Software Quality Journal (2022) 30:337–366
https://doi.org/10.1007/s11219-021-09556-z

Evaluate and control service and transaction dependability of complex IoT systems

Sina Niedermaier[1] · Thommy Zelenik[2] · Stefan Heisse[3] · Stefan Wagner[1]

Accepted: 7 April 2021 / Published online: 3 May 2021

Abstract
Observing and controlling the dependability of service provision of complex IoT systems is challenging. In practice, many organizations struggle to derive consumer needs related to quality and to observe and quantify the service provision in the context of the dynamic behavior of a complex distributed system. In this paper, we present an approach to define and evaluate the dependability of complex IoT systems. Our approach is an adaptation of the ISO/IEC 25040, an international standard for the evaluation process for system and software quality, which is part of the systems and software quality requirements and evaluation (SQuaRE) series. Our approach was designed and evaluated with action research in an industrial study at Robert Bosch GmbH. Based on the framework of the SQuaRE series, we integrated different elements of site reliability engineering (SRE) and combined them with distributed tracing as a promising measurement method. Our approach introduces the IoT transaction concept to reduce modeling and observation efforts while increasing operationalization to measure performance against dependability targets. Our adaption was effectively applied, consumer-centricity along different system stakeholders were enhanced, and negative consequences of organizational silos were reduced. This has improved the dependability evaluation of service provision to enable fast feedback cycles for service performance control and improvement.

✉ Sina Niedermaier
sina.niedermaier@iste.uni-stuttgart.de

Thommy Zelenik
t.zelenik@live.de

Stefan Heisse
stefan.heisse@de.bosch.com

Stefan Wagner
stefan.wagner@iste.uni-stuttgart.de

[1] Institute of Software Engineering, University of Stuttgart, Stuttgart, Germany

[2] University of Stuttgart, Stuttgart, Germany

[3] Bosch Engineering GmbH, Abstatt, Germany

1 Introduction

Quality is not a property of a product or service. The quality of a product or service can only be assigned in relation to its stakeholder, the intended task, the context of use, and the value it is providing Bevan (1995); ISO/IEC. Systems and software engineering (2011d). Therefore, it is essential to understand the stakeholder needs and expectations to transform them into requirements and to set measurable targets for evaluating its quality. In practice, quality requirements derived from a quality model do not get enough attention compared to the planning, design, and evaluation of functionalities Wagner (2013). However, the quality of value provision, like availability and time behavior, is critical to a company's success that offers services. With the concept of dependability, the "ability to perform as and when required" IEC (2015), we are able to aggregate and express quality characteristics from time-dependent and, therefore, dynamic consumer perspective in operation.

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" ISO/IEC. Systems and software engineering (2019) 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 ISO/IEC. Systems and software engineering (2019). This problem is intensified for developing and operating distributed IoT service systems with complex interactions of system components, dynamic changes, and a short time to market. Development and deployment paradigms of microservices, DevOps, and cloud are creating maximum independence and specialization of development, resulting in isolated observability and monitoring methods, not allowing to control a service from a customer and consumer-centric view. Neglecting quality requirement elicitation and lack of effective and efficient control loops for controlling service quality can result in failures related to the value proposition to consumers. In consequence, service failures are usually noticed by the consumers only Niedermaier et al. (2019).

The interview study of Niedermaier et al. (2019) with 28 participants from 16 companies identified that distributed systems are continually changing and produce massive amounts of isolated monitoring data from individual components, often without contextual information to a consumer service. Therefore, developers and operators are often overwhelmed to create meaningful conclusions out of anomaly data from monitoring systems. A method with high potential to observe the dynamics in the service provision of complex systems is distributed tracing. It uncovers the trace of an IoT transaction within distributed systems by capturing data at runtime to infer causal relationships Sambasivan et al. (2016).

For the evaluation and control of the service dependability from a consumer-centric view, an approach is necessary to derive quality characteristics and appropriate indicators and enable a context-dependent dynamic observation and control. To be suitable for practical utilization, the approach needs to reduce observation complexity.

We propose a lightweight method explicitly designed for the dependability evaluation of complex IoT service systems from a consumer-centric view to address this need. To reduce observation and control complexity, we focus on component effects in relation to high business value of an IoT transaction. Via distributed tracing, we are able to observe and verify the value provision of an IoT transaction in a dynamic context of use. As a result, time for fault detection can be reduced, and implementing feedback cycles for service performance control and improvement is enabled.

The research artifact – the transaction-based dependability evaluation approach – was developed and evaluated by performing action research in two stages. Based on industry requirements, in the first research stage, we designed the artifact, which is derived from the process model of ISO/IEC 25040 ISO/IEC. Systems and software engineering (2011b) and integrated approaches of Google's discipline of Site Reliability Engineering (SRE) Beyer et al. (2016) as well as distributed tracing as an observation method. In the second research stage, we investigated the quality in use of the artifact by the practical application at Robert Bosch GmbH, a company providing IoT solutions. By observing the application of the artifact and conducting a focus group discussion with the software practitioners applying the research artifact, we evaluate its quality in use.

This paper is structured as follows. After the introduction, we present the related work and fundamental concepts in Section 2. In this section, we introduce approaches that are the basis of our designed research artifact, the – transaction-based dependability evaluation approach. Furthermore, we present a systematic derivation of our understanding of the concepts of service, transaction, and monitoring for this paper. Section 7 describes our research objective and the two-staged research process. The results of stage one, the design of our transaction-based dependability evaluation approach is presented in Section 4. Section 5 further describes the instantiation of the research artifact by applying it at the case company. We present the results of the application and the evaluation of the quality in use of the artifact from the perspective of software practitioners. We interpret the results of the study and derive practical implications in the discussion in Section 6. Finally, Section 7 describes the limitation of this study and the last Section 8 summarizes this paper and suggests future work.

2 Concepts and related work

This section introduces approaches for quality operationalization which are the basis of our designed research artifact, the – transaction-based dependability evaluation approach. In addition, we present a systematic derivation of the terminology of the concepts of service, transaction, and monitoring for this paper.

2.1 Approaches for quality operationalization

The following sections present the structure of the quality models of ISO/IEC 25010 and the quality evaluation process of ISO/IEC 25040, from which we derived our transaction-based dependability evaluation approach. Furthermore, we describe Google's Site Reliability Engineering concepts of SLO, SLI, which we integrated into our approach to create quality feedback cycles.

2.1.1 SQuaRE: ISO/IEC 25010 and 25040

In research and practice, various quality models exist. Most well known is the international standard ISO/IEC 25010 – system and software quality models ISO/IEC. Systems and software engineering (2011d). ISO/IEC 25010 ISO/IEC. Systems and software engineering (2011d) is part of the systems and software quality requirements and evaluation (SQuaRE) ISO/IEC 25000 series, a framework for the requirement specification and evaluation of system and software quality.

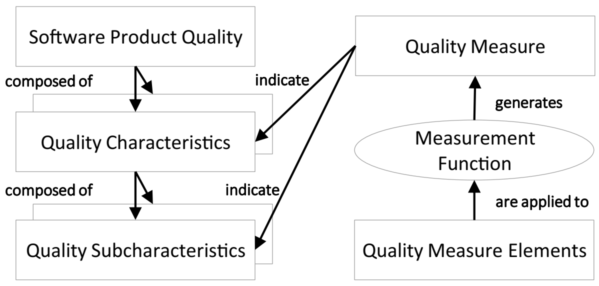


Fig. 1 Measurement of quality characteristics (based on ISO/IEC. Systems and software engineering (2011d))

ISO/IEC 25010 ISO/IEC. Systems and software engineering (2011d) includes a standard decomposition of quality characteristics and sub-characteristics, which can be indicated through quality measures. It contains two quality models: quality in use (QIU) and product quality (PQ), which can be used to specify requirements and derive measures and evaluation. The QIU model takes the consumer's perspective and focuses on the characteristics of their interaction with the product in a specific context of use. It comprises five quality characteristics: the effectiveness, efficiency, satisfaction, freedom from risk, and context coverage. The PQ model decomposes characteristics of the product into eight characteristics: functional suitability, performance efficiency, compatibility, usability, reliability, security, maintainability, and portability ISO/IEC. Systems and software engineering (2011d).

The concept of quality measurement of the SQuaRE series is illustrated in Fig. 1.

