Institute of Software Engineering
Software Quality and Architecture

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit

Evaluating Human-Computer Interfaces for Specification and Comprehension of Transient Behavior in Microservice-based Software Systems

Samuel Beck

Course of Study:	Softwaretechnik
Examiner:	Dr.-Ing. André van Hoorn
Supervisor:	Sebastian Frank, M.Sc., Dr. Leonel Merino
Commenced:	May 27, 2020
Completed:	November 27, 2020

Abstract

Modern software systems are subject to constant change while operating in production. New agile development methods such as continuous deployment and DevOps enable developers to deploy code changes frequently. Also, failures and self-adaptation through mechanisms such as elastic scaling and resilience patterns introduce changes into a system during runtime. For that reason, these systems that become more complex and distributed continuously exhibit transient behavior, the state that occurs while transitioning from one state to another. To make statements about a system's reliability and performance, it is imperative that this transient behavior is specified in non-functional requirements and that stakeholders can review whether these requirements are met. However, due to the complexity of this behavior and the accompanying specifications, only experts can achieve this.

This thesis aims to make the specification of non-functional requirements for, and the comprehension of, transient behavior in microservice systems more accessible, particularly for stakeholders that lack expert knowledge about transient behavior. To achieve this, novel approaches are explored that utilize modern human-computer interaction methods to facilitate this problem. At first, the state of the art in transient behavior in software systems, human-computer interaction, and software visualization is presented. Subsequently, expert interviews are conducted to understand how transient behavior is handled in practice and which requirements experts have to an envisioned solution. Based on this, a concept for a solution is proposed, which integrates different visualizations with a chatbot, and implemented as a prototype. Finally, the prototype is evaluated in an expert study. The evaluation shows that the approach can support software architects and DevOps engineers to create and verify specifications for transient behavior. However, it also reveals that the prototype can still be improved further. Furthermore, it was shown that the integration of a chatbot into the solution was not helpful for the participants. In conclusion, human-computer interaction and visualization methods can be applied to the problems of specifying and analyzing transient behavior to support software architects and engineers. The developed prototype shows potential for the exploration of transient behavior. The evaluation also revealed many opportunities for future improvements.

Kurzfassung

Moderne Softwaresysteme sind in Produktion ständigen Änderungen unterworfen. Neue agile Methoden, wie Continuous Deployment und DevOps, ermöglichen es Entwicklern häufig Codeänderungen zu deployen. Außerdem führen Ausfälle und Anpassungen durch das System selbst, wie etwa durch Mechanismen wie elastische Skalierung und Resilience Patterns, während der Laufzeit Änderungen am System durch. Aus diesem Grund weisen diese immer komplexer und verteilteren Systeme kontinuierlich transientes Verhalten auf. Der Zustand, der auftritt während von einem Zustand in den nächsten übergegangen wird. Um Aussagen zur Zuverlässigkeit und Performanz eines Systems zu treffen ist es unerlässlich, dieses transiente Verhalten in nicht-funktionalen Anforderungen zu spezifizieren, deren Einhaltung von Stakeholdern überprüft werden kann. Aufgrund der Komplexität dieses Verhaltens und der dazugehörigen Spezifikationen ist dies allerdings nur für Experten machbar.

Diese Arbeit zielt darauf ab, die Spezifikation von nicht-funktionalen Anforderungen für und die Analyse von transientem Verhalten zugänglicher zu machen, insbesondere für Stakeholder, denen es an Expertenwissen zu transientem Verhalten mangelt. Um dies zu erreichen werden neuartige Ansätze erkundet, die moderne Methoden aus dem Gebiet der Mensch-Computer Interaktion verwenden, um dieses Problem zu vereinfachen. Zuerst wird der aktuelle Stand der Wissenschaft in den Gebieten transientes Verhalten von Softwaresystemen, Mensch-Computer Interaktion und Softwarevisualisierung vorgestellt. Anschließend werden Experteninterviews durchgeführt, um zu verstehen, wie mit transientem Verhalten in der Praxis umgegangen wird und, welche Anforderungen Experten an eine ausgemalte Lösung stellen. Auf Grundlage davon stellen wir ein Konzept für eine Lösung vor, welche verschiedene Visualisierungen mit einem Chatbot verbindet. Dieses Konzept wird anhand eines Prototyps implementiert, dieser wird anschließend in einer Expertenstudie evaluiert. Die Evaluation zeigt, dass der vorgestellte Ansatz Softwarearchitekten und DevOps Ingenieure beim Erstellen und Verifizieren von Spezifikationen für transientes Verhalten unterstützen kann. Allerdings stellt sich auch heraus, dass es noch Verbesserungspotential für den Prototypen gibt. Des Weiteren ist eine Erkenntnis, dass die Studienteilnehmer die Integration des Chatbots in die Lösung nicht als hilfreich empfanden. Abschließend lässt sich sagen, dass sich Methoden aus der Mensch-Computer Interaktion und Softwarevisualisierung auf die Probleme der Spezifizierung und Analyse von transientem Verhalten anwenden lassen, um Softwarearchitekten und -ingenieure zu unterstützen. Der entwickelte Prototyp zeigt Potential für die Erkundung und Analyse von transientem Verhalten. Die Evaluation deckte darüber hinaus zahlreiche Gelegenheiten für zukünftige Verbesserungen auf.

Contents

1 Introduction **1**
1.1 Motivation and Problem . . . 1
1.2 Goal and Method . . . 2
1.3 Thesis Structure . . . 2

2 Foundations **5**
2.1 Microservice Architectural Pattern . . . 5
2.2 Transient Behavior . . . 5
2.3 Human-Computer Interaction . . . 9
2.4 Information Visualization . . . 13
2.5 Evaluating Human-Computer Interaction Methods . . . 18

3 Related Work **19**
3.1 Application of Novel Human-Computer Interfaces in Software Engineering . . . 19
3.2 Resilience Visualization . . . 20

4 Expert Interviews about Transient Behavior **23**
4.1 Goals . . . 23
4.2 Method . . . 23
4.3 Results . . . 26
4.4 Discussion . . . 34

5 Concept **37**
5.1 Use Cases . . . 37
5.2 Visualizations . . . 45
5.3 Conversational Visualization Approach . . . 48

6 Prototype **53**
6.1 Technologies . . . 53
6.2 System Design . . . 54
6.3 Data Structure and REST API . . . 55
6.4 Dashboard . . . 56
6.5 Transient Behavior Identification . . . 58
6.6 Chatbot . . . 59

7 Evaluation **63**
7.1 Evaluation Goals . . . 63
7.2 Evaluation Methodology . . . 64
7.3 Evaluation Setup . . . 65
7.4 Experiment Settings . . . 67

7.5 Description of Results 69
7.6 Discussion of Results 79

8 Conclusion **85**
8.1 Summary 85
8.2 Discussion 85
8.3 Future Work 86

Bibliography **89**

A Interview Resources **97**
A.1 Consent Form 98

B Evaluation Resources **101**
B.1 Consent Form 101
B.2 Information 101
B.3 Task Descriptions 107
B.4 Questionnaire 109

List of Figures

2.1 Simplified example microservice system. 6
2.2 Bruneau's resilience triangle model [BCE+03] on a resilience curve, adopted from Yin et al. [YDW+19]. 9
2.3 On the left, there is an intent definition consisting of an input and output context, entities and training phrases. On the right, there is an exemplary conversation between a chatbot and a user. 12
2.4 The visualization reference model, after [Car99]. 14
2.5 Example for preattentive processing. 15

3.1 Resilience curves visualized as hyperbolas, from [Zob10]. 20
3.2 A heat map that visualizes the resilience of a system against disruptions with different stress levels, from [DRB16]. 21

4.1 Interview Design 24

5.1 Structural hierarchy of a Goal Question Metric (GQM) model. 38
5.2 Visualization of the services and their dependencies. 45
5.3 Visualization of the quality of service over time. 46
5.4 Visualization of a specification for transient behavior. 47
5.5 Comparison between the transient behavior of a service and its specification. . . . 47
5.6 The orange area marks the loss of resilience of the service. 48
5.7 Visualization of the actual loss of resilience of a service and the specified loss of resilience. The specified maximum loss of resilience is represented by a red line. 48
5.8 Sketch of the conversational visualization. 49

6.1 The architecture of the prototype. 54
6.2 The dashboard of the prototype. 57
6.3 Add specification dialog. 58

7.1 Architecture of the mockup payroll accounting system, adopted from [KWK+20a]. 65
7.2 Architecture of the SockShop demo microservice system. 66
7.3 The time that it took the participants to solve the respective task. 70
7.4 Distribution of answers for the first six usability statements. 75
7.5 Distribution of answers for the last five usability statements. 76
7.6 Distribution of answers for the four questions concerning cognitive load. 77
7.7 Level of agreement with the statement *The integration of a chatbot in the tool was helpful*. 78
7.8 Deviation between the plotted specification and transient behavior. 80

A.1 Interview consent form page 1. 98

A.2 Interview consent form page 2. 99

B.1 User study consent form page 1. 102
B.2 User study consent form page 2. 103
B.3 Architecture of the mocked payroll accounting system. 104
B.4 Architecture of the SockShop system. 104
B.5 Resilience curve with indication of initial loss, recovery time, and resilience loss. 106

List of Tables

2.1 The five most effective visual variables for each measurement scale, after [Mac86]. 14
2.2 The seven categories of interactions techniques proposed by Yin et al. [YDW+19]. 16

4.1 Background information of the five interviewees. 27

5.1 GQM model for UC 1. 39
5.2 GQM model for UC 2. 40
5.3 GQM model for UC 3. 42
5.4 GQM model for UC 4. 43
5.5 GQM model for UC 5. 44

7.1 Endpoints of `payslip` and `payslip2`. 66
7.2 Endpoints of the services of SockShop. 67

B.1 Endpoints of payslip and payslip2. 104
B.2 Endpoints of the services of SockShop. 105

Listings

6.1 JSON representation of a service object. 55
6.2 JSON representation of a dependency object. 55
6.3 JSON representation of a measurement object. 56
6.4 JSON representation of a specification object. 56
6.5 Dialogflow request that shares information about a recognized intent with the backend. 60
6.6 WebSocket message containing the command to show the specification for the Cart service for transient behavior caused by failure. 61

Acronyms

API Application Programming Interface. 5

CRUD Create, Read, Update, Delete. 55

CSS Cascading Style Sheets. 53

DOM Document Object Model. 53

GQM Goal Question Metric. ix

HCI Human-Computer Interaction. 9

HTML Hypertext Markup Language. 53

HTTP Hypertext Transfer Protocol. 53

InfoVis Information Visualization. 13

JSON JavaScript Object Notation. 55

MRMM Microservice Resilience Measurement Model. 22

NASA National Aeronautics and Space Administration. 69

NUI Natural User Interface. 10

REST Representational State Transfer. 54

SDK Software Development Kit. 13

SV Software Visualization. 17

SVG Scalable Vector Graphics. 53

1 Introduction

This chapter is divided into three sections. The first, Section 1.1 explains the complexity of transient behavior and the need for its specification and assessment. Next, Section 1.2 outlines the goal of this thesis and describes how it approaches the problem of aiding different stakeholders in specifying and analyzing the transient behavior of microservice systems. Finally, Section 1.3 sets out how the remainder of the thesis is structured.

1.1 Motivation and Problem

Modern software systems are complex, highly distributed, and subject to constant change during runtime. To uphold their reliability is a great challenge under these circumstances. Nonetheless, a multitude of software systems is highly critical, and failures can cause enormous harm and costs [Kri10; Tho08; Wol18]. For this reason, non-functional requirements for the reliability and performance of software must be specified and continuously assessed during the runtime. However, the numerous changes that a system must endure complicate these task. For instance, DevOps [BWZ15] enables developers to frequently deploy new software releases. Microservice systems [New15] are susceptible to service failures that can propagate through the system. To prevent them, resilience mechanisms can automatically change the service behavior. Furthermore, systems are exposed to unpredictable workloads and attacks from malicious users.

