partial failures, and monitoring them is a significant challenge.

Tamburri et al. [TMD20] investigate with a quantitative industry study from 2020 the state of the practice in cloud application monitoring. With interviews and a web survey, including more than 140 practitioners from over 70 organizations, they analyzed applied monitoring practices, including solutions and challenges. The study found a lack in the implementation of structured approaches for cloud application monitoring and continuous quality improvement. Their results confirm our interview study's observation from 2019 that there are deficiencies in the mindset of industrial decision-makers, not considering monitoring as a company key asset. Furthermore, the study found a correlation of service failures with a lack of monitoring. The study also substantiates our qualitative observation of reactive implementation, where consumers instead of service providers discover failures. Moreover, the study explicates that while there are many technical solutions, organizational and technical standards are main limitations.

Analyzing the studies regarding monitoring and observability, they either focus on drivers and challenges or available solutions in science or from a tooling perspective but do not integrate them. We aimed to address this gap with our interview study, presented in Chapter 4, which takes a holistic view. We provide empirical industry-focused research with in-depth interviews, where we study individual problem instances. With our study, we combine different perspectives to find out which solutions and strategies are applied by the organizations and to what degree they overcome the existing challenges. This is necessary to apply appropriate existing technologies and approaches in practice and adapt them accordingly to increase their quality in use.

3.2 Classification of Anomalies

A widespread classification of anomalies is the taxonomy of dependable and secure computing of Avizienis et al. [ALRL04]. The authors address the threats to dependability and security by the concepts of *failure*, *error* and *fault*. Moreover, they discuss fault tolerance techniques by handling errors and faults. Their concept of *failure* is described as a deviation from the correct service. An *error*, for them represents “*the part of a system’s total state that may lead to a failure*” [ALRL04]. Their concept of *fault* characterized as the “*cause of the error*” [ALRL04].

In software engineering the term *error* is ambiguous as the term is applied for different concepts. ISO/IEC/IEEE glossary 24765 [ISO17b] presents three different concepts for it.

- “*Human action that produces an incorrect result.*”
- “*Difference between a computed, observed or measured value or condition and the true, specified or theoretically correct value or condition.*”
- “*Erroneous state of the system.*”

The view of Avizienis et al. [ALRL04] on the relationship between their concept of *error* and *fault* indicates the consideration of a linear chain of events, as described in Section 2.9. Furthermore, Avizienis et al. [ALRL04] present the concept of system function, behavior, and structure, which have similarities to our systems views stated in Chapter 5. In contrast to our classification of system anomalies, they do not relate their concepts of anomalies to system views. We will further discuss the similarities and differences of the work of Avizienis et al. [ALRL04] in relation to our contribution in Chapter 5.

The standard classification of software anomalies, IEEE 1044-2009 [IEE09], presented in Section 2.5 focuses on anomalies regarding the system element software and is not targeted to anomalies on the system level.

In summary, both classifications do not explicitly analyze the handling of system anomalies regarding different system views. They do not differentiate

the design and operation of fault tolerant systems and sustainable system correction. Hence, no explicit differentiation of the dynamic system view to control the quality of service respectively and the system structure view to assign anomalies and correction options for sustainable correction is possible. Therefore, we propose the adaption of the IEEE 1044-2009 [IEE09] with our contribution in Chapter 5. Our contribution introduces three system views: functional system view, dynamic system view, and system structure view and assigned the anomalies of IEEE 1044-2009 [IEE09]. This enables differentiation between the handling of fault detection and dynamic fault reactions respectively and the handling a defect and its options for sustainable defect correction, not only on system element software but generally on system level.

3.3 Evaluating Quality Requirements

Evaluating quality requirements includes the elicitation based on customer or consumer needs, their operationalization, and evaluation during operation. In the following, we discuss studies focusing on the definition of quality requirements and their evaluation. Finally, we present studies examining observation approaches that specifically apply distributed tracing.

3.3.1 Defining Quality Requirements

For services that offer different functionalities, not all are equally important from a customer or consumer viewpoint and do not have the same business value. For example, failing a sign-up workflow is more critical than failing polling new emails in the background [BJPM16]. To identify common tasks and critical activities of consumer and customer interaction with a service the concept of *customer, user or consumer journey* is applied [KPT18; MDC15; MF20]. In the context of services, the customer journey is the representation of the customer viewpoint of the process of service delivery [SSAL11]. It is the perception and experience of a service interface along with a time axis [MK09]. Lemon et al. [LV16] describe it as a dynamic flow. In the

context of SRE, the customer journey represents a sequence of tasks that is of high importance to a consumer, experiencing a service [BJPM16]. Within the SRE approach, SLIs and SLOs are defined from a customer journey perspective and measured at an observation point.

There exist different goal- [RS05; Van01] and task- [LK12; ROS98] driven requirement elicitation approaches. According to Fotrousi et al. [FFF14], the application of goal models does not sufficiently support the quantification of quality requirements and, therefore, is of limited help to make its conformance operationalizable through measures. One reason for neglecting the derivation of quality requirements is that *“unlike functional requirements, most quality requirements represent emergent properties of the system, which appear on a set of components, not on a specific component”* [ISO19b] and are dependent on the context of use. Therefore, it is challenging to define traceable quality requirements, which can be implemented and verified along the product lifecycle and validated in the context of use during operation [ISO19b]. Furthermore, goals are often defined on the abstraction level of a system or a service and not related to a specific functionality performed in a workflow [AP98; Van01].

The concepts of *task* and *goal* are also part of the SQuaRE series [ISO14]. In ISO/IEC 25065 [ISO19a], the common industry format for user requirements specification, *goals* are defined as *“the intended outcome(s) of use”* [ISO19a]. This corresponds to our view of the value provision of a service as described in Section 2.2.2. *“Tasks consist of one or more activities undertaken to achieve a goal [...] Tasks are described in terms of the activities undertaken by users to achieve an intended outcome”* [ISO19a]. However, the concept of *task* of the SQuaRE series does not include *“organizational procedures that describe how information and resources are interchanged within and across departments within an organization.”* Moreover, the SQuaRE series states that *“Organizational procedures include the users’ tasks.”* [ISO19a]. Therefore, the concept *task* does not represent all activities of a workflow to deliver value to a customer or a consumer that are performed by the provider. We aim to address this gap by conceptualizing an *IoT transaction* as a workflow, for

which we can observe and control the value provision to a customer or a consumer (see Chapter 6). Hence, our concept of IoT transaction includes the consumer task.

The work of Haindl et al. [HPK20] also takes a task-oriented perspective. Their paper presents the TAICOS (Task-Interest-Constraint Satisfying) approach to elicit and model user tasks, for which quantitative constraints within a context of use can be refined and evaluated during the software lifecycle. The paper presents the TAICOS approach and its evaluation with an exploratory interview study with eleven domain experts. The approach aims to elicit functional requirements from major customer tasks of an enterprise system and to specify quantitative non-functional requirements for them. Haindl et al. [HPK20] define the concepts of *task* as a “*sequence of actions that people perform to attain a goal*”, which fits the concept of *task* from the SQuaRE series [ISO19a]. Therefore, we refer to the discussion of the paragraph before, considering the task concept of the SQuaRE series. In presenting the TAICOS approach Haindl et al. [HPK20] extend their concept of *task* with different perspectives, including user’s intentions and interactions and system responsibilities supporting the intentions. This comes even closer to our concept of an *IoT transaction*. However, as our work is grounded in the concepts of the SQuaRE series, applying the concept of *task* would be contradictory, as a *task* is an element of a workflow, and thus an element of an *IoT transaction* in our work. The TAICOS approach contains eight steps that can be mapped on the activities of our transaction-based approach of dependability evaluation. We found several similarities to our concept of the dependability evaluation approach, but TAICOS does not focus on the value provision from a customer and consumer viewpoint. We will compare the procedure of the TAICOS approach in more detail in Chapter 7.

In [HPK19], Haindl et al. developed an approach to specify and evaluate non-functional requirements on feature level in the context of enterprise applications. The approach is based on the QUAMOCO model [WGH+15a], which is a meta quality model. They extended the QUAMOCO model with the concepts of features and constraints. Haindl et al. [HPK19] understand a *feature* as “*a unit of functionality of a software system that satisfies a re-*

quirement, represents a design decision, and provides a potential configuration option” [KA11]. Their concept of *feature* has similarities to our concept of *IoT transaction*, that we present in Chapter 6. For Haindl et al. [HPK19] features encompass functionalities of applications. In comparison, our *IoT transaction* concept is derived from the service concept; therefore, it represents the activities of a workflow, providing value to a customer or a consumer and originates from a business-oriented perspective. Besides their *feature* concept is more generic. It is not specifically derived from a viewpoint of a customer or a consumer value but generally from software engineers or other stakeholder viewpoints. Haindl et al. [HPK19] present some exemplary non-functional requirements. In comparison to our work, they do not discuss what “*invocation of the feature must be completed*” means and how to indicate goal attainment for their feature context. Furthermore, they do not consider and model degraded quality of a feature execution. With our concept of an *IoT transaction*, we aim to model the targeted completeness of a transaction, which is perceived as complete by customer or a consumer (see Chapter 6). In summary, similar to Haindl et al. [HPK19], with our concept of *IoT transaction*, we derive different requirements for different transaction types of a service, based on customer and consumer needs (see Chapter 7).

The concept of customer journey, including tasks, focuses on the touch-points of consumer interaction with a business process and does not conceptualize the value provision as a workflow. In contrast to the SRE approach, we define and indicate the quality characteristics transaction-based (Chapter 6) rather than at one observation point. Therefore, we are able to detect faults during the value provision and react to them. The work of Haindl et al. [HPK20] has several symmetries to our concept of *IoT transaction* (Chapter 6) but focuses on software development and operation generally, and do not focus on distributed development and operation of complex IoT service systems. Moreover, their approach is generic for different system stakeholder needs and does not specifically focus on customer or consumer value.

3.3.2 Evaluating Quality Requirements

The study of Olsson and Bosch [OB15] presents a conceptual model for the quantitative customer observation with operation monitoring to accelerate the feedback loop between the customer and the provider [OB15]. They inductively developed the “*Qualitative/quantitative Customer-driven Development*” (QCD) model, which combines qualitative customer feedback with quantitative customer observation. They identify qualitative customer feedback techniques, e.g., interviews or surveys, and quantitative customer feedback techniques, e.g., observation and monitoring of customers and product behavior to validate product functionalities. The study identifies that organizations cannot capitalize on the extensive amount of feedback data collected, as they fail to ask the “*right*” questions. However, Olsson and Bosch [OB15] do not conceptualize how to ask the “*right questions*”. From our perspective, this includes the derivation of the needs of a customer and consumer, including the dynamic properties and how to project these needs onto a quality model. Moreover, they do not explicitly describe and discuss observation methods and how to detect anomalies to validate the hypothesis.

The distributed system architectures and the large number of components with complex communication patterns produce a massive volume of monitoring data. Every new release may fix a fault but can also introduce new faults. With the rapid releases, controlling the dependability of value provision along different components becomes critical. The evaluation process model of ISO/IEC 25040 [ISO11b], described in the background in Section 2.3.5 lacks the integration of the observation of the operation stage with empirical data and validation of the operationalization of quality characteristics. Furthermore, we identified a deficiency in a continuous and iterative adaption of the quality model. We address these issues in our contribution in Chapter 7.

3.3.3 Observability and Monitoring Approaches

This section summarizes research related to monitoring and observability approaches.

Johng et al. [JKHC18] discuss approaches of benchmarking and simulation of the runtime behavior of cloud infrastructures that have shortcomings if they are used separately. Therefore, this paper introduces and validate a complementary approach. Their approach presents a process that maps benchmark ontologies of simulations. In experiments validating the ontological approach, they proved that their approach is leading to more reliable simulation results and better explainability of the simulations.

Lin et al. [LCZ18] propose a novel way to detect performance faults in microservice architectures utilizing causal graphs with the system called “Microscope”. Microscope automatically creates a causality graph without instrumentation of the application.

Gupta et al. [GMDS18] address runtime monitoring on continuous deployment in software development, which is a crucial task, especially for rapidly changing software systems. While current runtime monitoring approaches of previous and newly deployed versions lack in capturing and monitoring differences at runtime, they present an approach that automatically discovers a model of the execution behavior by mining execution logs.

Yang et al. [YWGL18] investigate the capturing of service execution paths in distributed systems. While capturing the execution path is challenging, as each request may cross many components and may be processed on several hosts, they introduce a generic end-to-end methodology to capture the entire request.

Thalheim et al. propose an approach to reduce the number of measures and infer measure dependencies of components applying a predictive-causality model [TRA+17]. The approach reduces overhead and can be applied for auto-scaling and diagnosing system failures. Their approach creates a dependency graph to identify inter-component relationships and infer causal relations among time series. However, as their approach is a black-box approach, it can only collect infrastructure measures of saturation, e.g., CPU

usage or disk I/O. For measures related to the time-behavior of a request or a transaction, they rely on the capability of the microservice applications to export them. In contrast to Thalheim et al. [TRA+17], our contribution reduces observation complexity coming from a business perspective with the purpose of evaluating and controlling dependability by selecting high value component effects.

These works indicate that research on closing the complexity gap between distributed systems and their observation and monitoring is ongoing. As described, there exist different observation and monitoring approaches and tools. However, these solutions are not effectively adopted in practice. To adopt these technologies in the industry, they need to be aligned with the conceptualization of quality of services to provide effective and efficient feedback loops. We identified a lack of concepts to relate the value provision and business logic with observation and quantification approaches to bring together business and technical perspectives.

INTERVIEW STUDY ON OBSERVABILITY AND MONITORING OF DISTRIBUTED SYSTEMS

This chapter contains our contribution of the interview study on the observability and monitoring of complex distributed systems. We relate challenges, requirements, and existing solutions from the viewpoints of different stakeholders of monitoring systems and monitoring tool providers. This study is the basis on which we have explored the problem context of organizations in order to design the artifacts of our further contributions. This chapter is an extension of the following publication:

- S. Niedermaier et al. “On Observability and Monitoring of Distributed Systems – An Industry Interview Study.” In: Service-Oriented Computing, 17th International Conference, ICSOC 2019, Toulouse, France,

4.1 Context and Goals

Different monitoring and observability approaches exist to deal with the complexity and the dynamics of complex distributed systems. We found that there is a lack of empirical studies matching solutions and approaches to challenges and requirements. While many technologies and approaches exist, organizations need to relate these technologies and approaches to the challenges they face. Therefore, this study takes a holistic approach. We provide empirical industry-focused research with semi-structured interviews from different stakeholders of monitoring systems and tool providers. We combine different perspectives to analyze organizations' problem context and relate challenges to requirements and to existing solutions. Our research goal can be defined as follows:

Analysis of the contemporary challenges of monitoring and operating distributed systems to derive requirements and mapping existing solutions and strategies from the viewpoint of different stakeholders of monitoring systems and tool providers.

With this interview study, we explored the problem context qualitatively. The results served as input for the contributions in the following chapters.

4.2 Research Design

To explore the problem context, we conducted an industry interview study among different stakeholders of monitoring, including service managers, DevOps engineers, product owners, managers, tool providers and consultants. The structure of our research is based on the five-step case study research process, as stated by Runeson and Hoest [RH08] (see Figure 4.1) that we describe in the following sections.

Derived from the research goal of Section 4.1, we formulated the following research questions (RQ) that guided our study (see Table 4.1).



Figure 4.1: Case Study Research Process (adapted from [RH09])

Table 4.1: Overview of the research questions

RQ1	Which contemporary challenges exist in monitoring distributed systems?
RQ2	Which requirements do stakeholders have for a monitoring and observability concept for distributed systems?
RQ3	What are technical and organizational strategies and solutions in the organizations?

To answer the research questions, we decided to conduct semi-structured interviews. The interviews allowed us to explore the problem context of the individual participants in detail and analyze the underlying relations of their environment. Besides, the semi-structured nature of the interviews allowed us to dynamically adapt the interview structure according to the individual background and the responses of the participants. As the interactive character corresponds more to a conversation flow [SSL08], the interviewees were willing to talk about the challenges and negative aspects of their organizations.

We conducted 28 interviews, which lasted between 45 and 90 minutes. The interviews took place between February and April 2019. For a balanced distribution of interviewees, we first considered consumers and tool providers of monitoring solutions (see Table 4.2). Second, we ensured that the consumers and tool providers were coming from different domains to obtain different viewpoints on monitoring solutions. We covered domains of IoT, software and IT service, telecommunication, and insurance. Furthermore, we included IT consulting and consultants from applied research. By supporting organizations in implementing observability and monitoring solutions, consultants provide in-depth sources of consolidated experience on the challenges, requirements, approaches and applied solutions of their customers. We selected the consumers or customers of monitoring solutions from different points of view in the application stack and with different roles.

These included DevOps engineers, architects, support engineers, product owners, and managers. All participants had at least six years of professional experience. The experts (E1-E28) were coming from 16 organizations of different sizes. All interviewees are located in Germany except of E28, who was located in the USA. We recruited the participants by personal industry contacts as well as by acquisition at developer and research conferences.

Table 4.2: Overview about participants and companies

CID*	Domain	Staff	EID**	Expert role	Focus
C1	IoT	> 100 Tsd	E1	Product Owner	APM Solution
			E2	Lead Architect	Cloud Infrastructure
			E3	Product Owner	Connectivity Backend
			E4	Service Manager	Support and Operation of IoT Solution
			E5	Cloud Architect	IoT Backend
C2	IoT	100-1 Tsd	E6	IoT Consultant/Architect	Consulting of IoT Projects
			E7	Open Source Developer	Cloud Service
			E8	DevOps Engineer	Cloud Service
			E9	DevOps Architect	Cloud Service
			E10	Product Owner	Cloud Service
			E11	Project Lead	IoT Project
C3	IoT	10 Tsd - 100 Tsd	E12	Manager	IoT Platform
C4	IoT	10 Tsd - 100 Tsd	E13	IoT Solution Owner	Cloud Service
C5	Telecommunications	10 Tsd - 100 Tsd	E14	Product Owner	Monitoring Platform
C6	Software and IT Services	> 100 Tsd	E15	Former Chief Technology Officer	Software Development and Operations Tool
			E16	Technical Lead IT-Operations	Operation Solution and Event Management
C7	Applied Research	100 - 1 Tsd	E17	DevOps Engineer	Insurance Service
			E18	Developer	Front- and Backend of Fleet Management
C8	Tool Provider	1 Tsd - 10 Tsd	E19	Sales Engineer	APM Solution
C9	Tool Provider	10 Tsd - 100 Tsd	E20	Strategic Officer	Infrastructure Monitoring Tool
			E21	Support	Infrastructure Monitoring Tool
C10	Tool Provider	100 - 1 Tsd	E22	Developer and Architect	Infrastructure Monitoring Tool
C11	Tool Provider	100 - 1 Tsd	E23	Developer	Monitoring Tool
C12	IT Service Insurance	1 Tsd - 10 Tsd	E24	Divisional Director Monitoring	Performance Monitoring
C13	IT Consulting	1 - 25	E25	Developer and Architect	IT Consulting Monitoring
C14	IT Consulting	100 - 1 Tsd	E26	Chief Executive Officer	Business Process Monitoring
C15	Software and IT Services	10 Tsd - 100 Tsd	E27	Solution Architect	Open Source Technology Provider
C16	Software and IT Services	10 Tsd - 100 Tsd	E28	Developer	Cross-Stack Instrumentation for Monitoring and Debugging

*CID = Company ID, **EID = Expert ID

4.3 Preparation for Data Collection

To structure and focus our interviews, we created an interview guide [NKFW19a]. The guide is divided into the following thematic sections:

1. Motivation
2. Requirements
3. Solutions
4. Strategy and operating process
5. Lessons learned, challenges and outlook

The first section (1.) aims to explore the setting of the interviewee and the initial motivation to observe and monitor systems. We started by asking the interviewee about his background and role, the organizational environment and the related product (system, service or process). Afterward, we investigated the purpose and goals of observing and monitoring distributed systems. With the second section (2.), we explored the needs and expectations of the stakeholders to derive requirements for monitoring and observing distributed systems. Moreover, we asked the interviewees to describe the implementation of their monitoring concept and system. In the third section (3.), we deepened the conversation to applied observability and monitoring solutions. The participants were asked about approaches and specific tools they apply, and the criteria for selecting them. Moreover, we explored which components they observe and the abstraction level of observability in their application stack. After that, we tried to discover whether they gain all the data required and in sufficient quality, including the latency of data, to monitor their systems. The fourth section (4.) aims to investigate the strategies in the organizational context and the operated monitoring practices. We placed open questions to examine the participant's procedure of monitoring and controlling service disruptions. If possible, we delved into their approach of deriving target values and thresholds for specific monitoring indicators. Furthermore, we tried to find out if their organization has a uniform monitoring strategy and governance. With the last part of the interview (5.), we

focused on lessons learned, challenges, and outlooks. Since the descriptions of challenges are sensitive information, we positioned this block at the end of the interview in order to first build the respondents' trust.

Before conducting the interviews, we pre-informed the participants about the scope and procedure of the interview sessions. Besides, informing them to treat their transcripts as confidential, we asked to record the interviews to create transcripts if permitted. Moreover, we communicated the possibility of reviewing their transcript to approve the information given in the interview.

4.4 Data Collection

In total, we conducted 28 interviews. Fifteen interviews were conducted “face-to-face” and thirteen via remote communication. Twenty-six interviews were held in German, and two in English. Twenty-one interviewees agreed with the audio recording. For the seven interviews without audio records, two researchers created protocols, which we cross-checked to reduce researcher bias. In the interviews, we loosely followed the interview guide according to the participants' answers and backgrounds. The audio recorded interviews were manually transcribed into text by the researchers. Afterward, we sent the transcripts to the participants for review. We invited the participants to correct unintended statements or remove sensitive data.

4.5 Analysis of Collected Data

To analyze the individual transcripts, we encoded the material to extract relevant categories regarding our research questions. We decided to follow Mayring's approach of qualitative content analysis [May14] as it allows a mixed approach of deductive and inductive coding on a sentence level. Based on the research questions (RQ1-RQ3), we deductively formulated the highest level of the category system as follows:

- Challenges
- Requirements

- Solutions

By passing through the interviews, we inductively constructed new categories for contents that could not be directly assigned to the main categories. Moreover, we inductively formulated sub-categories to differentiate aspects of the main categories further. Thus, we formulated hierarchies of categories containing codes for the main categories and sub-codes for the sub-categories during the analysis. In several iterations, the codes were revised, split, or merged.

4.6 Results

In this section, we present the aggregated findings from the interview analysis with the focus on our research questions defined in Section 4.2. In the following, we describe the sub-categories of the identified challenges, requirements, and solutions. We illustrate the categories with exemplary statements from the experts (see Expert ID (EID) Table 4.2).

4.6.1 Challenges (RQ1)

With RQ1, we aimed to analyze the challenges the participants perceive in monitoring and observing their distributed systems and to work out the implications that are further related to these challenges. We identified a set of nine technical and organizational challenges (Cx).

Increasing dynamics and complexity (C1): The emerging paradigms of microservice architectures, cloud deployments, and DevOps increase the complexity of distributed systems. While the individual complexity of a microservice is reduced, the operation complexity and operation efforts are increased. The component dependencies and dynamic behaviors within the distributed system are causing complexity in a way that system behavior cannot be deduced easily from the properties of its individual elements. This challenge regarding unplanned system behavior is stated by several

participants and corresponds to the concept of unwanted emergent system behavior. The challenge of increasing dynamics and complexity applies to cloud-native microservice architectures and historically grown systems, where an overview and model of the component dependencies are often missing. Besides, some participants stated an underestimation of the dynamic complexity of their systems. Consequently, in case of faults, the duration for fault detection and fault reaction to control the value provision takes too long. Besides, we identified a lack of knowledge about effect chains and of interaction concerning the multitude of components and their observation. Participants also stated that due to frequent changes in system and environment: “[...] *there is always something that fails*” (E16), therefore, it is challenging to control the service provision to consumers.

Heterogeneity (C2): The distributed systems consist of several layers: from application to infrastructure technologies like containers, virtual machines (VMs), or serverless environments. These layers are developed and operated by heterogeneous teams that are specialized in their context. Moreover, as stated by the participants, systems often contain legacy software, modern service technology, and third party components in parallel, where additional tooling is necessary to integrate all components. Concerning multi-tenant systems, some participants experienced a noisy-neighbor-effect, where one tenant monopolizes resources and negatively affects other tenants on the same infrastructure. However, in this case, the participants were not able to separate views among different tenants.

Regarding the high degree of technological heterogeneity and speed of innovation, we identified different positions. On the one hand, developers highlighted the advantages of using the most appropriate programming language and technology to develop, observe, and monitor their systems. On the other hand, the technological heterogeneity complicates the consistent application of monitoring approaches. In addition, some interviewees critically reviewed the speed of innovation, and some wished for a slow-down of technology hypes by defining regulations.

The heterogeneity of systems, technologies, and teams leads to a miss-

ing overview of the overall system, its components, and processed workflows.

Company culture and mindset (C3): Most of the participants affirmed that culture and mindset aspects regarding monitoring and collaboration are essential for operations. Several interviewees stated that with the growing complexity and the emergent behavior of the distributed systems communication and collaboration along different teams and organization is more challenging than technical aspects. Faults in the operation of the distributed systems are normal behavior, which requires observability to detect faults and respond to them with fault reactions. However, for detecting and handling unwanted effects across different teams, shared concepts and performance observation during operation are often missing. Some participants criticized the common industry mindset of just verifying the normative requirements' implementation rather than asking: Are consumers complaining? E16 depicted that: *“Many companies that are not web-native are lagging behind the importance of the problem, unlike Google, Netflix, and Spotify, for example, who had to have this discussion very early in order to stay successful.”*

Moreover, some interviewees mentioned that a holistic transparency to apply monitoring is often not intended by their organization. They described concerns for being responsible or rather blamed for a failures in retrospective. For further consideration what is preventing employees from communicating negative aspects transparently organizational information theory can be applied. One assumption is, that employees associate potential harmful consequences (threat to their job or career) with the communication of troubling information and confronting these [BA12].