It presents the relationship among characteristics and sub-characteristics, which can be measured by and are indicated through quality measures that are quantifications of quality (sub-) characteristics. A measure is defined by a measurement function that is composed of quality measure elements. The measurement elements are generated through a measurement method applied to a target entity ISO/IEC. Systems and software engineering (2011d, 2011c). A target entity is a "fundamental thing of relevance to the user, about which information is kept, and need to be measured." A target entity can be a work product or behavior of a system, software, or stakeholders ISO/IEC. Systems and software engineering (2012).

The quality evaluation process is described as a sequence of activities to determine the extent to which an entity meets its specified requirements ISO/IEC. Systems and software engineering (2011b). ISO/IEC 25040 ISO/IEC. Systems and software engineering (2011b) provides a generic process model (see Fig. 2) to evaluate system or software products, including five activities, which are described in the following.

1. Establish the evaluation requirements In this activity, the evaluation purpose, according to the entity, needs to be established. Furthermore, quality requirements need to be specified by applying a quality model.

2. Specify the evaluation Based on the quality requirements, measures need to be selected, and decision criteria like numerical thresholds or targets need to be determined.

3. Design the evaluation After specifying requirements and measures, the evaluation needs to be designed considering budget, methods, tools, and resources ISO/IEC. Systems and software engineering (2011b).

4. Execute the evaluation In the execution, the quality measurement is performed and needs to be assessed against the numerical thresholds or target values.

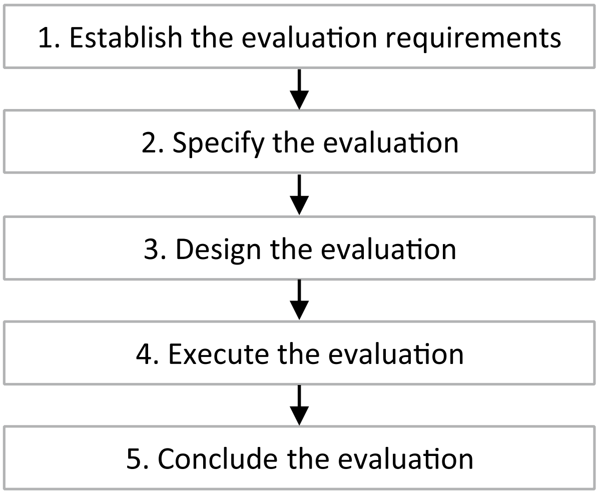

1. Establish the evaluation requirements
2. Specify the evaluation
3. Design the evaluation
4. Execute the evaluation
5. Conclude the evaluation

Fig. 2 Software product quality evaluation process defined by ISO/IEC 25040 ISO/IEC. Systems and software engineering (2011b)

5. Conclude the evaluation In the last activity, the evaluation is concluded. A review of the evaluation results and on the validity of the evaluation process needs to be performed to provide feedback to the organization.

While the SQuaRE series provides abstract guidance on how to decompose the quality of software and systems into characteristics and measure and evaluate them, the standards are criticized for being too generic Plöesch et al. (2015); Wagner et al. (2015). For the evaluation process of ISO/IEC 25040 ISO/IEC. Systems and software engineering (2011b) the same criticism concerning the abstract nature and lack of applicability is stated by Argotti et al. (2019). Therefore, our approach introduces the concept of the IoT transaction as a discrete target entity and applies the patterns of SQuaRE series while describing and evaluating its application and the operationalization of transaction quality characteristics.

2.1.2 Site reliability engineering (SRE)

SRE Beyer et al. (2016) is a discipline originated from Google, which applies aspects of software engineering to infrastructure and operations problems with a focus on consumer experience. It is an emerging framework, which many industries increasingly adopt to balance competing demands of functionality development and quality of a service Niedermaier et al. (2019); Beyer et al. (2016). An essential principle is to define which behavior matters for the value provision for a consumer and how to evaluate these behaviors. This is enabled by the concepts of service level indicators (SLIs) and service level objectives (SLOs). Product owners, developers, and operators select important quality characteristics that aim to represent consumer experience and identify SLIs to measure them. For the SLIs, SLOs with target values are defined and are verified against measurements running a control loop Beyer et al. (2016). IEEE Std. 15939 defines an indicator as a "measure that provides an estimate or evaluation of specified attributes derived from a model with respect to defined information needs" IEEE. Systems and software engineering (2008). In this paper, we apply the concepts of measure and indicator as synonyms.

- **SLI:** Defined quantitative measure to indicate the quality of the provided service
- **SLO:** Defined quality objective represented by a target value for the service quality which is to be achieved in a certain observation interval and verified against measurements at operation stage.

In SRE, specific SLIs, called the "golden signals of monitoring" are observed. These include traffic, latency, error rate, and saturation Beyer et al. (2016). In the context of this paper, we apply the more precisely defined quality characteristics and measures of the SQuaRE series ISO/IEC. Systems and software engineering (2011d) to formulate SLIs. It is possible to map our derived SLIs of this paper: throughput, turnaround time, and transaction completion ratio, described in Section 5.2, to the first three of the "golden signals" of SRE. We did not include saturation measures, as we are focusing on characteristics and related indicators from a consumer point of view and not from a provider view.

Within the SRE approach, SLIs and SLOs are defined from a customer journey perspective and measured at an observation point. In contrast, with our concept, we indicate the quality characteristics transaction-based rather than at one observation point and therefore are able to detect faults in the value provision and react to them.

2.2 Concepts of IoT service, IoT transaction, and IoT transaction monitoring

This section presents our terminology with relationships between the concepts of IoT service, IoT transaction, and IoT transaction monitoring which are part of our transaction-based dependability evaluation approach. There are different meanings for the concepts of service and transaction depending on the context in which they are applied. Thus, the purpose of the following sections is to facilitate a common understanding of the fundamental concepts of this paper to enable effective and efficient communication and collaboration among different organizational units working asynchronously and geographically separated. Following, we derive the concept of IoT service from common industry standards. Furthermore, we systematically deduce the concepts of IoT transaction and IoT transaction monitoring from our concept of IoT service and depict their relationships.

2.2.1 IoT service

Since different service concepts exist, which are ambiguous, we derive a more precise concept of an IoT service for this paper.

As stated by ISO/IEC/IEEE 24765:2017 systems and software engineering vocabulary ISO/IEC/IEE. Systems and software engineering (2017), 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." Further, it is stated that services are intangible. This service concept is symmetrical to the service concept provided by ISO/IEC 20000-1:2018 ISO/IEC. Information technology (2018): "means of delivering value for the customer by facilitating outcomes the customer wants to achieve." Instead of "result" ISO/IEC 20000-1:2018 ISO/IEC. Information technology (2018) applied the concept of "outcome".

We suggest applying the concept "process outcome" for our definition of the concept service, for which ISO/IEC/IEEE 24774 IEEE. Systems and software engineering (2012) provides the following definition. "An outcome is an 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."

The technical specification regarding service quality models ISO/IEC TS 25011:2017 ISO/IEC. Information technology (2017) refers to the concept of service by ISO/IEC 20000-1:2011 ISO/IEC. Information technology (2011a): "means of delivering value for the user by facilitating results the user wants to achieve" and changed "customer" to the concept "user". Whereupon ISO/IEC TS 25011:2017 ISO/IEC. Information technology (2017) differentiate customer and user by referring to the concept customer provided by ISO/IEC 20000-1:2011 ISO/IEC. Information technology (2011a), 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" ISO/IEC. Information technology (2017).

For the operationalization of service quality, we need to be more specific than ISO/IEC TS 25011:2017 ISO/IEC. Information technology (2017) by differentiating the concepts of customer and consumer in relation to the concept of service as follows:

IoT Service: performance of activities[1] for delivering

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

by facilitating

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

wants to achieve.

We differentiate the concepts of consumer and customer based on their different relationships on value and different feedback mechanisms for quality improvement.

A **customer** is an individual or organization that purchases products or service via financial business transactions. The customer value focuses on a **normative perspective** of the evaluation at the point in time of purchase Lai (1995). If we consider services and not one-time product purchases, this also includes performance-based billing such as per IoT transaction performed. The customer's feedback-mechanism is normative, following a dual logic based on terms and conditions of a contract, such as by retaining 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 highlights the **dynamic perspective** of the consumption in a specific consumption context Lai (1995). The consumer is interested in the **dependability**, "the ability to perform as and when required" IEC (2015) 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 the consumption of service in the context of use to indicate the value provided and detect anomalies that impact service dependability.

[1] We apply the concepts of activities and tasks from the ISO/IEC/IEEE 24774 IEEE. Systems and software engineering (2012).