All these sources of change imply that microservice systems are almost continuously in a state of transient behavior, i.e., in a transition from one state to another. For that reason, to assure a system's reliability and performance, it is imperative that this behavior is specified in non-functional requirements, which stakeholders continuously evaluate. However, due to the complexity of this transient behavior and the specifications, these demanding tasks can only be conducted by experts. Czepa and Zdun [CZ19] recently found evidence that both textual and graphical formal specification languages are difficult to use for novice developers. Furthermore, they discovered that most of their participants had no interest in using the investigated languages in the future. This serves as incentive to create novel solutions for capturing non-functional requirements, peculiarly those for transient behavior. Methods and technologies from the field of human-computer interaction have successfully been applied to problems in other domains to increase both user performance and user experience. For instance, Züllighoven et al. [ZNR+18] developed an application for the monitoring and planning of shipping traffic in the port of Hamburg that runs on a touch table and utilizes multi-touch to mimic the navigators' former process that comprised paper maps and rulers. To name another example, Shakil et al. [SLW19] exploit eye tracking to simplify code navigation within integrated development environments. They discovered that participants of their study mainly perceive this method for code navigation intuitive and natural. Furthermore, Merino et al. [MGAN17] created CityVR, an interactive software visualization tool that visualizes software in virtual reality as a city metaphor. They observed that developers felt immersed and excited when interacting with the visualization. Toxtli et al. [TMC18] deployed a chatbot to help users to create, assign, and keep track of tasks in their team. Participants of their study mostly reported that they felt more productive because of the chatbot and that they will use it in the future. The hypothesis that is investigated in this thesis is that these benefits of modern HCI methods could be transferred to software engineering and help to address the challenges of comprehending and capturing non-functional requirements for the transient behavior of software systems.

1.2 Goal and Method

This work aims to make the specification of non-functional requirements for transient behavior and the comprehension thereof more accessible, particularly for stakeholders that lack expert knowledge about transient behavior. Proposed requirements models [YDW+19] do not capitalize on state-of-the-art human-computer interaction methods to increase user experience and performance. Such methods could be employed to support stakeholders in these tasks.

To reach the goal, the state of the art in resilience engineering, human-computer interaction, and software visualization is researched. Also, a series of expert interviews is conducted to understand how architects and engineers deal with transient behavior in the industry. The interviews also serve as an opportunity to collect the requirements that stakeholders have for a solution that supports them in the specification and analysis of transient behavior. The literature research and interviews serve as a basis for creating a concept that involves using novel human-computer interfaces to facilitate the tasks mentioned above. Afterward, the concept is implemented as a prototype to evaluate its feasibility. The evaluation of the approach also examines to what extent and in which tasks the proposed approach supports architects and engineers.

1.3 Thesis Structure

The remainder of the thesis is structured as follows:

Chapter 2 – Foundations: Describes the foundations of the thesis. The concept of transient behavior and its causes are introduced before an overview of different human-computer interaction techniques with a focus on natural user interfaces and multimodal interaction is given. Subsequently, the fundamental concepts of information visualization are explained.

Chapter 3 – Related Work: Gives an overview of work that is related to the thesis and positions the thesis within it.

Chapter 4 – Expert Interviews about Transient Behavior: Presents a series of expert interviews about transient behavior and the requirements for a potential solution.

Chapter 5 – Concept: Outlines a solution that supports architects and engineers in capturing and comprehending non-functional requirements for transient behavior with the help of novel human-computer interfaces.

Chapter 6 – Prototype: Presents a prototype that was developed to evaluate the feasibility of the proposed approach for the analysis and specification of transient behavior. The focus is on the design and implementation of the prototype as well as on the used technologies.

Chapter 7 – Evaluation: Describes the evaluation of the developed prototype, including the experimental setup, the results that were obtained, and a discussion of the results.

Chapter 8 – Conclusion: Concludes the thesis by summarizing the main findings and discussing potential future work.

Supplementary material, like the used datasets and study results, is publicly available online [Bec20]. The codebase of the prototype that has been developed as part of this work is also publicly available [Bec].

2 Foundations

This chapter provides the technical foundation for the rest of the thesis. Section 2.1 outlines the microservice architectural pattern. Section 2.2 introduces the concept of transient behavior and its causes in microservice systems. Furthermore, it defines the concept of resilience and how it can be quantified. Afterward, Section 2.3 gives an overview of the field of human-computer interaction with a focus on two paradigms: natural user interfaces and multimodal interaction. Subsequently, Section 2.4 presents the fields of information visualization and software visualization. Finally, Section 2.5 presents state-of-the-art guidelines and best practices for evaluating human-computer interaction methods.

2.1 Microservice Architectural Pattern

The microservice architectural style [New15] has become popular throughout the past years, and many organizations are adopting it [BHJ16]. A microservice application is developed as a composition of decentralized self-contained services that are independently deployed, which communicate with each other through lightweight protocols, often HTTP. Furthermore, each service implements a single function and can be written in a different programming language, which allows every development team to choose their technology stack. Each service only has to adhere to the Application Programming Interface (API) requirements. Otherwise, the choice of technologies is flexible. Microservices are independently deployable through a fully automated continuous deployment pipeline [FL14].

The distributed nature of microservice systems introduces various failure points. Any service call could fail at any time due to the unavailability of the called service, e.g., high network latency. However, the failure of a single service does not necessarily entail the unavailability of the whole system. Instead, the failure might cascade from service to service, potentially degrading the performance of the system. To prevent this, services need to be designed in a way that allows them to respond to the failure of other services as gracefully as possible. The handling of such service failures adds additional complexity compared to traditional monolithic systems and is another disadvantage of the microservice architectural style. Consequently, development teams continuously reflect on the repercussions of service failures on the user experience of the application [FL14] and discuss the resilience of microservice systems.

2.2 Transient Behavior

The term transient behavior has its origin in electrical engineering. Venikov [Ven14] defines that transient behavior or a transient state “occurs [in an electrical power system] when the system is changing from one steady state to another”. This is caused by any switching operation that changes

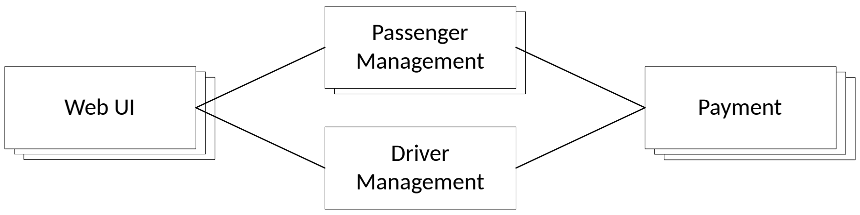


Figure 2.1: Simplified example microservice system.

the amount of power in an electrical circuit, which is called a normal transient process. However, accidental changes to the system, as the disconnection of a transmission line, might cause a fault transient process. Like electrical systems, software systems also have transitions between steady states and can exhibit transient behavior during a transition from one state to another.

Software systems are subject to constant change while operating in production. New agile development methods such as continuous deployment [HF10] and DevOps [BWZ15] enable developers to release new functionality frequently. Software architectures must be designed with this in mind, like the popular new microservice architectural style [New15]. Nonetheless, this new architectural style is susceptible to service failures, network latency, and failures cascading through the system. Resilience mechanisms [Nyg18] are employed to ensure service quality attributes and quick failure recovery. They allow the system to react automatically to failures. As a consequence, modern software systems are widely distributed and complex. They undergo constant changes during run-time, like varying user behavior, service failures, deployment of new features, and reconfiguration of the system's architecture (e.g., scaling). For that reason, they continuously exhibit transient behavior under which their non-functional requirements are not sufficiently specified [YDW+19].

Indeed, various mechanisms can cause transient behavior in software systems. They are presented in Section 2.2.1. Afterward, Section 2.2.2 introduces the concept or resilience.

2.2.1 Causes of Transient Behavior

Numerous causes can introduce change into a software system and lead to transient behavior during this transition from one state into another. Among them are deployments, failures, resilience mechanisms, and load balancing. This subsection presents different causes for transient behavior in software systems and exemplifies them using three scenarios in the simplified microservice system depicted in Figure 2.1. The system consists of four services: `Web UI`, `Passenger Management`, `Driver Management`, and `Payment`. The `Passenger Management` and `Driver Management` services are connected to the `Payment` service to be able to process payments. They are also connected to the `Web UI` service which provides a user interface through which users can interact with the system. Some of the services, such as `Web UI` and `Payment` experience more load than others. For that reason they run simultaneously in multiple service instances to balance the load.

Continuous Deployment and DevOps

In the past, even more than today, software deployments used to be accompanied by system downtime and a high risk of failure. Deployments were only conducted a few times per year due to the immense effort that had to be undertaken by developers, testers, and operators to create and configure tests and production environments [BWZ15]. However, the adoption of agile values and methods demands that developers continuously release new value to the customer [BBV+01]. One such method is DevOps [BWZ15], a collection of practices and tools that increase teams' ability to deliver software at a high speed. This led to the creation of new deployment processes like continuous delivery and continuous deployment [HF10], which are embraced by an ever-growing number of development teams. Continuous deployment allows fully automated deployments of small software changes that occur in high frequency [SDG+16]. Amazon, for example, stated in 2014 that they averaged on more than one deployment per second [Vog14]. When a new service version is deployed, the system is in a transient state until the newly deployed version handles all requests to this service.

Lets consider the example system from Figure 2.1. If a new version of the `Driver Management` service is deployed, pending requests should still be handled by the instances that run the old version of the service. However, new requests should be routed to instances that already run the new version. This is called a *rolling deployment*, which is commonly used to test new features on a subset of users and ensure that no bugs are introduced by a new software version with the possibility to reverse the deployment if problems arise [BWZ15]. Because both the old and new version of the service are running simultaneously, the system is in an undefined state of transient behavior during the deployment. This lasts until the new service version is completely rolled out, and requests are no longer routed to the old instances.

Failures and Resilience Mechanisms

According to Avižienis et al. [ALRL04], a failure occurs in a software system when the service provided by the system deviates from the service that is specified as correct. The complexity of software makes it impossible to prevent failures completely [Fri16]. Some measures can be taken to increase a system's reliability, thus diminishing the probability of a failure occurring [Lyu07]. Additionally, the implementation of resilience mechanisms can make a system more tolerant towards failures [Nyg18]. Nonetheless, the occurrence of a failure changes the state of a system, introducing transient behavior during this state transition. Equally, resilience mechanisms meant to weaken the effect of a failure can still change a system's state. For example, in the circuit breaker pattern, requests are answered with a default response if a threshold of failed requests to an endpoint is crossed until the dependency is responsive again [R4J]. When this circuit breaker takes effect, the system state deviates from the previous steady state and exhibits transient behavior until another steady state is reached.

In our example, system depicted in Figure 2.1, the `Payment` service is executed in three separate instances. If one of these instances fails while handling a request, new requests to the `Payment` service are routed to the remaining instances, and the failing instance is restarted. However, the system's quality of service is affected by the failure until it is running again. In this case, the system exhibits transient behavior. It would be desirable to know in advance whether users will be affected by this instance failure and whether payments can still be calculated.

Varying User Behavior

A software system might be exposed to different types of workloads. The constant utilization of a system with only minimal changes in the number of user requests is called *static workload*. Static workload is an ideal scenario as the amount of required resources is known and changes only minimally. Therefore, elastic scaling is not necessarily required to deal with this workload. Elastic scaling allows for rescaling a cloud application during runtime, i.e., dynamically adding or removing resources to adapt to a changing workload. This saves resources and money compared to static scaling methods, which do not permit the dynamic addition or removal of computing resources to an application. This is an advantage when the workload is not static but changes, as is the case with *periodic workload* and *unpredictable workload*. In a *periodic workload*, the number of requests peaks in a predictable reoccurring time interval. In an *unpredictable workload*, the load is random and cannot be predicted, as the name suggests. Through elastic scaling, resources can be provisioned and decommissioned as needed to adapt to these changing workloads [FLR+14]. However, the dynamic addition and removal of additional service instances is another source of transient behavior. There is a period from when a new instance is started until it serves requests in which the system is in a transition between two states, i.e., exhibiting transient behavior.

Another source of transient behavior are security attacks, such as DDoS attacks [Clo20].

The `Passenger Management` service, in our example application, is currently running in two service instances. Let us assume that it is early evening on a weekday. The service experienced high load in the last hours as people were ordering rides home from work. As it gets later, fewer rides are hailed, and the load on the service decreases. This leads to under-utilization of the two instances, and the elasticity engine decommissions one of them. Until the instance is completely shut down, the system is in a state of transient behavior as the system waits for pending requests to be either processed or transferred to the other service instance.