Furthermore, joint sensemaking [Wei93], the process performed by people making sense of ambiguous and complex situations in order to make decisions, is impeded because often the teams do not have an overview outside of their area, such as the business and consumer context of their microservice. They focus on normative requirements, especially on the verification of individual functionalities on component level. Therefore, they often neglect to control the dynamically linked provision of functionalities within a workflow to provide value to a customer or consumer. We also identified drivers

in some organizations' reward system: product owners (POs) and teams are primarily rewarded for developing functionalities; thus, the functional completeness of services gets greater attention than its dependability. Hence, *“a team will be able to read Google’s SRE book as often as they want, this will barely have an impact”* (E16).

This causes isolated monitoring and operation concepts without context to the value provision to a customer or a consumer and the corresponding quality requirements. Overall, collaboration and communication between teams and the perspective on developing and operating their components are often weakly pronounced. This illustrates the following statement (E22): *“It is usually not the ignorance or the inability of people in the company, but the wrong point of view. Often the developers are so buried in their problem environment, so engrossed in their daily tasks that they can no longer afford to change themselves.”*

Lack of central point of view (C4): Many participants described isolated monitoring of components with no or limited possibilities regarding the observability of dependencies to other components and a transaction providing value. The chains of effects of the transactions are not or too less known; thus, observing and monitoring transactions is often neglected. E6 describes the following situation: *“If it comes to problems such that there are many users complaints because the system is not working properly, everyone went for troubleshooting. Due to the lack of an overview, it was difficult to diagnose the faults. It took several escalation rounds and teleconferences to discuss where the fault is located.”* At the same time, we identified a lack of responsible persons in charge to generate overall views and enable individual teams to collaborate. Another point mentioned is the missing transparency about the impact of integrated third party components. As for third party components, performance parameters are usually not accessible, blind spots remain, which prevent an integrated monitoring.

Flood of data (C5): The distributed systems that are constantly in change bring a high amount of observation complexity and produce a high volume

and variety of data. Participants mentioned that they are overwhelmed by the flood of data. The data is often unstructured and with too little relational context to analyze and extract meaningful information to achieve joint sensemaking. This shows the following statement (E17): *“The volume and amount of alerts are currently challenging, we are not able to prioritize the consumer impacting ones.”*

One participant discussed the aspect of the emergent behavior of the distributed systems. It is fatal to conclude from the observation of individual microservices on the behavior of a transaction delivering value for a consumer, as *“The behavior of the forest cannot be derived from the behavior of the individual trees.”* (E15).

The isolated observation and monitoring of individual components and related flood of notifications and alerts lead to “alert fatigue”. Developers and operators become desensitized, not able to differentiate “signal from noise” and exclude false positives to detect anomalies impacting a value provision to a customer or consumer. Participants described difficulties in selecting measures and deriving sufficient target values for notification and alert thresholds due to a missing relation to transaction delivering value and the relevant quality characteristics. In terms of troubleshooting, where one transaction has to be handled by multiple components developed by independent teams, it is very complex to detect faults and defects, including the responsibilities to handle and correct them. Many participants described the complexity in the correlation of measures and timestamped logs from multiple components, often accompanied by insufficient metadata. Besides, participants stated the absence of a comprehensible information processing that enables navigable views. The handling of the flood of data and information processing is influenced by the challenge of sensemaking [Wei93; Wei95], where sufficient information is lacking to be able to make contextually appropriate decisions.

Dependency on experts (C6): The fault handling to control the value provision and the postmortem analysis procedures seem to be highly dependent on individual experts' knowledge about the design and behavior of the system. The participants highlighted experience-related and implicit expertise as essential. As the following statement from E11 illustrates, some experts appear as a “source to debug”: *“This form of troubleshooting depends highly on the expert knowledge and experience of the team members [...]. The knowledge about the service structure is mostly more crucial than a monitoring, which specifically indicates: search at this point.”* This challenge again outlines the complexity of the behavior and the interrelations in a distributed system. Furthermore, it highlights missing systematic development of monitoring approaches in supporting humans in sensemaking [Wei95] during situation management to handle faults and postmortem analysis to detect defects.

Lack of experience, time and resources (C7): As microservices architectures come with advantages for faster development and deployment, many participants described the challenge of mastering microservice technologies and the DevOps paradigm. As operational complexity is increased, they require additional effort and knowledge for operation control. At the same time, skilled DevOps or site reliability engineers are missing. Classical capabilities and mindset of operators of monolithic systems architectures are not sufficient anymore. People with knowledge in development and operation, applying software engineering and automation practices are needed. One participant also mentioned that from his perspective, software engineers' academic education is lagging behind the practical needs regarding operations topics. In addition, the high frequency of releases and the short time to market results in prioritizing the development of functionalities and a disregard of the dynamic behavior and quality requirements of the services. As monitoring is not considered directly value-adding compared to developing functionalities, limited resources and capacity are spent. Most participants mentioned the limited time and experience as reasons for an iterative, often reactive development of system observability and monitoring.

Unclear non-functional requirements (C8): According to the interviewees, non-functional requirements like dependability and performance of services, expressed by SLIs and SLOs, are often not defined and controlled. Some participants commented that teams are often unaware of the essential quality characteristics from a customer or a consumer view of value and how they should be observed and measured. *“It is very important to think about service levels or KPIs and to define them in a certain way. This is often underestimated. In many projects, it can be determined that the project managers only have a purely technical view of the system without being aware of the availability and performance that is needed.”* (E6). Reasons might be vague or imprecisely stated consumer needs and the context of use that are not explored yet. The participants also described a lack of awareness regarding the importance of non-functional requirements as a foundation for continuous feedback cycles for operation. As a result of unclear consumer needs and the complexity of system behavior, the interviewees struggle to define overall performance objectives, which need to be converted into performance objectives for components. This is further intensified by time constraints leading to reactive implementations, as illustrated in the following statement (E6): *“Many development teams are under pressure to bring the service to market as quickly as possible. So the teams usually start developing without specific customer requirements and end up in production without any systematically derived requirements.”* The lack of requirement definition leads in turn to missing feedback loops (E4): *“[...] where the quality control in the service provision is missing.”* Furthermore, due to the system’s complexity, participants cannot predict non-functional behavior like the time-based performance of their system to process transactions. Therefore, the participants stated that they prefer not to specify SLOs, as these could be violated. Lastly, several interviewees mentioned the operational complexity and missing attractiveness of operational tasks as a reason for neglecting non-functional requirements: *“[...] usually developers are excited about building functionalities and not about monitoring the operation.”* (E11)

Reactive implementation (C9): As stated in several interviews, the unclear requirements and insufficient indication and control often lead to faults propagating to failures. In these cases, the application of monitoring was triggered by a failure in production, where the teams recognized a lack of observability to detect faults. In fact, consumers often received inadequate quality of service. In several examples, the teams were occupied only with troubleshooting, which resulted in ad-hoc solutions, instead of creating systematically derived monitoring solutions. Moreover, we identified that teams run into the similar problems, where labor-intensive development of monitoring for individual components is created, and synergy effects of sharing knowledge, expertise, and good practices are not used. A further reason for reactive implementation is that during development, the developers did not have enough knowledge about the complex interactions in operations, and therefore blind spots remained until operation.

4.6.2 Requirements and Solutions (RQ2 and RQ3)

This section presents the results regarding RQ2, the requirements for monitoring distributed systems, and RQ3, the strategies, and solutions that practitioners applied or suggested. We identified a set of 14 requirements and 14 solutions. To relate the identified challenges with the requirements and the solutions, we created a matrix. At the end of this section, Figure 4.2 presents the mapping of the challenges (C1-C9), the requirements (R1-R14), and the solutions (S1-S14).

Holistic approach (R1): Associated with C4, the lack of central point of view, some participants characterize monitoring as “[...] *holistic problem and try to come up with a holistic approach to ensure observability [...]*” (E28). E22 noted that although microservices are designed as isolated and specialized entities, they are still a vehicle in a bigger context of a value provision to implement a business need and create value. To break down silos created through the high degree of specialization, we identified a need for a holistic approach to enable collaboration and communication across different sys-

tem layers, teams, and organizational units. The holistic view is especially required due to the constant changes and the dynamics of the distributed system. In case of a failure, system-wide fault detection, followed by fault reactions and consecutive postmortem analysis, are needed. One solution stated by interviewees is an event management system (S1), also referred to as “manager of managers”, that enables an overall view of the system state. It further allows to correlate events for event reduction. Other solutions mentioned are topology managers and architecture discovery modeling (S2). These enable to map transactions to underlying infrastructure components dynamically. Moreover, distributed tracing (S3) was emphasized as a solution that records the execution path of a transaction at runtime by propagating request IDs [SSM+16]. Therefore, tracing enables to infer causal relationship among events on the execution path. It enables to create a “[...] *bird’s eye view, to find out what is going on with a user request [...]*” (E28). Distributed tracing can also be applied to diagnose the dependencies to third-party components, representing black-boxes, without knowledge about their inner workings. However, the participants mentioned that the instrumentation to propagate trace IDs through individual microservices, developed by different teams, is not consistently assured.

Control from customer or consumer view (R2): Several participants described the trend moving from isolated monitoring of individual microservices to a context-dependent view from the perspective of a customer or a consumer value. Furthermore, some stated to apply Google’s SRE approach (S4) :*“With our SRE approach in mind, we care about the user experience and these are the golden paths we want to improve. I do not necessarily care about what is going on underneath, as soon as the user is not experiencing any anomalies.”* (E28). Moreover, several interviewees stated to perform synthetic probing. This is known as end-to-end monitoring (S5) that enables the emulation of real consumer behavior to measure and compare the transaction dependability. In general, many participants referred to apply APM solutions (S6) to continuously monitor the state of a system from a consumer-centric view. Further essential aspects of taking a consumer

perspective are approaches to validate assumptions about consumer needs, operationalize them, and identify anomalies experienced by the consumers.

Definition of indicators from customer or consumer view (R3): Several participants noted SLIs and SLOs as an approach to systematically define indicators and objectives for individual services as part of an SRE discipline. The interviewees expressed an underlying demand for a systematic definition of these indicators, but at the same time, they struggled with their implementation (E28): *“It is a lot of work to figure out the right SLO, which is a very long process. Not everybody is interested in this. [...] so you may end up with having a lot of toil and alerts if you haven’t set the SLO range sufficiently. So it is a kind of an experimental process.”* This corresponds to an inductive-iterative feedback-loop that continuously validates assumptions for requirements as well as SLIs and SLOs, based on individual consumer experiences.

Context propagation (R4): To provide a holistic view and to be able to detect faults that may propagate to failures, context propagation is needed. The system can propagate relevant context in the form of metadata, such as IDs or tags along a workflow. Besides distributed tracing (S3), adding metadata (S7) to measures and logs is a further example. During situation management, metadata enable to detect critical faults in the flood of data generated by the systems. This is necessary to perform quick fault reactions to prevent a failure and control the situation. Moreover, it enables to localize involved components and detect defects with potential correction options in a postmortem analysis.

Governance (R5): Several participants stated the need for governance that defines a strategy, including roles, responsibilities, processes, and technologies for monitoring and observability. This should comprise precise formulations of a minimal set of indicators that must be monitored for every service and component. A further requirement is to claim the observability of a component as an acceptance criterion for development and operation.

The participants mentioned that developing and implementing appropriate governance structures needs several iterations and has to be continuously adapted according to the company strategy. The participants-base was divided in terms of the introduction of tooling standards. On the one hand, some participants required standards, and on the other hand, other participants, especially developers, criticized a slow-down of development by oversized governance regulations and standards.

Some companies already implemented departments with specialists, who are responsible for the organization-wide monitoring strategy and implementation (S9). Some participants mentioned to align their governance structure to SRE principles and guidelines. At one company, component teams get support from specialized SRE teams only if they follow the SRE guidelines and collect at least the “golden signals”. Several participants highlighted creating a community of practice (S10) to share good practices and lessons learned across the organization.

Collaboration model (R6): The distributed systems are becoming increasingly heterogeneous, consisting of hardware, software, (Io)things, developed and operated by teams specialized in their context. Therefore, shared concepts for the organizing the division of labour between the teams are needed.

From a perspective of organizational information theory, an organization exists in an information environment. Therefore, it is crucial to recognize information processing as a social activity where communication to collaborate and make a collective sense of reality is needed in order to balance information complexity and practicability of actions [Col94].

Participants described a collaboration model as a base for effective and efficient communication in order to handle faults and failures during situation management and for the consecutive postmortem analysis, which needs to be performed across the specialized teams. This is particularly important when teams are geographically dispersed and are working asynchronously. Collaboration cannot be effective if teams work as black-box silos. Especially for postmortem analysis, it is necessary to bring together

specialists from different domains and enable inter-team communication about system anomalies. As there are different concepts for anomalies such as fault, failure, defect, error, or bug, in order to precisely communicate and exchange information, a classification for anomalies (S11) needs to be defined and agreed upon. Besides, to collaborate efficiently during situation management and postmortem analysis, metadata (S7) is needed to provide context in order to be able to infer the dependencies. Transparency is the basis for a holistic view, and some participants stated that a (E28): “[...] *common language, called SLO and SLI*” (S8) serves as a shared concept and the basis for their inter-team collaboration.

Monitoring platform (R7): Several interviewees demanded a unified monitoring platform, which offers solutions for different stakeholders, and which is independent of heterogeneous technology stacks to overcome siloed solutions. Its purpose is to increase operational efficiency between the different stakeholders. Hence it should include out of the box (S12), and standardized components that can be used modularly and are customizable for different stakeholder concerns. Monitoring and its default setup also need to enable the “*democratization of data*” (E19), with different stakeholders having access to required data in order to increase transparency. This can be enabled by offering standard APIs and deploying adaptors (S13) for different technologies. Consequently, a monitoring platform as default could “*create a kind of governance that is not strict.*” (E20).

Monitoring mindset (R8): With the increasing complexity and dynamics of the distributed system, the probability of faults and failures increases. Therefore, “*monitoring and observability are not a 'nice-to-have'*” thing (E19). A mindset to observe and monitor the operation to exclude unwanted effects during operation is a prerequisite for service control and improvement. Without increasing awareness, isolated ad-hoc solutions will remain. As observability and monitoring are perceived as strategic topics, which do not generate direct return-on-investment, management has to support these strategic investments.

One mentioned solution to increase the importance of dynamic non-functional characteristics is the derivation of SLIs and setting of SLOs (S8), which serve as means to control the value provision and to improve a service. Furthermore, the participants outlined a need to equalize the importance of functional and non-functional requirements. In order to balance the pace of functionality development versus the dependability of a service is to establish so-called “error budgets”. An error budget is part of the SRE discipline and represents the difference of the maximum possible value of an SLI and its SLO [BJPM16]. Hence, the setting of the target value and controlling the adherence of the error budgets is an instrument to balance the pace of functionality development versus the dependability.

New quality of operator (R9): A requirement highlighted by several interviews are extended capabilities of the developers and operators: “*With classical developers and classical operators, these systems can not be managed anymore.*” (E16). Operators need to increase their skills in terms of software engineering expertise, especially of being able to cope with automation tasks for implementing fault reactions to increase the fault tolerance of the system. Hence, organizations have to increase awareness of monitoring. To manage these systems, participants highlighted the need for site reliability engineers (SRE) (S4), who can work on operation and infrastructure tasks as well as software engineering aspects.

Detection of normal and anomalous patterns (R10): Almost every interviewee pointed out anomaly detection as an essential task for monitoring, which differentiates between normal and anomalous behavior. Different solutions are mentioned, such as event management (S1), to correlate events from different system parts. For correlating events, predictive analytics, and artificial intelligence (AI) (S14) are applied. Some participants discussed the difficulties of defining the expected behavior of a service. In this context, they considered distributed tracing (S3) to indicate performance measures and iteratively develop guarantees for their services by setting service level objectives (SLO). Most of the participants appreciated the enormous po-

tential of AI (S14) approaches to master the complexity and the flood of data generated by distributed systems. However, many interviewees pointed out that sufficient preconditions for applying AI are still missing in practice. Appropriate objectives need to be derived and operationalized, the right data has to be collected, the quality of data has to be ensured, the context needs to be propagated, and data has to be stored centrally. Concerns in terms of not being able to derive appropriate objectives and the reliability and cost-value ratio of AI approaches remain.

Automation of monitoring and fault reactions (R11): The increasing dynamics and complexity within distributed systems, caused by the upcoming microservice architectures and shorter lifecycles of components, is no more manually controllable. Therefore, automation is indispensable to observe and monitor a distributed systems. Agents (S15) are applied to automate the collection and analysis of data. This also includes the demands for automated instrumentation without manual configuration. Moreover, operations and fault reaction need to be automatized in a form *“that operators receive notifications that a problem has occurred and was solved by a bot and a runbook automation.”* Bots (S16) are increasingly applied to realize automation. In combination with AI (S14), bots and agents are able to perform more efficiently including collection and analysis of data and performing fault reactions.

Monitoring from the start (R12): Monitoring is a prerequisite for service control and improvement and therefore needs to accompany development and operation. Many interviewees indicated to consider monitoring from the start and use information from development to detect anomalies and validate quality requirements during operations. Some participants quote to integrate SLOs and SLIs (S8) *“[...] in the design time. As soon as there is a new service, you have a section in the design doc., where you can see these are the promises, formulated as SLOs. They may change over time to reduce toil. We start this conversation very early on.”* (E28). This might enhance the awareness for monitoring and can change the company culture towards

a monitoring mindset. While the teams should consider monitoring as an integrated part, the management needs to support a monitoring and observability mindset and make resources available.

All-in-one solution (R13): Participants require monitoring solutions to satisfy various needs, from consumer experience monitoring to application monitoring and heterogeneous infrastructure monitoring. However, the state of the practice indicates that all-in-one solutions do not exist and compromises are needed. Commercial solutions are especially adapting to stakeholder needs and provide a combination of basic functionalities for monitoring and capabilities to expand their solutions by combining and integrating other solutions. This can also comprise new standards and technologies. Hence, all-in-one solutions represent, in this context, a combination of several solutions and technologies. To realize such an encompassing solution, organizations need to substitute isolated solutions with open standards and modern technologies. Nevertheless, an overarching solution cannot substitute a monitoring strategy appropriate for the service.

Tool Capabilities (R14): This category summarizes different tool capabilities stated by the participants.

An often mentioned requirement is real-time monitoring, where transaction traces and system structure changes are being monitored with minimal delay. A further requirement addresses the avoidance of proprietary agents and applying open standards (e.g., JSON), motivated by being adaptable and flexible to new technologies. This also fosters the maintainability and portability of applications and monitoring solutions by being easily transferable to other distributed infrastructures and cloud providers. Besides, scalability is of particular importance to cope with large and dynamic distributed systems. While the management of the dynamics within a distributed system needs to be addressed, the dependability of the monitoring is highly demanded. For example, to view the current system status, health functionalities need to be accessible all the time. Moreover, tools need to support multi-tenant management. This requirement explicitly addresses the ability of tenant-

specific views and individual permission management. With the increasing application of agents, the more “backdoors” might be open, which may lead to a growing system fragility. Therefore, some participants required minimally invasive approaches in this context, where changes in an existing system are limited.

As tool capabilities do not constitute a requirement itself but represent different aggregated requirements, we integrated them as a row (without solutions) in the matrix of Figure 4.2.

Challenges		C1	C2	C3	C4	C5	C6	C7	C8	C9
		Dynamics and complexity	Heterogeneity	Culture and mindset	Lack of central point of view	Flood of data	Dependency on experts	Lack in experience, time and resources	Unclear non-functional requirements	Reactive implementation
R1	Holistic approach		Event management (S1), topology manager (S2), distributed tracing (S3)							
R2	Control from customer or consumer view		SRE (S4)		Synthetic probing (S5), APM (S6)					
R3	Definition of indicators from customer or consumer view					SLO, SLI (S8)			SLO, SLI (S8)	
R4	Context propagation	(S5) ITS 'OTS		Distributed tracing (S3), metadata (S7)				Central group of experts (S9)		SRE (S4)
R5	Governance				SRE (S4)					
R6	Collaboration model		SLO (S8) Classification for anomalies (S15)		SLO, SLI (S8)	Metadata (S7)				
R7	Monitoring platform		Adaptors (S13)				Out of the box (S12)			
R8	Monitoring mindset				SLO, SLI (S8)				SLO, SLI (S8)	
R9	New quality of operator							SRE (S4)		Distributed tracing (S3), AI (I4)
R10	Detection of normal and anomalous patterns		Event management (S1), AI (I4)					Distributed tracing (S3)		
R11	Automation of monitoring and fault reactions		Agents (S16), bots (S15), AI (S14)				Agents (S15), bots (S15), AI (S16)			
R12	Monitoring from the start				SLO, SLI (S8)					SLO, SLI (S8)
R13	All-in-one solution									
R14	Tool capabilities									

Available solution(s) No solution(s) needed

Figure 4-2: Challenges, Requirements, and Solutions. The arrows indicate grey fields for which the solutions are available.

4.7 Discussion

Our interview study identified several challenges in observing and monitoring distributed systems in order to control and continuously improve the value provision to customers and consumers. These challenges are of technical and organizational nature. Major challenges are the increasing complexity and dynamics of the systems. The complexity is exacerbated by paradigms of microservices, where functionalities are fragmented to be independently developed and operated with heterogeneous infrastructures running on worldwide distributed machines. Our study shows this “independence” of distributed architectures is somehow not the reality for real-world systems. Most consumer-relevant transactions involve several components and unexpected dynamic system behaviors remain undetected or get lost in the flood of data during production and hence may propagate to failures.

The different organizations already have various technical solutions in place. However, they often remain isolated and focused on local optimization so that organizations cannot control services from a customer or consumer viewpoint. The interviewees highlighted that service control and monitoring are not purely technical issues and participants referred to them as more cross-cutting and strategic topics. The continuous adaptation of these systems and the high frequency of releases need continuous observation and monitoring to control the dynamic system behavior. An essential aspect of monitoring, which seems to be difficult for several participants, is the derivation of quality characteristics and their operationalization across several teams and stakeholders.

Fundamental approaches are context propagation, enabled by distributed tracing to infer causal relationships between the various component interactions. To fully utilize such technical solutions across different teams, collaboration models with shared concepts are needed. These shared concepts are the basis for a “blameless” company culture, built on transparency to create feedback cycles for service, system, and organizational improvement. Shared concepts are especially needed for postmortem analysis, where specialists from different domains come together and need to perform inter-

team communication and collaboration.

4.8 Threats to Validity

This section discusses the limitations of the interview study. Concerning the **internal validity**, we see a risk that participants could not realistically reflect the situation in their organization or felt inhibited to talk about problems and challenges openly. We think this risk is relatively low, as we ensured the anonymity of the interviews, and the participants seemed not to be worried to talk about the negative aspects of their service or organization. Another threat to internal validity is the potential misunderstanding of concepts, which were used in the questions or answers. Therefore, we provided additional explanations for essential concepts applied by the researchers. Moreover, we asked questions to clarify concepts used by the participants with a potential domain- or company-specific meaning. The participants also took the chance to clarify questions if applied concepts were not clear to them.

At least one additional researcher reviewed the transcripts to reduce researcher bias and increase the interpretation validity. Furthermore, our participants were invited to review their transcripts to adjust unintended or incorrect statements and sensitive data.

A further threat is related to the coding and creation of categories, which highly depends on researchers' subjective interpretation. Therefore, the research team reviewed and discussed the category system after 15% of all interviews and again after full coding in order to reduce individual interpretation bias. After approximately 70% of all analyzed interviews, we reached a point of theoretical saturation, where the analysis did not lead to new main categories. The rest of the interviews only led to sporadic new sub-categories.

Concerning the **external validity**, we interviewed participants from German and international companies with diversity in terms of domain and size. Furthermore, with our participants' selection, we covered different

roles from different layers of the application stack. Moreover, we included providers of monitoring solutions and consultants advising organizations and teams in integrating monitoring solutions. As we focused on opinions and relations of the concepts concerning the research questions with this qualitative study, we did not count and quantitatively analyzed the number or distribution of mentioned codes. Hence, we do not claim generalizability of the results. Instead, our goal was to picture an overview of the complex relations in terms of technical and organizational challenges, requirements, and solutions.

4.9 Conclusion

This study's objective was to explore challenges, requirements, and contemporary good practices and solutions regarding monitoring and observability of distributed systems. Therefore, we conducted interviews with 28 software professionals from 16 organizations.

We identified that monitoring and the observability of distributed systems is not a purely technical issue anymore but is becoming a more cross-cutting and strategic topic, critical to a company's success. Development and deployment paradigms of microservices, DevOps, and cloud are creating maximal independence and specialization, resulting in isolated monitoring and observability solutions, where the participants struggle to control service from a customer or consumer-centric view. Most companies already have solutions and good practices in place, but in many cases, they remain isolated approaches due to siloed company structures. Therefore, practical and holistic concepts and approaches to enable customer and consumer-oriented development and operation are needed, which we tried to cover with our IoT transaction concept in Chapter 6. With reference to the findings of the contemporary state of practice, we see a need for further work on good practices and real-world examples of how to derive quality requirements based on customer and consumer needs and how to evaluate the dynamic system behavior to detect anomalies. We will address this need with our transaction-

based approach of dependability evaluation in Chapter 7. Furthermore, shared concepts to break down silos and enable efficient development and operation are needed. This includes a classification of anomalies to control the dynamic system behavior and analyze the service and organization's deficiencies to propose correction options for continuous improvement. With the following contribution in Chapter 5, we propose a classification of system anomalies to address this need.