2.2.2 IoT transaction

As a service is intangible and has a continuous nature, we introduce the concept of the IoT transaction which is a procedure 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 cannot be partially complete Gray and Reuter (1992). With our concept of the IoT transaction, we distinguish ourselves from the concept of database transaction and the ACID properties as our concept allows us to consider transactions with degraded quality and partial completeness (see Section 5.2, Table 4).

Sigelman, author of one of the early tracing frameworks – Dapper Sigelman et al. (2010), describes the concept of transaction as a "single, logical unit of work in its entirety" Blumen (2019).

For our concept of the IoT transaction, we are more restrictive than Sigelman Blumen (2019) and deduce it from the concept of IoT service, whereby an IoT service is decomposed into IoT transactions. As described in the IoT service concept, the customer has a dual perspective and evaluates a transaction as completed or not completed. In contrast, the consumer has a dynamic perspective on transaction dependability, which can be evaluated by a projection on a subjective quality model.

IoT Transaction: performance of activities of a workflow that starts and ends for delivering

a) a quantum of value to the **customer** or
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** and
b) to target dependability for the **consumer** via a quality model for the consumer.

An IoT system decomposes into a set of components that provide functionalities. A component-effect is the performance of a function in the dynamic system view related to a quantum of business value of an IoT transaction (see Fig. 3). To reduce observation and control complexity, we focus with the concept of component effects on business-relevant component behaviors, which, in case of faults, can propagate to a failure. Thereby, they 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.

2.2.3 Distributed tracing

Distributed tracing is a method to approximate a transaction path in a distributed system. Fundamental for this method is the concept of a span, a named, and timed operation representing a part of a workflow that needs to be processed to complete a transaction. Every span propagates span context, which enables the reconstruction of an instance of a

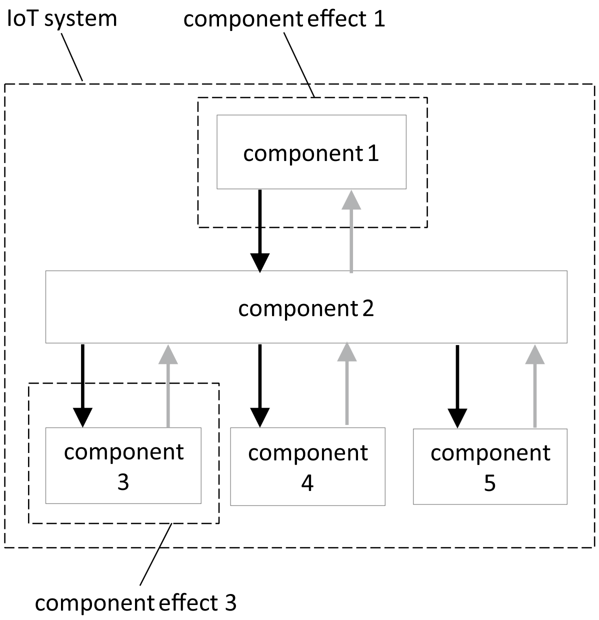


Fig. 3 Component effects in relation to a transaction of an IoT system

transaction, 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 Sigelman et al. (2010). A trace can be represented as causal relationship with a directed acyclical graph (DAG) (see Fig. 4) and as temporal relationship with a time bar (see Figs. 5 and 6).

The concept of distributed tracing became of growing interest with the Dapper paper Sigelman et al. (2010), which further motivated open-source implementations like Jaeger Jaeger (2020). Developers and operators use tracing to gain insight into distributed system behaviors, including troubleshooting and diagnosing distributed system behaviors.

2.2.4 IoT transaction monitoring

The IoT transaction concept comprising of component effects has observable and quantifiable properties that enable us to indicate transaction completion and indicate transaction dependability by IoT Transaction-Monitoring.

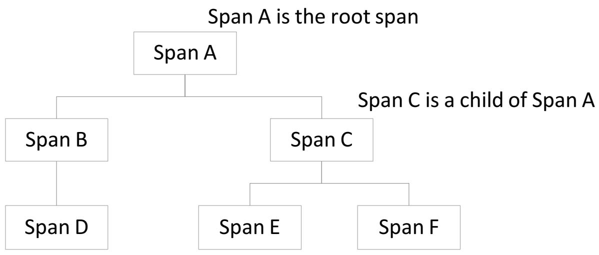


Fig. 4 Causal relationship – DAG adapted from OpenTracing (2020)

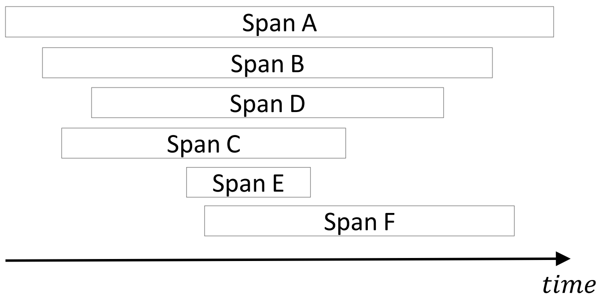


Fig. 5 Temporal relationship – time bar adapted from OpenTracing (2020)

IoT Transaction Monitoring: performance of activities of an IoT-System for delivering (a quantum of) information

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

by providing

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

According to the IoT transaction concept, a) corresponds to the customer view, applying dual logic to the value provision and b) the consumer view, differentiating quality characteristics 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 (Fig. 6). Therefore, it is possible to indicate IoT transaction completion and indicate transaction dependability via distributed tracing.

Distributed Tracing:
performance of activities for delivering (a quantum) of information

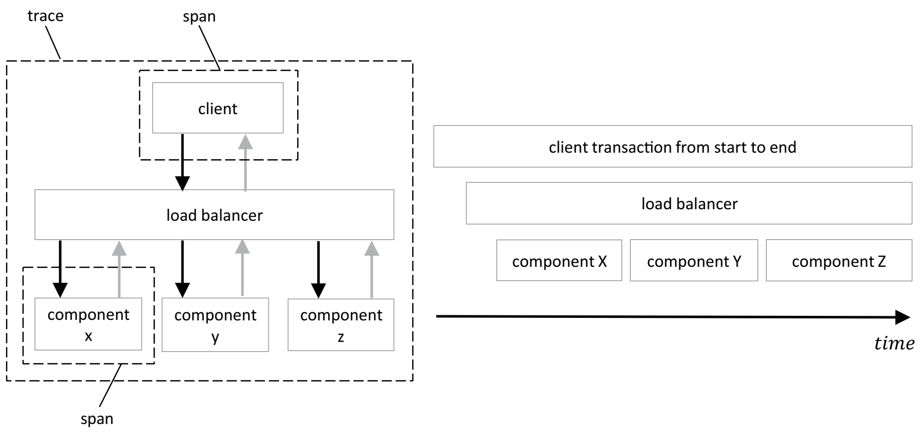


Fig. 6 Relationship spans and trace adapted from Whitmore (2019)

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

by providing

a) a method to observe spans
b) a method 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. IoT transaction monitoring enables us to indicate IoT transaction completion from a dual customer view, e.g., for billing and indicate IoT transaction dependability from a dynamic consumer view. As we focus in this paper on the consumer view and IoT transaction dependability, we do not further consider the customer view. Therefore, we take the IoT transaction concept and project the quality characteristics of a transaction onto the quality models of the SQuaRE series ISO/IEC. Systems and software engineering (2014). This enables us to implement control loops with SLIs and SLOs, which can be quantified in operation through distributed tracing. We integrated this idea by adapting the software quality evaluation process defined by ISO/IEC 25040 ISO/IEC. Systems and software engineering (2011b) for our research artifact accordingly. Furthermore, we evaluated the quality in use of our artifact from the perspective of software practitioners applying it within an action research project.

3 Research design

We followed the method of action research, which combines research and practice in a cyclical approach Reason and Bradbury (2001). While it focuses on improving industrial practices, it enables iteratively to develop solutions for complex problems Staron (2019). Action research is appropriate for the solution of our design problem, where we aim to improve the dependability evaluation with our research artifact – the transaction-based dependability evaluation approach – and simultaneously studying the experience of applying it in real-world industry use case Davison et al. (2004).

3.1 Research objective

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

Enable the evaluation of transaction dependability in dynamic operations for complex systems by designing an approach for dependability evaluation using distributed tracing as a measurement method. The approach should satisfy the needs and expectations of software practitioners towards its quality in use, with a focus on effectiveness, efficiency, and satisfaction.

The following research questions in Table 1 guided our study.

We designed an artifact – the transaction-based dependability evaluation approach – in a design science project to address the research questions. Furthermore, we evaluated its quality in use, focusing on effectiveness, efficiency, and satisfaction in an action research project at Robert Bosch GmbH.