2.2.2 Resilience

The New International Webster's Comprehensive Dictionary [96] defines resilience as "the ability to recover quickly from illness, change, or misfortune. Buoyancy. The property of a material that enables it to assume its original shape or position after being bent, stretched, or compressed. Elasticity." It is a well-established concept in many domains. As such, it is employed in the context of disaster recovery [BCE+03; CRB10], engineering [HBR16; YW16], and ecology [Hol73]. In disaster recovery, resilience is the ability to contain the effects of disasters while they take place and a swift recovery that minimizes their impact afterwards [BCE+03]. In engineering, systems are exposed to unpredictable environments and conditions, similar to software systems where requirements and user behavior can unexpectedly change and services fail. The performance of a nonresilient-engineered system might decline after an unexpected disruption. Depending on the system it may keep operating steadily at a worse performance level or even continuously deteriorate until the system fails completely. A resilient-engineered system, on the other hand, will recover from the disruption over time and return to its previous performance level [YW16].

A common way to visualize resilience are resilience curves [YDW+19]. Figure 2.2 illustrates the resilience triangle model proposed by Bruneau et al. [BCE+03] on a resilience curve. The x-axis stands for the time t and the y-axis stands for the quality of the system $Q(t)$. According to Bruneau

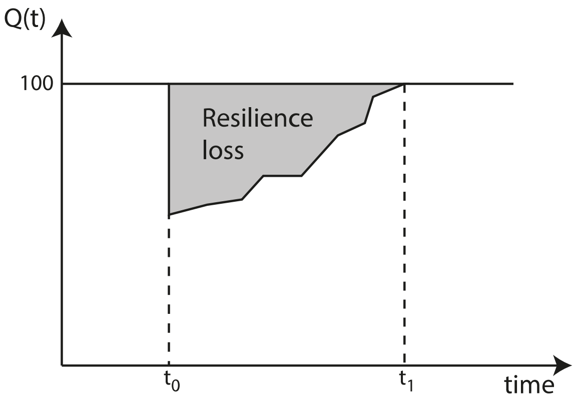


Figure 2.2: Bruneau's resilience triangle model [BCE+03] on a resilience curve, adopted from Yin et al. [YDW+19].

et al., resilient systems are more robust against failures and when they fail, a failure has less severe consequences than in a nonresilient system. Furthermore, resilient systems recover faster from failures. In Figure 2.2 robustness is measured on the y-axis, rapidity on the x-axis and the loss of resilience (R) by the shaded area, which roughly resembles a triangle. A disruption at t_0 causes a reduction of quality. Afterwards the system recovers from the disruption until t_1, where it resumes normal operation. R can be quantified by the reduction of quality over the recovery time:

$$R = \int_{t_0}^{t_1} Q(t_0) - Q(t_1) dt \tag{2.1}$$

Indeed, resilience curves can be suitable visualizations to support the analysis of resilience in microservice systems [YDW+19]. The analysis of resilience is a challenging task. However, Human-Computer Interaction (HCI) methods have successfully been applied to other domains to increase both user performance and experience [MGAN17; SLW19; TMC18; ZNR+18].

2.3 Human-Computer Interaction

HCI can be defined as "a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them" [HBC+92].

In their beginnings, computers provided limited interaction capabilities, for instance, through command-line interfaces, which constrained their use to professional users. However, thanks to the application of methods from psychology, design, and other domains in the field of computer science, researchers, designers, and developers have been able to create new user experiences that make computers accessible and empower also non-experts to solve a wide range of problems [SPC+16].

HCI is an interdisciplinary area. According to Hewett et al. [HBC+92], the main concerns of HCI are (i) the joint performance of humans and machines in a task, (ii) the structure of communication between human and machine, (iii) human capabilities to use machines, (iv) algorithms and the programming of interfaces, (v) engineering concerns about designing and building interfaces, and (vi) the process of specification, design, and implementation of interfaces. Computer science emphasizes on the design and implementation of applications and human interfaces within the field of HCI. In addition, psychology provides theories on cognitive processes and methods for empirical analysis of user behavior. Also, industrial and graphic design deal with designing interactive products [HBC+92].

In effect, various modern HCI methods have proven effective by utilizing humans' inherent capabilities. In the following, Section 2.3.1 will introduce the notion of natural user interfaces. Afterward, Section 2.3.2 presents the concept of multimodal interaction. Subsequently, Section 2.3.3 describes chatbots in detail before Section 2.3.5 gives a short introduction about virtual and augmented reality and Section 2.3.4 introduces voice assistants.

2.3.1 Natural User Interfaces

Natural User Interfaces (NUIs) allow users to interact with a computer like they normally interact with the real world [JLW11]. Hence, Wigdor and Wixon describe that "in the natural user interface, natural refers to the user's behavior and feeling during the experience" [WW11]. The goal of NUIs is to provide a smooth user experience where technology is invisible to the user by combining inputs and outputs that users experience as natural. In WIMP-style interaction (i.e., windows, icons, menus, pointer), humans interact with a computer primarily through only a few modes. For example, a keyboard might serve as input mode and text on a screen as an output mode [Tur14]. NUIs, on the other hand, utilize various modes of interaction for input such as gesture, body language, proximity and location, eye gaze, and facial expression. Audio and visual output, tactile and object location, smell, and others are employed for output [JLW11]. To create an interaction that feels natural, the chosen technique must meet the user's needs, fit their task and environment, and take full advantage of their abilities [WW11]. As humans interact with the real world using multiple modalities (e.g., sight and hearing), NUIs are often expected to combine multiple modes to provide users a seamless experience [JLW11]. This is called multimodal interaction.

2.3.2 Multimodal Interaction

Although HCI historically focuses on unimodal interactions, humans interact with the world in a multimodal manner. We utilize multiple senses, both sequentially and in parallel, to perceive information, like sight, hearing, and touch. Through these multiple sense, we collect an enormous amount of information that helps us interact with the world.

Multimodal interfaces attempt to leverage natural human communication and interaction abilities for HCI by combining modalities such as speech, gesture, touch, facial expressions. Multimodal interaction supports the recognition of such modalities, based on technologies, to boost user experience and user performance [Tur14]. The first example of a system that utilizes a multimodal interface is the "Put That There" system by Bolt in 1980 [Bol80]. Recent research conjectures that users may prefer multimodal interfaces over unimodal alternatives. For instance, multimodal interfaces can offer more flexibility to users and can be more reliable. Using them, it could be possible to present a user with multiple interaction alternatives. Additionally, multimodal interaction might increase task efficiency slightly and enables humans to process information faster and better [Tur14].

Designing a system that employs multimodal interfaces is a challenging task as each environment might require to take different design decisions [Tur14]. Reeves et al. [RLL+04] proposed a set of guidelines to support designers in the following tasks: (i) requirements specification, (ii) designing multimodal input and output, (iii) enabling interfaces to adapt to needs and abilities of different users, (iv) ensuring consistency, (v) giving feedback to the user, and (vi) preventing and handling errors. A technical key challenge when designing multimodal interfaces is their integration, i.e., integrating multiple interaction modes with each other (either sequentially or in parallel) [Tur14]. Nigay and Coutaz [NC93] classified multimodal interfaces into four categories, depending on the fusion method and the use of modalities in a system: (i) exclusive, (ii) concurrent, (iii) combined, and (iv) synergistic. Whereas other types of multimodal systems have benefits, synergistic is a main concern of multimodal interaction as it integrates modalities and allows their use in parallel. In particular, a technology that allows users to interact with software systems through natural language is chatbots.

2.3.3 Chatbots

Bots are a software tool that automates repetitive tasks [PV18]. They are becoming more common in software engineering due to the increase in efficiency they bring with them. There are different types of bots, some that support developers with coding or testing, others used in a DevOps context to speed up deployments [SZ16]. However, chatbots distinguish themselves from these bots that are meant to automate tasks. Instead, they simulate written or spoken conversations and attempt to give the users the impression that they are interacting with another human being [PV18]. Chatbots are used in different contexts and are growing increasingly popular. They can answer questions and provide information or act as a conversational interface to a more extensive service in the background [Goa19]. It is not surprising that they are also being adopted in software engineering [AS].

One of the first attempts to create a program that converses with users in a human-like manner was ELIZA, a so-called “chatterbot” created in 1966 by Weizenbaum [Wei66]. ELIZA answered utterances entered by the user with vague questions, giving the impression of talking to a psychotherapist [KDMB17]. Another well-known chatbot is ALICE, short for Artificial Linguistic Internet Computer Entity, developed in 1995 by Wallace [Wal09]. ALICE, which is based on ELIZA but expanded its capabilities by far, won the Loebner prize, an annual prize for the most human-like system, in 2000, 2001, and 2004 [KDMB17]. It is based on a simple pattern-matching algorithm that uses roughly 41,000 templates to represent different input and output categories.

Nowadays, chatbots rely on natural language processing methods and language models to understand user input. They can easily be integrated into messaging systems such as Slack[1], Telegram[2], Kik[3], or Facebook Messenger[4]. Through these platforms they are conveniently accessible to a wide range of users and can be used to interact with services that are used in everyday life [KDMB17]. There are chatbots for ordering food, booking restaurants or hotels, ordering a ride, or for simple customer service in the web [CU2020].

[1]https://slack.com/
[2]https://telegram.org/
[3]https://www.kik.com/
[4]https://www.messenger.com/

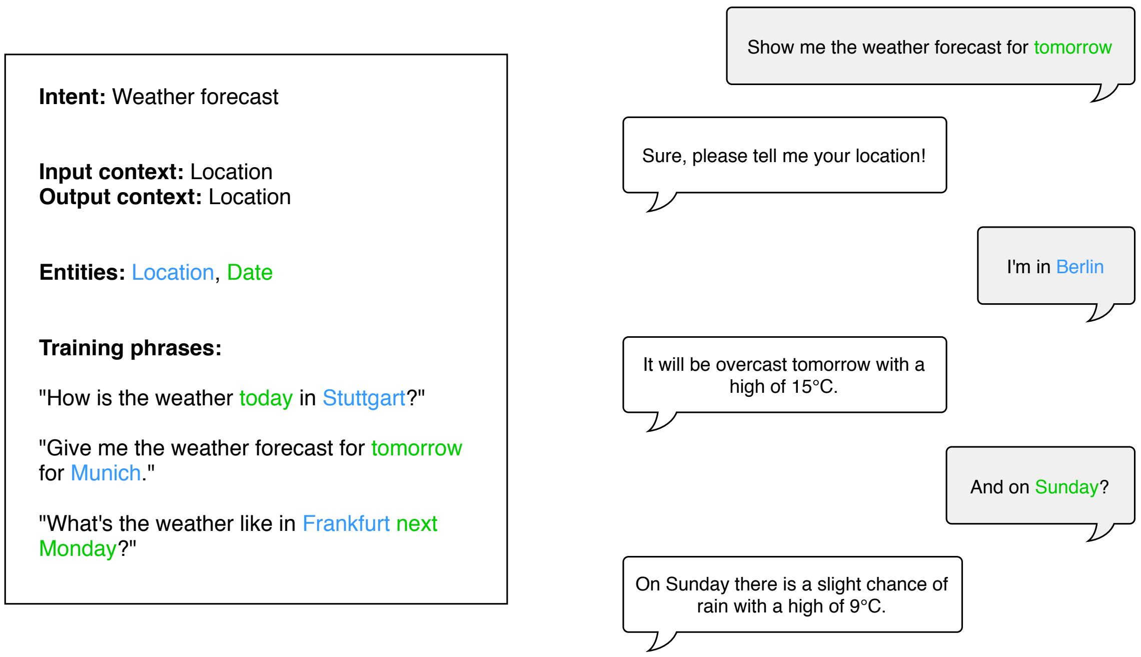

Intent: Weather forecast
Input context: Location
Output context: Location
Entities: Location, Date
Training phrases:
"How is the weather today in Stuttgart?"
"Give me the weather forecast for tomorrow for Munich."
"What's the weather like in Frankfurt next Monday?"
Show me the weather forecast for tomorrow
Sure, please tell me your location!
I'm in Berlin
It will be overcast tomorrow with a high of 15°C.
And on Sunday?
On Sunday there is a slight chance of rain with a high of 9°C.

Figure 2.3: On the left, there is an intent definition consisting of an input and output context, entities and training phrases. On the right, there is an exemplary conversation between a chatbot and a user.