CORRECT & CONTROL COMPLEX IOT SYSTEMS: A CLASSIFICATION OF SYSTEM ANOMALIES

The results of the interview study in Chapter 4 revealed that there is a lack of shared concepts to break down silos and enable effective and efficient development and operation between different teams and organizational units. This includes a classification of anomalies that enables individuals from different teams to collaborate and deal with the dynamic system and situation complexity during operations. Furthermore, the classification of anomalies needs to support their iterative and inductive learning in retrospect, as service and organizational deficiencies have to be analyzed to propose correction options for continuous improvement of the system, service, and organization.

This chapter describes our contribution to the field of classifications of system anomalies to address this need. We present our adaption of the classification of software anomalies of the IEEE 1044-2009 [IEE09] standard. Our approach is based on three system views, to support inter-team communication across different organizational units. We applied and evaluated our classification of system anomalies in a two-staged action research project of a postmortem analysis at the Bosch Group. To evaluate the quality in use of our artifact, we performed semi-structured interviews with three of the stakeholders of the postmortem analysis. This chapter is an extension of the following publication:

- S. Niedermaier et al. “Correct and Control Complex IoT Systems: Evaluation of a Classification for System Anomalies” In: Proceedings of the IEEE 20th Conference on Software Quality, Reliability and Security (QRS 2020). IEEE, 2020, pp. 321-328. [NHW20]

5.1 Context

Developing and operating IoT service systems constitutes a socio-technical area, which needs to combine technical and organizational perspectives. The distributed complex systems are compiled of components, integrating hardware, software, and mechanical system elements in interaction with humans. In operations, providers of IoT service systems need to control the dynamic behavior in a permanently changing context of use. The system components and elements are developed and operated by different teams from different organizational units [NGSR16]. The individual teams specialize according to the context of their component organizational silo. For communication within an organizational silo, the teams apply concepts optimized for their specific intra-team context.

In case of failure, a failed function of an IoT service, the different teams operating the system components have to form a collective to control the situation collaboratively.

First, in situation management, facing dynamics, the objective is to regain

control by performing fault reactions as quickly as possible. The purpose of a fault reaction is to avoid a failure influencing customer or consumer satisfaction or, if not possible, at least to provide limited operation to demonstrate controlling the system and situation for keeping the service's trust. The handling of anomalies in situation management is performed in a tactical mode. Its optimization goal is "to be effective before to be efficient" (see Section 2.9).

Second, once the system is transferred into a controlled state, the anomalies in the effect chain and the actions of responding to dynamic cascades of consecutive anomalies are investigated within a postmortem analysis. The objective is to detect defects and subsequently remove them and increase the fault tolerance of the system. With this analysis, actions are initiated from a strategical long-term business perspective. These actions are performed with the goal of profitability. The leading principle is "to be efficient" (see Section 2.9).

As Blohowiak et al. [BBHR16] from Netflix describe, modern distributed systems have a level of complexity that is chaotic, and that there is no chief architect who can keep "*all of the system's moving parts in their head*" anymore. Consequently, to perform situation management and postmortem analysis, several developments and operational teams from different organizational silos need to form a collective. As identified in the interview study in Section 4.6.1, the different teams perform specialized and isolated observations and monitoring of their system component or system element. From an organizational psychology perspective Weick [Wei93] describes how different perspectives of individuals impede "*to make common sense*". Deficiencies on shared concepts supporting communications about anomalies often lead to misunderstandings and different interpretations regarding anomalies. Therefore, individuals from different teams are not able to act as a collective and deal with the system and situation complexity. In addition, missing shared concepts may be used to direct "responsibility" for failure, compensation of loss of value, and related effort for defect correction to a specific organizational unit [NKFW19b].

Root cause analysis, e.g., performing the method of 5xWhy [Ohn88] may

reflect this social need to assign blame to a specific organizational unit. When people ask “why” (within the concept of the 5Why method), they often tend to point in the direction of “who”. This oversimplification of a single root cause and a single responsibility often results in a culture of blaming.

In contrast, a retrospective is an instrument of a learning organization performing feedback cycles to improve product and service quality as well as organization quality [DIN15; For94; Sen06]. The feedback cycles process nonconformities related to product and service quality and defects related to the performance of the organization which include corrective actions and actions to increase fault tolerance.

Furthermore, we detected that if the tactical mode of effective but costly performance of work is not differentiated from the strategical mode of long-term and efficient performance of work, an organization tends to remain in an effective tactical mode.

Accordingly, if a failure can be controlled by a tactical fault reactions, an interest in the defect and its sustainable correction often remains secondary. Hence, defects with their options for a correction are not identified and listed and they accumulate to technical debt.

5.2 Goals

To address the need for a precise differentiation in order to enhance the effectiveness and efficiency of the collaboration capabilities of developers, operators, and organizations, and to reduce the assignment of blame, we propose a shared classification of system anomalies. It is based on the common logical concept of the IEEE Std. 1044-2009 [IEE09] to classify software anomalies.

We propose an adaptation of the IEEE Std. 1044-2009 [IEE09] which applies to the system element software. Our adaptation is an extension which applies to a system composed of a set of interacting system elements. We introduce three system views: functional system view, dynamic system view,

and system structure view and we assign the concept of anomalies: failure, fault, and defect of IEEE 1044-2009 to them. Furthermore, we extended the concept of the defect to be assigned to a set of components with options for defect correction. Therefore, our adaptation enables differentiation between the handling of fault detection and dynamic fault reactions and respectively the handling of defects and its options for sustainable defect correction.

We investigated its quality in use [ISO11c] by conducting an action research project in two stages. In the first stage, we applied our adaptation of the IEEE Std. 1044-2009 to support the inter-team communication of the stakeholders of the postmortem analysis to class the detected anomalies accordingly. To further evaluate the quality in use of our adaptation, we conducted semi-structured interviews with three of the stakeholders of the postmortem analysis in stage two.

At the time of the study, we did not differentiate the concept of service regarding the customer and consumer value, as described in Section 2.2.2 and further examined in our contribution in Chapter 6. For the postmortem analysis (described in Section 5.5), the customer corresponds to the consumer. Nevertheless, we further extended our concepts in Section 5.3 by differentiating customer and consumer value.

5.3 Concepts

This section introduces three system views and our assignment and adaptation of the concepts of anomalies, namely failure, fault, and defect, based on IEEE 1044-2009 standard [IEEE09]. In the context of fault tolerance and situation management, we focus on the relationship between fault and failure to keep the system in a controlled state. Furthermore, we investigate the relationship between a tactical quick fault fixing and a sustainable defect correction which is optimized supporting strategic interest in the context of defect correction and quality management.

5.3.1 Concepts of Classification of System Anomalies

Our adaptation of the concepts of the IEEE Std. 1044-2009 [IEE09] is general enough to include anomalies related to complex IoT systems. For this dissertation, an (IoT) system is defined by its components, providing functionalities. The components are compiled in a structural arrangement, interacting in component effect relationships. The components themselves can be composed of interacting system elements such as software, hardware, and mechanics [ISO18c]. Our concept of system considers a human, providing functionality, as a component, and as part of the IoT system. For developing and operating complex IoT systems, three architectural views are motivated. In the following, we describe them and state related system concerns, according to ISO/IEC/IEEE 42010 [ISO07a].

- **System structure view:** The system structure view addresses provider concerns. It represents the components of an IoT system in their structural arrangement. The system structure view supports organizing the division of labor.
- **Functional system view:** The functional system view addresses customer concerns. It represents the customer value, related to a value proposition and its verification if the promise of value is kept or not (dual logic). The functional system view supports the representation of the offered functions. The functions can be decomposed into sub-functions and assigned to system components.
- **Dynamic system view:** The dynamic system view addresses consumer concerns. It presents the value and value loss of component effect interactions of an IoT system in their temporal propagation. A component effect is the performance of a function in the dynamic system view. The dynamic system view supports situation management to control the system's dynamic behavior.

To avoid conflicts with other classifications of anomalies, we choose a deductive approach and motivate our classification of system anomalies out of the three system views. The views contain conformities and nonconformities.

A nonconformity related to a system view is an anomaly. We added the following classification of system anomalies to the three system views, which is an adaptation of the anomaly classes of IEEE Std. 1044-2009 [IEEE09], which we described in Section 3.2.

- **Failure:** A failure is related to a loss of value by:
 - a failed function, where a system does not perform a required function within specified limits. This concept of anomaly is assigned to and optimized for the **functional system view**.
 - a termination of the ability of a system to perform a required functionality. This concept of anomaly is assigned to and optimized for the **dynamic system view**.
- **Fault:** A fault is a break in a component effect chain and has a temporal extension. This concept of anomaly is assigned to and optimized for the **dynamic system view**. A fault can be classified as permanent, transient or intermittent [ISO18a].
- **Defect:** The defect in IEEE Std. 1044-2009 is an imperfection or deficiency in a work product, which needs to be either repaired or replaced. We assign the concept of defect to a set of components in their structural arrangement and identify *potential option(s) for correction*. An option for a correction is related to a modification on:
 - a set of components and their structural arrangement
 - a component itself
 - a system element of a component.

The trivial option for a correction is the exchange of the set of components, the defect is assigned to. An assignment of option(s) for a correction to a set of components does not imply poor quality of work. We assume that every team and employee acts with the best intention based on information of the context at that time of the insertion of the defect. It is the context and emergent system behavior which may activate a “latent” or “dormant” fault. The concept of the anomaly

defect and its option(s) for a correction are assigned to and optimized for the **system structure view**.

As we consider not only software anomalies but also system anomalies, we extended the software change request of IEEE 1044-2009 to a system modification request (SMR) [IEE06], assigned to a set of components in their structural arrangement (see Figure 5.1). Therefore, the system modification request comprises a software change request.

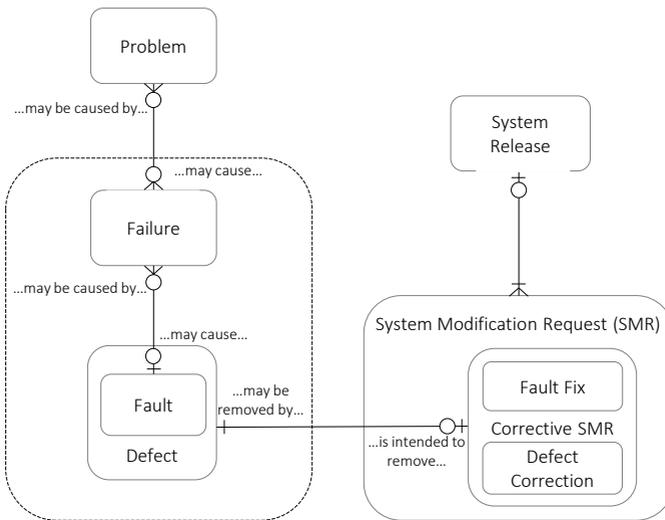


Figure 5.1: Classification of System Anomalies as an entity relationship diagram (adapted from [IEE09])

As a “*fault is [related to] a defect*” [IEE09], there is a:

- first, corrective SMR resulting in a (quick) **fault fix** and a
- second, corrective SMR resulting in a **defect correction**.

This is depicted in Figure 5.1.

5.3.2 Fault Tolerance and Situation Management

We focus on the relationship between the concept fault and concept failure from the dynamic system view with this section of fault tolerance and situation management. In engineering and operating complex IoT systems, we aim for fault tolerance to continue providing functionality in the presence of one or more faults [ISO18a]. Ideally, a system is designed fault tolerant. If the system is not tolerant to a fault, the organization must face the situation: a fault reaction is needed to control the situation and keep the customer's trust. What matters in situation management is time; the organization switches into a reaction mode focusing on effectiveness.

Analogous to ISO 26262 – Functional safety [ISO18a], we differentiate the following points in time and time intervals in the **dynamic system view** (see Figure 5.2).

Points in time:

- t_1 : Occurrence of fault, fault is not detected.
- t_2 : Time when fault is detected.
- t_3 : End of fault reaction.
- t_4 : End of fault tolerant time interval. Occurrence of failure.

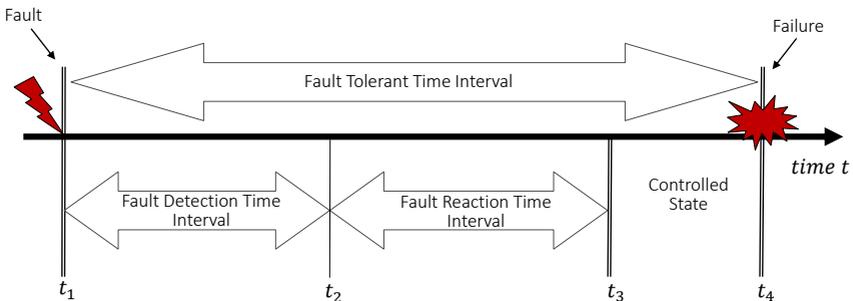


Figure 5.2: Fault tolerant time interval (adapted from [ISO18a])

Time intervals:

- The **fault tolerant time interval** is the minimum time span from the occurrence of a fault in a component effect chain of a system which propagates to a failure (in the dynamic system view) if a quality mechanism is not effective.
- The **fault detection time interval** is the time span from the occurrence of a fault to its detection, comprising the exclusion of false positives and false negatives.
- The **fault reaction time interval** is the time span from the fault detection and the execution of a fault reaction until the system is transitioned into a controlled state.

The controlled state of this contribution corresponds to the safe state of ISO 26262 [ISO18a], where a functionality is switched-off, or a degraded functionality is provided.

Fault Tolerance: A fault reaction is performed to prevent the fault propagating to a failure. For enabling fault tolerance, a fault reaction needs to be automated. Therefore, with the concept of fault tolerance, we include automated technical fault reactions. To enable fault reactions, the faults need to be detected in the fault detection time interval. A fault reaction has to be performed in the fault reaction time interval and needs to be successful before the fault tolerant time interval ends.

Situation management: If the system is not tolerant to a fault, when detecting the fault, the organization is switching into a tactical mode by performing an OODA loop [Boy96]. Effectiveness dominates efficiency. We **Observe, Orient, Decide and Act (OODA)**, supported by the dynamic system view. In situation management, sensemaking is needed. The organization has no full information transparency but needs to react. In this context, it is less about the accuracy and completeness of information than of sufficiency; meaning knowing enough for being able to make a contextually appropriate decision and therefore enable action-in-context [Wei95].

- **Observe:** We observe the system behavior to detect anomalies.

- **Orient:** Based on the context-dependent observation, supported by the dynamic system view, we orient ourselves. We need to assess: How critical is the situation? What are our options for fault reaction?
- **Decide:** Based on a set of prepared tactical options, we decide for a fault reaction to control the system and situation.
- **Act:** We act with the trained fault reaction.

The purpose of a fault reaction is to control the value provision, by avoiding a failure or, if not successful, to transfer the system into a controlled state for keeping the customer's trust. In situation management, facing dynamics, the organization must demonstrate that it is able to control the system. This is a continuous action supported by the dynamic system view. Tactical options for fault reaction have to be engineered. The controlled states need to be predefined to the customer or consumer needs and must be prepared to degrade or regrade the value provision.

Controlled states can include:

- **redundancy activated:** no loss in value, no failure in customer and consumer context of value by activating a homogeneous or a heterogeneous redundancy [[ALRL04](#)].
- **degraded:** less value by reduction of functionality or reduced level of performance related to the value proposition.
- **regraded:** alternative value by alteration of the grade of a nonconforming service in order to make it conform to requirements differing from the initial requirements which are related to the value proposition [[DIN15](#)].
- **deactivated:** no negative value by precluding consumption of service by discontinuing it [[DIN15](#)] in order to maintain or achieve a safe state [[ISO18a](#)].
- **compensated:** compensation of the loss of value or compensation of accumulation of negative value, e.g. reimbursement.

Quality mechanisms represent not only technical fault reactions but also organizational fault reactions. The fault reaction needs to be well-trained to be performed effectively during situation management. However, fault reactions can also include novel combinations or new creations of fault reactions [Coo98]. As trust needs to be preserved in the present [Luh14], the servee observes and assesses the organizational behavior and communication to continue placing trust in the organization. Transparent communication about anomalies based on common concepts is the basis for collaboration with the servee.

With an effective (quick) fault fix, the system is transferred in a controlled state. The situation is under control, and the servee's trust remains placed on the organization. The organization is now independent of time. Consequently, it switches from the effective tactical mode, which is supported by the dynamic system view, to an efficient strategic mode to support defect management.

5.3.3 Defect Correction and Quality Management

Once the system is transitioned into a controlled state, sustainable defect correction can be performed. Quality management of the organization process defects in a PDCA loop (Plan, Do, Check, Act) [DIN15]. By performing a postmortem analysis, on the one hand, we aim to improve fault reaction in order to increase fault tolerance. On the other hand, we aim to improve tactical options in order to increase the capability for situation management.

Hence, defect correction is optimized for strategic interest and therefore is not equivalent to a (quick) fault fix, which is optimized for tactical interest. Within this section, we focus on the relationship between the concept fault and the concept defect. The organization is in an efficient mode with the goal of profitability.

In postmortem analysis, anomalies related to the different system views can be identified:

- In the functional system view an anomaly (**failure**) with need for

correction is identified, a defect is detected and is to be assigned to system structure view.

- In the dynamic system view an anomaly (**fault**) with the need for correction is identified, a defect is detected and is to be assigned to system structure view.
- In the system structure view an anomaly (**defect**) with the need for correction is detected and is to be assigned to a set of components.

As described in Section 5.3, a defect and its option(s) for correction are assigned to the system structure view and a set of components in their structural arrangement. The detected defects and option(s) for defect correction, including defects related to the IoT system and defects related to situation management, are transferred to defect management. The defect management decides whether and which option for defect correction, addressed by an SMR, is to be executed for strategic and efficiency interest.

The lifecycle of a defect is illustrated in Figure 5.3. We adapted the UML statechart diagram of the defect life cycle of the IEEE Std. 1044-2009 [IEE09].

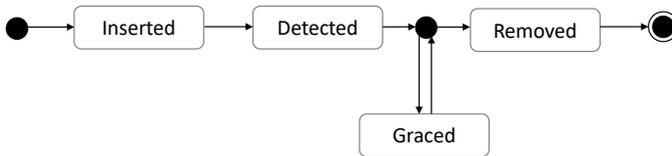


Figure 5.3: Defect life cycle (adapted from [IEE09])

The states of a defect *inserted*, *detected*, *removed* have been extended with an additional state *graced*. We propose, that after a fault is detected, there is the additional option to grace a defect for a certain period of time until it is removed. The decision to grace a defect is based on an assessment that a defect can be accepted for one of the following circumstances:

- The decision can take place due to economical reasons, when the organization decides to remain in a fault reaction mode (as described in Section 5.3.2) instead of sustainably remove the defect.
- The decision to grace a defect can be influenced by the assumption that there will be acceptable customer or consumer impact.
- The decision to grace a certain defect may be necessary in the context of a safety vs. security vs. privacy discussion. For example, a defect from security perspective – a security defect – may be graced for safety or privacy reasons.

Despite the decision to grace a defect, defects and the context in which they represent defects have to be documented in a defect list and need to be transferred to defect management.

5.3.4 Communicating Anomalies

By enriching the classification of anomalies with the concept of fault reaction and defect correction, we can differentiate between fault detection and fault reaction in the context of situation management and respectively the defect and its options for defect correction in the context of defect management.

The relationship between the different concepts of Section 5.3 can be used to communicate about the anomalies across different organizational units as follows:

*The servee got a **problem** caused by a **failure** (a failed function which is required, which is promised with a value proposition), indicating the presence of a permanent, intermittent or transient **fault** (activated by the context of use of the servee).*

The fault has to be detected and a fault reaction has to be executed to transfer the system into a controlled state.

*The fault is indicating the presence of a **defect**, which has to be assigned to a set of components and is intended to be removed by a completed defect correction (of a corrective SMR).*

Furthermore, for the communication about anomalies and the organization's reaction, we propose to apply two anomaly models – the fault model and the failure model, described in the following section.

5.3.4.1 Anomaly (Fault and Failure) Model

The purpose of an anomaly model is to support communications about anomalies related to the value provision and document reusable options for handling of the anomalies. The anomaly model enables an organization to collect and provide reusable anomaly-solution patterns for the operation of an IoT system providing service. Furthermore, it is an instrument to obtain feedback from a servee about anomalies in the value provision and the context of use they appear, which the organization did not observe and/or detect.

For an IoT-System providing service, we propose the following predefined anomaly models:

- The **failure model** addresses customer concerns. It is optimized to support the communication about anomalies representing a loss of value related to the value proposition in the functional system view. The failure model enables to collect the situational context of the failure and includes reusable options to handle the failure.
- The **fault model** addresses operations concerns. It is optimized to support the communication about anomalies representing a loss value in the dynamic system view. The fault model enables to collect the situational context of the fault and includes reusable options for a fault fix to transfer the IoT system in a controlled state.

With their reusable solution patterns, the failure and the fault model serve as a runbook for the operation of the IoT system and can be iteratively adapted through the exploration of the context of use. Moreover, the provider can collect anomalies and the context of use, where a servee's needs have not been satisfied. Thus, needs and expectations can be identified that were implied or unaware during the requirement stage. With the information

received from a servee, feedback loops to explore the context of use and therefore validate and adapt the quality model, including measures and target values (as described in Section 7.3.2) can be performed.

5.4 Research Design

To structure our action research project with the Bosch Group, we applied the case study research process, as proposed by Runeson and Hoest [RH09]. We addressed the following research objective.

5.4.1 Research Objective

Evaluate the quality in use (with limitation to the sub-characteristics: effectiveness, efficiency, and satisfaction) of the adaptation of the standard classification for software anomalies, the IEEE 1044-2009, by applying it to a postmortem analysis of an IoT system.

This work is intended to enhance the effectiveness and efficiency of the collaboration capabilities between developers and operators along organizational units by providing a common logical concepts of anomalies. The concepts of anomalies enable precise and “blameless”¹ inter-team communication to differentiate between actions for controlling the system and actions for its sustainable correction.

The following research questions in Table 5.1 guided our study.

Table 5.1: Overview of the research questions – Classification of System Anomalies

RQ1	Is the approach effective to differentiate between fault handling and defect handling?
RQ2	Can professionals understand and apply the classification efficiently?
RQ3	Are professionals satisfied with the classification?

We extended the scope of IEEE Std. 1044-2009 from anomalies related to software only to anomalies related to complex IoT systems providing

¹With the concept of “blameless” communication, we refer to the concept of blameless culture and blameless postmortems, as described in Section 2.9.2.

service to a servee (customer and/or consumer). We differentiate between the handling of:

- first, **fault** detection and **fault reaction** in the context of **situation management**
- second, **defect** and its options for **defect correction** in the context of **defect management**.

The following definitions of effectiveness, efficiency, and satisfaction are presented and interpreted for this study, in order to evaluate the quality in use of our artifact.

- **Effectiveness:** *“Accuracy and completeness with which users achieve specified goals”* [ISO98]. We evaluate the stakeholders’ ability to differentiate between the handling of a fault, including fault reactions and respectively the handling of a defect and its option(s) for defect correction, applying our classification.
- **Efficiency:** *“Resources expended in relation to the accuracy and completeness with which users achieve goals”* [ISO98]. Resources include temporal and also mental effort performing the task. We did not track the time consumption of performing the task since we conducted qualitative research. Therefore, we qualitatively evaluated the estimated efficiency of the interview participants compared to other postmortem analyses.
- **Satisfaction:** *“Degree to which user needs are satisfied when a product or system is used in a specified context of use”* [ISO11c]. This sub-characteristic includes a positive attitude towards the application of the artifact. We evaluated it by asking the interview participants about their subjective experiences and opinions towards applying the classification.

5.4.2 Research Process

We conducted an action research project in 2019 at the Bosch Group, a German company providing IoT solutions, in order to evaluate the quality in use of our classification of system anomalies.

The action research study is based on a postmortem analysis related to a customer problem in the domain of IoT condition monitoring. We designed the study in a two-staged procedure. In stage one, we applied the adapted classification on a real-world postmortem analysis of the IoT solution. In stage two, we performed semi-structured interviews to evaluate the quality in use of the classification with the stakeholders of the postmortem analysis.

5.4.2.1 Stage 1: Application - Data Collection and Analysis

The data collection and analysis of stage one was conducted between February and May 2019 at the Bosch Group. Specialists from the different organizational units, representing the stakeholders' concerns, formed a collective to perform the postmortem analysis. The method of 5xWhy is part of the organizational postmortem procedure and was executed to assign root causes. Selected stakeholders of the collective applied our classification of anomalies to enable intra-collective communications. Two of the researchers were in direct relation with the Bosch Group and the application of the classification, thus the data collection technique can be determined as first degree [RH09]. One researcher acted as a coach to ensure the application of our classification during the postmortem analysis. Another researcher took the role of an independent observer. For data triangulation reasons [Sta95], we analyzed data from multiple sources, including incident report data, notes from postmortem analysis meetings, observations, and emails discussing the results of the classed anomalies. As the data is sensitive, it is not provided within this work. By applying member checking, the classification results were validated by the stakeholders of the postmortem analysis and have been simplified for the purpose of this dissertation in Section 5.5.3.

5.4.2.2 Stage 2: Quality in Use Interviews - Data Collection and Analysis

To ensure interpretation validity, besides performing observations, we conducted semi-structured interviews with one participant of each stakeholder group of the postmortem analysis. The interviews aimed to explore the stakeholders' individual experiences and qualitatively evaluate the quality in use of our classification. The three stakeholders were actively involved in the postmortem analysis and were responsible for the resulting classification. All of the three interviewees were employees of the Bosch Group but coming from different organizational units. We interviewed the product manager (S1) of the IoT solution, a quality manager of the cloud infrastructure provider (S2), and a systems analyst and quality expert (S3), who has coached the other stakeholders in applying the classification (see Table 5.2).