Table 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 fulfil requirements of software practitioners?
RQ3	Can software practitioners understand and use the approach easily and are they satisfied with the approach? (efficiency, satisfaction)

3.2 Case context

We conducted the research project at Robert Bosch GmbH, a company providing IoT services in two stages (see Fig. 7). Objective of stage one is the design of the artifact – transaction-based dependability evaluation approach – which addresses RQ1. The evaluation of the artifact was conducted in stage two and focused on answering RQ2 and RQ3.

3.3 Stage 1

The challenges, requirements, and possible solutions of the interview study of Niedermaier et al. (2019) served as input for the requirement stage. We further analyzed existing approaches to define and evaluate quality characteristics. These included the ISO/IEC SQuaRE series ISO/IEC. Systems and software engineering (2014) and Google's SRE approach for defining and evaluating SLOs and SLIs Beyer et al. (2016), which was highlighted as a dominated industry practice in Niedermaier et al. (2019). In collaboration with seven industry experts coming from the case company and other companies providing services, we discussed the approaches and the requirements of our artifacts. Following Table 2 provides an overview of the roles of the experts and their experience in software development and operation.

After a revision, these results were used in the next stage to design the initial artifact (RQ1). Stage 1 lasted from February 2019 till May 2019.

3.4 Stage 2

In Stage two, the application and the evaluation of the artifact were conducted between June 2019 and March 2020 to answer RQ2 and RQ3 at an IoT service team at Robert Bosch GmbH. The evaluation of the research artifact was performed in the context of a service that was not yet released at the time of evaluation. Therefore, to create load on the system we emulated consumer behavior and produced a synthetic workload.

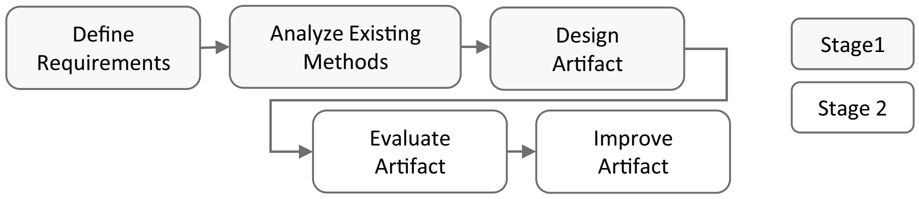


Fig. 7 Research Process

Table 2 Expert information for discussing design requirement of the research artifact

Expert role	Expert experience
Chief Technology Officer	26 years
Senior Technical Lead	14 years
Chief Architect Microservice Solution	7 years
Service Manager of IoT Solution	15 years
Software Quality Manager	17 years
DevOps Engineer	11 years
Product Owner	14 years

As summarized in Table 3, we performed an observation of the application of the approach and a focus group discussion to triangulate the observations with opinions of the participants.

Observation of Application The artifact was applied by an IoT service team of the case company to define and evaluate dependability requirements for transactions of an IoT service. Different team members, like the product owner, architect, service manager, and DevOps engineers, were involved in the different stages of applying the transaction-based dependability evaluation approach. The application was guided by one researcher and a second researcher accompanied the execution to validate the quality in use (with a focus on effectiveness, efficiency, and satisfaction) of the approach in practice.

Focus Group Discussion After completing the application of the approach, we performed a focus group that lasted 2,5 hours, where we reflected the artifact's application to reduce a threat to internal validity. The focus group setting enabled us to explore the experiences of the participants with the artifact throughout an interactive setting Kontio et al. (2008).

Participants were the system architect and a DevOps engineer who were actively involved in all of the activities performing the dependability evaluation approach. Both had experience in developing and operating service-based systems and knowledge in monitoring and distributed tracing. We reviewed the application of the approach by executing them exemplarily with the participants again. One researcher acted as a moderator who guided the participants throughout the approach. After each activity, the researcher asked for feedback with a focus on effectiveness, efficiency, and satisfaction as documented in the interview guide Niedermaier et al. (2020). The second researcher took the role of an independent observer and noted the participants' answers and reactions related to the artifact.

Table 3 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 retrospective
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

Afterward, the notes were analyzed using qualitative content analysis Mayring (2014) with a focus on the quality in use of the artifact. The expert opinions led to further improvements in the approach which are described in Section 5.3.

4 Stage 1: design of transaction-based dependability evaluation approach

In this section, we present the transaction-based dependability evaluation approach. We start outlining the requirements for the approach. Based on the results of the analysis of existing approaches provided in Section 2, we designed the artifact.

4.1 Define Requirements for the Design of the Artifact

The outcomes of the interview study of Niedermaier et al. (2019), our experiences developing IoT services and a literature review resulted in the following requirements and characteristics for the dependability evaluation approach.

- **Targeted at distributed Service Systems.** The approach is tailored for distributed systems providing consumer services. We focus on containerized systems that use RESTful HTTP or lightweight messaging (e.g., AMQP) and the integration of (Io) things.
- **Management of dependability from consumer-centric view.** The goal is to provide a concrete approach that enables to evaluate the abstract concept of consumer perceived dependability: "perform as and when required." Therefore, dependability requirements need to be defined by a cross-functional perspective where business, technical, and operation perspectives are aligned. Furthermore, the dependability is to be assessed in the dynamic context of use in operation. The approach aims to enable fault detection, which is the base to control the value provision with fault reactions. Moreover, the artifact intends to enable continuous product improvement by observing anomalies in the dynamic context of use.
- **Dynamic view and context propagation.** The approach needs to support the continuous observation of the dynamic context of use of consumer transactions; therefore, context propagation, along with components of an IoT system, providing component effects, is necessary. Applying distributed tracing enables to propagate context of consumer transactions and to derive quality and performance measurements. It further enables us to observe anomalies in the high frequency of system changes and detect faults in service operation to enable fast fault reaction.
- **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 complexity, our approach focuses on business-value component effects for evaluating the dynamic behavior of service provision in operation.
- **Relies on existing and proven approaches and measures: ISO/IEC 25040 and SRE.** Due to the criticism of the quality evaluation process of ISO/IEC 25040 ISO/IEC. Systems and software engineering (2011b) for being too abstract, we enhanced the generic

reference model with the concept of the IoT transaction as an entity of interest and distributed tracing as an observation and measurement method. To complement industry needs to focus on consumer-centricity Niedermaier et al. (2019), we integrated control loops with the concepts of SLI and SLO.

4.2 Transaction-based Dependability Evaluation Approach (RQ1)

This section presents the results for RQ1, the designed artifact of the transaction-based dependability evaluation approach, which includes the different activities and tasks to guide the transaction dependability evaluation (see Fig. 8).

1. Decompose IoT service into IoT transactions An IoT service is decomposed into types of IoT transactions delivering a quantum of value for the customer and consumer. Focus on IoT transactions with high business value, which in case of faults are propagating to significant quality deficiencies (failures) a consumer experiences and cares about.

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 for 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 25010 ISO/IEC. Systems and software engineering (2011d)). Therefore, suitable quality characteristics and sub-characteristics need to be derived from consumer needs with regard to the quantum of value to be delivered. The IoT transaction dependability can be specified 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 IoT transaction dependability specification has to be documented using the following items:

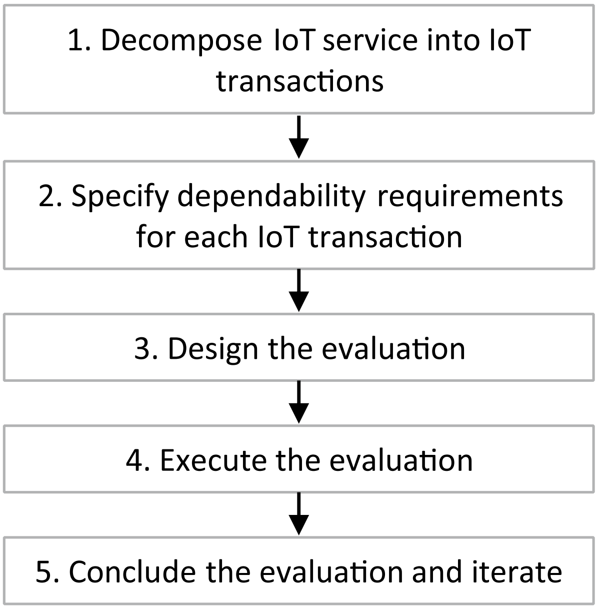

1. Decompose IoT service into IoT transactions
2. Specify dependability requirements for each IoT transaction
3. Design the evaluation
4. Execute the evaluation
5. Conclude the evaluation and iterate

Fig. 8 Activities of the transaction-based dependability evaluation approach

- transaction type with reference to consumer
- 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 should be reflected in the specification. Often there are trade-offs needed between different stakeholder groups. The following aspects, inspired by SRE practice Beyer et al. (2016) and ISO/IEC 25030 ISO/IEC. Systems and software engineering (2019) shall assist the specification of SLOs and SLIs.