To understand the natural language input, chatbots mainly use two concepts: *intents* and *entities* [DFC]. The intent of an input is what the user wants to know or do. A chatbot attempts to identify an intent for each phrase that the user inputs into the system. This is achieved by training a machine learning model with training phrases mapped to intents. The chatbot is able to map similar input phrases to the respective intent. Entities can be thought of as parameters for an intent. They can have different types, such as numbers, dates, locations, or custom types defined by the chatbot developers. Some chatbot frameworks make it possible to mark an entity as mandatory in an intent and automatically prompt the user for it if the entity is missing in the input. Furthermore, if a conversation consists of multiple messages, a chatbot can keep track of the user input context to accomplish more complex actions and avoid repeatedly asking for the same information.

An example of an intent definition of a chatbot that provides weather forecasts is shown in Figure 2.3. The two entities needed for the weather forecast are location and date, highlighted in blue and green respectively. For brevity, the intent is trained with three training phrases. The entities location and data are marked to train the chatbot to extract them from the user input. Typically, a real chatbot would be trained with a much larger number of training phrases to improve its accuracy. The intent also has an input and output context specified. The input context is a parameter that could have been specified in previous user interactions. The output context is a parameter that the chatbot will add to the context after the current interaction. In this case, the intent would take a previously specified location as input and save the user's location from the current interaction as the output context. The figure's right side illustrates how a conversation between a user and the chatbot might play out. The chatbot recognizes the weather forecast intent from the user input and can extract the date entity (`tomorrow`). However, the location entity, which is mandatory for the chatbot to answer, is missing from the input. Therefore, the chatbot prompts the user for the missing location. Once it has been provided, the chatbot queries a weather service in the background and provides the forecast in natural language to the user. The user follows up with a question about the forecast of the weather on another date. Because the chatbot saved the previously mentioned location in the conversation's context, it can reply with the according weather forecast without having to prompt the user for the location again.

There are numerous chatbot frameworks such as Microsoft Bot Framework [Mic19], Amazon Lex [Ama20], IBM Watson Assisstant [IBM20], and Dialogflow [Goo20] that provide a simple way to develop a chatbot. They do not only enable the design of intents and conversation flows in web interfaces, but they also provide pre-trained entities such as location, duration, length, and many more. Furthermore, they automatically train a language model with the provided training phrases, taking away the complexity of creating a new machine learning model from scratch. Finally, they provide software development kits (SDKs) that make it possible to implement advanced functionalities like communicating with other systems.

2.3.4 Voice Assistants

Voice interfaces work in a similar way as chatbot with the exception that the interaction with the users is slightly different. Voice assistants use artificial intelligence and deep learning algorithms to determine the best response. Furthermore, they employ big data and natural language generation to understand and respond to human speech. The main characteristic of voice assistants is that they respond to spoken language commands while chatbots interact with users through a written chat interface [Mar20].

2.3.5 Virtual and Augmented Reality

Augmented reality refers to a technology that allows the combination of digital and physical information in real time. "Augmented Reality supplements the real world with virtual (computer-generated) objects that appear to coexist in the same space as the real world" [VP10]. This environment created by augmented reality allows the reality observer to simultaneously interact with real and virtual elements [SH16].

In the case of virtual reality, on the contrary, there is a substitution of data, the virtual data substitutes the physical one. This allows the user to immerse himself in a completely computer-generated environment [SH16].

2.4 Information Visualization

Card [Car99] defines Information Visualization (InfoVis) as "the use of computer-supported interactive visual representations of abstract data to amplify cognition". This definition highlights the two most important tasks of InfoVis: (i) finding suitable visual representations for data, and (ii) creating adequate interactions that facilitate the use of visualizations in order to analyze data. Card describes the complete process of getting from raw data to visualization with his reference model for visualization illustrated in Figure 2.4. Data transformations like the application of filters or aggregations are used to structure raw data. The attributes of this structured data are then mapped

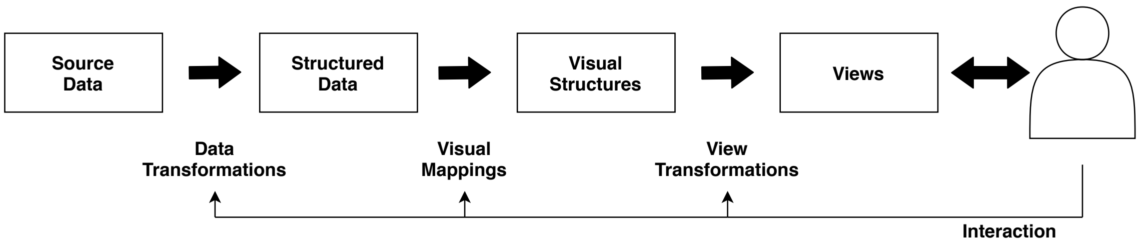


Figure 2.4: The visualization reference model, after [Car99].

Quantitative	Ordinal	Nominal
Position	Position	Position
Length	Density	Color Hue
Angle	Color saturation	Texture
Slope	Color Hue	Connection
Area	Texture	Containment

Table 2.1: The five most effective visual variables for each measurement scale, after [Mac86].

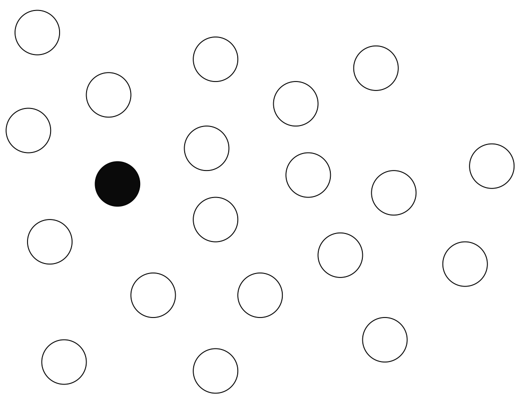
to visual variables to create visual structures. These can be changed via view transformations to create single views. User interactions can affect each step of this process in order to change the visualization.

Section 2.4.1 presents details of visual variables, after that, Section 2.4.2 discusses different interaction taxonomies. As we want to visualize data related to software, Section 2.4.3 presents the field of software visualization.

2.4.1 Visual Variables

Data attributes must be mapped to visual variables in order to create a visualization from a dataset. Depending on the type of data, different variables are more or less suited to represent it [Mac86].

Stevens [Ste+46] classified data into four scales of measurement. The most unrestricted is the *nominal scale*, which only assigns data items labels to classify them. An example of this scale could be car brands. The *ordinal scale* introduces order to the data. The difference between single values, however, does not have to be equal. Military ranks are a good example of this type of scale. The next scale is the *interval scale*, which is the first quantitative scale. In this scale, the difference between values is fixed. That allows operations such as addition or subtraction on the data. A commonly used interval scale is degrees Celsius. Like all interval scales, the Celsius scale does not have a meaningful zero. That means zero degrees Celsius is not the lowest temperature value. Since interval scales have only an arbitrary zero, ratios of their values are not meaningful. For this reason, an object with a temperature of 30 °C does not possess twice the heat of an object with a temperature of 15 °C. If an absolute zero exists, then the scale is a *ratio scale*. A ratio scale always has a meaningful zero and, therefore, meaningful ratios. The Kelvin scale for temperature and length are two examples of ratio scales. The existence of a meaningful zero makes ratios possible: a line that is two meters long is twice as long as a line that is only one meter long.


(a) The target can be identified preattentively because it has the unique property *filled*.

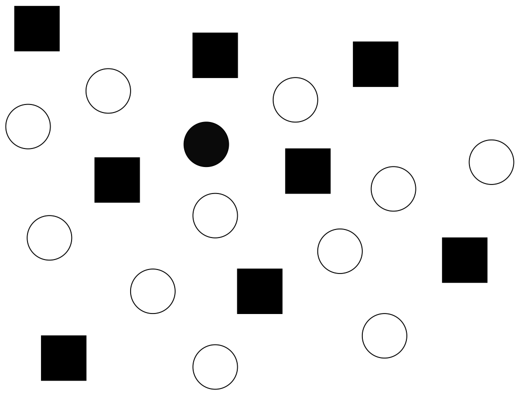

(b) The target cannot be identified preattentively because it has no visual property that is unique from the other elements.

Figure 2.5: Example for preattentive processing.

Visual variables are used to represent data attributes in visualizations. Prominent examples of such variables are position, size, shape, and color [Ber83]. Interestingly, such variables are not equally effective when used in a visualization, as observed by Cleveland and McGill [CM84]. They recognized that different quantitative perceptual tasks are accomplished with a varying degree of accuracy. Mackinlay [Mac86] extended this observation with a raking of visual variables for different measurement scales, which are listed in Table 2.1. He discovered that the most effective variables for quantitative scales are position, length, angle, slope, and area. The most effective visual variables for ordinal scales are, on the other hand, position, density, color saturation, color hue, and texture. Finally, for nominal scales, they are position, color hue, texture, connection, and containment. It can be seen that no matter the scale of the measurements, position is a suitable choice for a visual variable, and therefore, widely used in common charts.

Some visual properties can be processed very quickly in just around 200 milliseconds. This is called preattentive processing because it is done without the user focusing their attention on the visualization. Preattentive tasks require little effort and can be solved rapidly. For that reason, it is recommendable to utilize visual properties that can be processed preattentively as often as possible. They include form, color, motion, and spatial position [HBE96]. Figure 2.5 shows examples of tasks that can (and cannot) be processed preattentively. The task depicted in Figure 2.5a is to detect a filled circle among a group of empty circles. Because the target has the distinct visual property *filled* while the distractors do not, viewers can detect the target at a glance. This is not possible in the task presented in Figure 2.5b. Here, the target object has two properties: it has a circular shape and is filled. One of these properties is also present in each of the distractor objects: some are empty circles while others are filled squares. This constitutes a conjunction search that cannot be solved preattentively [Tre85].

Next to finding visual mappings for data, the second integral part of InfoVis is interaction [Few09].

Category	Description
Select	Mark something as interesting
Explore	Show something else
Reconfigure	Show a different arrangement
Encode	Show a different representation
Abstract/Elaborate	Show more or less detail
Filter	Show something conditionally
Connect	Show related items

Table 2.2: The seven categories of interactions techniques proposed by Yin et al. [YDW+19].

2.4.2 Interaction

As seen in the information visualization reference model in Figure 2.4, interactions are involved in the various steps of the visualization pipeline [Car99]. Dix et al. [DDF+03] define interaction as "the communication between users and the system". Interactions allow the user to mitigate limitations of static visualizations like spatial or temporal display constraints [DE98]. Through interacting with *data transformations*, the user can change the data displayed in a visualization. Operations like filtering or aggregating data help to get an overview of large datasets. Interactions can also be used to change the *visual mappings* between data attributes and visual variables to gain insights from a different visual representation. Finally, *view transformations* like zooming or panning over the visual structure allow users to change the view that they have of the visualization [Car99].

An often cited approach for designing interactions in InfoVis applications is the visual information-seeking mantra by Shneiderman [Shn96]: "overview first, zoom and filter, then details-on-demand". Overview first means that the first view to present should give an overview of the whole dataset. Zoom and filter signify that the subsequent interaction should be zooming in on items of interest and filtering out unimportant items. Last, details-on-demand involves selecting one or multiple data items and presenting them with more detail if needed.

Multiple taxonomies of interaction techniques have been proposed [BM13; Shn96; YDW+19]. After a comprehensive review of InfoVis applications, Yin et al. [YDW+19] proposed seven categories of prevalent interaction techniques, listed in Table 2.2. *Select* interactions allow the user to mark a data item to keep track of it in a large dataset. They are often followed by other interactions to facilitate data exploration. Due to technological and cognitive limitations, users can only view a limited number of data items simultaneously. *Explore* interactions permit to focus on a different subset of data items by exchanging currently visualized items with previously not visible ones. *Reconfigure* techniques are used to provide different perspectives on the same dataset by changing the visual representations' spatial arrangement. A prominent example of this category would be sorting the bars of a bar chart. *Encode* interactions allow the user to change the underlying visual mapping from data to visual variables to improve their understanding of the data, the relationships between individual data items, and their distribution. Through *Abstract/Elaborate* interaction techniques, users can modify the level of abstraction of the data representation. This might range from an overview of the whole dataset to the detailed presentation of individual data items. *Filter* techniques allow the user to restrict which data items might be shown in the visualization. Typically, a condition or range is used to narrow down which data items should be presented. Items that are not covered by this specification are not shown in the visualization. Finally, the category of *Connect* interactions is used to highlight relationships between data items that are already visible or to display hidden data items associated with a selected item. An established technique from this category is brushing and linking. Brushing is an interaction that allows the selection of the representations of multiple data items; linking is the highlighting of representations of the same items in another view.