Table 5.2: Interviewee Information

Stakeholder ID	Role	Focus in the Analysis
S1	Product Manager of IoT Solution	Representing customer perspective
S2	Quality Manager of Cloud Provider	Lessons learned
S3	Quality Expert Consultant	Systems analysis

To conduct the interviews, we created an interview guide [NKFW19a], that we structured as follows:

- general questions about classifications for anomalies
- questions regarding the effectiveness, efficiency, and satisfaction of the application of our classification of system anomalies
- open questions to discover the pros and cons of our classification
- open questions to identify the potential for optimization.

We loosely followed the questions and audio recorded the three face-to-face interviews of approximately 40 minutes, which were all held in German. After transcribing, we sent the interviews to the participants for review. To evaluate the quality in use of the classification, we analyzed the semi-structured interviews by performing Mayring's qualitative content analysis [May14]. We performed a mixed approach of deductive and inductive coding

by encoding the transcripts with the predefined quality in use criteria of Section 5.4.1 and creating further categories for contents that could not be directly assigned to the existing categories. The transcripts were analyzed on a sentence level. During analysis, we formed hierarchies of codes and sub-codes. Through several iterations, the codes were revised, split, or merged. The results of the interviews are described in Section 5.5.5.

5.5 Results

This section presents an abstract description of the IoT solution and the context of the customer problem. Furthermore, we provide the results of the classification (stage 1) and the interview analysis (stage 2) and discuss them.

5.5.1 IoT System Description

The IoT solution is based on a distributed system architecture, with relation to different stakeholders: customer, solution provider, cloud infrastructure provider, sensor gateway provider, and sensor provider. The mission of the IoT solution can be stated as follows:

Generate and provide condition monitoring data of a physical customer asset to increase efficiency in managing it.

5.5.2 Situation

The solution provider did not detect faults in the effect chain or the instantiated failure of the complex IoT system during operations. By calling the service desk, the customer informed the organization that a required function had failed. The customer's failure notification was the only indicator of faults. The customer's notification initiated the incident-specific problem-solving procedure. The problem-solving procedure includes the companies' intervention with fault reactions to transfer the system into a controlled state.

A team of specialists representing the stakeholders' interests was executing postmortem analysis, applying the method of 5xWhy to assign root causes. During the analysis, the stakeholders applied our classification of system anomalies to their communications.

5.5.3 Classification Results – Stage 1

In the following, we describe the results of the postmortem analysis, applying the classification of anomalies to differentiate between the handling of:

- first, fault detection and fault reaction
in the context of situation management
- second, defect and its options for defect correction
in the context of defect management.

Problem:

- Customer view: customer does not receive up-to-date asset condition information. Loss of control with risk of damage of customer asset.
- Solution provider view: customer perceives that solution provider lacks competency to control the situation. The relationship of trust between customer and solution provider is at risk.

Failure:

- Customer (**functional system view**): failure of solution. Information related to asset condition inconsistent due to message transmission fault.
- Provider (**functional system view**): failure of load balancer (single point of failure) with fail passive due to load test on cloud infrastructure. The load balancer failed by a temporary overload and reacted as expected by switching to its controlled state: The system is down. The server and its service are not available.

Fault (→ fault reaction):

- Customer (**dynamic system view**): Customer is only offered functional system view. Complexity of dynamic system view is hidden from the customer.
- Provider (**dynamic system view**): failure and fault detection by customers via emergency call to service desk. The organization did not detect intermittent transaction message faults. In consequence, there was no fault reaction. Both, technical (redundancy), as well as an organizational (manual) fault reaction were missing. Incident-based problem solving process did not start until customer calls or sends an email to the service desk.

Defect (→ option(s) for defect correction):

Defects related to IoT system (**system structure view**):

- Defect 1: malconfigured load balancer.
- Defect correction option 1: correction of load balancer malconfiguration.

Defects related to situation management capability (**system structure view**):

- Defect 2: missing technical option for fault reaction of load balancer (single point of failure): no failing active and operational, by activating a redundant component.
- Defect correction option 2: implement failing active and activating redundancy.
- Defect 3: missing organizational options for fault reaction: no tactical options for situation management. On organizational level the solution provider did not set up controlled states for degrading the value proposition.
- Defect correction option 3: implement degrading value proposition, including customer information of quality degradation.

5.5.4 Discussion – Stage 1 (RQ1)

By observing and reflecting the application of the classification to the post-mortem analysis, we identified the following aspects:

- By differentiating the concepts fault and failure, we improve in engineering and operating fault tolerant systems.
- By differentiating the concepts fault and defect, after effective fault fixing, we are able to assign a defect to a component and identify its options for defect correction for transfer to defect management. This activity is done in an efficiency mode to gain efficiency yield. The organization is customer-oriented. The customer's interest is focused on fault fixing and not on defect correction and decreasing the technical debt. We identified that the organization must therefore motivate the defect correction itself.
- By differentiating the dynamic system view, we can assign defects related to the IoT system and defects related to situation management capability.
- During postmortem analysis, we tended to assign the defect to a single component and not to an interacting set of components. Therefore, we missed options for correction, with a potentially higher efficiency yield and sustainability.
- For preparing appropriate tactical options in situation management, we could assign additional defects (related to situation management) to a set of components. Therefore, evaluating the completeness of detected defects is not necessary because the number of possible defects depends on the variable assignment of a defect to components for correction options.
- The observability in the dynamic system view needs to be improved, with the focus on the IoT transaction (presented in Chapter 6) in order to control the value provision for the customer.

In this case, the correction of load balancer malconfiguration (defect 1) covers defect correction and fault fix. Defect 2, the elimination of the single point of failure, was transferred to the backlog. Improvement to increase fault tolerance and improvement to increase situation management capability have been identified and transferred to defect management. The defect management decides whether and which options for correction are to be executed.

Our classification concept has several symmetries to the concepts of anomalies of Avizienis et al. [ALRL04]. We can roughly map their concepts fault, error and failure (described in Section 3.2) to our concepts of defect, fault and failure. However, we identified a strong difference of our concept of defect to their concept of fault [ALRL04]. Avizienis et al. [ALRL04] define a fault as a cause of an error. Implicitly they mix in their consideration of the concept fault the system structure view (e.g., physical hardware faults, which may correspond to defective components) as well as the dynamic system view (e.g., interaction fault, which occur during operation). In contrast, we detect a fault as a break in the effect chain from a dynamic system view and assign a defect to a set of components from a system structure view, for which we identify options for correction. Thus, we can assign a set of the faults of Avizienis et al. [ALRL04], that means the interaction faults, to our concept of fault, as the break in the cause effect chain of the dynamic system view. We can summarize that with the concepts of Avizienis et al. [ALRL04] we cannot differentiate between a quick fault fix and a sustainable defect correction.

5.5.5 Interview Results – Stage 2 (RQ1, RQ2, RQ3)

This section provides the results of the participant interviewees. We further investigate the quality in use (as described in Section 5.4.1) of applying our adaptation. Following the structure of the interview guide, we present the highest level of the coding system, including the purpose of the classification, the evaluation criteria (effectiveness, efficiency, satisfaction), and further optimization potential for our classification. We describe the different codes

with exemplary statements from the stakeholders (S1, S2 and S3) of the postmortem analysis.

Purpose of classification: Beginning the interviews, we asked about the purpose of a classification of anomalies in the context of a postmortem analysis. All of the three participants described the purpose of it related to existing challenges.

S1 and S2 outlined from experience with other postmortem analyses that discussions between different organizational units are often very imprecise if no shared concepts exist. Furthermore, S3 stated that: *“In deciding conflicts of interest, it is advantageous to have the power of interpretation over the concepts of anomalies. This sometimes leads to the rhetorically strongest person taking over the interpretation sovereignty over the concepts and, for example, directing the effort for corrections to other organizational units.”* In terms of interpretation sovereignty a classification of anomalies allows to: *“competently exchange about anomalies with colleagues from other organizational units [...] and thus ensures that you have a common view of the situation and the system.”* (S2). Further a commonly accepted classification, serves as *“[...] a common language that enables to improve mutual understanding.”* (S2). For *“[...] only yourself you do not necessarily need a classification.”* (S2).

S1 summarized the benefits of our classification as follows: *“The differentiation of fault and defect is very important. When I detect a fault, I have to do the hotfix immediately. It’s essential that I can get this to work at all. To correct it in a sustainable way, working on the defect, probably makes sense out of a business perspective.”* To proceed with the defects from a business perspective, the reusability of the classification results is necessary and was highlighted by all three stakeholders. *“The information status for a decision, for dealing with the anomalies is available in a known and reusable schema. It can be decided quickly and comprehensibly, also at a later time, whether and how defects to be corrected are transferred back to development.”*(S3). S2 confirmed that the differentiation enables to create a transparent and comprehensive *“[...] documentation that you can pull out even after a year or later [...]”* Hence, the documentation, including detected defects which may

have been graded for economically reasons, can be used to set up actions for latent faults “[...] at an early stage or at least you can be aware that something can go wrong.”(S2).

Effectiveness: In terms of effectiveness, all three stakeholders of the postmortem analysis confirmed that it is possible to differentiate between fault detection and fault reaction and respectively the defect and its defect correction options. S2 stated that: *“With the classification, one can achieve high accuracy in differentiation.”* Moreover, S1 describes: *“The classification enables us to distinguish more precisely between a quick fix and sustainable correction.”*

However, all three stakeholders conclude that the current organizational framework of the postmortem analysis is not yet an environment that is sufficiently conducive to learning. This illustrates the following statement (S1): *“I think it would have been much easier if the organizational frame would have been different. The questioning of 5xWhy, was just limitedly compatible with it. [...] It has disrupted the regular procedure of the postmortem analysis, which is why it was not as efficient and took longer. However, it was definitely valuable in terms of content.”* This statement illustrates that the substantive goal was nevertheless achieved. The participants recognized an added value; concurrently, they referred to aspects of time consumption and degree of difficulty, which is discussed in the next evaluation criterion efficiency.

Efficiency: The interviewees described that the application of the classification across different stakeholders and organizations was challenging and complicated. *“You have individual views and a global view, which is presented simplified in retrospective. However, it is not that easy in the analysis process because the reality is nested with different responsibilities”* (S2). Besides, S2 noted that at the beginning of an analysis, appropriate data on anomalies at different system levels are not immediately available. Only through several iterations can the data be converted into a form with sufficient information.

As described in the criterion effectiveness, the efficiency was limited due

to the organizational frame. Therefore, we further discuss the relationship of the efficiency of applying the classification to the procedure of asking 5xWhy, mentioned by all three stakeholders. S3 reflected that applying the 5xWhy method goes beyond the boundaries between the effect chain and the tactical fault reactions during operation, without explicitly differentiating them. As a consequence: *“With troubleshooting, outstanding unwanted effects ‘faults’ are reacted to with a fault reaction. When these unwanted effects are under control through a fault reaction, an interest in further actions related to sustainable correction decreases, even if the fault reaction has to be operated persistently.”* (S3).

Overall, our classification performs worse than the previous postmortem analysis in terms of time consumption and increasing difficulty to differentiate. However, the application of our classification enabled us to identify faults with fault reactions and defects with options for sustainable defect correction. We detected defects related to the product quality of the IoT system correction as well as defects related to the organizational capability for situation management and fault reaction. From a strategic business perspective, the list of detected defects and correction options allows us to consider, now and in the future, in which activities to invest.

Furthermore, S3 highlighted that consumption of resources must be considered in relation to the effect achieved. *“Symmetry and reuse of work is not the primary optimization goal of those who are involved in a postmortem analysis”*. This is where S3 identified a high potential for return on efficiency (see category *“further Optimization”*).

Throughout the interviews, we identified that further practical executions are needed to increase the time efficiency of the application. We assume that the effort of applying the classification decreases with increasing usage, and therefore, efficiency can be further increased.

Satisfaction: All of the three stakeholders of the postmortem analysis stated to have a positive attitude towards the logic of the classification. S1 describes feeling secure in applying our concepts and further distributing it across his team: *“I like it a lot. That’s why I use the concepts and terminology in*

our team. Also, to get people used to it, because I find the model very useful.” S3 commented on the question of satisfaction by relating it to cultural aspects of performing postmortem analysis: *“What I particularly like is the fact that the classification refers to anomalies in the IoT system and not to people who are to blame for something.”*

Further Optimization: We identified that the classification could be a foundation for an analysis that is not focused on blaming. However, it is not the solution to the cultural problem (S1): *“With the concept, you can also look for ‘culprits’ just as well as with the postmortem analysis before. At the end of the day, it is crucial what is done with the analysis result.”* S2 also discussed this point and stated: *“If you disassemble everything in detail and you take a close look at what went wrong and where a defect is located, that is what people do not really enjoy. At this point, you have to be especially careful not to be destructive.”*

S3 reflected how the execution of the 5xWhy in combination with the classification, could be improved in terms of efficiency (S3): *“Once the defect is detected or reported, so assigned to a set of components in an arrangement, asking ‘why’ can be stopped. Which option for correction is finally instructed, can be [strategically] planned outside of the postmortem analysis meetings, [in the efficiency paradigm]. Currently, in the postmortem meetings, the options for correction are ‘quickly found’, regardless of whether these corrections find a higher strategic use.”*

It turned out that further improvement in training the classification, including practical instructions on how to apply it to a real-world customer problem, is necessary. At this point, it has to be ensured that people are not *“[...] overwhelmed by the differentiating nature of the classification applying different system views”* (S2).

5.5.6 Discussion – Stage 2

The participants’ answers indicate that if our classification is established and sufficiently practiced by the participants, it can be effectively and efficiently

applied. The classification reduces ambiguities and misunderstandings in communicating about anomalies across different organizational units. With the concept of defect and its options for defect correction, we take away the assignment of blame to a specific responsibility. We assume that every team and employee acted with the best intention based on their information and situation at that time. Moreover, the participants accept anomalies in these complex systems as normal and gain sensibility for an increasingly dynamic environment and the need for collaboration. Instead of just detecting defects, we focus on improvement with a collaborative identification of defect correction options, revealing multiple options for defect correction. This fosters transparency and organizational learning culture to remove anomalies in the IoT system and improve the organization's capabilities for fault reaction. With the differentiation of the effective handling of faults in the dynamic system view and the sustainable and efficient defect handling in the retrospective, teams can situationally focus on what is important. Besides, the reusability of the classification results fosters continuous learning culture for iterative and inductive product development and operations.

The application of our concepts was challenging in various degrees to different stakeholders. *“The colleagues with contact to the customer saw a benefit for themselves and quickly completed communication and thinking with the concepts that were new to them.”* (S3). We assume that, depending on the different organizational roles, responsibilities, and motivations, the challenges concerning communication and information processing are reflected in different degrees. The stakeholders in contact with the customer seem to have the highest intention to create an overall view to control their value proposition. Furthermore, they have a high interest that defects are sustainably corrected to avoid further failures in their value provision to the customer. For colleagues at the infrastructure level, with a more complex and dynamic view of the system, it is challenging to relate faults and defects at the infrastructure level with failures at the application level. The overwhelming complexity of data generated by the system makes it challenging to make sense of the information and draw implications as described in our interview study in Section 4.6.1.

We conclude that the classification for system anomalies can increase an open learning culture; however, there is a need to adapt it to the organizational frame. To improve efficiency, further practical instructions, applying the classification, and how to improve the integration in the company-specific procedure of the postmortem analysis are needed.

5.6 Threats to Validity

We designed the artifact for application in an industry context and documented the adaptation based on IEEE Std. 1044-2009 [IEEE09]. To reduce a threat to **construct validity**, we performed the application and evaluation on real-world anomaly data of an industrial action research case.

According to the qualitative nature of the interviews, the results are based on the three stakeholders' personal experience and judgment. This weakness introduces the possibility of subjective bias influencing **internal validity**. To reduce this threat, we triangulated the answers of one interview of each stakeholder group. We see another threat to internal validity as the stakeholders may tend to answer in confirmation of our artifact. This could have led to confirmation bias. To reduce the threat of subjective bias and confirmation bias, we triangulated the interview answers with the observation of the artifact's application.

In terms of **external validity**, we see a threat that the classification and the evaluation results are specific to the context of the Bosch Group. As we documented our adaptation on the IEEE Std. 1044-2009 for the system level and described the case context, we assume that the application can be transferred to similar IoT contexts outside of this specific case and company. This threat needs to be further decreased by applying it to other postmortem analyses at different organizations and comparing the results.

We described the procedure of data collection and analysis. Hence, we consider that the study can be reproduced and a threat to **reliability** has been reduced. However, specific details about the incident report are sensitive and cannot be provided.

5.7 Conclusion

Developing and operating IoT service systems in an open system context needs iterative and inductive learning to improve services. This is enabled by excluding unwanted effects, anomalies in the dynamics of the customer and consumer context of use by fault reaction and/or defect correction. Precise communication about anomalies with common concepts is the basis for collaboration across different organizational units of a distributed system. With this artifact, we proposed and applied an adaptation of the IEEE Std. 1044-2009 for classifying system anomalies according to three systems views: functional system view, dynamic system view, and system structure view. We assigned the anomaly concepts failure, fault, and defect to the system views. Our adaptation allows us to differentiate between the handling of fault detection and fault reaction in the context of situation management and enables us to increase fault tolerance. Furthermore, it allows us to differentiate the handling of defects and their sustainable defect correction options in the context of defect management. We evaluated the artifact's effectiveness, efficiency, and satisfaction by applying it to a postmortem analysis at the Bosch Group. In terms of **effectiveness**, we were able to differentiate between the concepts of failure and fault. This enabled us to identify missed fault reactions for controlling the system to keep the customer's trust during situation management. These missed fault reactions also represent defects that can be corrected to improve the IoT system's fault tolerance and the organizations' capability for situation management in the future. Moreover, we were able to differentiate between the concepts of fault and defect. This enabled us to differentiate between actions for tactical fault fixing and actions for sustainable defect correction. The results of the postmortem analysis with the classed anomalies are available in a reusable scheme and allow us to decide whether and how defects have to be corrected.

The **efficiency** of applying our classification was limited. The stakeholders described the application of the classification across different organizations as challenging, also due to the organizational frame. However, all interviewees

highlighted the valuable content-related contribution. We anticipate that the effort and time of applying the classification will decrease as usage rises, and thus the efficiency will increase.

The overall **satisfaction** of applying the classification was positive. The stakeholders stated a positive attitude towards the constructive usage of the concepts to foster continuous learning. With our classification, we provide a foundation for “blameless” and transparent handling with anomalies. This is an opportunity to enable organizational feedback cycles to learn and improve. People with different organizational backgrounds can be encouraged to communicate about anomalies transparently.

We identified that in industry, there is a focus on the functional system view of the “happy path”. A dynamic system view to control temporal effects, which enables fault detection and fault reaction, is often missing. To be able to implement tactical options for fault reaction, we identified the need to enhance system modeling.

In order to sufficiently react to anomalies and keep the system in a controlled state performing operations, we have to observe and quantify the dynamic system behavior to indicate and detect faults. Therefore, our further contributions deal with approaches to indicate and detect faults along the component-effect chains of a service throughout different organizational units, performing development, and operations. This should allow us to improve fault reaction and fault tolerance of complex IoT systems.

Furthermore, the differentiation between security, safety, and privacy defects allows deciding domain conflicts collaboratively and documenting defects from the individual domain’s perspective.

We think practitioners and researchers can take these concepts into account for developing and implementing collaborative concepts in handling system anomalies across different organizational units.

CHAPTER
6

THE CONCEPT OF IoT TRANSACTION

In this chapter, we present the concept of the IoT transaction, which we derived from our concept of an IoT service, presented in Section 2.2.2. With the concept of an IoT transaction, we target to conceptualize a discrete, observable entity to evaluate the value provision of a service from a customer or a consumer view. This chapter is part of the following publication:

- S. Niedermaier et al. “Evaluate and Control Service and Transaction Dependability of Complex IoT Systems”. In: *Software Quality Journal*. Springer, 2021.

6.1 Context and Goal

As described in Section 3.3, paradigms of microservices and cloud are creating maximum independence and specialization of teams. In industry, this often leads to isolated observability and monitoring approaches focused

on individual components or elements, not allowing to control the value provision from a customer and consumer view.

Neglecting to elicit quality requirements and a lack of feedback loops for controlling the value provision can result in failures. In practice, the operationalization of customer or consumer needs is often performed on service or component level, but not on the level of a workflow of a performed functionality or task. Furthermore, our interview study of Chapter 4 showed that developers and operators are overwhelmed by the massive amount of isolated monitoring data. Therefore, an approach to reduce observation complexity is needed. To address these issues, we introduce the concept of an IoT transaction, which is defined from a business perspective and has a discrete and observable nature.

6.2 Concepts

6.2.1 IoT Transaction

In Section 2.2.2, we defined our IoT service concept based on the different views of a customer or a consumer respectively, regarding value. Services are intangible and have a continuous nature. Hence they cannot be directly measured (see Section 2.2.1). Nevertheless, we want to evaluate the value provision of services from a consumer and customer view. Therefore, we introduce the concept of an IoT transaction, which is a workflow with discrete and observable properties, which can be evaluated.

A transaction in the context of a database is a sequence of operations following the ACID properties: atomicity, consistency, isolation, and durability. According to this concept, a transaction must be executed completely or fail as a unit and can not be partially complete [GR92]. With our concept of the IoT transaction, we distinguish ourselves from the concept of database transactions and the ACID properties. We come from the business value and, therefore, take the perspective of a business transaction. Furthermore, our concept of IoT transaction allows us to consider transactions with degraded quality and partial completeness, which we describe in our following

contribution in Chapter 7.4.

The concept of transaction finds increasing adoption in domains of APM and the distributed tracing community. AppDynamics, one of the commercial APM market leaders, is defining a transaction as “*the end-to-end, cross-tier processing path used to fulfill a request for a service by a user*” [App20]. Sigleman, the author of the early tracing framework Dapper [SBB+10], describes the concept of transaction as a “*single, logical unit of work in its entirety*” [Blu19].

For our concept of the IoT transaction, we are more restrictive than Sigelman [SBB+10] and tool providers like AppDynamics and deduce it from the concept of an IoT service. An IoT service can be decomposed into IoT transactions. Compared to an IoT service, which delivers value, an IoT transaction consequently delivers a quantum of value.

IoT Transaction:

performance of activities of a workflow that starts and ends, delivering

- a) a quantum of value to the **customer**
- b) a quantum of value to the **consumer**

by providing component effects of an IoT system

- a) to target the completion of the workflow for the **customer**
- b) to target dependability for the **consumer** via a quality model for the consumer.

As described in the IoT service concept in Section 2.2.2, the customer has a dual perspective and evaluates a transaction as completed or not completed. In comparison, the consumer has a dynamic perspective on transaction dependability, which can be evaluated by a projection on a subjective quality model.

An IoT system decomposes into a set of components that provide functionalities (see Figure 6.1). As described in the contribution of the classification of anomalies in Section 5.3, a component effect corresponds to the performance of a function in the dynamic system view. We further define that

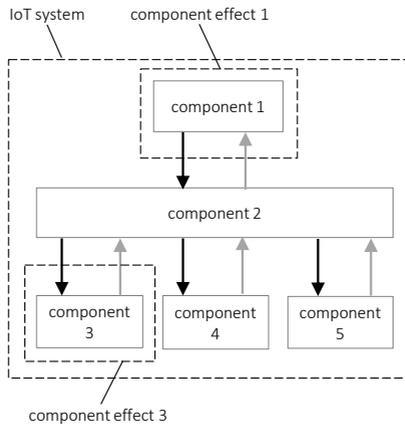


Figure 6.1: Component effects in relation to a transaction of an IoT system

a component effect is related to a quantum of business value of an IoT transaction. To reduce observation and control complexity, we focus with the concept of component effects on business or value relevant component behaviors, which, in case of faults, can propagate to a failure. Thereby, component effects are means to define from the set of possible behaviors to be observed, a smaller subset of elements, and represent points of observation and control.

6.2.2 IoT Transaction Monitoring

The concept of IoT transaction, encompassing component effects, has observable and quantifiable properties that enable us to indicate transaction completion and transaction dependability with the concept of IoT transaction monitoring, as described below.

IoT Transaction Monitoring:

performance of activities of an IoT system delivering (a quantum of) information

- a) to indicate IoT transaction completion
- b) to indicate IoT transaction dependability

by providing a method

- a) to observe component effects and classify the observation: completed/not completed
- b) to observe component effects and quantify the observation.

According to the IoT transaction concept, a) corresponds to the customer view regarding value, applying dual logic to the value provision and b) corresponds to the consumer view, differentiating quality characteristics regarding dependability of value provision in time. Applying the method of distributed tracing, component effects can be indicated (observed and quantified) through spans and reconstructed into a trace (see Figure 6.2). Therefore, it is possible to indicate IoT transaction completion and indicate transaction dependability via distributed tracing, as described in the following.

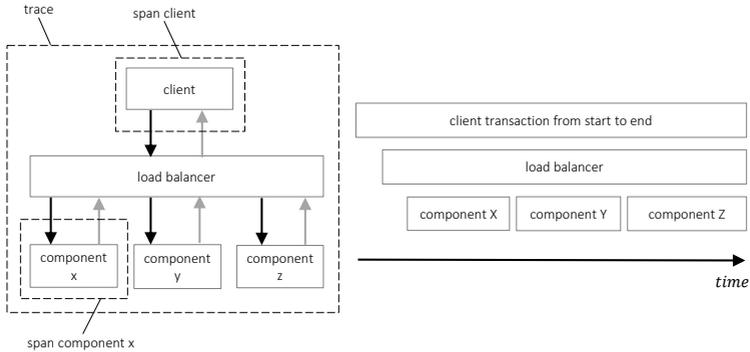


Figure 6.2: Relationship spans and trace (adapted from [Whi19])

Distributed Tracing:

performance of activities of an IoT system delivering (a quantum) of information

- a) to indicate IoT transaction completion
- b) to indicate IoT transaction dependability

by providing a method

- a) to observe spans
- b) to observe spans and quantify the observation.