- **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 execution flow: Identify and model components providing component effects to be included to indicate IoT transaction dependability. The component effects shall be described from the perspective of outcome, providing a quantum of value for the consumer. Model the high-level conceptual execution flow of the IoT transaction.

Instrument components: As we consider distributed tracing as the measurement method, instrument the components providing component effects, allowing them to propagate metadata to log the start and end time of a span and reconstruct the corresponding trace.

4. Execute the evaluation The measurements are taken with the initiation and processing of IoT transactions. Trace, span, and parent ID are propagated and span duration is 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.

5. Conclude the evaluation and iterate To conclude the evaluation, the results need to be reviewed, and the limitations of the evaluation procedure need to be discussed.

Furthermore, this activity is representative of the tasks of operation, where the evaluation is executed iteratively. Continuous observations to find deviation in the SLI measurements compared to SLOs are necessary 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. Through the increasing exploration of the context of use, the model of component effects and, therefore, the points of observation and control have to be validated iteratively. Moreover, the suitability of SLIs and SLOs need to be continuously evaluated and adapted to the changing environment in terms of requirements and system changes.

5 Stage 2: evaluation of the artifact

To evaluate the designed artifact, we applied it in the context of the IoT solution – remote measurement service – at the business unit of Connected Mobility Solutions of Robert Bosch GmbH. This section provides an abstract description of the IoT solution and its main components. Furthermore, we present the application of the transaction-based dependability evaluation approach in the context of the remote measurement service and describe the challenges which appeared in the different activities and tasks during execution. Finally, we delineate the focus group results to evaluate the effectiveness, efficiency, and satisfaction of the approach (RQ2 and RQ3).

5.1 System description - remote measurement

The system under scope provides over the air access to vehicle measurements, e.g., the engine temperature or the vehicle speed with different protocols. A consumer can define a measurement job, configuring the data to be extracted from the vehicles. The systems architecture, containing the essential components and communication flows, is abstracted in Fig. 9.

The component **Zuul** Netflix (2020) 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 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 Fig. 9) are developed by the team.

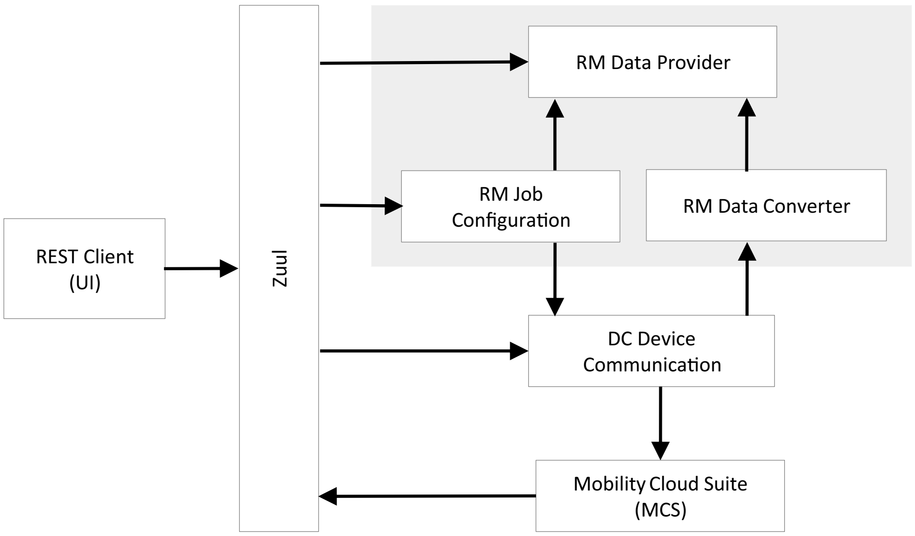


Fig. 9 Overview Remote Measurement system

5.2 Application of the transaction-based dependability evaluation approach

In the following, we present the application of our artifact, by exemplary describing the execution of the five activities of the dependability evaluation approach.

1. Decompose IoT service into IoT transactions The first task is to decompose the IoT service into IoT transactions providing quantifiable value to the customer. Consumers for this service are original equipment manufacturer (OEM) engineers, who want to gain access to vehicle lifecycle measurement data. The consumer need can be defined as follows.

Consumer need: An engineer (consumer) of an OEM (customer) needs to have access to required vehicle field data 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 result (T3)
- Download measurement result (T4)

For the scope of this work, T3 – provide access to measurement result – was used to apply the approach. As the service was still under development at the evaluation stage, IoT transaction three was the only one for which dependability measurements could be derived. The quantum of value for T3 corresponds to the access to the requested vehicle measurement data in the specified data format.

2. Specify dependability requirements for each IoT transaction To define dependability requirements for T3, characteristics of the transaction have been projected onto quality models with related measures of ISO/IEC 25000 series ISO/IEC. Systems and software engineering (2014). Essential quality (sub-) characteristics have been selected together with the chief product owner, two system architects, and the service manager. The interpretation of the concept of dependability "perform as and when required" IEC (2015) yielded to the following requirements, including the specification defining SLIs, their measurement function, and SLOs (Table 4 and 5). As the specific target values for the SLOs are confidential company data, they cannot be published in this paper. Therefore we have abstracted the SLO syntax.

Table 4 Effectiveness SLIs and SLOs for T3 (individual IoT transaction view)

SLI	Measurement function	SLO
Transaction completeness (C_1) (derived from objectives achieved measure)	$C_1 := \max(0, B) B = (1 - \sum_e P_e)$	target $\leq C_1$ per transaction
	(P_e) : Proportional value of each missing or incorrect component effect (e) (with a fault) of the transaction (max. value of each $(P_e = 1)$)	
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

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 **efficiency**, the ability to quickly process the measurement result and making it accessible, as well as the performance of how many transactions are successfully processed in an observation interval.

The **effectiveness** of an IoT transaction can be indicated by the objective achieved measure of the QIU model, which is: "The proportion of the objectives of the task that are achieved correctly without assistance" ISO/IEC. Systems and software engineering (2011c). As we cannot observe the sum of the objectives achieved, which corresponds to the quanta of value in this paper, we propose to indicate the effectiveness of T3 through its completion by observing and quantifying the provided component effects. Therefore, we adapt the objective achieved measure as transaction completeness with the following description.

– **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 form of the SLO are defined in Table 4. For each missing or incorrect component effect, representing a fault, a proportional value P_e can be assigned.

In the context of effectiveness indication, we also discussed the characteristic of functional appropriateness, which is a sub-characteristic of the functional suitability of the PQ model. The functional appropriateness is defined as the "degree to which the functions facilitate the accomplishment of specified tasks and objectives" ISO/IEC. Systems and software engineering (2011d). The functional appropriateness measure is the "[...] proportion of the functions required by the user [that] provides appropriate outcome to achieve a specific usage objective." The measurement function of functional appropriateness ISO/IEC. Systems and software engineering (2016) is similar to the measurement function of objectives achieved measure ISO/IEC. Systems and software engineering (2011c), but represents a special case in which all component effects are equally weighted. 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.

Table 5 Efficiency SLIs and SLOs for T3 (across several instances of T3)

SLI	Measurement function	SLO
Transaction throughput (Q_j)	$(Q_j = N_{c,j}/\Delta t_j)(N_{c,j})$: 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)	$(R_j = N_{c,j}/N_{i,j})(N_{c,j})$: Number of transactions completed during observation interval (j) $(N_{i,j})$: Total number of transactions initiated during observation interval (j)	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) ($\leq$) target (in, e.g., ms)

The target value (SLO) for the proportion of completeness introduces a class boundary to class instances of IoT transactions into complete and not complete. A proportional value of a missing and incorrect component effect can be indicated via anomalous span patterns of a component effect as part of a directed acyclic graph.

Identifying the component effects is described in activity 3 of the dependability evaluation approach, where the evaluation is designed. Furthermore, 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 time efficiency measure of ISO/IEC 25022 ISO/IEC. Systems and software engineering (2011c) and the mean throughput measure of ISO/IEC 25023 ISO/IEC. Systems and software engineering (2016).

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

The transaction throughput target 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.

To assess the quantity of completed transactions relative to the initiated transactions in an observation interval, we formulated a SLI of transaction completion ratio adapted from the task completed measure of ISO/IEC 25022 ISO/IEC. Systems and software engineering (2011c).