2.4.3 Software Visualization

Software visualization is a subfield of InfoVis [BTP]. Its purpose is to support developers by providing them interactive visual representations of software systems that facilitate to understand a system [BE96]. Software systems are complex and difficult to comprehend, which is necessary for maintenance tasks. Maintaining them is time-consuming and requires a deep understanding of their code, architecture, and behavior.

There are different definitions for software visualization that vary in scope. Gračanin et al. [GME05] state that “the field of Software Visualization (SV) investigates approaches and techniques for static and dynamic graphical representation of algorithms, programs (code), and processed data. SV is concerned primarily with the analysis of programs and their development.” Diehl [Die07] extends this definition and defines that software visualization is “the visualization of artifacts related to software and its development process.” This deliberately includes requirements, documentation, and software structure and evolution.

Summarized, software visualization is concerned with visualizing the following three aspects of software:

Structure Static parts of the system that can be inferred without executing the code. Includes code, data structures, and architecture of the software.

Behavior Execution of the program.

Evolution Development process of the system and changes to the code.

The aspects relevant for this thesis are the structure, the behavior, and the performance of software systems — explicitly after changes occurred to the system.

Software Architecture Visualization

The architecture of a software system involves its structure. This, for instance, includes the various components that a system comprises, the relationships between these components, and different metrics that relate to the static aspects of systems [BCK03]. Software architecture visualization deals with the visualization of these entities and how they evolve [GC08].

As in all disciplines of visualization, much of the work in software architecture visualization amounts to finding suitable mappings from the elements that comprise the software architecture to visual representations. In this regard, two different approaches are pursued. Some techniques map architectural entities to abstract metaphors like spheres or cubes, while others prefer real metaphors such as buildings and islands. Some researchers argue that the familiarity of the human brain to real objects' properties can be exploited to increase the understanding of software. However, these visualizations' effectiveness might be negatively impacted by the complexity of these real metaphors [GC08]. A popular metaphor for software architecture is the city metaphor proposed by Wettel and Lanza [WL07]. They represent modules as city districts and the classes belonging to these modules as buildings in the respective district. Associated metrics, like the number of methods and attributes of a class, are mapped to the height and footprint of a building.

2.5 Evaluating Human-Computer Interaction Methods

Merino et al. [MGAN18] found, in a systematic literature review of software visualization evaluation approaches, that 62% of the reviewed papers lacked a strong evaluation of the proposed software visualization method. They argue that this lack of evaluation is one of the main reasons to explain why software developers do not broadly adopt novel software visualizations. Furthermore, they define guidelines for evaluating software visualizations. They argue that the goal of evaluations should be explicitly and unambiguously defined. Also, other factors, besides user performance and user experience, should be regarded in the evaluation, too. Those include engagement, recollection, and emotions. Likewise, the tasks in an experiment must fit the goal of the evaluation. Another guideline is to conduct surveys to identify the requirements for a software visualization because the authors found a disconnect between developers' real-life problems and the visualization approaches developed by researchers.

Popular methods for empirical evaluations are surveys, experiments, and case studies. Surveys are conducted to collect data about the current status of a situation. They are taken from a sample to create generalized conclusions about the whole population. Experiments, on the other hand, allow a high degree of control. To compare outcomes, they involve multiple treatments that are controlled by researchers [WRH+12]. Merino et al. [MGAN18] recommend that the experiment participants include a random sample of a visualization's target audience and a real-world software system to allow for meaningful conclusions. Finally, a case study is conducted to investigate a phenomenon in its real-life context. For that reason, researchers have less control over it than over an experiment. However, case studies allow for a more realistic evaluation in the environment in which a phenomenon occurs and for a deeper analysis [WRH+12].

Merino et al. observed several obstacles for better evaluations. The first is "the lack of ready-to-use evaluation infrastructure" [MGAN18], e.g., a pool of baseline visualization tools against which a new approach can be compared. The second one is, "the lack of benchmarks that ease comparisons against tools" [MGAN18], e.g., quality metrics that can be compared between different visualization approaches.

Modern HCI methods, like NUIs and multimodal interfaces, pose challenges for traditional evaluation approaches. The recognition of natural interaction modes is often error-prone as natural input modes can be ambiguous. Thus, Poppe et al. [PRD07] argue that an assessment of the input reliability of an HCI method must be part of the evaluation, too. Furthermore, the communication lexicon that humans use in natural interaction, i.e., speech, gestures, and gaze movement, is enormous. An evaluation must consider this fact. When multiple modes are combined in a multimodal approach, the recognition of the integrated information must be evaluated as well as the recognition of a single input mode. Additionally, context is essential for the interpretation of human interaction. Therefore, it must also be considered when evaluating modern HCI methods. Finally, Poppe et al. explain that there is a lack of reference tasks on which new HCI approaches can be evaluated [PRD07].

3 Related Work

This chapter presents work that is related to the topic of the thesis. Section 3.1 presents approaches that employ novel human-computer interfaces in the domain of software engineering. Afterward, Section 3.2 introduces work that dealt with the visualization of resilience.

3.1 Application of Novel Human-Computer Interfaces in Software Engineering

Much research has been done on visualizing software architectures in virtual reality. Often, researchers choose metaphors as a representation for software components. For instance, Fittkau et al. [FKH15] propose ExplorViz, which uses a city metaphor to create 3D software visualizations of execution traces and software architectures. The originally web-based tool was extended by a virtual reality component that allowed the exploration of software visualizations in virtual reality through a head-mounted display and gesture-based interaction. The city metaphor for software architectures was first introduced by Wettel and Lanza [WL07] in 2007. They represented software packages as city districts and classes as buildings and explored different city layouts for the visualization. Merino et al.[MBN18] investigated whether the application of immersive augmented reality to 3D software visualizations solved usability issues and increased users' effectiveness. They found out that immersive augmented reality facilitated navigation in 3D visualizations and enhanced the user experience.

Seipel et al. [SSS+19] combined software visualization in augmented reality with a conversational interface that can understand spoken natural language. This allows users to explore 3D software visualizations through gestures and speech simultaneously.

Bieliauskas and Schreiber [BS17] propose interaction with software visualizations through a conversational interface. Their approach had several benefits. They showed that the interface was able to support group conversations with relevant visualizations. Furthermore, the visualizations were more accessible to new team members. They could also ask natural language questions about the visualizations. They claim that their chatbot shows potential for accessing software visualizations more naturally and providing fitting information at the right time.

Züllighoven et al. [ZNR+18] implemented an interactive tool for the planning of the harbor traffic in the port of Hamburg that runs on a multi-touch table. Their solution was able to replace the old paper map-based approach completely and convinced because of its ease of use.

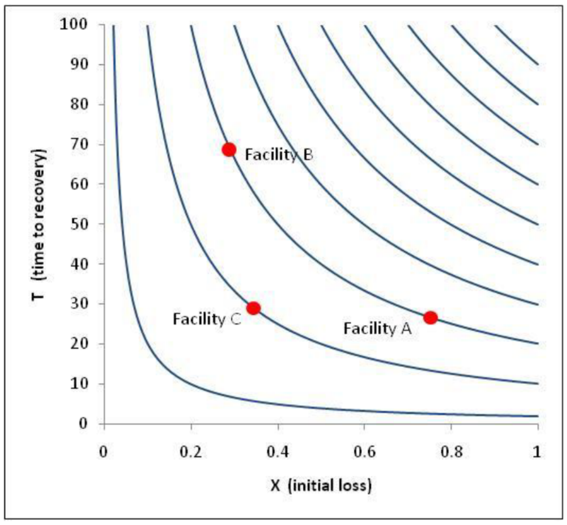


Figure 3.1: Resilience curves visualized as hyperbolas, from [Zob10].

3.2 Resilience Visualization

Zobel [Zob10] extends the resilience triangle from Bruneau et al. [BCE+03] with a new approach for visualizing the relationship between the predicted initial loss that a facility suffers during a disaster and the predicted time it needs for recovery. Both of these factors are important when considering the resilience of a facility. The new visualization is intended to support decision makers in disaster planning in comparing the resilience of different critical facilities. Instead of using the integral above the resilience curve for computing the resilience loss suffered during a disruption, Zobel suggests to approximate the loss of resilience by calculating the area of the triangle formed by the initial drop in quality and the time needed for full recovery. However, with this approach it is possible that different combinations of initial loss and time to recovery result in the same loss of resilience. These combinations represent very different scenarios, thus it is desirable for decision makers to be able to differentiate between them. To make that possible, Zobel first introduces a definition for predicted resilience $R(X,T)$ that depends on the predicted initial loss X, which is measured as a percentage of the ideal quality, and the the predicted time to recovery T. The predicted resilience is the difference of the area determined by the ideal quality and $T^* >= max(T)$ and the area of the resilience triangle, i.e., the loss of resilience ($\frac{XT}{2}$):

$$(3.1)\quad R(X,T) = \frac{T^* - \frac{XT}{2}}{T^*} = 1 - \frac{XT}{2T^*}$$

The difference of the two areas is divided by T^* to get a percentage value. With this formulation of resilience it is possible to represent an instance of the predicted resilience as pair (X,T). For fixed R and T^*, the relationship between X and T is given by:

$$(3.2)\quad XT = (X-0)(T-0) = M, \text{where } M = (2T^*)(1-R)$$

This equation describes a rectangular hyperbola that represents the possible values for X and T that result in the given resilience. Such hyperbolas for different R are illustrated in Figure 3.1. The larger the resilience value, the closer the hyperbola is to the origin. Observations plotted in the lower right

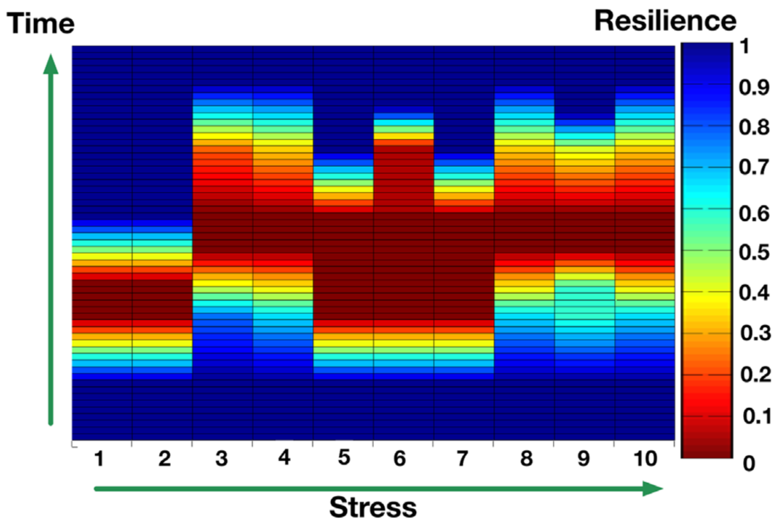


Figure 3.2: A heat map that visualizes the resilience of a system against disruptions with different stress levels, from [DRB16].

portion represent scenarios with high initial loss and a short recovery time, while observations in the upper left portion represent scenarios with small initial loss and a long recovery time. Observations in the middle of a hyperbola are more evenly balanced between the size of the initial loss and the length of the recovery time. Zobel argues that the benefit of this visualization is that it allows to compare the resilience of multiple facilities not only with respect to their overall resilience but also their robustness (size of initial loss) and rapidity (time to recover).

Dessavre et al. [DRB16] propose an extended model for system resilience that introduces stress as a new dimension. The inspiration for this approach came from material resilience, where stress describes how well a material returns to its original shape after having been deformed. They argue that the addition of stress as a dimension of disruptive events enables the comparison of the impacts of different disruptions as well as of the resilience of different systems. In their model, they use either quotient resilience or integral resilience as underlying resilience function. The quotient resilience function represents the proportion of quality of service that has been recovered after a disruption, while the integral resilience function is a ratio of the area under the resilience curve for the experienced quality of service and the area under the ideal quality of service during a given time period. They define a multidimensional resilience function from time, stress, and quality of service to the real numbers that associates the stress of a disruption with the effect it has on the system. Here, the stress function assigns each disruptive event a stress that is described by a real number. The multidimensional resilience function can be used to compute the overall resilience of a system against a set of disruptions. Dessavre et al. also created a heat map visualization that allows for an effortless comparison of the resilience of different systems. The visualization is created by mapping the values of a resilience curve to a color scale. Multiple such heat maps can be arranged in one visualization, as shown in Figure 3.2, to display information about the overall system resilience behavior during a set of disruptions. Furthermore, the authors claim that their model and visualization approach can also be used to appraise the dependency between two systems. To create a heat map that visualizes this estimation, first, the resilience curves of both systems have to be created. Then the values of the dependent system are subtracted from the values of the system that it is dependent on. The obtained values can be visualized in a heat map, to visualize the dependencies between the systems regarding the set of disruptions.