With our concept of IoT transaction, we can model the service needs and expectations of a consumer and a customer by defining IoT transaction completion and IoT transaction dependability requirements. On the one hand, IoT transaction monitoring enables us to indicate IoT transaction completion from a dual customer view, e.g., for billing. On the other hand, IoT transaction monitoring enables us to indicate IoT transaction dependability from a dynamic consumer view, e.g., to detect faults and failures in operations and perform fault reactions in order to bring the system in a controlled state (see Chapter 5).

6.3 Conclusion

In this chapter, we introduced the concept of the IoT transaction, which is defined from a business perspective and has a discrete and observable nature. An IoT transaction represents workflow activities to provide a quantum of value to a customer or a consumer. With the observation approach of distributed tracing, we are able to observe and control the value provision to a customer or a consumer. In the following contribution (Section 7), we applied the concept of IoT transaction monitoring to evaluate the dependability of IoT transactions related to an IoT solution of the Bosch Group from a consumer view of value. Future work could focus on evaluating IoT transaction completion from a customer view, e.g., for billing.



CHAPTER
7

EVALUATE & CONTROL SERVICE & TRANSACTION DEPENDABILITY OF COMPLEX IoT SYSTEMS

The interview study of Chapter 4 showed that due to maximal independence and specialization of individual teams, practitioners struggle to derive quality requirements from a customer and a consumer view on value. As a consequence, monitoring and observability implementations are often not sufficient to evaluate and control the value provision of services in a dynamic context of use. Therefore, this chapter presents our contribution regarding the evaluation and control of the value provision from a consumer view. It applies and extends the previous Chapter 6 of the concept of the IoT transaction. As we focus on the value provision of an IoT system from a consumer view, we indicate the dependability of an IoT transaction to

deliver a quantum of value. In an action research project at the Bosch Group, we designed and validated an approach to evaluate the dependability of IoT transactions. We guided and observed the application of the approach in the company and conducted a focus group to evaluate its quality in use. This chapter is part of the following publication:

- S. Niedermaier et al. “Evaluate and Control Service and Transaction Dependability of Complex IoT Systems”. In: *Software Quality Journal*. Springer, 2021.

7.1 Context and Goals

For the evaluation and control of IoT transaction dependability from a consumer’s view, an approach to derive quality characteristics and appropriate indicators to enable context-dependent dynamic observation and control is necessary. As the large number of components, with complex and dynamic communication patterns, produce a massive volume of monitoring data, the approach needs to reduce observation and control complexity.

We propose a lightweight approach, explicitly designed for the dependability evaluation of complex IoT service systems from a consumer-centric view on value. To reduce observation and control complexity, we focus on relevant component effects that are related to a quantum of value of an IoT transaction. Via distributed tracing, we can observe and verify the value provision of an IoT transaction in a dynamic context of use. As a result, the time for fault detection can be reduced. Moreover, we are able to implement control loops for service performance control and service improvement. Furthermore, through observing the dynamic context of use and the system behavior, we can inductively and iteratively validate and adapt the quality model.

The research artifact – the transaction-based approach for dependability evaluation – was developed and evaluated by performing action research in two stages. We designed the artifact in the first stage based on industry requirements, which we derived from the process model of ISO/IEC

25040 [ISO11b]. For our approach, we extended the scope of ISO/IEC 25040 [ISO11b] to a continuous observation context of operations. We take the IoT transaction concept and project the quality characteristics of a transaction onto the quality models of the SQuaRE series [ISO11d]. This enables us to implement control loops with SLIs and SLOs, which can be quantified in operation through distributed tracing. In the second stage, we applied the artifact at the Bosch Group. We observed the artifact's instantiation and evaluated its effectiveness, efficiency, and satisfaction through a focus group discussion by reviewing the application in detail with two of the stakeholders.

7.2 Research Design

We followed action research and design science principles to develop an artifact and validate it in an industry context. With the artifact, we aimed to improve the evaluation of service and transaction dependability within the scope of continuous operation of IoT service systems. Hence, we studied the experience of applying it in a real-world industry use case [DMK04].

7.2.1 Research Objective

The research objective of this contribution can be defined as follows:

*Enable the evaluation of transaction dependability
in dynamic operations of complex systems
by designing an approach for dependability evaluation
applying distributed tracing as a measurement method that satisfies
expectations towards quality in use of software practitioners,
with a focus on effectiveness, efficiency, and satisfaction.*

The following research questions in Table 7.1 guided our study.

Table 7.1: Overview of the research questions

RQ1	What is a feasible and efficient approach for dependability evaluation and control? (design)
RQ2	How effective is the approach to fulfill stakeholder requirements?
RQ3	Can software professionals understand and use the approach efficiently and are they satisfied with the approach? (efficiency, satisfaction)

7.2.2 Research Process

We designed and evaluated our artifact – the transaction-based approach for dependability evaluation – in an action research project to address the research questions. We conducted the action research project at the Bosch Group in two stages (see Figure 7.1). The objective of stage one was the design of the artifact, which addresses RQ1. The evaluation of the artifact was conducted in stage two and focused on answering RQ2 and RQ3.

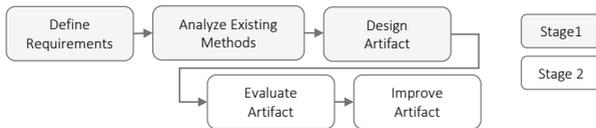


Figure 7.1: Research Process

7.2.2.1 Design of the Artifact – Stage 1

The challenges, requirements, and possible solutions of the interview study of Chapter 4 served as input for the requirement stage of our the transaction-based approach for dependability evaluation. We further analyzed existing approaches to define and evaluate quality characteristics (see Chapter 3.3). Two approaches ISO/IEC SQuaRE series [ISO14] (see Section 2.3) and Google’s SRE approach (see Section 2.8) for defining and evaluating SLOs and SLIs [BJPM16] are the basis for our research artifact. We integrated SRE practices of applying SLI and SLO as they were highlighted as a dominant industry practice in the interview study (see Section 4.6). In collaboration with different software practitioners in the case company and other companies providing services, we discussed the requirements of our artifacts (see

Section 7.3). After several iterations, the results were used in the next stage to design the initial artifact, which we will present in detail in Section 7.3.2. Stage one was performed during February 2019 and May 2019.

7.2.2.2 Application and Evaluation of Artifact – Stage 2

We conducted the evaluation of the artifact at the Bosch Group in stage two, which lasted from June 2019 until March 2020, to answer RQ2 and RQ3. An IoT service team of the Bosch Group applied our artifact to define and evaluate dependability requirements for several IoT transactions of their service. We involved different system stakeholders, like the product owner, the system architect, the service manager, and DevOps engineers, to perform the particular activities of the transaction-based approach for dependability evaluation. One researcher guided the application and a second researcher accompanied the application to validate the quality in use (with a focus on effectiveness, efficiency, and satisfaction) of the approach (see Table 7.2).

Table 7.2: Research Approach of Stage 2

	Observation of Application	Focus Group Discussion
Description	Observing the product team in executing the activities of the dependability evaluation approach	Interviewing team members for getting detailed feedback of the application of the approach by reflecting the application activities in retrospect
Participants	Chief Product Owner, System Architect, Service Manager, DevOps Engineers	System Architect, DevOps Engineer
Researcher Roles	One researcher as coach, second researcher as independent observer	One researcher as moderator, second researcher as minutes-taker

In addition, we performed a focus group with two of the team members (see Table 7.2), with whom we reviewed and discussed the artifact’s application to reduce a threat to internal validity. The focus group discussion lasted 2.5 hours. The two team members were actively involved in all of the performed activities of the transaction-based approach for dependability evaluation. With the interactive setting of the focus group, we were able to study the participants’ experiences, and reactions in detail [KBL08] and triangulate the results with our observation of the application of the artifact.

Furthermore, the focus group setting allowed us to interactively discuss two different stakeholder concerns, as one participant was the system architect, and the other one was a DevOps engineer. Both participants had several years of experience in developing and operating service-based systems. The system architect was responsible for different IoT services and had experience in monitoring, testing, and applying SLOs and SLIs. The DevOps engineer had experience in testing and monitoring, particularly in the application of distributed tracing.

We examined the application of the approach again by exemplarily going through the activities and reviewing the application. One of the researchers acted as moderator for the focus group and guided the participants throughout the approach. The moderator enabled the conversation and further ensured that the discussion stayed focused. After performing and reviewing each activity, the moderator asked for feedback, focusing on effectiveness, efficiency, satisfaction, as well as ideas for improvement (as documented in the interview guide [NZW20]).

The second researcher took the role of an independent observer and noted the participants' answers and reactions related to the artifact. The focus group protocol was analyzed applying qualitative content analysis [May14]. We decided to apply Mayring's approach for qualitative content analysis as it allows to perform a connection of deductive and inductive coding. Based on the research questions, we deductively formulated the highest level of the category system with a focus on the participant's statements regarding effectiveness, efficiency, and satisfaction. Moreover, we inductively formulated two additional first-level categories of challenges improvements and further sub-categories to differentiate aspects of the main categories.

We triangulated the observation of the application with the results of the focus group. The participant's opinions led to further improvements of the approach. In particular, we adapted the description of the concepts of service, transaction, SLO, and SLI and provided them via a company internal terminology management system.

We performed the evaluation in the context of an IoT service that was not released at the time of evaluation. Therefore, to evaluate the performance

of the IoT system under load, we emulated consumer behavior by producing synthetic workload (further described in Section 7.4)

7.3 Transaction-Based Approach for Dependability Evaluation - Stage 1

In this section, we describe the requirements for designing the artifact. Furthermore, we present the resulting artifact of the transaction-based approach for dependability evaluation (RQ1).

7.3.1 Define Requirements for the Design of the Artifact

Based on the outcomes of the interview study of Chapter 4 and the input from specific problem instances at the Bosch Group, we defined the following requirements and characteristics for the transaction-based approach for dependability evaluation.

- **Targeted at complex IoT Service Systems.** The approach is tailored for distributed IoT systems providing services. We consider IoT systems that are developed and operated by different and specialized teams. We focus on cloud-based or containerized systems that use RESTful HTTP or lightweight messaging (e.g., AMQP) and the integration of (Io)things.
- **Management of dependability from consumer view of value.** The goal is to provide a concrete approach that enables to evaluate the abstract concept of dependability, “perform as and when required” from a consumer-centric view. Therefore dependability requirements need to be defined by a cross-functional perspective where business, technical and operation perspectives are aligned. Furthermore, the dependability needs to be evaluated in the dynamic context of use in operation. The approach aims to enable fault detection in order to control the value provision to consumers with fault reactions. Moreover,

the artifact intends to enable continuous product improvement by observing anomalies in the dynamic context of use.

- **Dynamic system view and context propagation.** The approach needs to support the continuous observation of the dynamic context of use of IoT transactions from a consumer view on value. Therefore, it is necessary to propagate context across the components of an IoT system which provide component effects. Applying distributed tracing enables to propagate context of consumer transactions and to execute quality and performance measurements. It further enables us to observe anomalies in the high frequency of system changes and detect faults during operations to perform quick fault reactions.
- **Efficient for dynamic dependability evaluation.** The approach should address the industry context and needs to be efficient and easy to understand. This includes the efficiency with which practitioners achieve the specified goals in relation to expenditure of resources, like mental effort. Following the goal of reducing observation and control complexity, our approach focuses on component effects with high business value for evaluating the service provision from a dynamic system view.
- **Relies on existing and proven approaches: ISO/IEC 25040 and SRE.** As there are already existing standards and established industry approaches, they should form the basis for our artifact. We adapted them with concepts according to our problem context. Due to the criticism of the quality evaluation process of ISO/IEC 25040 (described in Section 2.3.6) for being too abstract and not focusing on operations, we enhanced the generic reference model with the concept of the IoT transaction as the concrete entity of interest. Furthermore, we integrated distributed tracing as an observation and measurement method enabling context propagation. To complement the needs for consumer focus and observation, we integrated control loops with the concepts of SLI and SLO.

7.3.2 Transaction-Based Approach for Dependability Evaluation (RQ1)

This section presents the results for RQ1, the designed artifact of the transaction-based approach for dependability evaluation. The approach includes the different activities and tasks to guide the execution (see Figure 7.2).

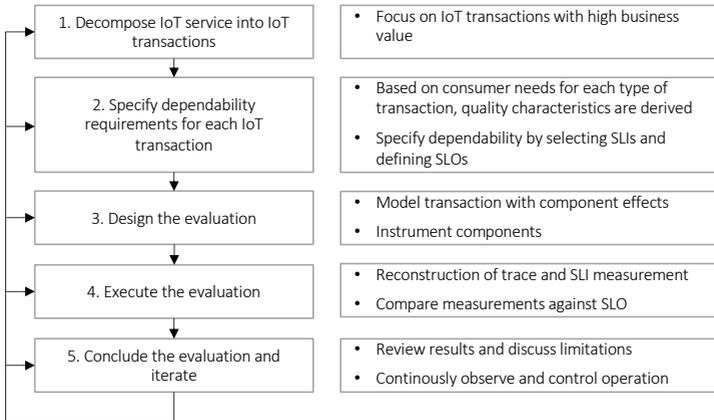


Figure 7.2: Activities of the Transaction-based Approach for Dependability Evaluation

1. Decompose IoT service into IoT transactions

In this first activity, we define IoT transaction types that need to be analyzed. Hence, we decompose an IoT service into IoT transactions, delivering a quantum of value to the consumer. From a business model view, we focus on IoT transactions with high business value. These include transactions for which faults may propagate to significant quality deficiencies (failures) for a consumer. We document the types of IoT transactions and their quantum of value that they deliver to a consumer.

2. Specify dependability requirements for each IoT transaction

The target entity for evaluation is the IoT transaction, which needs to be controlled to deliver a quantum of value to the consumer. For each type

of IoT transaction, a set of requirements to indicate dependability can be defined by applying a quality model (like ISO/IEC [ISO11d]). Therefore, we derive suitable quality characteristics and sub-characteristics from consumer needs with regard to the quantum of value to be delivered by each of the IoT transactions. We specify IoT transaction dependability by selecting quality measures (SLIs) and defining quality objectives represented by target values (SLOs). Target values and acceptable range of value with a numerical threshold indicate the need for further investigation or intervention. The specification of the IoT transaction dependability has to be documented using the following items:

- transaction type with reference to consumer value
- quality characteristic and sub-characteristic
- quality measure (SLI)
- quality objective with target values (SLO)
 - target value for SLI or
 - acceptable range of value for SLI

The selection of appropriate quality characteristics and definition of SLIs and SLOs is an interdisciplinary activity with consumer, business, and product implications, which have to be reflected in the specification. Often there are trade-offs needed between different stakeholder groups. The following aspects, inspired by SRE practice [BJPM16] and ISO/IEC 25030 [ISO19b] shall assist the specification of SLIs and SLOs.

- **Simplicity:** Have as few SLOs as possible. Choose relevant characteristics and measures reflecting IoT transaction dependability.
- **Realistic SLOs:** Analyze whether SLOs are consistent with other quality requirements and whether prioritization is needed. Set realistic SLOs that can be achieved regarding constraints such as business constraints.

- **Safety margin:** Start with a conservative SLO and tighten it iteratively. If SLO is offered to consumers and customers, set a tighter internal SLO for having options for tactical intervention.

3. Design the evaluation

Model system structure view and dynamic transaction flow: Identify components that provide component effects to indicate IoT transaction dependability. The component effects shall be described from the perspective of the outcome, providing a quantum of value to the consumer. Model the high-level conceptual workflow of the IoT transaction, including components and component effects.

Instrument components: To apply distributed tracing as the observation method, instrument the components providing component effects. Metadata can be propagated through the system, including span context and the start and end time of a span to reconstruct the corresponding trace.

4. Execute the evaluation

The measurements are performed with the initiation and processing of IoT transactions. Trace, span, and parent ID are propagated, and span time stamps and duration are logged. The tracing agent collects the span data that a tracing backend receives to reconstruct the trace out of the corresponding spans. In the following, the trace data has to be aggregated and analyzed. The trace measurements need to be compared against the targets of SLOs to evaluate the fulfillment of requirements.

5. Conclude the evaluation and iterate

To conclude the evaluation, review the results, and discuss the limitations of the evaluation procedure. Furthermore, this activity represents the iterative evaluation during operation. Continuous observations to find deviation in the SLI measurements compared to SLOs are necessary in order to decide whether or not investigation or intervention is required. The information from distributed trace data enables us to detect faults and respond to them

with fault reactions. With an increasing exploration of the context of use, further iteration of the activities 1-5 is needed. For example, the model of component effects as observation and control points have to be continuously validated and if necessary adapted. Moreover, continuously evaluate the suitability of SLIs and SLOs, which must be adapted to the changing environment, including changes in consumer needs and the system.

7.4 Evaluation of the Transaction-Based Approach for Dependability Evaluation - Stage 2 (RQ2 and RQ3)

To evaluate the designed artifact, we applied it in the context of an IoT solution, the Remote Measurement Service, at Bosch Mobility Solutions. In the following section, we provide an abstract description of the IoT solution and its main components. Furthermore, we present the application of the transaction-based approach for dependability evaluation in the context of the Remote Measurement Service. Besides, we describe the challenges we perceived by performing the different activities and tasks of the artifact. Finally, we present the results of the focus group to evaluate the effectiveness, efficiency, and satisfaction of the approach (RQ2 and RQ3).

7.4.1 System Description

The Remote Measurement Service (RMS) is a cloud-based service as part of the vehicle management solution, which offers an access and communication channel to the vehicle. The RMS provides over the air access to vehicle measurements, e.g., the engine temperature or the vehicle speed with different protocols (e.g., CAN raw and XCP). A consumer can configure a measurement job to extract data from the vehicle. The system architecture, including the essential components and communication flows, is abstracted in Figure 7.3.

The component **Zuul** [Net20] serves as an API gateway and routes a consumer request to corresponding components. The **Remote Measurement (RM) Job Configuration** is responsible for the events triggered by

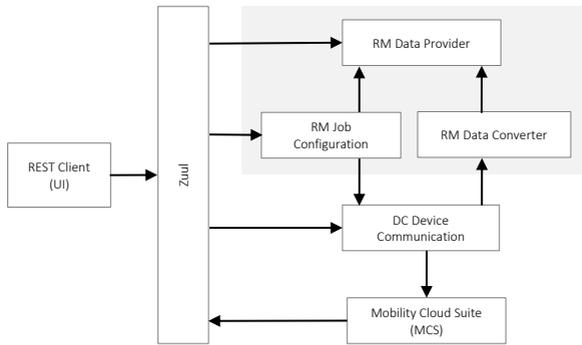


Figure 7.3: Overview Remote Measurement System

the consumer, e.g., the activation or deactivation of a measurement job configuration. The measurement job configuration is passed to the **DC Device Communication** component, which acts as a gateway to route job configurations and job results (vehicle data) to other components. The measurement job configuration is routed to the **Mobility Cloud Suite (MCS)**, an external component that provides the vehicle data. The measurement job results (in the form of binary data) are sent to the Zuul, which routes the data to the DC Device Communication. The **RM Data Converter** converts the data into a consumer-defined format and passes it on to the **RM Data Provider** component, which stores and provides the data to the consumer. The components in the grey box (see Figure 7.3) are developed by the team.

7.4.2 Application of the Transaction-Based Approach for Dependability Evaluation

In the following, we present the application of our artifact by exemplarily describing the execution of the five activities of the transaction-based approach for dependability evaluation.

1. Decompose IoT service into IoT transactions

The first task is to decompose the IoT service into IoT transactions providing

quantifiable value to the consumer. Consumers for the RMS are original equipment manufacturer (OEM) engineers who want to get access to measurement data from the vehicle's lifecycle. The consumer need can be defined as follows.

Consumer need: An engineer (consumer) of an OEM (customer) needs to have access to the required field data of the vehicle that can be downloaded from central storage to process or analyze data using a client UI. Four types of IoT transactions could be defined.

- Activate a measurement job configuration (T1)
- Deactivate a measurement job configuration (T2)
- Provide access to measurement results (T3)
- Download measurement results (T4)

For the scope of this work, T3 – provide access to measurement results – was applied to evaluate the approach. As the service was still under development at the evaluation stage, T3 was the only transaction type for which we could perform dependability measurements. The quantum of value for T3 corresponds with the access to the requested vehicle measurement data in a specific data format.

2. Specify dependability requirements for each IoT transaction

For defining dependability requirements for T3, we projected the transaction characteristics onto the quality models and related measures of ISO/IEC 25000 series [ISO14]. Together with the chief product owner, two system architects, and the service manager, we selected essential quality characteristics. The interpretation of the concept of dependability “*perform as and when required*” led to the requirements presented in the following paragraphs. The corresponding specification defining SLIs, their measurement function, and SLOs are summarized in Table 7.3 and 7.4. The specific target values for the SLOs cannot be published for this dissertation; therefore, we have abstracted the SLO syntax.

A critical quality in use characteristics for T3 is the **effectiveness**, the ability to successfully provide access to measurement results accessible in the specified data format by transaction completion. Furthermore, the characteristic **efficiency**, was derived from consumer needs. It includes the system's ability for quick processing of measurement results and providing access to them, as well as the performance of how many transactions the system successfully processes in an observation interval.

The **effectiveness** of an IoT transaction can be indicated by the *objective achieved* measure of the QIU model. ISO/IEC 25022 defines the *objective achieved* measure as the “*proportion of the objectives of the task that are achieved correctly without assistance*” [ISO11c]. We cannot directly observe the sum of the objectives achieved, which corresponds to our concept of a quantum of value of an IoT transaction, as they require consumer feedback and inspecting the outcome in order to assess the correctness of the measurement. An exemplary approach to assess the correctness of the outcome is synthetic probing, by injecting synthetic data, for which we know the correct outcome. However, synthetic probing cannot represent the full consumer context of use in operation. Therefore, we propose to indicate the *effectiveness* of T3 to deliver a quantum of value through its completion by observing and quantifying the provided component effects in operation. We adapted the definition of the *objective achieved* measure as *transaction completeness* accordingly.

- **Transaction completeness:** The proportion of the component effects of a transaction that are provided to facilitate the accomplishment of a quantum of value of a transaction.

The corresponding measurement function and the SLO form are defined in Table 7.3. For each missing or incorrect component effect, representing a fault, a proportional value P_e can be assigned. If the sum of proportional values exceeds the value of one, the *transaction completeness measure* C_1 is set to zero, as we do not consider negative transaction completeness.

In the context of *effectiveness* indication, we also discussed the charac-

Table 7.3: Effectiveness SLIs and SLOs for T3 (for an individual instance of T3)

SLI	Measurement Function	SLO
Transaction completeness C_1 (derived from objectives achieved measure)	$C_1 := \max(0, B)$ $B = (1 - \sum_e P_e)$ P_e : Proportional value of each missing or incorrect component effect e (with a fault) of the transaction (max. value of each $P_e = 1$)	target $\leq C_1$ per transaction
Transaction completeness C_2 (derived from functional appropriateness measure)	$C_2 = 1 - N_f / N_r$ N_f : Number of component effects missing or incorrect (with a fault) among those that are required for delivering the quantum of value N_r : Number of component effects required for delivering the quantum of value	target $\leq C_2$ per transaction

teristic of *functional appropriateness*, which is a sub-characteristic of the *functional suitability* of the PQ model. ISO/IEC 25023 [ISO16] defines the *functional appropriateness* as the “degree to which the functions facilitate the accomplishment of specified tasks and objectives”. The *functional appropriateness* measure is the “[...] proportion of the functions required by the user [that] provides appropriate outcome to achieve a specific usage objective”. We derived a second *transaction completeness* measure C_2 . The measurement function C_2 , derived from the *functional appropriateness* [ISO11c] is similar to the measurement function C_1 , derived from the *objectives achieved* measure [ISO16], but represents a special case in which all component effects are equally weighted. We assume this measurement function is suitable when a provider does not have enough knowledge about the consumer context to set a proportional value for an anomalous component effect.

The target value (SLO) for the proportion of completeness, introduces a class boundary to classify instances of IoT transactions into complete and not complete. We can indicate a proportional value of a missing and incorrect component effect via an anomalous span pattern of the component effect.

Identifying the component effects is described in the following activity (3. Design the evaluation) of our approach. Furthermore, in the following

activity three, we identified expected span patterns representing component effects.

To indicate the **efficiency** of the value provision from a consumer view, the number of completed transactions in a time interval needs to be observed. We apply the SLI of *transaction throughput*, which is similar to the *time efficiency* measure of ISO/IEC 25022 [ISO11c] and the *mean throughput* measure of ISO/IEC 25023 [ISO16].

- **Transaction throughput:** Number of the transactions that are completed within an observation interval.

The *transaction throughput* target (see Table 7.4) is formulated for a particular observation interval, a standardized time unit, whose interval must be large enough to quantify a difference in the observed property.

Table 7.4: Efficiency SLIs and SLOs for T3 (across several instances of T3)

SLI	Measurement Function	SLO
Transaction throughput Q_j	$Q_j = N_{e,j} / \Delta t_j$ $N_{e,j}$: Number of transactions completed during observation interval j Δt_j : observation interval j	transactions completed per time unit (e.g. number of transactions per min): target $\leq Q_j$
Transaction completion ratio R_j	$R_j = N_{e,j} / N_{i,j}$ $N_{e,j}$: Number of transactions completed during observation interval j $N_{i,j}$: Total number of transactions initiated during observation interval j	proportion of transactions completed per time unit: target $\leq R_j$
Transaction turnaround time t_k	$t_k = (t_{k,c} - t_{k,i})$ $t_{k,i}$: Time of initiating a transaction k $t_{k,c}$: Time of completing a transaction k	proportion p of transactions completed in t_k with $t_k \leq$ target (in e.g. ms)

In order to assess the quantity of completed transactions relative to the initiated transactions in an observation interval, we formulated the SLI of *transaction completion ratio* based on the *task completed* measure of ISO/IEC 25022 [ISO11c].

- **Transaction completion ratio:** Proportion of the transactions that are

completed within a time interval in comparison to the total number of transactions initiated.