– **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 transaction turnaround time, which is adapted from the task time measure of ISO/IEC 25022 ISO/IEC. Systems and software engineering (2011c) and the mean turnaround time measure of ISO/IEC 25023 ISO/IEC. Systems and software engineering (2016).

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

The distribution of the transactions turnaround times can be indicated by the nth percentile turnaround time under expected load conditions, which we adapted from the time behavior measures of ISO/IEC 25023 ISO/IEC. Systems and software engineering (2016).

Challenges: When interpreting the abstract definition of dependability: “perform as, and when required” IEC (2015), the team had difficulties identifying essential characteristics. Initially, they struggled to determine which characteristics have a significant impact on the consumer perceived quality. DevOps engineers and PO tend to suggest product quality properties from a provider view. As the team was prone to take the provider view instead of the consumer view, we identified the importance of differentiating and highlighting the viewpoint and the purpose for which quality characteristics, related measures, and targets are formulated and linking the views appropriately. In this case of defining requirements and evaluating operation by emulating the behavior of one consumer, the performance measures from provider and consumer view are similar. Only the assignment of measures to the characteristics would change for this case (e.g., the transaction throughput can be assigned to indicate provider effectiveness). Besides, for a provider view, additional performance measures would be of interest, e.g., saturation.

Furthermore, the research team and participants discussed the interpretation of the quality characteristics of QIU and PQ characteristics and measures of ISO/IEC 25000 series ISO/IEC. Systems and software engineering (2014). Therefore, we compared suitable measures of QIU and PQ standards and adapted them to our formulation and measurement functions for the SLIs.

Moreover, we identified difficulties related to the terminology and 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 included a terminology page into the company internal glossary for the concepts of SLA, SLO, and SLI, which also handles information about the purpose and viewpoint from which they should be developed.

3. Design the evaluation Model System structure view and dynamic IoT transaction flow: We modeled the system structure view with the components that provide component effects to the IoT transaction and further added the (high-level) dynamic transaction flow in Fig. 10. Each of the component effects needs to be successful (without a fault) to consider the IoT transaction as complete.

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

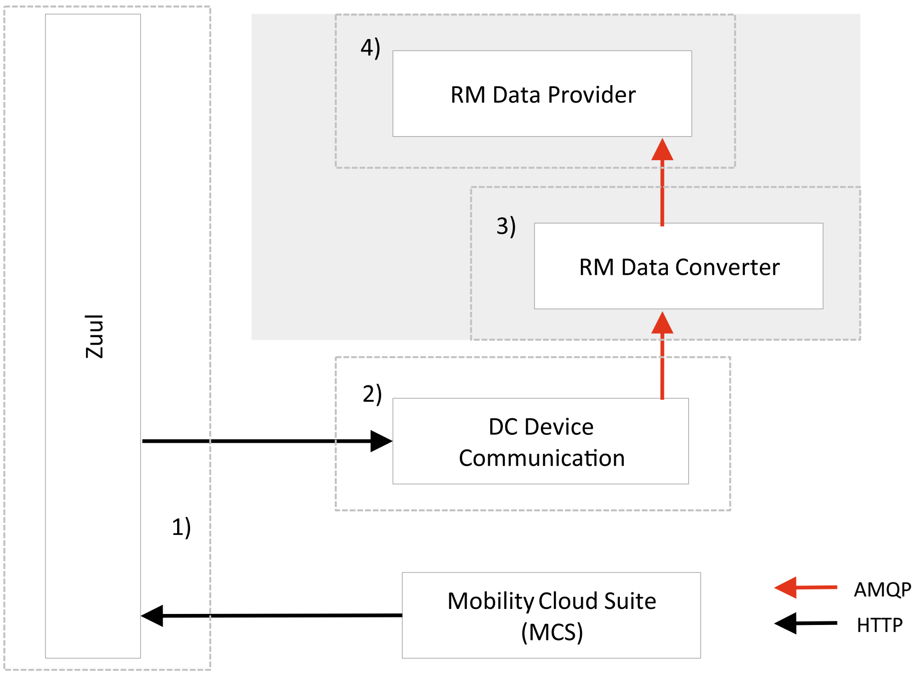


Fig. 10 T3: Provide access to measurement result

As the remote measurement service was not in production yet, and the MCS is an external service, we performed load tests by emulating measurement data. For the load test, we used JMeter Foundation (2020), which is an application to perform functional and performance load tests. The load test scenario included requirements provided by the customer: connection of ten vehicles and each of the ten vehicles sending a message every 120 seconds to the DC Device communication. We also needed to set up a MCS mock service, which emulated the external real MCS that forwards the processed vehicle data and passes it to the DC Device Communication.

Instrument components: For this action research project, two distributed tracing systems had been available for the product team. One system was Jaeger Jaeger (2020), which is an open-source distributed tracing tool and is inspired by Dapper Sigelman et al. (2010). As the Zuul, RM Data Provider, RM Data Converter, and the DC Device Communication have already implemented a Jaeger tracer. Therefore, no modifications to instrument components were required. Furthermore, we deployed New Relic (NR) NewRelic (2020), which is an application performance management tool and supports the analysis of distributed tracing data with its extensive UI.

4. Execute the evaluation As the Remote Measurement system was not yet in production, we performed a load test according to the customer specification.

Reconstruction of trace and measurement of SLIs: We used Jaeger and NR to reconstruct and display the traces. With NR, we could not reconstruct the whole trace due to the communication protocol of AMQP. The trace context was not attached to AMQP at the time of the evaluation, and therefore we could not gain full trace visibility. With Jaeger, we were able to produce the overall trace, but due to its limited aggregation capability, we imported the data to NR for SLI measurement and visualization. Since the measurements of runtime data generated in the research project are confidential, in the following, 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 related span patterns, as illustrated in Fig. 11. It presents an exemplary trace of an instance of a transaction type T3 that we abstracted from Jaeger. According to the completeness measure C_1, we assigned a proportional value P_e for each component effect, in case a fault is detected. As described in Activity 3 – Design the evaluation –, each component effect needs to be successful to consider a transaction as complete, therefore $P_1 = P_2 = P_3 = P_4 = 1$. In this example, no faults for the component effects of the trace of an instance of T3 are detected, consequently $C_1 = 1$. This represents the maximum value possible for the measure of transaction completeness. Thus this instance of T3 can be classed as complete.

For a more complex type of transaction, where software practitioners do not have sufficient knowledge about the operations and the component interactions, it would be possible

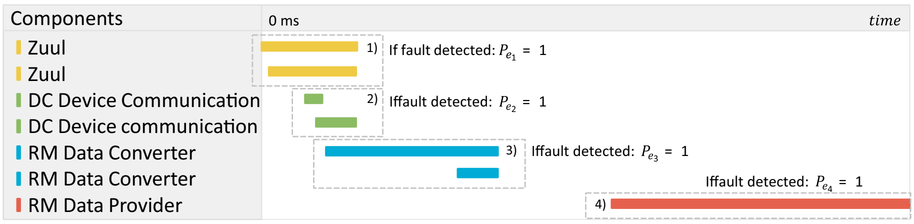

Components
0 ms
time
Zuul
Zuul
DC Device Communication
DC Device communication
RM Data Converter
RM Data Converter
RM Data Provider
1) If fault detected: $P_{e_1} = 1$
2) Iffault detected: $P_{e_2} = 1$
3) Iffault detected: $P_{e_3} = 1$
Iffault detected: $P_{e_4} = 1$
4)

Fig. 11 Exemplary trace including component effects abstracted from Jaeger (2020)

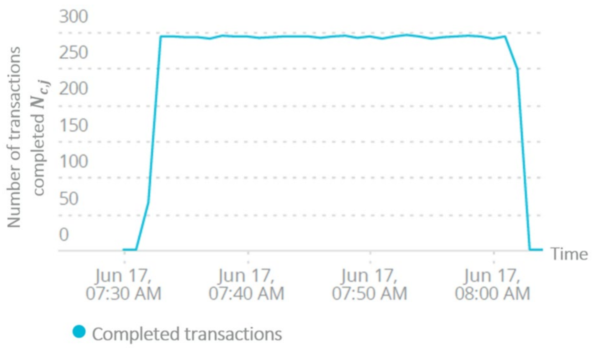


Fig. 12 Number of transactions as a function of time

to indicate a completed transaction inductively through specific trace patterns (by the number and the compilation of span patterns). This indication could be performed by labeling them in a machine learning context.