Yin et al. [YDW+19] propose the Microservice Resilience Measurement Model (MRMM) as a quantitative metric to measure resilience in microservice systems. Furthermore, they introduce a model for representing resilience requirements that is based on the MRMM. Resilience requirements consist of (i) a service resilience goal, (ii) disruptions that cause a violation of the goal, and (iii) an implemented resilience mechanisms that mitigate the impact of these disruptions. These requirements are elicitated through an iterative 3-step process. The first step is System Identification in which the system's services and service dependencies are identified. In the second step, resilience goals are set by creating a performance benchmark and linking them with goals. Finally, in the last Disruption Analysis step, service degradations in the system are analyzed, and corresponding resilience mechanisms are established. While Yin et al.'s work makes it possible to elicit and model resilience requirements, they highlight that manually generating resilience requirements for complex microservice systems is challenging. Also, they do not employ modern HCI methods to make the specification of requirements more accessible.

4 Expert Interviews about Transient Behavior

In addition to reviewing the state of the art in research on transient behavior in software systems, we sought to gain a better understanding of the problems and challenges that software developers, architects, and operators face with elicitating and comprehending non-functional requirements for transient behavior in practice. To this end, we were also interested in collecting requirements that a solution, which employs novel HCI methods to support the mentioned stakeholders in this context, would need to fulfill to be applicable in the industry.

4.1 Goals

The purpose of the interviews was to explore how developers, architects, and operators handle transient behavior in practice. We were especially interested in learning whether and how they specify transient behavior in non-functional requirements and which challenges they face in this task. Furthermore, we sought to learn how these requirements are later verified and which problems these stakeholders have when elicitating and analyzing the transient behavior of their software systems. Finally, we were also interested in analyzing the tools and methods that they use for these tasks and collecting requirements they have for a new envisioned solution that employs novel human-computer interfaces to support them.

4.2 Method

To obtain the desired insights, we interviewed software architects from five different companies that are all in the IT consulting business. We selected candidates that already accumulated a considerable amount of experience in this field and that work regularly on projects that involve a cloud and microservice environment. As such, they were in an ideal position to give us valuable information about the topics on which we interviewed them. We contacted seven software architects and invited each of them to an interview. Five out of those seven accepted the invitation. The interviews were held in 2020 in a semi-structured manner in the form of video calls, each lasting around one hour. The questions of the interview were split into four categories of (i) general questions about the interviewees' background, (ii) their familiarity and experience with transient behavior, (iii) the specification and comprehension of transient behavior, and (iv) tools and methods for the specification and comprehension of transient behavior and the interviewees' requirements for such. We also created five hypotheses, formulated based on our survey of the research literature, which we wanted to confirm or reject based on the interviews. Figure 4.1 gives an overview over the design of the interviews and shows which hypotheses were created for which category of questions. The hypotheses are presented in Section 4.2.1 and the questions are listed in Section 4.2.2. The question categories were time-boxed to ensure that each category is covered sufficiently in the interview. The

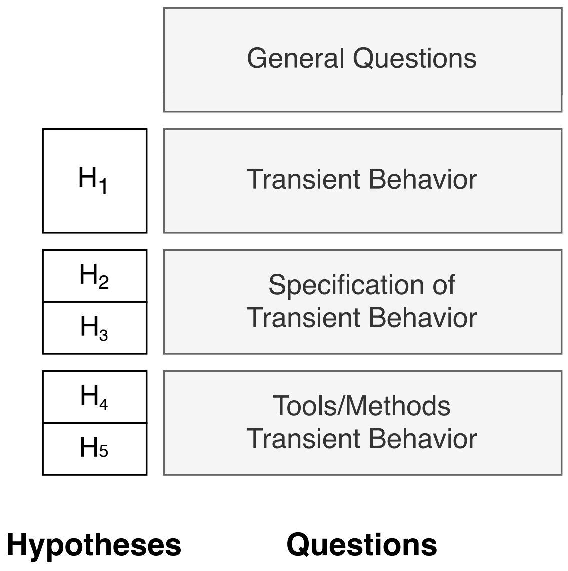

General Questions
H1
Transient Behavior
H2
H3
Specification of Transient Behavior
H4
H5
Tools/Methods Transient Behavior
Hypotheses
Questions

Figure 4.1: Interview Design

semi-structured nature of the interviews ensure that the interviewees' statements can be compared while allowing us freedom to go into more details based on the answers of participants. We think that this design promoted a more natural discourse in which interviewees can express their thoughts freely. The audio was recorded and a verbatim transcript of each interview produced, which later served as the basis for the analysis. Each interviewee got a short explanation of the concept of transient behavior or time-dependent behavior in software systems and its causes if he was not yet familiar with it.

To analyze the answers in the transcripts, we first classified them into the categories presented in Section 4.3. Then, the answers in each category were summarized for each interviewee, comprising their main points. For each category, the summaries of all five candidates were then compiled into one report that gives a reflection of the interviewees' general opinion on a topic.

4.2.1 Hypotheses

Before we conducted the interviews, we created five hypotheses about how engineers think about and handle transient behavior and sought to confirm or reject them through the interviews. The hypotheses are:

H_1**:** Engineers are familiar with the concept of transient behavior in software.

H_2**:** Engineers think that the transient behavior of software should be specified in non-functional requirements.

H_3**:** Engineers specify the transient behavior of their software systems in non-functional requirements.

H_4**:** Engineers are satisfied with existing methods and processes for analyzing and specifying non-functional requirements for the transient behavior of software.

H_5: Engineers collaborate when they are analyzing or specifying non-functional requirements for the transient behavior of software.

4.2.2 Questions

The following questions were sent to the interviewees in advance of the interviews. If the interviewee answered in Question 3.2 that transient behavior is not specified in non-functional requirements in their organization, Questions 3.3 to 3.6, which enquire about how these requirements are specified and verified, were skipped in the interview. Otherwise, Questions 3.7 and 3.8, which enquire why transient behavior is not specified in non-functional requirements, were skipped.

1. **General Questions**
   - 1.1. What is your role in your organization and how many years of experience do you have in that role?
   - 1.2. To what extent do you work on microservices and on non-functional requirements of software systems?
   - 1.3. Did your team specify non-functional requirements for projects you have worked on in the past?
2. **Transient Behavior**
   - 2.1. Are you familiar with the concept of transient behavior (time-dependent behavior) in software systems and its causes?
   - 2.2. Did you previously observe transient behavior in one of your software systems? If yes, are your systems often in a state of transient behavior? What are, in your opinion, the main causes for transient behavior in your system?
   - 2.3. In your experience, are the developers in your organization able to identify and analyze the transient behavior in your software systems?
   - 2.4. In your opinion, what are the main challenges for developers and other stakeholders in analyzing transient behavior?
3. **Specification of Transient Behavior**
   - 3.1. Do you think transient behavior should be specified in non-functional requirements? If so, how do you think such requirements could be captured?
   - 3.2. Do you already specify transient behavior in non-functional requirements in your organization?
   - 3.3. How do you specify non-functional requirements for the transient behavior of your software systems?
   - 3.4. What challenges do you face when you are specifying the non-functional requirements for the transient behavior of your software systems?
   - 3.5. How do you verify if the non-functional requirements for the transient behavior of your software systems are satisfied?

3.6. What challenges do you face when you verify the non-functional requirements for the transient behavior of your systems?

3.7. For which reasons do you not specify the transient behavior of your software systems in non-functional requirements?

3.8. Do you specify the transient behavior of your software systems in another way than non-functional requirements? If yes, how?

4. Tools/Methods for the Specification of Transient Behavior

4.1. How satisfied are you with existing tools/methods that support developers in comprehending transient behavior and specifying it in non-functional requirements? How could they be improved?

4.2. In your opinion, in which ways could the employment of novel human-computer interfaces (VR/AR, multitouch tables, voice assistants, chatbots, etc.) make it easier to analyze transient behavior and to specify it in non-functional requirements?

4.3. In your opinion, which interface or combination of interfaces would be best suited to support developers? And, why?

4.4. Which requirements would a solution that uses novel human-computer interfaces to support developers need to fulfill for you to use it?

4.5. How important is it that developers can collaborate when they analyze a system's transient behavior and specify it in non-functional requirements? And, why?

4.6. In which ways does your organization already utilize hardware like VR/AR glasses or multitouch tables to support its developers in their work? If it does not employ such hardware, do you think your organization would be willing to acquire new hardware to support its developers in analyzing your systems' transient behavior and specifying it in non-functional requirements?

4.3 Results

This section describes the results of the conducted interviews. To preserve the privacy of the identity of the interviewees, the names that are used below are fictional. First, we give an overview of the interviewees backgrounds and their practices for elicitating non-functional requirements. Then, we present their answers that regard the analysis and specification of transient behavior and finish with an overview of their answers about currently employed tools and methods, requirements for an envisioned solution, and their views on novel human-computer interfaces.

Interviewee	Position	Field
Torsten	Software architect	Tax consulting software
Michael	Managing director/principal consultant	IT consulting
Wolfgang	Chief technical officer/software architect	IT consulting
Sebastian	Head of cloud innovation/software engineer	IT consulting
Andreas	Senior managing consultant	IT consulting

Table 4.1: Background information of the five interviewees.

4.3.1 Background of Interviewees

All five interviewees hold senior positions at different companies in the software consulting industry that mainly work on enterprise applications. All interviewees are experienced software architects and have a deep understanding of microservice architectures. Table 4.1 presents an overview of the background of the five interviewees.

Torsten holds a PhD in computer science. His thesis focused on software performance engineering. He has been working as a software architect for one year at a software and IT consulting company whose main focus is on the tax consultant market. At the moment he is mainly concerned with software performance aspects, microservice systems, and maintainability. He is also directly involved with the elicitation and documentation of non-functional requirements in his projects.

Michael is managing director and principal consultant at a software consulting company. He is involved in customer projects, key account management, workshop organization, and architecture consultancy. His company has been actively advocating the microservice architectural style for five to six years.

Wolfgang is chief technical officer at an IT consulting company and responsible for its technological and methodical strategy. He is active as a software architect and develops solutions tailored to the needs of his customers. He has been working with service-oriented architectures in the last two decades and now works with microservices since 2014. Furthermore, he is directly involved in the elicitation and documentation of non-functional requirements in his projects.

Sebastian was recently appointed as the head of cloud innovation at an IT consulting company. He also works as a software engineer, as he has been doing since the last four years. His company develops and maintains multiple microservice systems.

Andreas has been a senior managing consultant at an IT consulting company for four years. Before that, he worked for almost 17 years at one of the worldwide leading hardware, software, and IT consulting companies. He is mostly concerned with cloud-native software development. He mainly works in customer projects and also gives workshops and lectures on this topic. Furthermore, Andreas is the organizer of a local meetup for a popular cloud platform. His experience with microservices is mostly limited to transforming monolithic legacy applications into microservice applications. He highlighted during the interview that he is not directly involved with the elicitation and documentation of non-functional requirements in his projects.

4.3.2 Specification of Non-functional Requirements

All five interviewees stated that they specify some form of non-functional requirements for their projects. However, the thoroughness with which requirements are captured at the different companies varies. Torsten and Michael explained that they usually capture non-functional requirements in formal quality scenarios. The three other interviewees explained that they specify non-functional requirements informally, citing the complexity of formal requirements and scenarios as the reason. Sebastian expressed that they document non-functional requirements in user stories, together with functional requirements.

They all described that it is a challenge to obtain concrete non-functional requirements from their customers. A lack of general understanding of these requirements and their importance among business stakeholders were generally stated as the main obstacles for a more thorough and precise specification of non-functional requirements. Wolfgang and Andreas pointed out beyond this that they are often left to guess the non-functional requirements that their customers have for a system. The only way to validate if these requirements meet the customer's expectations is through frequent testing.