The **time efficiency** of the transaction completion can be indicated by the *transaction turnaround time*, which is based on the *task time* measure of ISO/IEC 25022 [ISO11c] and the *mean turnaround time* measure of ISO/IEC 25023 [ISO16].

- **Transaction turnaround time:** The time taken to successfully complete a transaction.

The distribution of the *transaction turnaround times* can be indicated by their *n*th percentile under expected load conditions, which we adapted from the *time behavior measures* of ISO/IEC 25023 [ISO16].

Challenges: Initially, the team had difficulties interpreting the abstract definition of the concept dependability: “*perform as, and when required*” [IEC15]. They struggled to determine which characteristics have a significant influence on the consumer perceived quality. DevOps engineers and PO tended to suggest product quality properties from their provider view. They suggested measures related to security and maintainability characteristics from an provider view, which do not directly contribute to the value provision from a consumer view. As the team was prone to take the provider view instead of the consumer view of value, we identified the need to strengthen the differentiation of the the concerns and views for which quality characteristics, related measures, and targets are formulated. For operations, the provider and consumer view with the characteristics and SLIs and SLOs need to be linked appropriately. For this study of defining requirements and evaluating operations by emulating the behavior of one consumer, the performance measures from provider and consumer view are similar. From a provider viewpoint, only the assignment of measures to the characteristics of *effectiveness* and *efficiency* would change (e.g., the transaction throughput can be assigned to indicate provider effectiveness). Besides, for a provider view, additional performance characteristics like *resource utilization* and related measures would be of interest.

Moreover, we identified difficulties related to the definitions of the concepts of SLOs and SLIs. Some participants had the impression that all SLOs have to be part of a contractual agreement like a service level agreement (SLA). Therefore, we created a terminology part in the company’s internal glossary for the concepts of IoT service, transaction, SLA, SLO, and SLI. In the glossary, we explain the concepts and highlight the importance of the different stakeholder concerns from which they should be developed.

3. Design the evaluation

Model System structure view and dynamic IoT transaction flow: We modeled the (high-level) IoT transaction flow applying the system structure view. We abstracted the components that provide component effects to the IoT transaction and modeled the dynamic transaction flow in Figure 7.4. For T3, each of the component effects needs to be successful (without a fault) to consider the IoT transaction as complete.

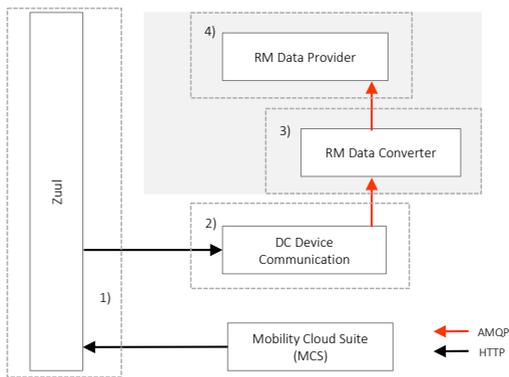


Figure 7.4: T3: Provide access to measurement results

- 1) Zuul: receives data and routes data to DC Device Communication
- 2) DC Device Communication: receives data in binary data format and inserts data into messaging queue
- 3) RM Data Converter: converts data into a consumer specified format

and inserts data into messaging queue

4) RM Data Provider: persists data in data base

Since the Remote Measurement System was not in production at the time of the study, and the MCS is an external service system, we performed load tests by emulating measurement data. For the load test, we used JMeter [Fou20], an application to perform functional and performance load tests. The load test scenario was based on requirements provided by the customer (OEM). Including the connection of ten vehicles, whereupon each of the ten vehicles sending a message every 120 seconds to the DC Device communication. Since the MCS is an external service system, we needed to set up an MCS mock component, which emulated the external real MCS that forwards the processed vehicle data and passes it to the DC Device Communication.

Instrument components: Two distributed tracing systems had been available for the product team at the time of the study, Jaeger [Jae20] and New Relic [Rel20]. Jaeger [Jae20], is an open-source distributed tracing tool and is inspired by Dapper [SBB+10], described in Section 2.7.2. As the Zuul, RM Data Provider, RM Data Converter, and DC Device Communication have already implemented a Jaeger tracer, no modifications to instrument components were required. New Relic (NR) [Rel20], is an application performance management tool and supports the analysis of distributed tracing data with an extensive UI.

4. Execute the evaluation

We performed a load test according to customer requirements described in Activity 3.

Reconstruction of trace and measurement of SLIs: We applied Jaeger and NR to reconstruct and visualize the traces. With NR, we could not reconstruct the whole trace due to AMQP as the communication protocol. At the time of the evaluation, in NR the trace context was not attached to AMQP; therefore we could not gain full trace visibility. With Jaeger, we were

able to produce the overall trace. However, due to Jaeger’s limited aggregation capability and visualization, we imported the data in NR. Since the measurements of runtime data generated in the research project cannot be published, below, we provide some exemplary abstraction and visualizations for the SLIs and SLOs, produced via NR.

- Transaction completeness:** We were able to deductively assign component effects to the span patterns, as shown in Figure 7.5. It illustrates an exemplary trace of an instance of a transaction type T3 that we abstracted from the Jaeger UI. According to the completeness measure C_1 , we assigned a proportional value P_e for each component effect, in the case a fault is detected. As described in Activity 3 – Design the evaluation –, each component effect has to be successful in order to consider a transaction as complete; therefore we set $P_1 = P_2 = P_3 = P_4 = 1$. For the instance of T3 in Figure 7.5 no faults for the component effects were detected, consequently $C_1 = 1$. This represents the maximum value possible for the measure of transaction completeness. Thus, we can classify this instance of T3 as complete.

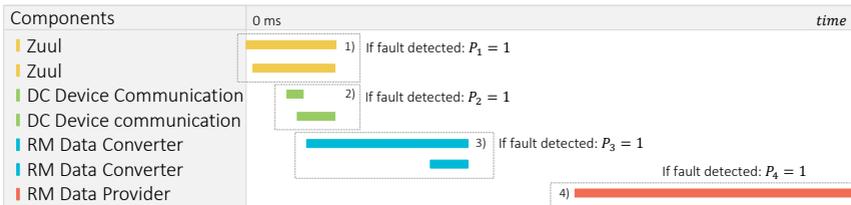


Figure 7.5: Exemplary trace including component effects abstracted from Jaeger [Jae20]

For a more complex type of transaction, for which software practitioners do not have sufficient knowledge about the operations and the component interactions, it would be possible to indicate a completed transaction inductively through particular trace patterns (by the number and the compilation of span patterns). We could perform this indication by labeling the span patterns in a machine learning context.

- **Transaction throughput:** Figure 7.6 presents the SLI measurements for the transaction throughput. It illustrates the number of completed transactions as a function of time plotted over the load test interval.



Figure 7.6: Number of transactions as a function of time

- **Transaction completion ratio:** As we performed a load test and did not simulate faults, 100% of the transactions of the load test were completed. Hence, we refrain from visualizing the transaction completion ratio as it corresponds to Figure 7.6, the number of completed transactions plotted as a function of time.
- **Transactions turnaround time percentiles:** Figure 7.7 visualizes the 99th, 98th, 95th, and 50th percentile transactions turnaround time t_k plotted as function of time of the load test.

5. Conclude the evaluation and iterate

Since the system was not in production at the time of the artifact evaluation, we conducted a load test. However, the load test only emulates consumer load but cannot represent the consumer’s operation real-world context. Therefore, we see a threat to the interpretation validity of the results concerning the service dependability from a consumer view of value. Nevertheless, the

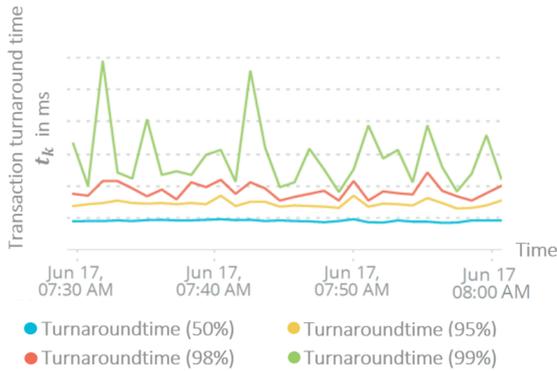


Figure 7.7: Transactions turnaround time t_k as a function of time for different percentiles

results were valuable in terms of the IoT system’s performance capabilities, especially concerning the system’s time-behavior to process transactions. We identified that the RM Data Provider and the RM Data Converter consume the largest part of the transaction turnaround time. Besides, we detected a fault, a bottleneck between these two components (see Figure 7.5). Within the relatively large time gap, the enqueued data were waiting for being consumed by the data provider. This time interval increases with increasing load. We detected a defect in the conception of the underperforming data provider, which the team subsequently redesigned to decrease the bottleneck.

In the future, we will compare the results from the load test with real-world consumer load results in operation. Deviations from expected and normal system behavior can indicate anomalies that require further investigation. In addition, we identified and discussed the following errors related to modeling and measurement [Sar10], which could lead to false positives and false negatives.

- Modeling errors, which occur because the conceptual models constructed provide a simplified or insufficient description of the phenomena in reality. This includes modeling the transaction quality based on assumptions of the consumer needs that we need to validate by

exploring the context of use in operation.

- Observational errors, which arise through inaccuracies of the measurement instrument.
- Errors related to distributed tracing as the measurement instrument, impacting the system performance.

Once the system is in production and we can obtain consumer feedback, we have to further investigate the implications of the discussed errors. This mainly includes the potential modelling errors. Thus, we have to validate the suitability of SLIs and SLOs. Moreover, the model of component effects, the selection of observation points through distributed tracing, and the assigned proportional value for the component effects have to be evaluated. If we detect deviations in the conceptual model, we need to revise the model and perform the model validation again.

7.4.3 Focus Group

In this section, we provide the focus group results with two stakeholders involved in all the activities of executing the approach. We present the results of the qualitative content analysis, as described in Section 7.2.2.2, regarding the participants' answers about the characteristics of effectiveness, efficiency, satisfaction, as well as the challenges and improvements to our approach.

Effectiveness: The discussion of the focus group acknowledged that the approach with the concept of dependability supports the operationalization of the consumer view of value. The participants stated that the application of the approach promotes early discussion and alignment of different stakeholder perspectives. The decomposition of a service in IoT transactions with component effects is a suitable abstraction to bring together business, technical, and operation domains and provide a common understanding of essential quality characteristics from a consumer view of value. The participants highlighted the application of the concepts SLO, SLI as they allow them to formulate measurable targets with regard to consumer value. Through

the increased comprehension of consumer needs and the differentiation of them regarding value, organizations can take decisions on an explicit information base. This includes the decisions about prioritizing consumer concerns against operational and business concerns, and operationalizing them. Therefore, the approach supports clarifying ambiguities about the concept of quality from a consumer and a provider view and how to integrate them.

Moreover, the artifact supported the understanding of how the IoT system needs to be optimized to enable the transaction's effective and efficient execution. The participants stated that without the approach, they tended to focus on the functional view and monitored individual components while neglecting the dynamic transaction view. The synthetic probing, which the team applies, is also capturing a dynamic perspective. However, synthetic probing approximates the consumer experience, unlike distributed tracing, which is able to capture real-world consumer transactions during operation. The participants further highlighted that the approach supports getting an overview of the component dependencies to identify critical paths.

Efficiency and satisfaction: The participants explained that without the approach, their procedure was less structured and differentiated. Previously, they focused on the service or system level and did not differentiate at the level of individual IoT transactions. The approach provides systematic guidance to focus on high-value transactions and quality characteristics a consumer cares about. Furthermore, as the approach focuses on identifying relevant component effects of an IoT transaction, it is possible to reduce the observation space while at the same time getting a holistic overview of the dynamic processing of an IoT transaction.

Moreover, the participants stated that the communication efficiency in collaborating with the different system stakeholders was increased with the systematic approach and the shared understanding of the concepts. Thus also trade-offs could be made explicit. Instead of defining generic service requirements on system or component level, the stakeholders highlighted the transaction-based quality requirements, which are more concrete for the

different stakeholders. We further discussed with the participants that the breakdown of target values on component level could be iteratively derived through operation data (but respectively in the context of a transaction type and in relation to a consumer context).

In terms of understandability, the participants noted that the approach is well-defined and explained in a comprehensible way and that they are satisfied with it. While there was a discussion on whether the approach introduces additional specification efforts, participants also expressed that the expected benefits outweighed the compromises inherent in any structured method, especially as the precise specification is primarily done for high-value transactions and not for every type of transaction a system provides. The incorporation of SLOs and SLIs initially confused the participants, as the terminology was too abstract, and concrete examples of applications were missing. By creating the internal documentation with the concepts of SLA, SLO and SLI, and related examples, the experts confirmed that a clear distinction and application could be ensured.

Challenges and improvements: Initially, the stakeholders perceived the specification of transaction dependability requirements and the interpretation of the abstract definition: *“perform as and when required”* as challenging. The large number of possible projections onto quality models with characteristics and measures of the SQuaRE series was difficult. We have therefore adapted the approach by distinguishing more precisely between consumer and provider concerns. Furthermore, the researchers suggested starting the projection of the concept of dependability onto the quality model of ISO/IEC 25010 with the trivial quality characteristics of efficiency and effectiveness.

Regarding the SLI measurement, the stakeholders described that the aggregation of traces was challenging. The traces are raw data and have to be sufficiently processed to quantify transaction properties and compare them against SLOs. Since Jaeger was not sufficient for trace aggregation and NR could not trace the whole transaction due to AMQP, the traces had to be converted and imported to NR for performing SLI measurement. Therefore, we recommend applying the trace context specification of the World Wide Web

Consortium (W3C) for future projects. It is a de facto standard, developed by open-source and commercial tool providers, which defines a unified format for propagating tracing context between components [W3C20]. Most of the open-source and commercial tracing tools support this context specification. Applying a standardized trace context reveals the potential to include further external components like the mobility cloud suite for this research study. Lastly, for the application of distributed tracing as the measurement method, the participants noted that with the increasing application of distributed tracing, an evaluation of data storage and sampling strategies is needed.

7.5 Discussion

The study reveals that our transaction-based approach for dependability evaluation facilitates a shared understanding of consumer needs regarding value and how to prioritize and operationalize them. The concept of IoT service and its decomposition into IoT transactions makes the derivation of quality requirements regarding consumer value tangible to the different system stakeholders. Instead of generally defining dependability requirements on service system and component level, with our approach, we determine them in a context of an IoT transaction and the required component effects providing value to a consumer. The artifact's application indicates that it can improve collaboration and communication between teams and align the perspective from which they develop and operate services. Besides, the approach fosters the importance of continuous feedback cycles through observation, evaluation, and control of the transaction dependability with SLIs and SLOs. The approach enables us to detect anomalies in the dynamic context of use and to react to them. For example, changes in the quality of value provision can be observed when the system is modified. Furthermore, it can be applied to improve service quality gradually. Therefore, the approach increased the team's awareness of the validation of the operationalization and iterative exploration of the consumer context of use. Besides, the team confirmed to get valuable insight into the system's performance that they

can use to adapt the SLOs iteratively.

In the future, we see a high potential for the indication of system performance anomalies from a provider viewpoint by quantifying the rate of change (gradient) of transaction completeness across the instances of transactions processed by the system. We also identified that it is possible to extend the concept of faults (as missing or incorrect component effects) to faults, including anomalies in the time behavior of providing component effects (e.g., if a component effect takes too long). Consequently, we could assign proportional values P_e for timing faults of component effects. Moreover, we can apply the completeness indicator to assess completed transactions from a customer view of value. This information can be applied to assess compliance with a contract and perform billing, as stated in Section 6.2.2.

Although modern distributed tracing systems like Jaeger are designed for minimal performance overhead, they still create additional load. Therefore, the usage of these systems needs to be carefully evaluated.

Following, we discuss similarities and differences of the TAICOS [HPK20] approach (described in Section 3.3.1) to our transaction-based approach for dependability evaluation of this chapter, as it presents a closely related work. In the first activity of the TAICOS approach, user tasks are elicited. This activity is similar to our first activity of decomposing an IoT service into IoT transactions and selecting IoT transactions with high business relevance. The collection of task-relevant data such as dependencies, constraints, and the determination of the task goal corresponds to the definition of the quantum of value delivered by the transaction and is covered by activity one of our the transaction-based approach for dependability evaluation. Furthermore, the determination of subtasks can be mapped to activities two and three of our approach, where we select high-value component effects. Haindl et al.'s [HPK20] conception of subtasks can be roughly mapped to our concept of component effects, which are provided by people, acting as components. However, as our concept of IoT services takes the view towards a workflow, including component effects from business process relevance, we have a broader perspective than their understanding of a task (as a sequence of actions performed by people [HPK20]). With an IoT transaction, we con-

sider component effects, which are provided by components that include but are not limited to people. In further detailing the elicitation of task details, Haindl et al. [HPK20] include a second perspective and go beyond the task concept of user's interactions during execution. They introduce the concept of *system responsibilities* supporting user intentions, reflecting software functionalities. This extension of subtasks approximates parts of our concept of component effects. Furthermore, the task detailing step, which targets to identify a “*minimal and satisfactory technical solutions to execute the task effectively*” [HPK20], is similar to our idea of modeling transaction completeness and designing a class boundary classifying instances of IoT transactions into complete and not complete. After the task elicitation, with step three, Haindl et al. [HPK20] target to model the control flow between the subtasks, similar like we instruct with activity three the modeling of the IoT transaction's dynamic execution flow. However, we combine a system structure view and dynamic execution of the IoT transactions. In comparison Haindl et al. [HPK20] model effects, without reference to the components providing the effects. Similiar to our selection of IoT transactions with high business value in activity one, Haindl et al. [HPK20] select the most important user tasks by considering the business model. Next, with step four, stakeholder interests are elicited, which include non-technical objectives, such as “*quality-related, operational, business, legal and other non-behavioral interests*” [HPK20] that influence how the software system must support user tasks. The elicited stakeholder interests are documented to record objectives and expectations towards the software system in step five. Furthermore, the stakeholder interests are refined and documented with relevant stakeholders to generate a common understanding (step six). These tasks can be mapped to the the activities one and two of our approach. The stakeholder interests and user tasks are analyzed in a two-dimensional task-interest matrix with step seven, which does not yet include qualitative fulfillment criteria. Haindl et al. [HPK20] target to assure that stakeholder interests are analyzed for each user task individually. The task-interest matrix then serves as the basis to specify quantitative constraints by selecting measures and defining thresholds to evaluate the fulfillment of stakeholder interests. The steps seven and

eight are similar to our activities two of deriving quality characteristics from consumer needs of an IoT transaction and specify requirements by selecting quality measures (SLIs) and defining quality targets with SLOs.

We identified several symmetries of the TAICOS approach to our IoT transaction concept and the transaction-based approach for dependability evaluation. Their concept of user task can be roughly mapped onto our concept of IoT transaction and some of their stakeholder interest to the quantum of value a transaction delivers. The work of Haindl et al. [HPK20] focuses on software development and operation in general. In comparison, our approach is conceptualized for distributed system architectures like complex IoT systems, which are developed and operated by different teams and organizations. Therefore, our IoT transaction approach targets to reduce the observation complexity by modeling relevant component effects and operationalizing them. Besides, the approach of Haindl et al. [HPK20] is more generic and includes other stakeholder views (like development), where we focus on customer and consumer expectations regarding value and their operationalization. As our work deals with evaluating the dynamic system behavior to serve the dependability needs of a consumer, our approach mainly focuses on their operationalization and observation through distributed tracing during operations. Nevertheless, the idea of both approaches is to model and evaluate individual quality requirements for IoT transactions/tasks and not on a service/system/application level. Besides, both approaches target to align different system stakeholders and generate a common understanding of stakeholder needs and expectations.

7.6 Threats to Validity

We anticipate some personal bias as a threat to **internal validity** regarding the participant's statements and answers during the focus group session. Furthermore, participants may tend to comment and answer in confirmation of our approach, which could have led to confirmation bias. In addition, we

see the possibility that the focus group results are the phenomenon of group-think. We assume these threats as rather low, as the participants seemed not to be worried about pointing out negative aspects of the artifact and moreover held partially different opinions. Furthermore, we tried to mitigate the implication of these threats by triangulating the results of the focus group with our observations during the application of the transaction-based approach for dependability evaluation. Since we did not audio-record the focus group, we see a risk of researcher bias for analyzing and interpreting the results. We tried to control this risk as one researcher acted as the moderator, making some notes, and the second researcher took the role of an independent observer who was responsible for recording the reactions, statements, and answers. We counterchecked the minutes to increase interpretation validity.

Since we have described, documented, and attached an exemplary application of the artifact, we consider that the study can be reproduced, and a threat to **reliability** has been reduced. However, we see a threat to reliability regarding a selection bias of the researchers and participants towards the choice of the measures for the quality characteristics. We regard this threat as relatively small as we can map the selected measures onto the common “golden signals” (see Section 2.8), without taking them as an explicit pattern. This is different for the completeness measure, which is highly dependable on what a consumer perceives as minimally complete. However, for this case study, we think the identified component effects are necessarily required, and therefore the selection of the measure and target value seems to be clear. However, we do not claim that other researchers applying our approach would select the same characteristics and measures. Furthermore, we regard this threat as negligible because this study’s focus was on the evaluation of the quality in use of the approach and not the suitability of the IoT transaction effectiveness and efficiency measures.

Although our IoT transaction concept is designed for IoT service systems, we think it can also be applied to other service-based or monolithic applications. However, further empirical studies for other IoT service systems and transfer to other domains are needed. In terms of **external validity**, we

have the limitation that the application of the artifact in one action research project is too little to claim generalizability on a quantitative level. Hence, further studies are needed.

7.7 Conclusion

The quality of value provision is a complex concept. It causes challenges for evaluating and controlling the value provision to consumers of IoT services. Therefore, we designed and evaluated the transaction-based approach for dependability evaluation in an action research project. We enabled the definition and evaluation of service and transaction dependability from a consumer view of value. We designed our artifact based on emerging industry approaches like SRE [BJPM16] and traditional frameworks like the ISO/IEC 25000 [ISO14] series. Moreover, we integrated into our approach the concept of IoT transaction comprising component effects as a discrete entity, which can be made observable by distributed tracing. Hence, by the systematic derivation of quality measures and comparison of actual values against previously defined target values, the organization is able to detect anomalies in the temporal behavior of the value provision. We evaluated the quality in use of the approach, focusing on effectiveness, efficiency, and satisfaction by applying it to an IoT service of the Bosch Group. Our action research project demonstrated that the artifact enables consumer-centricity across different system stakeholders. It transforms the complexity of observation and control of service provision into discrete observable and controllable IoT transactions, for which dependability characteristics and targets can be defined, indicated, and evaluated in a dynamic context of use.

The results of applying the artifact indicate a cross-functional alignment on consumer concerns across different teams and the potential to break down silos. The artifact focuses on observing high-value contributing components and their behavior in the context of a transaction, emphasizing characteristics that are of importance for consumers. Therefore, it fosters early discussion about important transaction characteristics from a consumer view while

involving different system stakeholders like developers, operators, product owners, architects, and service managers. The artifact enabled us to create a shared understanding of transaction and service dependability aspects and its related observability and control needs. This again promotes a mindset and awareness according to observability and quality control.

The participants experienced the quality in use of the artifact as positive. They considered the concepts and the approach as understandable. Nevertheless, it remains challenging to identify appropriate quality characteristics representing consumer needs and appropriate measures with target values to indicate transaction dependability from a consumer view of value. The evaluation led to further improvements. We described the concepts IoT service and IoT transaction and the relationship among the quality model, with quality characteristics, measures (SLIs), and target values (SLOs) precisely, and provided them via a company internal terminology management system.

The aggregation of individual trace data to generate indicators for the operational performance of the systems was challenging. Due to the use of AMQP as the messaging protocol, we had to convert the Jaeger trace data to import them into NR. For future projects, we prefer applying the standardized W3C context specification, which has the potential of integrated trace observability, including external components. Besides, strategies for trace data storage and trace sampling are needed.

The approach enables detecting anomalies in the dynamic context of use with which fault reaction can be performed to control the value provision. Our approach fosters the importance of continuous feedback cycles and adaptation of quality models, including measures and target values by iterative and inductive exploration of the context of use. Future work should focus on further conducting empirical data. This includes application and implementation for projects that are already in production and under consumer load.

Researchers can use the insights to further develop industry-oriented observation methods and procedures. Practitioners receive guidance on developing consumer-centered indicators and observing and quantifying dynamic system behavior to evaluate the dependability of the value provision.



CONCLUSION

This chapter summarizes the contributions of this dissertation. We put our three main contributions – the classification of software anomalies, the concept of IoT transaction, and the transaction-based approach for dependability evaluation – into context and discuss their implications and limitations. Furthermore, we describe our current work and illustrate how the contributions can be further evaluated and improved.

8.1 Summary of Contributions

At the beginning of this work, the area of observability, monitoring, and the handling of anomalies of complex distributed systems formed a diffuse structure of challenges, requirements, and existing solutions. Following a design science paradigm, we investigated this area and specific problem instances in detail with our industry interview study **On Observability and Monitoring of distributed Systems (C1)**. We interviewed 28 industry experts from 16 organizations. With this study, we holistically analyzed technical and organizational perspectives and extracted challenges, requirements, and applied practices and solutions. These served as further input for the problem concep-

tualization and solution design for our three main contributions, targeting to improve real-world software and system engineering practice. The interview study identified that monitoring and handling anomalies are not purely technical issues but organizational and strategic topics that strongly influences a company's success. Shared concepts and collaborative observation of the value provision of an IoT service system are needed to detect anomalies in the context of use. This allows the organization to create feedback cycles to control service, system, and organization.