Transaction throughput: Figure 12 illustrates the SLI measurement for the transaction throughput, representing the number of the completed transactions plotted over a specific observation interval.

Transaction completion ratio: As we performed a load test and did not simulate faults, 100% of the transactions were completed in the observation interval.

Transactions turnaround time percentiles: Figure 13 visualizes the 99th, 98th, 95th, and 50th percentile transactions turnaround time t_k for a specific observation interval.

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 that cannot represent the consumer's operation real-world context. Therefore, we see a threat to the interpretation validity of the results for the dependability from a consumer's perspective. Nevertheless, the results were valuable in terms of the IoT system's performance capabilities, especially concerning the 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. Moreover, we detected a fault, a bottleneck between these two components (see Fig. 11). Within the relatively large time gap, the enqueued data is waiting for being consumed by the data provider. This time interval grows with increasing

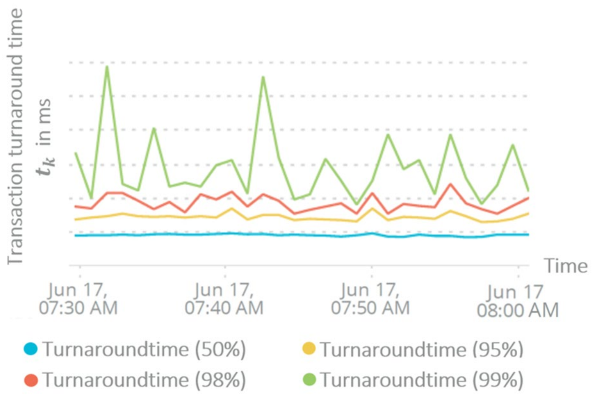


Fig. 13 Transactions turnaround time (t_k) in percentiles

load. Thus, we detected a defect in the conception of the underperforming data provider, which the team redesigned subsequently to decrease the bottleneck.

We need to 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 have to be further investigated. Moreover, we identified and discussed the following errors related to modeling and measurement:

- modeling errors, which arise when the conceptual models used provide a simplified or insufficient description of the phenomena in reality,
- observational errors, which arise through inaccuracies of the measurement instrument and
- errors related to distributed tracing as measurement instrument, impacting the system performance.

We need to investigate the implications of the discussed errors in consumer operation when consumer feedback can be obtained. This also includes the validation of the suitability of SLIs and SLOs. Furthermore, 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.

5.3 Focus group

This section provides the results of the focus group with two software practitioners who applied the approach. Following the interview guide's structure, we present expert opinions on the effectiveness, efficiency, satisfaction, and challenges of the approach by reviewing and discussing its application.

Effectiveness: The focus group discussion confirmed that the approach assists the operationalization of the concept dependability from a consumer perspective. The participants stated that the application of the approach promotes early discussion and alignment of different system stakeholder perspectives. The concept of transaction and its component effects are suitable abstractions to bring together business, technical, and operations domains and provide a common understanding of essential quality characteristics from a consumer perspective. Furthermore, the artifact increased a better understanding of how the IoT system needs to support the transaction's effective and efficient execution and how to operationalize the evaluation of the transaction dependability. The participants pointed out the application of the concepts SLO, SLI as they enable them to formulate measurable targets from a consumer's perspective. Moreover, they stated that they tend to focus on the functional perspective and the monitoring of individual components and neglected the dynamic transaction view without the approach. The synthetic probing, which the team applies, is also capturing a dynamic perspective. However, synthetic probing approximates the consumer experience, unlike distributed tracing, which captures real-world consumer transactions during operation. The participants further highlighted that the capability of the approach of getting an overview of the component dependencies helping to identify critical paths.

Efficiency and satisfaction: The experts outlined that their procedure would have been less structured without the approach and would have taken longer. The approach gives systematic guidance to focus on high-value transactions, characteristics a consumer cares about, and component effects that contribute to the value provision of a transaction.

Therefore, it is possible to reduce the observation space while at the same time getting a holistic overview of the dynamic processing of a transaction. Furthermore, the experts stated that it is easier to derive dynamic requirements transaction-based than on system level. 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, since the precise specification is primarily done for high-value transactions and not for every transaction possible. The incorporation of SLOs and SLIs created confusion initially, 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, the experts confirmed that a clear distinction and application can be ensured.

Challenges and improvements: The specification of transaction dependability requirements and interpretation of the abstract definition: "perform as and when required." was stated as challenging. The large number of possible projections onto quality models and measures of SQuaRE series was perceived as complicated by the participants. It was effective to adapt the approach to differentiate the consumer perspective from the provider perspective and start the projection of the concept of dependability onto the quality model of ISO/IEC 25010 ISO/IEC. Systems and software engineering (2011d) with the trivial quality characteristics of efficiency and effectiveness. Furthermore, the participants stated that the aggregation of traces was challenging. The traces are raw data and need to be processed sufficiently to quantify transaction properties and compare them against SLOs. As Jaeger was not sufficient for trace aggregation and NR could not trace the whole transaction due to the AMQP, the traces needed to be converted and imported to NR for performing SLI measurement. We prefer to apply the World Wide Web Consortium (W3C) trace context specification 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 W3C (2020). 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 discussions about the data storage and sampling are needed.

6 Discussion

Service design and evaluation should be performed from the perspective of customer or consumer needs and expectations. Based on these needs, quality requirements with quality characteristics operationalized through measures (SLIs) with appropriate targets (SLOs) are necessary. The approach's application demonstrated that the definition and measurement of SLIs and SLOs could be the base for continuous feedback cycles for service performance evaluation and control. The approach enables us to detect anomalies and to react to them. Furthermore, it can be applied to improve a service providing consumer value gradually.

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, it enables the detection of anomalies in the dynamic context of use, and changes in the quality of value provision can be observed when the system is modified. In addition, the team confirmed to get valuable insight into the system's performance that can be utilized to iterate the SLOs. Besides, we see a high potential for the indication of system performance anomalies 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 faults of component effect, regarding missing component effects to anomalies in the time behavior of providing component effects. Moreover, we can apply the completeness indicator to assess completed transactions for the customer billing process, as stated in the concept in Section 2.2.2.

Even though modern distributed tracing systems like Jaeger are designed for minimal performance overhead, they still create additional load. Therefore, the usage of such an approach with a critical production system needs to be carefully evaluated.

7 Threats to validity

Since we conducted a focus group to evaluate effectiveness, efficiency, and satisfaction with the artifact, we anticipate some personal bias in the participants' statements and answers and a threat to **internal validity**. Furthermore, participants may tend to comment and answer in confirmation of our approach. This could lead to confirmation bias. We consider this threat as reduced as we triangulated the answers with the observation of the application. Moreover, we see the possibility that the results of the focus group are the phenomenon of groupthink. We think this risk is low, as the participants seemed not worried about pointing out negative aspects of the artifact. We tried to control this risk by triangulating the observation results of the application of the dependability evaluation approach with the results of the focus group. As we did not audio-recorded the focus group, we see a risk of researcher bias. We tried to mitigate this risk as one researcher took the moderator's role, making some notes, and the other researcher acting as an independent observer who was dedicated to recording the reactions, statements, and answers. To increase interpretation validity, the minutes were counter checked by the researchers.

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. Furthermore, generalizability to other domains and further cases is not possible and needs to be evaluated.

Lastly, 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.

8 Conclusion and future work

We designed and evaluated an artifact to define and evaluate service and transaction dependability from a consumer-centric view in an action research project. The artifact design is based on industry approaches like SRE and traditional frameworks like the ISO/IEC 25000 series. We evaluated its quality in use with a focus on effectiveness, efficiency, and satisfaction by applying it to an IoT service of Robert Bosch GmbH. Our action research project demonstrated that the artifact enables consumer-centricity and transforms the complexity of observation and control of service provision into discrete observable and controllable 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 to create cross-functional alignment on consumer-centricity and have the potential to break down silos. It focuses on observing high-value contributing components and their behavior in the context of a transaction with an emphasis on characteristics that are of importance for consumers. Therefore, it fosters early discussion about important transaction characteristics from a consumer perspective while involving different system stakeholders like developers, operators, product owners, architects, and service managers. This enabled us to create a shared understanding of transaction and service dependability aspects and its related observability and control needs.

The participants experienced the quality in use of the artifact as positive. It was perceived as easy to understand. 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. The evaluation led to further improvements, where we adapted our concepts of service and transaction and differentiated these respectively from the view of a customer and a consumer. Furthermore, we described the relationship among a quality model, with quality characteristics, quality 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 measurements for the efficiency SLIs 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, if consistently applied, has the potential of full 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, and changes in the quality of value provision can be observed when the system is modified. Future work should focus on developing a concept for training and collecting further empirical data including consumer feedback to validate the suggested quality measures.