4.3.3 Familiarity with the Concept of Transient Behavior

Michael, Wolfgang, and Andreas were not familiar with the concept of transient behavior in software systems and its causes. Sebastian stated that he was already familiar with the basic concept but enquired about the need for a specification in non-functional requirements thereof. Torsten said that he was already familiar with the concept and its causes. Besides him, the other interviewees asked for a short explanation of the concept. We observe that transient behavior is not yet a generally known concept in the software industry.

4.3.4 Causes of Transient Behavior

All participants identified deployments as the biggest cause of transient behavior in their projects. Torsten, Sebastian, and Andreas expressed that failures and resilience mechanisms are, in their opinion, potentially the second biggest cause for transient behavior. While this case happens less frequently than deployments, they estimated that the consequences on the quality of service of the system might be more critical. However, as Torsten highlighted, the transient behavior that is introduced in a system by resilience mechanisms is difficult to detect and trace. None of the interviewees claimed that they analyze behavior introduced this way in detail.

Michael and Andreas explained that they minimize transient behavior caused by deployments by employing continuous deployment methods. Although frequently deploying small code changes into production means that the system is regularly in a state of transient behavior, this behavior usually has only a small impact on the quality of service of the system. Big traditional deployments that are carried out every four to six months on the other hand can cause transient behavior that substantially impacts the quality of service and potentially lasts for a much longer time. For example, Michael recounted instances in which the caused transient behavior lasted for weeks.

Andreas pointed out that the use of modern cloud and containerization technologies combined with new architectural styles, like microservices or serverless functions, minimize both the duration and the magnitude of transient behavior, which is an improvement over previous technologies. It is for this reason that he thinks that transient behavior is not as big a problem for developers, architects, and stakeholders anymore as it used to be in the past. Wolfgang remarked that the underlying cause for all undesired transient behavior is that engineers often ignore the fact that most actions take a certain amount of time before they are completed and instead assume that they are completed instantaneously.

4.3.5 Identifying, Understanding, and Analyzing Transient Behavior

Torsten and Sebastian observed that in their experience the engineers at their company are able to identify, understand, and analyze the transient behavior in their software systems. Andreas stated that he thinks the operators who are tasked with analyzing and understanding the current transient behavior of a system are able to do so. Wolfgang, on the other hand, estimated, from his experience, that only around 20% of engineers would be able to complete the task with and that only 1% of engineers are actually analyzing the transient behavior of their software systems. Thus it can be assumed that the skills and abilities of engineers in this domain vary between companies.

Current Situation

All interviewees mentioned that they are collecting data about the system behavior and its impact on the system during runtime through the means of monitoring and log aggregators. It seems like a large body of data is already available that can be analyzed to understand the various transient behaviors that occur in a system, especially their causes and impacts on the quality of service of the system. However, the interviewees also described that the amount of collected monitoring data is so large that it becomes overwhelming to examine it closely and carry out the difficult analyses that would be needed to fully understand the transient behavior. Usually it is not considered as critical or at least critical enough to warrant the required effort. Therefore, such analysis does not take place often. Torsten was the only one who claimed that his company actively attempts to understand and analyze the transient behavior in their software systems.

Torsten, Wolfgang, and Andreas stated that visualizations reduce some of the complexity when it comes to analyzing monitoring data. However, Torsten said that the integration of such visualizations into their application monitoring tools is still in a prototypical state. Wolfgang expects that it is a challenging task to develop visualizations that support engineers in understanding transient behavior in complex scenarios and systems, too.

Sebastian explained that their analysis of monitoring data is error-driven. This means they only look at the monitoring data after a failure occurred that significantly impacted the system. The negative impact it has on the system are accepted by both engineers and customers.

Torsten, Michael, Sebastian, and Andreas said that such behavior should ideally already be tested during development to predict and understand its consequences. However, Torsten, Michael, and Sebastian also pointed out that such tests require complicated setups and are time-consuming. Thus, such behavior is only sparsely covered by test scenarios. Michael explained that they often only discuss certain scenarios in theory, for the same reason. They try to reason about if the system would be able to meet the specified quality requirements under these scenarios. Andreas thinks that test-driven development and the scaling and monitoring capacities of modern cloud platforms can be used to keep the impact of transient behavior under control. Small services that can be restarted quickly and straightforwardly mean that stopping and restarting a service that displays undesired behavior is a simpler solution than attempting to find the underlying cause of the behavior.

When Torsten and Sebastian walked us through their process for analyzing the monitoring data after a failure occurred, they mentioned that the first step is usually to exchange ideas with other engineers, highlighting that collaboration is critical in this task.

Challenges

The interviewees mentioned that the most common challenge for analyzing and understanding transient behavior is that transient behavior is not perceived as critical enough to merit analyzing a large amount of monitoring data to understand its causes and consequences. Engineers have too many other responsibilities that have a higher priority and will be taken care of first before transient behavior is even considered. It seems like better tools are needed to reduce the effort that this task takes to make it worthwhile for engineers to undertake it. Indeed, Torsten stated that the biggest challenge that they face is inefficient tooling, which makes it too time-consuming to analyze transient behavior. Especially lacking filter configurations for monitoring notifications and alerts mean that relevant events are either not reported or that they are overlooked between a myriad of notifications on irrelevant events. The abundance of notifications also leads to engineers eventually outright ignoring reports. To improve this situation, the monitoring tool should only send notifications about events that are both critical and in the recipient's area of responsibility.

Another challenge that Wolfgang observes is, as mentioned earlier, to create visualizations of the monitoring data that are still helpful and easy to understand for complex systems and scenarios. Sebastian also remarked that it is challenging to reproduce and test many scenarios that cause transient behavior like deployments. Michael thinks this is complicated by the fact that some transient behaviors cannot be located with the help of monitoring tools.

Wolfgang thinks the main challenge in analyzing transient behavior is that parallel transient behaviors affect each other and form interrelationships between them, which are too complex to be easily understood by the human mind.

4.3.6 Specification of Transient Behavior

At the beginning of this part of the interview, we asked the interviewees if they think that transient behavior should be specified in non-functional requirements. Only Torsten agreed, in principle. Michael, Wolfgang, and Andreas on the other hand think that it is not necessary to explicitly specify requirements for this kind of software behavior in most projects. However, Michael and Wolfgang also added that it depends on the criticality of the respective system. For some projects, it might pay off to conduct a risk assessment of the expected transient behavior concerning its occurrence probability and potential impact on the system. If such an assessment reveals that the system could be critically impacted by transient behavior, this behavior should be specified. Nonetheless, for the enterprise systems that they are developing and operating, they do not think such a specification necessary.

Current Situation

Torsten claimed that they already specify some aspects of transient behavior in quality scenarios, the same way as they capture other non-functional requirements. These formal scenarios are usually created by the software architect or the test engineers. The other four interviewees stated that they do not explicitly specify non-functional requirements to capture transient behavior. However, they do specify other non-functional requirements that sometimes implicitly also have an influence on the transient behavior of a system. Michael, for example, said that they define informal quality scenarios to capture non-functional requirements that might also affect which transient behavior is acceptable. Sebastian also explained that they define requirements like the maximum number of simultaneous users that a system must be able to handle, but they do not specify quality requirements that the system must fulfill while scaling service instances in or out to comply with the other requirement. Andreas thinks that some requirements for transient behavior might be more interesting than others. His example was that startup time and scaling behavior is an important factor of serverless functions which get started and stopped frequently. The behavior during deployment on the other hand is less critical in his opinion because modern platforms and methods already reduce the impact and extend of transient behavior during that phase.

Challenges

The biggest challenge by far for specifying transient behavior that was named by all five interviewees is that customers and business stakeholders are not aware of it and lack the knowledge to understand what it is, why it should be specified, and what their requirements for it are. Thus, it is difficult to find acceptance for the topic of transient behavior among these stakeholders. This corresponds to what they already said about non-functional requirements in general towards the beginning of the interviews. Torsten explained that non-technical stakeholders often have difficulties with defining concrete numbers for non-functional requirements in general and transient behavior. Furthermore, they have problems assessing the trade-off between the development and operational costs of a better solution and the impact of unspecified transient behavior. Architects and engineers often have to infer the requirements of their customers themselves.

Another challenge that Torsten and Sebastian described is that it is difficult to define a response measure for quality scenarios that involve transient behavior, i.e., how the system should react if a certain behavior occurs. Here, input from the business stakeholders is needed again, which is challenging because of the issues mentioned previously.

Furthermore, Michael, Wolfgang, and Andreas perceive other non-functional requirements as more important than those that specify transient behavior. Since it is already difficult to elicit these from their customers, they would rather spend their effort on them than on additional requirements for transient behavior. They simply have other problems that have a higher priority than the specification of transient behavior and are willing to accept the consequences that unspecified transient behavior causes.

Finally, Torsten brought up that it can be difficult to identify quality scenarios for transient behavior and to be confident that a scenario captures all aspects of a specific behavior. The fact that it is challenging to trace the causes of transient behavior impedes its formal specification.

4.3.7 Tools and Methods

We asked the interviewees how satisfied they are with the tools and methods that they use for both analyzing and specifying transient behavior and how these tools could be further improved. Afterwards we also inquired about the requirements that a new solution for this tasks would have to meet that employs novel human-computer interfaces to support engineers.

Existing Tools

All five described that they take advantage of application performance monitoring tools to collect data about their software systems during the runtime. Sometimes they use proprietary solutions, other times the toolset is determined by the customer. In general, none of them has a fixed set of tools that they use in every project. To emphasize this, Andreas explained that they constantly search and try new tools to be able to consult their customers with the latest technologies and methods. These monitoring tools are usually configured to issue alerts when unexpected behavior arises to notify the responsible operator. Torsten also revealed that they use log aggregators and data analytics engines like Elasticsearch[1] to monitor the behavior of an application. In his opinion, these tools lack adequate filters to notify engineers only about events that are relevant to them.

Torsten, Michael, Wolfgang, and Sebastian elaborated about their method for documenting non-functional requirements. All of them use standard tools that are simple to use to capture their requirements, like different text processing applications and spreadsheets. Torsten stated that the important factor in the specification of non-functional requirements is good communication with non-technical stakeholders, not tooling. Before the COVID-19 pandemic in 2020, they simply documented these quality requirements with post-it notes. Sebastian said they collect non-functional requirements together with functional requirements in user stories that are tracked in a ticket management tool. However, this makes it difficult to find a specific requirement later on because they are collected in an unstructured fashion and their tool is lacking search functionality. The interviews exposed that an important aspect of the specification is that it must be simple to capture requirements and to change them later on.

An interesting approach was presented by Michael, who mentioned a method based on EventStorming[2] that was developed by one of his co-workers for the collaborative elicitation of quality requirements.

Requirements for a Potential Solution

A requirement that all interviewees had for a potential solution that takes advantage of novel human-computer interfaces is that it must allow collaboration. For once, virtual collaboration in the form that multiple users can access and edit the same model or file at the same time is a must. When multiple engineers are working together in person, it would also be helpful if there is a way to collaborate in that scenario, too. Especially, since the importance of communication and collaboration have been repeatedly highlighted during the interviews.

[1]https://www.elastic.co/enterprise-search
[2]https://www.eventstorming.com/

Michael established that a solution must be practice-oriented if it should be adapted in the industry. By that, he means that the tool and the employed interfaces should be easy to learn and to use. Wolfgang stressed this point by remarking that a solution approach should not worsen the current situation by adding additional complexity to the tasks. Furthermore, it must be possible to integrate a solution into the existing processes that surround the concerned activities. Otherwise, it would find no adoption in practice. Since a big problem in dealing with transient behavior seems to be the lack of acceptance that the topic has from business stakeholders, a solution would have to create value that is directly visible to them. Finally, most projects are completely managed in version control systems like Git. Any solution would have to support this by making sure that all created artifacts can be managed there without issues as well.

For Torsten, a solution must improve the problems that he sees with the current tools that are in use. It needs to include filtering of data and incidents to present only relevant data to the engineers that use it. Moreover, the representation and visualization of data must promote easy and fast identification of relevant behavior, its causes, and its impacts on the system. Sebastian would like a filtering or search function not for monitoring data but for the requirements that specify transient behavior. He also thinks it would be helpful if a solution supports engineers in the creation of scenarios for automatic tests and verification of these requirements.

Wolfgang already mentioned that a solution would need to include visualizations that are still helpful for complex scenarios and systems. Andreas especially wants visualizations for the duration of transient behavior, a service's or the system's recovery time from it, and the services that are affected by it.