Our second contribution, the **Classification of System Anomalies (C2)**, serves as a shared concept to support communication about system anomalies and collaboration across different specialized organizational units. Our classification enabled us to differentiate the purposes of handling different classes of anomalies and optimize the organizations actions accordingly. We motivated three architectural system views, the functional system view, the dynamic system view, and the system structure view and assigned the concepts of the anomalies failure, fault, and defect of IEEE 1044-2009 standard [IEE09] to the system views. Furthermore, we extended the concept of defect of IEEE 1044-2009 [IEE09] in a way that it can be assigned to a set of components for which different options for defect correction can be identified. By focusing on the defect correction options and selecting the most appropriate one, our concept can serve as valuable means to foster a transparent organizational culture of learning.

Furthermore, our classification of system anomalies enables differentiation between the handling of faults to control the system and keeping the customer's or consumer's trust and respectively, the handling of defects and its options for sustainable defect correction. In addition, we extended the view from anomalies of the system element software to anomalies on the system level, where also organizational anomalies can be detected and handled.

We investigated the quality in use of our classification of system anomalies by conducting an action research project, applying the concepts to a postmortem analysis at the Bosch Group. By observing the application and performing semi-structured interviews with the stakeholders of the post-

mortem analysis, we evaluated the artifact's effectiveness, efficiency, and satisfaction.

Regarding the effectiveness, the interviews confirmed that our classification enables precise communication about anomalies and supports collaboration across different organizational units. It supports individuals from different system context to act as a collective and handle system and situation complexity. On the one hand, the artifact's application showed that by the differentiation between the concepts of failure and fault, we were able to identify missed fault reactions for controlling the system to keep the customer's and consumer's trust. This is the basis to improve fault tolerance and capability for situation management in the future. On the other hand, the artifact supports sustainable defect correction. By differentiating between the concepts of fault and defect, actions for tactical fault fixing and actions for sustainable and strategic defect correction can be separated and individually optimized. In addition, the results of the postmortem analysis, including the classed anomalies, are available in a reusable scheme and allow the organization to decide whether and how defects have to be corrected.

The efficiency of applying our classification was limited at the time of the study. The stakeholders described the application across different organizational silos as challenging, which was also negatively influenced by the organizational frame. We anticipate that the effort and time of applying the classification will decrease with increasing application and experience. Although the efficiency was limited, the stakeholders appreciated the constructive approach to foster an transparent learning culture.

In summary, the application of the classification of system anomalies supports iterative and inductive learning to improve the quality of the system, service, and organization. Our concepts of anomalies further form the basis for the operationalization of quality from different system views by assessing the quality through anomaly detection.

To operationalize the quality of the value provision of complex IoT service systems from the views of customer and consumer value, we introduced the **concept of IoT transaction (C3)**. An IoT transaction represents a workflow

from a business perspective with discrete observable properties, that allows indicating the delivered customer and consumer value. With the concept of an IoT transaction, we focus on observing business-relevant component behaviors to reduce observation and control complexity. We proposed to apply the observation method of distributed tracing to indicate customer value by transaction completion and consumer value by IoT transaction dependability. To indicate customer value, we apply a dual logic to the value provision. For the indication of consumer value, we further differentiate the quality characteristics of value provision in time. This contribution was the basis for observing and controlling the value provision to a consumer, which we investigated with contribution C4 to this dissertation.

With contribution **Evaluate and Control Service and Transaction Dependability of Complex IoT Systems (C4)**, we applied the concept of IoT transaction to evaluate the value provision from a consumer view of value by indicating IoT transaction dependability. In an action research project at the Bosch Group, we designed and evaluated the transaction-based approach for dependability evaluation. We guided and observed the application of the approach by employees of the Bosch Group. Besides, we conducted a focus group to assess its quality in use, focusing on the effectiveness, efficiency, and satisfaction. In the first stage, we designed the artifact based on industry requirements. We derived the artifact from the process model of ISO/IEC 25040 [ISO11b] as part of the SQuaRE series [ISO11d] and extended its scope to the continuous observation context of operations. Therefore, we integrated the IoT transaction concept to project its quality characteristics onto the quality models of the SQuaRE series [ISO11d]. This allows an organization to implement control loops with SLIs and SLOs, which can be quantified via distributed tracing. In the second stage, we applied the artifact at the Bosch Group. We observed the artifact's instantiation and evaluated its effectiveness, efficiency, and satisfaction through a focus group discussion by reviewing the application in detail with two of the system stakeholders.

The study revealed that our transaction-based approach for dependability

evaluation can be effectively applied. The approach facilitates the different system stakeholders to commonly understand consumer concerns and support them in prioritizing and operationalizing them.

The concept of IoT service and its decomposition into IoT transactions makes the derivation of quality requirements from consumer view of value tangible to the different system stakeholders. Instead of generally defining dependability requirements on system and component level, our approach determines them in a context of an IoT transaction and the required component effects providing value to a consumer. The artifact's application indicates to improve collaboration and communication between different stakeholders and teams, as it aligns the perspective from which they develop and operate their components. Besides, the approach fosters the importance of continuous feedback loops through observation, evaluation, and control of the transaction dependability with SLIs and SLOs. Therefore, it allows us to detect anomalies in the dynamic context of use and react to them. Changes in the quality of the value provision can be observed when the system is modified. Besides, it can be applied to gradually improve the quality of the value provision, by adapting it to the consumer needs. Consequently, the approach increased awareness of the validation of the operationalization and iterative exploration of the consumer context of use. Furthermore, the participants noted that the systematic approach, with its shared concepts, increased communication efficiency in collaborating with different system stakeholders and teams.

In summary, this thesis supports organizations in quantifying the delivered value of an IoT service and controlling the value provision with effective actions. Furthermore, the trust of a customer and a consumer in the IoT service provided by an IoT system and in the organization can be kept and further increased by applying feedback loops for quality improvement of the system, the IoT service, and the organization.

8.2 Limitations and Future Work

In this section, we discuss the limitations of our contributions and present suggestions for future work to address them or extend the scope of our contributions.

For the evaluation of the **Classification of System Anomalies (C2)**, we performed action research in collaboration with industry. We demonstrated the quality in use of our classification. However, we see the limitation that our results are not generalizable with the sample of one action research project. Therefore, we are currently discussing the concept with different interest groups at the Bosch Group, including expert groups for monitoring and safety and security concerns. Besides, we are further applying and evaluating the classification in the context of different IoT solutions from different business units. However, we see the risk that the application and the results are specific to the organization's context of the Bosch Group. Therefore, we suggest applying our classification to other action research projects and case studies at different organizations to further evaluate and refine it.

We see high potential in the assignment of domain-related defects to differentiate safety, security, and privacy defects. Future research can be related to the implications of the additional status graced in the defect lifecycle. This may include how stakeholders of different domains with different concerns can analyze, communicate, and document trade-offs of gracing individual domain defects. Furthermore, if, e.g., a security defect is graced on behalf of a safety requirement, tactical options and fault reaction in operations are needed to handle the potential effects of the security defect. Therefore, future research regarding inter-team communication, collaboration, and handling of (safety, security, and privacy) defects is needed.

In addition, as the information management and communication about anomalies and the handling of them in organizations is a highly complex and social activity, we see a high potential for further investigation from an organizational information theory perspective (see the work of Weick et al. [WA01; Wei93; Wei95]). In particular, we see the need to investigate

how to support people in the dynamic execution of OODA loops where the organization must react despite complete information transparency. We think, an important condition is the implementation of a blameless culture, based on a concept of psychological safety [Edm99], in which individuals feel safe to take action without the fear of being blamed in retrospect.

Concerning the evaluation of the **Concept of IoT Transaction (C3)**, we assessed the IoT transaction dependability in the action research project of the contribution **(C4) Evaluate and Control Service and Transaction Dependability of Complex IoT Systems**. At this time, we did not apply and evaluate the transaction completion from a customer view regarding value. Nevertheless, we see a high potential to apply distributed tracing and the indication of transaction completion for the billing process. The idea is to enable transaction-based revenue streams, at which the customer does not pay for uptime but for a transaction-based discrete value provision. With direct transaction-based revenue streams, it could be possible to optimize the value provision to a customer and the revenue model accordingly. Furthermore, we see a high potential to optimize the cost structure of the IoT system operations by introducing a transaction-based unit-cost model. With a unit-cost model, the revenue stream could be opposed to the resources consumed from the individual components, enabling to optimize the operations and revenue model. Generally, we see future research directions in aligning our IoT transaction concept with business model development and evaluation approaches. The integration of the concept of IoT transaction into approaches like the Business Model Canvas [OPOF11] or the Business Model Navigator [GFC13] has the potential to further link the economic perspective of a business model with the technical development and operations of IoT service systems.

In our action research study **Evaluate and Control Service and Transaction Dependability of Complex IoT Systems (C4)** the IoT system was not yet in production, therefore we simulated consumer behavior with a load test. Further studies are needed to investigate the approach in the context

of real-world consumer loads. Similar to the limitations of the study of the Classification of System Anomalies (C2), we evaluated the quality in use of the artifact in one action research study. Due to the limited sample size, we propose to conduct further studies to evaluate the artifact and further refine it. We are currently discussing and implementing the transaction-based approach for dependability evaluation in the context of other IoT solutions at the Bosch Group and are performing further feedback loops for the quality in use of our IoT transaction concept, especially for the dependability evaluation. Moreover, we plan further investigation and evaluation of the suitability of the suggested dependability measures in Section 7.4.2. Especially regarding the measure of transaction completeness, we see a high potential to work on machine learning approaches for complex types of transactions. For example, inductive labeling of anomalous span patterns to indicate faults could be beneficial when software practitioners do not have sufficient knowledge about the component interactions of a transaction.

In addition, future work needs to support the continuous exploration of customer and consumer needs regarding the value provision. This includes the continuous validation of the quality model, including the defined measures (SLI) and target values (SLO) with dynamic and inductive methods. We also see the potential for future work in the iterative development of SLOs from operations data and the inductive enrichment of requirement engineering.

A general research direction for future work could be the combination of deductive and inductive approaches and the appropriate transition between both working modes. Many organizations are still strongly focused on static and deductive approaches like checklists and process capability assessment, following a normative and dual perspective of verification. These approaches have a too high feedback latency and do not adapt fast enough to complex and dynamic systems and contexts of use. Therefore, we think further investigation of standard quality management approaches and practices are needed, to adapt to an increasingly complex and dynamic environment. Emerging approaches are needed that follow the principle of observation-based deci-

sion and action loops to respond to the dynamic context and allow action apart from a normative perspective of conformance or nonconformance. An example could be to examine the triage method for situation management, at which sometimes anomalies cannot be immediately classified as ok or not ok.

In summary, we believe that concurrently with implementing methods towards a perspective of dynamic value provision, incentives for product owners and developers need to be adjusted accordingly. Incentives should not only be based on performance measures for quantifying the number of functionalities developed and their development speed but need to be aligned with targets and measures to validate the value provided by the service from a consumer perspective, i.e. with service consumption measures and measures of service dependability.

To support alignment and engagements towards value provision goals and outcomes, impacting customer and consumer satisfaction, the method of OKRs, objective and key results, could be beneficial. OKR is an operational management system approach that is recently gaining high attention due to its successful implementation at leading tech companies like Google. It is an agile and iterative approach, which represents a combination of top-down and bottom-up practices for goal setting and definition of key results, which are measures, to track goal achievement [NL16]. Therefore, OKR allows reducing the latency of feedback loops of traditional strategical and operative target setting and performance measurement. We see potential in applying OKRs to set focused and ambitious service quality objectives and targets that impact organizational performance measures and enable short feedback cycles.

BIBLIOGRAPHY

- [ABC+16] T. M. Ahmed, C.-P. Bezemer, T.-H. Chen, A. E. Hassan, W. Shang. “Studying the effectiveness of application performance management (APM) tools for detecting performance regressions for web applications: an experience report.” In: *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*. IEEE. 2016, pp. 1–12 (cit. on p. 58).
- [ABDP13] G. Aceto, A. Botta, W. de Donato, A. Pescapè. “Cloud monitoring: A survey.” In: *Computer Networks* 57.9 (2013), pp. 2093–2115 (cit. on pp. 20, 57, 68).
- [ABE19] Y. Argotti, C. Baron, P. Esteban. “Quality quantification in Systems Engineering from the Qualimetry Eye.” In: *2019 IEEE International Systems Conference (SysCon)*. IEEE. 2019, pp. 1–8 (cit. on pp. 21, 48).
- [ALRL04] A. Avizienis, J.-C. Laprie, B. Randell, C. Landwehr. “Basic concepts and taxonomy of dependable and secure computing.” In: *IEEE transactions on dependable and secure computing* 1.1 (2004), pp. 11–33 (cit. on pp. 35, 49, 51, 55, 56, 71, 117, 130).
- [AP98] A. I. Antón, C. Potts. “The use of goals to surface requirements for evolving systems.” In: *Proceedings of the 20th international conference on Software engineering*. IEEE. 1998, pp. 157–166 (cit. on p. 73).
- [App20] AppDynamics. *Business Transaction*. 2020. URL: <https://www.appdynamics.com/product/how-it-works/business-transaction> (visited on 09/01/2020) (cit. on p. 141).

- [BA12] R. S. Bisel, E. N. Arterburn. “Making sense of organizational members’ silence: A sensemaking-resource model.” In: *Communication Research Reports* 29.3 (2012), pp. 217–226 (cit. on p. 88).
- [BB15] P. Bosch-Sijtsema, J. Bosch. “User involvement throughout the innovation process in high-tech industries.” In: *Journal of Product Innovation Management* 32.5 (2015), pp. 793–807 (cit. on p. 20).
- [BBHR16] A. Blohowiak, A. Basiri, L. Hochstein, C. Rosenthal. “A platform for automating chaos experiments.” In: *2016 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. IEEE. 2016, pp. 5–8 (cit. on pp. 21, 57, 109).
- [Bev95] N. Bevan. “Measuring usability as quality of use.” In: *Software Quality Journal* 4.2 (1995), pp. 115–130 (cit. on pp. 39, 40, 48).
- [BIMN03] P. Barham, R. Isaacs, R. Mortier, D. Narayanan. “Magpie: Online Modelling and Performance-aware Systems.” In: *HotOS*. 2003, pp. 85–90 (cit. on p. 58).
- [BJPM16] B. Beyer, C. Jones, J. Petoff, N. R. Murphy. *Site Reliability Engineering: How Google Runs Production Systems*. " O’Reilly Media, Inc.", 2016 (cit. on pp. 31, 50, 57, 60, 62, 64, 72, 73, 98, 150, 156, 178).
- [Blu19] R. Blumen. “Ben Sigelman on Distributed Tracing [Software Engineering Radio].” In: *IEEE Software* 36.01 (Jan. 2019), pp. 98–101 (cit. on pp. 32, 141).
- [BMR+18] B. Beyer, N. R. Murphy, D. K. Rensin, K. Kawahara, S. Thorne. *The site reliability workbook: Practical ways to implement SRE*. " O’Reilly Media, Inc.", 2018 (cit. on p. 61).
- [Bos12] J. Bosch. “Building products as innovation experiment systems.” In: *International Conference of Software Business*. Springer. 2012, pp. 27–39 (cit. on p. 20).
- [Boy96] J. R. Boyd. “The essence of winning and losing.” In: *Unpublished lecture notes* 12.23 (1996), pp. 123–125 (cit. on pp. 63, 116).

- [BSN18] R. Benabidallah, S. Sadou, M. A. Nacer. “Using system of systems’ states for identifying emergent misbehaviors.” In: *2018 IEEE 27th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*. IEEE. 2018, pp. 66–71 (cit. on p. 30).
- [BWZ15] L. Bass, I. Weber, L. Zhu. *DevOps: A software architect’s perspective*. Addison-Wesley Professional, 2015 (cit. on p. 31).
- [CEC90] L. A. Crosby, K. R. Evans, D. Cowles. “Relationship quality in services selling: an interpersonal influence perspective.” In: *Journal of marketing* 54.3 (1990), pp. 68–81 (cit. on p. 36).
- [Cha91] W. Chambers. *Chambers’s Encyclopedia: A Dictionary of Universal Knowledge*. Vol. 8. JB Lippincott & Company, 1891 (cit. on p. 36).
- [Che80] R. C. Cheung. “A user-oriented software reliability model.” In: *IEEE transactions on Software Engineering* 2 (1980), pp. 118–125 (cit. on p. 51).
- [CKF+02] M. Y. Chen, E. Kiciman, E. Fratkin, A. Fox, E. Brewer. “Pinpoint: Problem determination in large, dynamic internet services.” In: *Proceedings International Conference on Dependable Systems and Networks*. IEEE. 2002, pp. 595–604 (cit. on p. 58).
- [Clo18] G. Cloud. *USENIX - SLO Workshop*. 2018. URL: <https://www.google.com/search?client=firefox-b-e&q=usenix+slo+workshop> (visited on 10/01/2020) (cit. on p. 56).
- [Col94] I. Colville. “Searching for Karl Weick and reviewing the future.” In: *Organization* 1.1 (1994), pp. 218–224 (cit. on p. 96).
- [Coo98] R. I. Cook. “How complex systems fail.” In: *Cognitive Technologies Laboratory, University of Chicago. Chicago IL* (1998) (cit. on pp. 64, 118).
- [CS20] R. Charley, F. D. Silva. “Gartner Magic Quadrant for Application Performance Monitorings.” In: (2020) (cit. on p. 58).
- [Dan14] S. Danilo. *CanaryRelease*. 2014. URL: <https://martinfowler.com/bliki/CanaryRelease.html> (visited on 05/02/2020) (cit. on p. 57).

- [DD90] G. S. Day, G. S. Day. *Market driven strategy: Processes for creating value*. Free Press New York, 1990 (cit. on p. 52).
- [Dek12] S. Dekker. *Just culture: Balancing safety and accountability*. Ashgate Publishing, Ltd., 2012 (cit. on p. 64).
- [Dey01] A. K. Dey. “Understanding and using context.” In: *Personal and ubiquitous computing* 5.1 (2001), pp. 4–7 (cit. on p. 40).
- [DGG+08] R. De Lemos, F. Giandomenico, C. Gacek, H. Muccini, M. Vieira. *Architecting Dependable Systems V*. Vol. 5135. Springer, 2008 (cit. on p. 56).
- [DHWC08] S. Dekker, E. Hollnagel, D. Woods, R. Cook. “Resilience Engineering: New directions for measuring and maintaining safety in complex systems.” In: *Lund University School of Aviation* (2008) (cit. on pp. 56, 64).
- [DIN15] DIN. “Quality management systems - Fundamentals and vocabulary (ISO 9000:2015).” In: (2015) (cit. on pp. 39, 40, 110, 117, 118).
- [DMK04] R. Davison, M. G. Martinsons, N. Kock. “Principles of canonical action research.” In: *Information systems journal* 14.1 (2004), pp. 65–86 (cit. on p. 149).
- [Edm99] A. Edmondson. “Psychological safety and learning behavior in work teams.” In: *Administrative science quarterly* 44.2 (1999), pp. 350–383 (cit. on p. 187).
- [Esa13] K. Esaki. “System quality requirement and evaluation, Importance of application of the ISO/IEC 25000 series.” In: *Global Perspectives on Engineering Management* 2 (2013), pp. 52–59 (cit. on p. 45).
- [FEH+14] K. Fatema, V. C. Emeakaroha, P. D. Healy, J. P. Morrison, T. Lynn. “A survey of Cloud monitoring tools: Taxonomy, capabilities and objectives.” In: *Journal of Parallel and Distributed Computing* 74.10 (2014), pp. 2918–2933 (cit. on pp. 57, 68).
- [FFF14] F. Fotrousi, S. A. Fricker, M. Fiedler. “Quality requirements elicitation based on inquiry of quality-impact relationships.” In: *2014 IEEE 22nd International Requirements Engineering Conference (RE)*. IEEE, 2014, pp. 303–312 (cit. on p. 73).

- [FLD06] A. S. Frankel, M. W. Leonard, C. R. Denham. “Fair and just culture, team behavior, and leadership engagement: The tools to achieve high reliability.” In: *Health services research* 41.4p2 (2006), pp. 1690–1709 (cit. on p. 64).
- [FLR+14] C. Fehling, F. Leymann, R. Retter, W. Schupeck, P. Arbitter. *Cloud computing patterns: fundamentals to design, build, and manage cloud applications*. Springer, 2014 (cit. on p. 31).
- [For94] J. W. Forrester. “System dynamics, systems thinking, and soft OR.” In: *System dynamics review* 10.2-3 (1994), pp. 245–256 (cit. on p. 110).
- [Fou20] A. S. Foundation. *Interview Guide: Quality in use of Anomaly Classification*. 2020. URL: <https://jmeter.apache.org/index.html> (cit. on p. 166).
- [Gar84] D. A. Garvin. “What does “hltoduct quality” really mean.” In: *Sloan management review* 25 (1984) (cit. on p. 39).
- [GFC13] O. Gassmann, K. Frankenberger, M. Csik. “The St. Gallen business model navigator.” In: (2013) (cit. on p. 187).
- [GMDS18] M. Gupta, A. Mandal, G. Dasgupta, A. Serebrenik. “Runtime Monitoring in Continuous Deployment by Differencing Execution Behavior Model.” In: *Service-Oriented Computing*. Cham: Springer International Publishing, 2018, pp. 812–827 (cit. on p. 77).
- [Goo17] Google. *Google/SOASTA Research 2017*. 2017. URL: <https://www.thinkwithgoogle.com/data/mobile-page-speed-new-industry-benchmarks-load-time-vs-bounce/> (visited on 02/02/2020) (cit. on p. 50).
- [Gop93] M. Gopal. *Modern Control System Theory*. 2nd. New York, NY, USA: Halsted Press, 1993 (cit. on p. 57).
- [GR92] J. Gray, A. Reuter. *Transaction processing: concepts and techniques*. Elsevier, 1992 (cit. on p. 140).
- [HHK+17] R. Heinrich, A. van Hoorn, H. Knoche, F. Li, L. E. Lwakatara, C. Pahl, S. Schulte, J. Wettinger. “Performance engineering for microservices: research challenges and directions.” In: *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion*. 2017, pp. 223–226 (cit. on pp. 33, 56, 58).

- [HHMO17] C. Heger, A. van Hoorn, M. Mann, D. Okanovic. “Application Performance Management: State of the Art and Challenges for the Future.” In: *Proceedings of the 8th ACM/SPEC International Conference on Performance Engineering (ICPE '17)*. ACM, 2017 (cit. on pp. 20, 58, 69).
- [HMPR04] A. R. Hevner, S. T. March, J. Park, S. Ram. “Design science in information systems research.” In: *MIS quarterly* (2004), pp. 75–105 (cit. on p. 22).
- [HPK19] P. Haindl, R. Plösch, C. Körner. “An Extension of the QUAMOCO Quality Model to Specify and Evaluate Feature-Dependent Non-Functional Requirements.” In: *2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE. 2019, pp. 19–28 (cit. on pp. 74, 75).
- [HPK20] P. Haindl, R. Plösch, C. Körner. “Tailoring and Evaluating Non-Functional Interests Towards Task-Oriented Functional Requirements.” In: *Software Engineering (Workshops)*. 2020 (cit. on pp. 74, 75, 174–176).
- [Hud11] J. W. Hudson. “Risk Assessment and Management for Efficient Self-Adapting Self-Organizing Emergent Multi-Agent Systems.” MA thesis. Graduate Studies, 2011 (cit. on p. 30).
- [IEC15] IEC. *IEC Electropedia Dependability*. 2015. URL: <http://www.electropedia.org/iev/iev.nsf/display?openform&ievref=192-01-22> (visited on 02/02/2020) (cit. on pp. 38, 49–52, 164).
- [IEE06] IEEE. “Standard for Software Engineering – Software Life cycle Processes – Maintenance (IEEE Std 14764-2006).” In: (2006) (cit. on p. 114).
- [IEE09] IEEE. “IEEE Standard Classification for Software Anomalies (IEEE Std. 1044-2009).” In: (2009) (cit. on pp. 24, 52–54, 71, 72, 108, 110–114, 119, 136, 182).
- [IEE12] IEEE. *Systems and Software Engineering – Life Cycle Management – Guidelines for Process Description (IEEE Std 24774)*. 2012 (cit. on p. 36).

- [IEE90a] IEEE. *IEEE standard glossary of software engineering terminology*. 1990. URL: <https://ieeexplore.ieee.org/document/159342> (cit. on p. 57).
- [IEE90b] IEEE. *IEEE Standard Glossary of Software Engineering Terminology (IEEE Std 610.12-1990)*. 1990 (cit. on p. 51).
- [Ins08] P. M. Institute. “A Guide to the Project Management Body of Knowledge (PMBOK® Guide).” In: (2008) (cit. on p. 53).
- [ISO01] ISO/IEC. *Software engineering — Product quality — Part 1: Quality model (ISO/IEC 9126)*. 2001 (cit. on p. 41).
- [ISO05] ISO/IEC. *Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Guide to (SQuaRE) (ISO/IEC 25000)*. 2005 (cit. on p. 53).
- [ISO07a] ISO/IEC. “Systems and software engineering – Recommended practice for architectural description of software-intensive systems (ISO/IEC 42010).” In: (2007) (cit. on pp. 34, 112).
- [ISO07b] ISO/IEC/IEEE. *Systems and Software Engineering — Measurement Process (ISO/IEC/IEEE 15939)*. 2007 (cit. on p. 42).
- [ISO08a] ISO/IEC. *Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Data quality model (ISO/IEC 25012)*. 2008 (cit. on p. 43).
- [ISO08b] ISO/IEC/IEEE. *Systems and software engineering – System life cycle processes (ISO/IEC/IEEE 15288)*. 2008 (cit. on p. 44).
- [ISO11a] ISO/IEC. *Information technology – Service management – Part 1: Service management systems requirements (ISO/IEC 20000-1:2011)*. 2011 (cit. on p. 37).
- [ISO11b] ISO/IEC. *Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Evaluation process (ISO/IEC 25040)*. 2011 (cit. on pp. 25, 46, 76, 149, 184).
- [ISO11c] ISO/IEC. “Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Measurement of quality in use (ISO/IEC 25022).” In: (2011) (cit. on pp. 40, 42, 43, 111, 123, 161–164).