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.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

Argotti, Y., Baron, C., & Esteban, P. (2019). Quality quantification in systems engineering from the qualimetry eye. In *2019 IEEE International Systems Conference (SysCon)*, pages 1–8. IEEE.

Bevan, N. (1995). Measuring usability as quality of use. *Software Quality Journal, 4*(2), 115–130.

Beyer, B., Jones, C., Petoff, J., & Murphy, N. R. (2016). *Site Reliability Engineering: How Google Runs Production Systems*. "O'Reilly Media, Inc.".

Blumen, R. (2019). Ben sigelman on distributed tracing [software engineering radio]. *IEEE Software, 36*(01), 98–101. https://doi.org/10.1109/MS.2018.2880598

Davison, R., Martinsons, M. G., & Kock, N. (2004). Principles of canonical action research. *Information Systems Journal, 14*(1), 65–86.

Foundation, A. S. (2020). *Interview guide: Quality in use of anomaly classification*. https://jmeter.apache.org/index.html

Gray, J., & Reuter, A. (1992). *Transaction processing: concepts and techniques*. Elsevier.

IEC. (2015). *International electrotechnical vocabulary – part 192 dependability* (IEC 60050-192).

IEEE. (2008). *Systems and software engineering – measurement process* (IEEE Std 15939).

IEEE. (2012). *Systems and software engineering – life cycle management – guidelines for process description* (IEEE Std 24774).

ISO/IEC. (2011a). *Information technology – service management – part 1: Service management systems requirements* (ISO/IEC 20000-1:2011).

ISO/IEC. (2011b). *Systems and software engineering – systems and software quality requirements and evaluation (SQuaRE) – evaluation process* (ISO/IEC 25040).

ISO/IEC. (2011c). *Systems and software engineering – systems and software quality requirements and evaluation (SQuaRE) – measurement of quality in use* (ISO/IEC 25022).

ISO/IEC. (2011d). *Systems and software engineering – systems and software quality requirements and evaluation (SQuaRE) – system and software quality models* (ISO/IEC 25010).

ISO/IEC. (2012). *Systems and software engineering – systems and software quality requirements and evaluation (SQuaRE) – quality measure elements* (ISO/IEC 25021).

ISO/IEC. (2014). *Systems and software engineering – systems and software quality requirements and evaluation (SQuaRE) – guide to (SQuaRE)* (ISO/IEC 25000).

ISO/IEC. (2016). *Systems and software engineering – systems and software quality requirements and evaluation (SQuaRE) – measurement of system and software product quality* (ISO/IEC 25023).

ISO/IEC. (2017). *Information technology – systems and software engineering – systems and software quality requirements and evaluation (SQuaRE) – service quality models* (ISO/IEC TS 25011).

ISO/IEC. (2018). *Information technology – service management – part 1: Service management systems requirements* (ISO/IEC 20000-1:2018).

ISO/IEC. (2019). *Systems and software engineering – systems and software quality requirements and evaluation (SQuaRE) – quality requirements framework* (ISO/IEC 25030).

ISO/IEC/IEE. (2017). *Systems and software engineering - vocabulary* (ISO/IEC/IEE 24765).

Jaeger. (2020). *Jaeger: open source, end-to-end distributed tracing*. https://www.jaegertracing.io/

Kontio, J., Bragge, J., & Lehtola, L. (2008). The focus group method as an empirical tool in software engineering. In *Guide to advanced empirical software engineering*, pages 93–116. Springer.

A. W. Lai. Consumer values, product benefits and customer value: a consumption behavior approach. Advances in consumer research, 22:381–381, 1995.

Mayring, P. (2014). *Qualitative content analysis: Theoretical foundation, basic procedures and software solution*.

Netflix. (2020). *Zuul*. https://github.com/Netix/zuul

NewRelic. (2020). *Distributed tracing*. https://docs.newrelic.com/docs/understanddependencies

Niedermaier, S., Koetter, F., Freymann, A., & Wagner, S. (2019). On observability and monitoring of distributed systems–an industry interview study. In *International Conference on Service-Oriented Computing*, pages 36–52. Springer.

Niedermaier, S., Zelenik, T., & Wagner, S. (2020). *Interview guide: Quality in use of dependability evaluation approach*. http://doi.org/10.5281/zenodo.3826099

OpenTracing. (2020). *The open tracing semantic specification*. https://opentracing.io/specification/

Plöesch, R., Schuerz, S., & Koerner, C. (2015). On the validity of the it-cisq quality model for automatic measurement of maintainability. In *2015 IEEE 39th Annual Computer Software and Applications Conference*, volume 2, pages 326–334. IEEE.

Reason, P., & Bradbury, H. (2001). *Handbook of action research: Participative inquiry and practice*. Sage.

Sambasivan, R. R. , Shafer, I., Mace, J., Sigelman, B. H., Fonseca, R., & Ganger, G. R. (2016). Principled workflow-centric tracing of distributed systems. In *Proceedings of the Seventh ACM Symposium on Cloud Computing*, pages 401–414.

Sigelman, B. H., Barroso, L. A., Burrows, M., Stephenson, P., Plakal, M., Beaver, D., et al. (2010). *Dapper, a large-scale distributed systems tracing infrastructure*. Technical Report, Google: Technical report.

Staron, M. (2019). Action research in software engineering: Metrics' research perspective (invited talk). pp. In B. Catania, R. Královič, J. Nawrocki, & G. Pighizzini (Eds.), *SOFSEM 2019: Theory and Practice of Computer Science* (pp. 39–49). Cham: Springer International Publishing.

W3C. (2020). *Trace context*. https://www.w3.org/TR/2020/REC-trace-context-1-20200206/trace-id

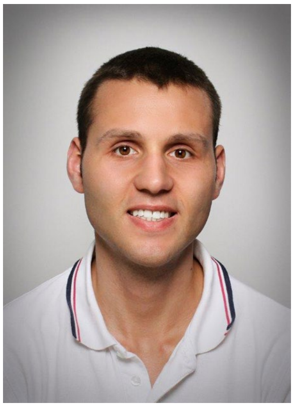
Wagner S. (2013). *Software product quality control*. Springer.

Wagner, S., Goeb, A., Heinemann, L., Kläs, M., Lampasona, C., Lochmann, K., et al. (2015). Operationalised product quality models and assessment: The quamoco approach. *Information and Software Technology, 62,* 101–123.

Whitmore, R. (2019). *Understand distributed tracing*. https://docs.lightstep.com/docs/understand-distributed-tracing

Sina Niedermaier is a PhD student at the Institute of Software Engineering at the University of Stuttgart in Germany since 2018. Besides, she is an external PhD student at Robert Bosch Group in the domain of Software Quality Management. She obtained her master`s degree in Industrial Engineering, focusing on Information Systems at Friedrich-Alexander University of Erlangen-Nuremberg, Germany. Since 2017 her research is focused on the observability and monitoring of complex distributed systems for quality evaluation and control. She investigates anomalies from a dynamic system view to control the quality in use with fault reactions. In case study research, she develops, implements and evaluates concepts for quality evaluation and control in collaboration with industry.

Thommy Zelenik obtained his bachelor`s degree in software engineering at the University of Stuttgart in 2016 by conducting research on the relationship between stress and attention by using thermal imaging techniques. He also obtained his master`s degree in software engineering in 2020 by researching the applicability of distributed tracing for deducing quality measures in distributed systems. Furthermore, he is an open-source contributor of the winery project, licensed by eclipse and has experience in full-stack software development.



Stefan Heisse is a Senior Systems Analyst at the Bosch Engineering GmbH Abstatt, Germany. Currently, he is working on ontology and terminology management to support cooperation and collaboration relationships of inter- and intra-team communications of different departments at Robert Bosch Group in the context of Bosch IoT-Services. Since 1991 he owns a Master`s Degree in Numerical Mathematics and Computer Science. He has professional experience in financial accounting and implementingfront- and back-offce systems and service management. He changed to Systems and Software Engineering for automotive ECU with functional safety, now focusing on models and standards of the Joint Technical Committee ISO/IEC JTC 1, Information technology, Subcommittee SC 7, Software and systems engineering. He researches on the in influence of strategic quality management topics and organization of work.

Stefan Wagner is a full professor of empirical software engineering at the University of Stuttgart, Germany. His research interests include software quality, requirements engineering, safety and security, and agile and continuous software engineering. He is member of ACM, IEEE and the German GI.