- [ISO11d] ISO/IEC. *Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models (ISO/IEC 25010)*. 2011 (cit. on pp. 20, 39–42, 45, 48, 49, 149, 156, 184).
- [ISO12] ISO/IEC. *Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Quality measure elements (ISO/IEC 25021)*. 2012 (cit. on pp. 43, 44).
- [ISO14] ISO/IEC. *Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Guide to (SQuaRE) (ISO/IEC 25000)*. 2014 (cit. on pp. 73, 150, 160, 178).
- [ISO15] ISO/IEC. *Information technology – Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Measurement of data quality (ISO/IEC 25024)*. 2015 (cit. on p. 44).
- [ISO16] ISO/IEC. *Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Measurement of system and software product quality (ISO/IEC 25023)*. 2016 (cit. on pp. 44, 49, 162–164).
- [ISO17a] ISO/IEC. *Information technology – Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Service quality models (ISO/IEC TS 25011)*. 2017 (cit. on pp. 37, 43).
- [ISO17b] ISO/IEC/IEEE. “Systems and software engineering - Vocabulary (ISO/IEC/IEEE 24765).” In: (2017) (cit. on pp. 35, 38, 53, 71).
- [ISO18a] ISO. “Road vehicles — Functional safety (ISO 26262-1:2018).” In: (2018) (cit. on pp. 55, 56, 113, 115–117).
- [ISO18b] ISO/IEC. *Information technology – Service management – Part 1: Service management systems requirements (ISO/IEC 20000-1:2018)*. 2018 (cit. on p. 35).
- [ISO18c] ISO/IEC/IEEE. “Systems and software engineering – Life cycle management Part1: Guidelines for life cycle management (ISO/IEC/IEEE 24748-1).” In: (2018) (cit. on p. 112).

- [ISO19a] ISO/IEC. *Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Common Industry Format (CIF) for Usability: User requirements specification (ISO/IEC 25065)*. 2019 (cit. on pp. 73, 74).
- [ISO19b] ISO/IEC. *Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Quality Requirements framework (ISO/IEC 25030)*. 2019 (cit. on pp. 46, 73, 156).
- [ISO98] ISO. “Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on usability (ISO 9241-11).” In: (1998) (cit. on p. 123).
- [Jae20] Jaeger. *Jaeger: open source, end-to-end distributed tracing*. 2020. URL: <https://www.jaegertracing.io/> (visited on 06/19/2020) (cit. on pp. 58, 166, 167).
- [JBL04] G. Jakobson, J. Buford, L. Lewis. “Towards an architecture for reasoning about complex event-based dynamic situations.” In: *Third International Workshop on Distributed Event-Based Systems*. Citeseer. 2004, p. 62 (cit. on p. 63).
- [JBL07] G. Jakobson, J. Buford, L. Lewis. “Situation management: Basic concepts and approaches.” In: *Information Fusion and Geographic Information Systems*. Springer, 2007, pp. 18–33 (cit. on p. 63).
- [JG03] M. D. Johnson, A. Gustafsson. *Competing in a service economy: how to create a competitive advantage through service development and innovation*. Vol. 37. John Wiley & Sons, 2003 (cit. on p. 49).
- [JKHC18] H. Johng, D. Kim, T. Hill, L. Chung. “Estimating the Performance of Cloud-Based Systems Using Benchmarking and Simulation in a Complementary Manner.” In: *Intl Conference on Service-Oriented Computing*. Springer. 2018, pp. 576–591 (cit. on p. 77).
- [JLM+05] G. Jakobson, L. Lewis, C. J. Matheus, M. M. Kokar, J. Buford. “Overview of situation management at SIMA 2005.” In: *MILCOM 2005-2005 IEEE Military Communications Conference*. IEEE. 2005, pp. 1630–1636 (cit. on p. 63).

- [JS16] S. L. Jones, P. P. Shah. “Diagnosing the locus of trust: A temporal perspective for trustor, trustee, and dyadic influences on perceived trustworthiness.” In: *Journal of Applied Psychology* 101.3 (2016), p. 392 (cit. on p. 36).
- [KA11] C. Kästner, S. Apel. “Feature-oriented software development.” In: *International Summer School on Generative and Transformational Techniques in Software Engineering*. Springer, 2011, pp. 346–382 (cit. on p. 75).
- [Kah11] D. Kahneman. *Thinking, fast and slow*. Penguin Books, 2011 (cit. on p. 64).
- [KBL08] J. Kontio, J. Bragge, L. Lehtola. “The focus group method as an empirical tool in software engineering.” In: *Guide to advanced empirical software engineering*. Springer, 2008, pp. 93–116 (cit. on p. 151).
- [KBS14] G. Kim, K. Behr, K. Spafford. *The phoenix project: A novel about IT, DevOps, and helping your business win*. IT Revolution, 2014 (cit. on p. 31).
- [KH19] H. Knoche, W. Hasselbring. “Drivers and Barriers for Microservice Adoption—A Survey among Professionals in Germany.” In: *Enterprise Modelling and Information Systems Architectures (EMISAJ)—International Journal of Conceptual Modeling: Vol. 14, Nr. 1* (2019) (cit. on p. 70).
- [Kot88] P. Kotler. *Marketing Management. Analysis, Planning, and Control*. New Jersey: Prentice-Hall: Englewood Cliffs, 1988 (cit. on p. 36).
- [KP96] B. Kitchenham, S. L. Pfleeger. “Software quality: the elusive target [special issues section].” In: *IEEE software* 13.1 (1996), pp. 12–21 (cit. on p. 45).
- [KPT18] J. Kietzmann, J. Paschen, E. Treen. “Artificial intelligence in advertising: How marketers can leverage artificial intelligence along the consumer journey.” In: *Journal of Advertising Research* 58.3 (2018), pp. 263–267 (cit. on p. 72).
- [Lai95] A. W. Lai. “Consumer values, product benefits and customer value: a consumption behavior approach.” In: *Advances in consumer research* 22 (1995), pp. 381–381 (cit. on pp. 37, 38).

- [Lam78] L. Lamport. “The implementation of reliable distributed multiprocess systems.” In: *Computer Networks (1976) 2.2* (1978), pp. 95–114 (cit. on p. 30).
- [Lam87] L. Lamport. *Distribution*. Email message sent to a DEC SRC bulletin board at 12:23:29 PDT on 28 May 87. May 1987. URL: <https://www.microsoft.com/en-us/research/publication/distribution/> (cit. on p. 30).
- [LCZ18] J. Lin, P. Chen, Z. Zheng. “Microscope: Pinpoint Performance Issues with Causal Graphs in Micro-service Environments.” In: *Service-Oriented Computing*. Cham: Springer International Publishing, 2018, pp. 3–20 (cit. on pp. 32, 77).
- [LK12] S. Lauesen, M. A. Kuhail. “Task descriptions versus use cases.” In: *Requirements engineering 17.1* (2012), pp. 3–18 (cit. on p. 73).
- [LROR07] S. M. Lee, D. Ribeiro, D. L. Olson, S. Roig. *The importance of the activities of service business in the economy: welcome to the Service Business. An International Journal*. 2007 (cit. on p. 34).
- [Luh14] N. Luhmann. “Vertrauen. Ein Mechanismus der Reduktion sozialer Komplexität. 5. Aufl. Konstanz, Stuttgart: UVK-Verl.” In: *Ges.[ua]* (2014) (cit. on pp. 36, 118).
- [LV16] K. N. Lemon, P. C. Verhoef. “Understanding customer experience throughout the customer journey.” In: *Journal of marketing 80.6* (2016), pp. 69–96 (cit. on pp. 49, 72).
- [LWRS12] S. Leminen, M. Westerlund, M. Rajahonka, R. Siuruainen. “Towards IOT ecosystems and business models.” In: *Internet of things, smart spaces, and next generation networking*. Springer, 2012, pp. 15–26 (cit. on pp. 19, 33).
- [Mac18] J. Mace. *A Universal Architecture for Cross-Cutting Tools in Distributed Systems*. dissertation. 2018 (cit. on p. 58).
- [May14] P. Mayring. *Qualitative content analysis: Theoretical foundation, basic procedures and software solution*. 2014 (cit. on pp. 85, 125, 152).

- [MDC15] J. J. Marquez, A. Downey, R. Clement. “Walking a mile in the user’s shoes: Customer journey mapping as a method to understanding the user experience.” In: *Internet Reference Services Quarterly* 20.3-4 (2015), pp. 135–150 (cit. on p. 72).
- [MF20] J. McCoy, N. Forsgren. *SLO Adoption and Usage in Site Reliability Engineering*. O’Reilly Media, Inc., 2020 (cit. on pp. 60–62, 72).
- [Mid07] M. Middelfart. “Improving business intelligence speed and quality through the OODA concept.” In: *Proceedings of the ACM tenth international workshop on Data warehousing and OLAP*. 2007, pp. 97–98 (cit. on p. 63).
- [MK09] S. Miettinen, M. Koivisto. *Designing services with innovative methods*. Savonia University of Applied Sciences, 2009 (cit. on p. 72).
- [MLLL+15] S. Madakam, V. Lake, V. Lake, V. Lake, et al. “Internet of Things (IoT): A literature review.” In: *Journal of Computer and Communications* 3.05 (2015), p. 164 (cit. on p. 33).
- [Mog06] J. C. Mogul. “Emergent (mis) behavior vs. complex software systems.” In: *ACM SIGOPS Operating Systems Review* 40.4 (2006), pp. 293–304 (cit. on p. 30).
- [MRF18] J. Mace, R. Roelke, R. Fonseca. “Pivot tracing: Dynamic causal monitoring for distributed systems.” In: *ACM Transactions on Computer Systems (TOCS)* 35.4 (2018), p. 11 (cit. on p. 58).
- [MS95] S. T. March, G. F. Smith. “Design and natural science research on information technology.” In: *Decision support systems* 15.4 (1995), pp. 251–266 (cit. on p. 22).
- [Net20] Netflix. *Zuul*. 2020. URL: <https://github.com/Netflix/zuul> (visited on 06/18/2020) (cit. on p. 158).
- [New15] S. Newman. *Building microservices: designing fine-grained systems*. " O’Reilly Media, Inc.", 2015 (cit. on p. 31).
- [NGSR16] M. Natu, R. K. Ghosh, R. K. Shyamsundar, R. Ranjan. “Holistic Performance Monitoring of Hybrid Clouds: Complexities and Future Directions.” In: *IEEE Cloud Computing* 3.1 (2016), pp. 72–81 (cit. on pp. 20, 21, 69, 108).

- [NHW20] S. Niedermaier, S. Heisse, S. Wagner. “Correct and Control Complex IoT Systems: Evaluation of a Classification for System Anomalies.” In: *Proceedings of the IEEE 20th Conference on Software Quality, Reliability and Security (QRS 2020)*. IEEE. 2020, pp. 321–328 (cit. on p. 108).
- [NKFW19a] S. Niedermaier, F. Koetter, A. Freymann, S. Wagner. *Interview Guide-line ‘On Observability and Monitoring of Distributed Systems’*. July 2019. URL: <https://doi.org/10.5281/zenodo.3346579> (cit. on pp. 84, 125).
- [NKFW19b] S. Niedermaier, F. Koetter, A. Freymann, S. Wagner. “On Observability and Monitoring of Distributed Systems—An Industry Interview Study.” In: *Service-Oriented Computing, 17th International Conference, ICSOC 2019, Toulouse, France, October 28-31, 2019 Proceedings*. Springer. 2019, pp. 36–52 (cit. on pp. 80, 109).
- [NL16] P. R. Niven, B. Lamorte. *Objectives and key results: Driving focus, alignment, and engagement with OKRs*. John Wiley & Sons, 2016 (cit. on p. 189).
- [NZW20] S. Niedermaier, T. Zelenik, S. Wagner. *Interview Guide: Quality in use of dependability evaluation approach*. 2020. URL: <http://doi.org/10.5281/zenodo.3826099> (cit. on p. 152).
- [OB15] H. H. Olsson, J. Bosch. “Towards continuous customer validation: A conceptual model for combining qualitative customer feedback with quantitative customer observation.” In: *International Conference of Software Business*. Springer. 2015, pp. 154–166 (cit. on pp. 20, 21, 68, 76).
- [Ohn88] T. Ohno. *Toyota production system: beyond large-scale production*. crc Press, 1988 (cit. on p. 109).
- [Ope20] OpenTracing. *The Open Tracing Semantic Specification*. 2020. URL: <https://opentracing.io/specification/> (visited on 06/18/2020) (cit. on p. 59).
- [OPOF11] A. Osterwalder, Y. Pigneur, M. A.-Y. Oliveira, J. J. P. Ferreira. “Business Model Generation: A handbook for visionaries, game changers and challengers.” In: *African journal of business management* 5.7 (2011), pp. 22–30 (cit. on p. 187).

- [PPM+18] R. Picoreti, A. Pereira do Carmo, F. Mendonça de Queiroz, A. Salles Garcia, R. Frizera Vassallo, D. Simeonidou. “Multilevel Observability in Cloud Orchestration.” In: *2018 IEEE 16th Intl Conf DASC/PiCom/DataCom/CyberSciTech*. 2018, pp. 776–784 (cit. on p. 57).
- [Rel20] N. Relic. *Distributed Tracing*. 2020. URL: <https://docs.newrelic.com/docs/understand-dependencies/distributed-tracing/get-started/how-new-relic-distributed-tracing-works> (visited on 01/02/2020) (cit. on p. 166).
- [RES20] P. Runeson, E. Engström, M.-A. Storey. “The Design Science Paradigm as a Frame for Empirical Software Engineering.” In: *Contemporary Empirical Methods in Software Engineering*. Ed. by M. Felderer, G. H. Travassos. Cham: Springer International Publishing, 2020, pp. 127–147. URL: https://doi.org/10.1007/978-3-030-32489-6_5 (cit. on pp. 22, 24).
- [RH08] P. Runeson, M. Höst. “Guidelines for conducting and reporting case study research in software engineering.” In: *Empirical Software Engineering* 14.2 (2008), p. 131 (cit. on p. 80).
- [RH09] P. Runeson, M. Höst. “Guidelines for conducting and reporting case study research in software engineering.” In: *Empirical software engineering* 14.2 (2009), p. 131 (cit. on pp. 81, 122, 124).
- [RKW+06] P. Reynolds, C. E. Killian, J. L. Wiener, J. C. Mogul, M. A. Shah, A. Vahdat. “Pip: Detecting the Unexpected in Distributed Systems.” In: *NSDI’06: 3rd Symposium on Networked Systems Design Implementation*. Vol. 6. 2006, pp. 9–9 (cit. on p. 58).
- [ROS98] J. Richardson, T. C. Ormerod, A. Shepherd. “The role of task analysis in capturing requirements for interface design.” In: *Interacting with computers* 9.4 (1998), pp. 367–384 (cit. on p. 73).
- [RPD06] F. Rosenberg, C. Platzer, S. Dustdar. “Bootstrapping performance and dependability attributes of web services.” In: *2006 IEEE International Conference on Web Services (ICWS’06)*. IEEE. 2006, pp. 205–212 (cit. on p. 50).
- [RS05] C. Rolland, C. Salinesi. “Modeling goals and reasoning with them.” In: *Engineering and managing software requirements*. Springer, 2005, pp. 189–217 (cit. on p. 73).

- [Sar10] R. G. Sargent. “Verification and validation of simulation models.” In: *Proceedings of the 2010 winter simulation conference*. IEEE, 2010, pp. 166–183 (cit. on p. 169).
- [SBB+10] B. H. Sigelman, L. A. Barroso, M. Burrows, P. Stephenson, M. Plakal, D. Beaver, S. Jaspan, C. Shanbhag. “Dapper, a large-scale distributed systems tracing infrastructure.” In: (2010) (cit. on pp. 58–60, 141, 166).
- [Sen06] P. M. Senge. *The fifth discipline: The art and practice of the learning organization*. Broadway Business, 2006 (cit. on p. 110).
- [Sim07] J. A. Simpson. “Foundations of interpersonal trust.” In: *Social psychology: Handbook of basic principles 2* (2007), pp. 587–607 (cit. on p. 36).
- [SML18] N. Sfondrini, G. Motta, A. Longo. “Public Cloud Adoption in Multinational Companies: A Survey.” In: *2018 IEEE International Conference on Services Computing (SCC)*. 2018, pp. 177–184 (cit. on p. 69).
- [Som04] I. Sommerville. *Software Engineering*. International computer science series. Pearson/Addison-Wesley, 2004 (cit. on p. 51).
- [SSAL11] M. Stickdorn, J. Schneider, K. Andrews, A. Lawrence. *This is service design thinking: Basics, tools, cases*. Vol. 1. Wiley Hoboken, NJ, 2011 (cit. on p. 72).
- [SSL08] J. Singer, S. E. Sim, T. C. Lethbridge. “Software engineering data collection for field studies.” In: *Guide to Advanced Empirical Software Engineering*. Springer, 2008, pp. 9–34 (cit. on p. 81).
- [SSM+16] R. R. Sambasivan, I. Shafer, J. Mace, B. H. Sigelman, R. Fonseca, G. R. Ganger. “Principled workflow-centric tracing of distributed systems.” In: *Proceedings of the Seventh ACM Symposium on Cloud Computing*. ACM, 2016, pp. 401–414 (cit. on p. 94).
- [Sta19] M. Staron. “Action Research in Software Engineering: Metrics’ Research Perspective (Invited Talk).” In: *SOFSEM 2019: Theory and Practice of Computer Science*. Ed. by B. Catania, R. Kráľovič, J. Nawrocki, G. Pighizzini. Cham: Springer International Publishing, 2019, pp. 39–49 (cit. on p. 24).
- [Sta95] R. E. Stake. *The art of case study research*. sage, 1995 (cit. on p. 124).

- [Swe20] M. Sweney. *Amazon Prime Video to slow streaming to fight broadband overload*. 2020. URL: https://www.theguardian.com/media/2020/mar/20/amazon-prime-video-to-slow-streaming-to-fight-broadband-overload?CMP=Share_iOSApp_Other (visited on 02/03/2020) (cit. on p. 56).
- [TMD20] D. A. Tamburri, M. Migliarina, E. Di Nitto. “Cloud applications monitoring: An industrial study.” In: *Information and Software Technology* 127 (2020), p. 106376 (cit. on pp. 49, 70).
- [TRA+17] J. Thalheim, A. Rodrigues, I. E. Akkus, P. Bhatotia, R. Chen, B. Viswanath, L. Jiao, C. Fetzer. “Sieve: actionable insights from monitored metrics in distributed systems.” In: *Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference*. 2017, pp. 14–27 (cit. on pp. 77, 78).
- [TV07] A. S. Tanenbaum, M. Van Steen. *Distributed systems: principles and paradigms*. Prentice-Hall, 2007 (cit. on p. 30).
- [Ulw02] A. W. Ulwick. “Turn customer input into innovation.” In: *Harvard business review* 80.1 (2002), pp. 91–98 (cit. on pp. 20, 40, 48).
- [Uni01] I. T. Union. “Global Information Infrastructure, Internet Protocol Aspects and Next-Generation Networks.” In: (2001) (cit. on p. 33).
- [Van01] A. Van Lamsweerde. “Goal-oriented requirements engineering: A guided tour.” In: *Proceedings fifth ieee international symposium on requirements engineering*. IEEE. 2001, pp. 249–262 (cit. on p. 73).
- [W3C20] W3C. *Trace Context*. 2020. URL: <https://www.w3.org/TR/2020/REC-trace-context-1-20200206/trace-id> (visited on 05/01/2020) (cit. on p. 173).
- [WA01] K. E. Weick, S. J. Ashford. “Learning in organizations.” In: *The new handbook of organizational communication: Advances in theory, research, and methods* 704 (2001), p. 731 (cit. on p. 186).
- [Wag08] S. Wagner. “Defect Classification and Defect Types Revisited.” In: *Proceedings of the 2008 Workshop on Defects in Large Software Systems*. DEFECTS ’08. Seattle, Washington: ACM, 2008, pp. 39–40. URL: <http://doi.acm.org/10.1145/1390817.1390829> (cit. on p. 52).

- [Wag13] S. Wagner. *Software product quality control*. Springer, 2013 (cit. on pp. 41, 45).
- [Wei93] K. E. Weick. “The collapse of sensemaking in organizations: The Mann Gulch disaster.” In: *Administrative science quarterly* (1993), pp. 628–652 (cit. on pp. 88, 90, 109, 186).
- [Wei95] K. E. Weick. *Sensemaking in organizations*. Vol. 3. Sage, 1995 (cit. on pp. 90, 91, 116, 186).
- [WGH+15a] S. Wagner, A. Goeb, L. Heinemann, M. Kläs, C. Lampasona, K. Lochmann, A. Mayr, R. Plösch, A. Seidl, J. Streit, et al. “Operationalised product quality models and assessment: The Quamoco approach.” In: *Information and Software Technology* 62 (2015), pp. 101–123 (cit. on p. 74).
- [WGH+15b] S. Wagner, A. Goeb, L. Heinemann, M. Kläs, C. Lampasona, K. Lochmann, A. Mayr, R. Plösch, A. Seidl, J. Streit, A. Trendowicz. “Operationalised Product Quality Models and Assessment.” In: *Inf. Softw. Technol.* 62.C (June 2015), pp. 101–123. URL: <https://doi.org/10.1016/j.infsof.2015.02.009> (cit. on p. 48).
- [Whi19] R. Whitmore. *Understand Distributed Tracing*. 2019. URL: <https://docs.lightstep.com/docs/understand-distributed-tracing> (visited on 05/19/2020) (cit. on p. 144).
- [Wie14] R. J. Wieringa. *Design science methodology for information systems and software engineering*. Springer, 2014 (cit. on p. 22).
- [WKH98] R. Wolak, S. Kalafatis, P. Harris. “An investigation into four characteristics of services.” In: *Journal of Empirical Generalisations in Marketing Science* 3.2 (1998) (cit. on p. 35).
- [YWGL18] Y. Yang, L. Wang, J. Gu, Y. Li. “Transparently Capturing Execution Path of Service/Job Request Processing.” In: *Service-Oriented Computing*. Cham: Springer International Publishing, 2018, pp. 879–887 (cit. on p. 77).
- [Zip20] Zipkin. *OpenZipkin: As distributed tracing system*. 2020. URL: <https://zipkin.io/> (visited on 05/10/2020) (cit. on p. 58).
- [ZZ17] R. Zager, J. Zager. “OODA loops in cyberspace: A new cyber-defense model.” In: *Journal Article October* 20.11 (2017), 33pm (cit. on p. 63).

LIST OF FIGURES

1.1	Research approach	23
2.1	Quality in use model (based on[ISO11d])	41
2.2	Product quality model (based on[ISO11d])	42
2.3	Relationship among quality characteristic, measure and target entity (adapted from [ISO12])	43
2.4	Conceptual approaches to quality (based on [ISO12])	44
2.5	Relationship among quality requirements and evaluation (adapted from [Esa13])	45
2.6	Software product quality evaluation process (adapted from [ISO11b])	46
2.7	Relationships of concepts of anomalies modeled as an entity relationship diagram (based on [IEE09])	54
2.8	Safety relevant time intervals (adapted from [ISO18a])	56
2.9	Causal relationship as DAG (adapted from [Ope20])	59
2.10	Temporal relationship as time bar (adapted from [Ope20])	59

4.1	Case Study Research Process (adapted from [RH09])	81
4.2	Challenges, Requirements, and Solutions. The arrows indicate grey fields for which the solutions are available.	102
5.1	Classification of System Anomalies as an entity relationship diagram (adapted from [IEE09])	114
5.2	Fault tolerant time interval (adapted from [ISO18a])	115
5.3	Defect life cycle (adapted from [IEE09])	119
6.1	Component effects in relation to a transaction of an IoT system	142
6.2	Relationship spans and trace (adapted from [Whi19])	144
7.1	Research Process	150
7.2	Activities of the Transaction-based Approach for Dependability Evaluation	155
7.3	Overview Remote Measurement System	159
7.4	T3: Provide access to measurement results	165
7.5	Exemplary trace including component effects abstracted from Jaeger [Jae20]	167
7.6	Number of transactions as a function of time	168
7.7	Transactions turnaround time t_k as a function of time for different percentiles	169

LIST OF TABLES

2.1 Relationships of concepts of anomalies of IEEE Std 1044 (based on [IEE09])	54
4.1 Overview of the research questions	81
4.2 Overview about participants and companies	83
5.1 Overview of the research questions – Classification of System Anomalies	122
5.2 Interviewee Information	125
7.1 Overview of the research questions	150
7.2 Research Approach of Stage 2	151
7.3 Effectiveness SLIs and SLOs for T3 (for an individual instance of T3)	162
7.4 Efficiency SLIs and SLOs for T3 (across several instances of T3)	163

LIST OF ABBREVIATIONS

ACID Atomicity, Consistency, Isolation, Durability.

AI Artificial Intelligence

AMQP Advanced Message Queuing Protocol

API Application Programming Interface

APM Application Performance Management

CAN Controller Area Network

CID Company ID

CPU Central Processing Unit

DAG Directed Acyclical Graph

EID Expert ID

I/O Input/Output

IoT Internet of Things

ITU International Telecommunication Union

NR New Relic

OODA Observe, Orient, Decide, Act

PO Product Owner

PQ Product Quality

QIU Quality in Use

QME Quality Measurement Element

QoS Quality of Service

REST Representational State Transfer

RMS Remote Measurement Service

RQ Research Question

SID Stakeholder ID

SLA Service Level Agreement

SRE Site Reliability Engineering

SLI Service Level Indicator

SLO Service Level Objective

SQuaRE System and Software Quality Requirements and Evaluation

UI User Interface

VM Virtual Machine

W3C World Wide Web Consortium

XCP Universal Measurement and Calibration Protocol