Multiple interviewees remarked that a solution that supports stakeholders in the specification of transient behavior must also be easy to use for non-technical stakeholders that are involved in this process. While engineers also expect rich options for configuring a solution, especially for analyzing transient behavior, usability should be the focus for a solution that supports various stakeholders in the specification of transient behavior.

Thoughts on Novel Human-Computer Interfaces

Michael, Wolfgang, and Andreas worry that the utilization of novel human-computer interfaces for a solution would not help to increase its usability and effectivity. Instead, in their opinion, the problems of modeling and visualizing transient behavior need to be solved first, as there is still a lack of capable solutions in this regard. Michael is concerned that the utilization of such HCI methods would make a solution less practical and harder to adapt in the industry.

Torsten, Michael, Wolfgang, and Sebastian are wary of a solution that employs VR or AR because the technology is not yet prevalent. Furthermore, they are afraid that the technology could restrict communication between stakeholders, and there is a context switch when putting on a head-mounted display to start a VR application. Wolfgang, who is aware that it is difficult to create useful visualizations for complex transient behavior, is not convinced that this problem would be solved by the application of VR or AR.

Whereas Sebastian also has concerns about VR in its current state, he thinks that a solution that employs this paradigm could have potential in the future, when VR is more broadly adopted and the flaws of the technology are alleviated. The technology that has the most potential for a solution right now is, in his opinion, chatbots. He argues that bots are already commonly used in software engineering, for example, for notifications about failures or other important system events. For him, it would be a useful extension of these bots if they allow him to interact with the information that they present, e.g., restarting a service or the presentation of suggestions for possible next steps from the chatbot.

Wolfgang thinks that natural user interfaces and multimodal user interfaces are the most promising HCI approaches. In his opinion, these user interface paradigms will see broad adoption in the future, as they allow more natural interaction, similar to how humans communicate with each other. For this reason, they could eventually be used to facilitate the handling of such complex topics as transient behavior. However, he fears that many engineers would reject such a solution at the moment until these methods are more commonly employed.

Finally, Sebastian and Wolfgang think that speech as a mode is less suited for such a use case because it has a low information density and the user experience with voice assistants is still lacking, at least for now.

4.4 Discussion

In this section, we discuss if the hypotheses presented in Section 4.2.1 can be rejected or confirmed on the basis of the results of the interviews. The sample size of our interviews was neither big enough to make statements about the general population of engineers nor were the means with which we collected and analyzed the results scientific; however, the insights that we gained through the interviews helped us to have an impression of the requirements that software engineers have towards transient behavior and knowledge of how they currently handle it. For the same reasons we are only discussing the hypotheses in this section, without formally rejecting or accepting them.

While some interviewees, like Torsten, were more familiar with the concept of transient behavior than others, the concept was mostly new to them. When it was explained to them, they quickly grasped it. However, most of them have not heard the term “transient behavior” before we invited them to the interview. Therefore, it seems fairly safe to say that H_1 should be rejected.

There is no clear answer to the second hypothesis. While Torsten thinks that transient behavior should generally be specified, most of the other interviewees made clear that this is often not necessary. However, they stated that for critical systems, which could severely be impacted by transient behavior, a specification thereof might make sense. Therefore, H_2 was neither confirmed nor can it be rejected , as this question is too complex to be covered by a single hypothesis that does not distinguish between different kinds of systems.

The division between the interviewees seems similar concerning the third hypothesis. Torsten claimed that they attempt to specify transient behavior in non-functional requirements in their projects. The other four interviewees stated that they do not capture non-functional requirements for transient behavior explicitly. While several of them mentioned that some of the non-functional requirements that they do specify also influence transient behavior, these requirements are not created with transient behavior in mind. For this reason, it seems like H_3 should mostly be rejected.

There is again no clear answer to hypothesis four. Torsten stated that tooling is one of their biggest problems when it comes to analyzing transient behavior. Similarly, Sebastian expressed that reproducing and testing scenarios that involve transient behavior is challenging. On the other hand, Wolfgang thinks that communication with non-technical stakeholders is a greater challenge while tooling plays less of a role. He claimed to be satisfied with the current tools and methods that they employ.

The situation is different with hypothesis five. All interviewees agreed that collaboration is important when dealing with the different aspects of transient behavior. It especially plays a big part in the specification because both non-technical and technical stakeholders have to work together. However, engineers also collaborate when they analyze transient behavior and its impacts on a system. Torsten and Sebastian explained that exchanging ideas with other engineers is often the first step when unexpected system behavior occurs. Therefore, H_5 was confirmed by our interviews.

5 Concept

This chapter conceptualizes an approach to support software architects and DevOps engineers both in the creation and verification of specifications for transient behavior. The concept is based on the background knowledge presented in Chapter 2, relies on related work from Chapter 3, and uses the findings of the expert interviews discussed in Chapter 4. Special attention was given to the usability and practicability of the solution. One insight from the interviews was that technologies that utilize novel human-computer interfaces, such as virtual and augmented reality, are not commonly employed in software engineering. Furthermore, the barrier for the purchase of specialized hardware is high. For this reason, the proposed approach does not take advantage of any technologies that require specialized hardware but considers the possibility to include them in a later iteration of the soluton.

In the following, Section 5.1 presents the functional requirements for the proposed solution based on five use cases. Subsequently, Section 5.2 describes various visualizations that have been designed to facilitate the specification and analysis of the transient behavior exhibited by microservices. The section also discusses interactions to facilitate data exploration. Finally, Section 5.3 discusses how the defined use cases can be satisfied with a conversational interface that integrates the visualizations and a chatbot to provide architects and engineers with the means to specify and assess transient behavior in microservice systems.

5.1 Use Cases

This section proposes five use cases for a software solution intended to support software architects and DevOps engineers in the specification and analysis of transient behavior in microservice systems. The use cases are motivated on the knowledge of the state of the art presented in Chapter 2 and the findings of the expert interviews discussed in Chapter 4. The use cases are defined using the Goal Question Metric (GQM) approach by Basili et al. [VBCR02].

In practice, DevOps engineers regularly conduct postmortem analyses of incidents that affected the performance or reliability of their software systems [BJPM16; Mon15]. During such an analysis, engineers document the root cause of the incident, its impact, the actions taken to resolve it, and the measures put into place to prevent similar incidents in the future. For this reason, it is reasonable to assume that the following use cases are realistic for a solution that regards itself with the specification and analysis of transient behavior.

The methodology used to capture the use cases is explained in Section 5.1.1, followed by the individual use cases.

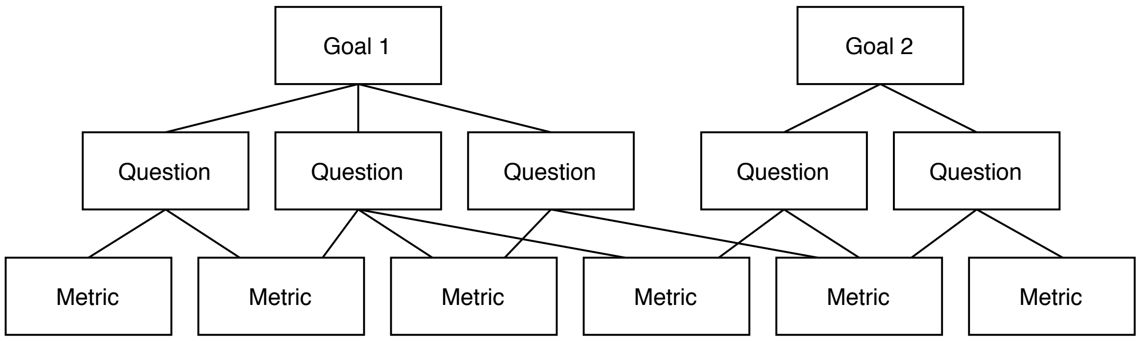


Figure 5.1: Structural hierarchy of a GQM model.

5.1.1 Goal Question Metric Method

The GQM method is a popular approach for the creation of measurement models in software engineering. The model has three levels [VBCR02]:

Conceptual level (Goal): A goal is defined for an object with respect to different quality models and different viewpoints.

Operational level (Question): Questions attempt to characterize the object of measurement with respect to one quality issue and from a selected viewpoint.

Quantitative level (Metric): Metrics are associated with each question in order to answer it in a quantitative way.

These three levels have a hierarchical structure, as illustrated in Figure 5.1. Each GQM model has a goal at the top that defines the purpose of the measurement, the issue that is measured, the object that is measured, and the viewpoint from which the measurement is taken. The goal is refined into multiple questions that describe the major factors of the issue. Subsequently, metrics are defined that describe how a question can be answered. They can be objective or subjective, and each metric can be used to answer multiple questions. Different GQM models can also have various questions or metrics in common.

Whereas the GQM approach has been developed to facilitate the creation of quality models, we are using it to define use cases and describe each use case's components. Goal-based requirements engineering [Van01] follows a similar approach of using goals to specify requirements. However, the use of the GQM model is not generally employed in that context. We define use cases as goals in the *purpose*, *issue*, *object*, and *viewpoint* notation. Each use case is dissected into numerous questions that represent the concerns related to it. The metrics defined for each question detail how they can be answered and specify which information is needed to satisfy the use case.

5.1.2 UC 1: Specification of Acceptable Transient Behavior

The first use case's goal is the specification of the acceptable transient behavior of a service or service instance from the software architect's viewpoint. The GQM model of the use case is presented in Table 5.1.

Goal	Purpose	Specify
	Issue	the acceptable transient behavior
	Object	of a service/service instance
	Viewpoint	from the software architect's viewpoint
Question	Q1	Which causes potentially introduce transient behavior into the service?
Metrics	M1	Potential causes of transient behavior
Question	Q2	How high is the expected quality of service during normal execution?
Metrics	M2	Quality of service function
	M3	Expected value for quality of service function
Question	Q3	How high is the initial loss acceptable for transient behavior?
Metrics	M3	Acceptable initial loss (%)
Question	Q4	What is the maximum duration for transient behavior?
Metrics	M4	Maximum duration (s)
Question	Q5	What is the maximum loss of resilience still acceptable during transient behavior?
Metrics	M5	$max(\int_{t_0}^{t_1} [Q_e(t) - Q_a(t)]\, dt)$

Table 5.1: GQM model for UC 1.

Transient behavior is a complex concept. The findings of Chapter 4 showed that non-functional requirements for transient behavior are usually not captured in most software projects. This has different reasons. The first is a lack of knowledge of this concept, especially among business stakeholders. The second is the low priority given to transient behavior and its specification. Coupled with the complex nature of transient behavior and the effort required to specify requirements for it, stakeholders do not see it as worthwhile to devote much effort to this task when it could be spent on more urgent problems instead.

Facilitating the specification through modern human-computer interfaces and suitable visualizations might reduce the effort to a level where this practice becomes more heavily adopted throughout the industry. The first question that concerns this use case is: *Which causes potentially introduce transient behavior into the service?* There might be different expectations for transient behavior introduced by a failure than for behavior caused by the deployment of a new service version. This should be taken into account when specifying transient behavior. The metric that answers this question is the cause of the transient behavior that we want to specify.

The next question that needs to be answered regarding this use case is: *How high is the expected quality of service during normal execution?* To specify which degradation of the quality of service is still acceptable, we need to know the baseline quality of service expected during regular operation. The first metric used to answer this question is a function of the quality of service. Quality of service is a utility function that maps specific performance metrics to a quality value. The function can vary depending on which performance aspects of the system are of interest to the software architect or DevOps engineer. For this reason, a specification of transient behavior should define how the quality of service function is used to compute the expected quality value. The expected value for the quality of service under regular operation is the second metric (M2) belonging to this question.

Goal	Purpose	Verify
	Issue	the compliance to the requirements for transient behavior
	Object	of a service/service instance
	Viewpoint	from the DevOps and test engineer's viewpoint
Question	Q1	What are the specifications for the transient behavior of this service?
Metrics	M1	Quality of service function
	M2	Cause of transient behavior
	M3	Maximum initial loss (%)
	M4	Maximum duration (s)
	M5	Maximum loss of resilience during transient behavior
Question	Q2	Has the transient behavior within a specified period been within the specification?
Metrics	M2	Cause of transient behavior
	M6	Difference initial loss and specified maximum initial loss
	M7	Difference duration and specified maximum duration
	M8	Difference loss of resilience and maximum loss of resilience during transient behavior

Table 5.2: GQM model for UC 2.

Having defined the expected quality of service and the cause for the transient behavior that we want to specify, the next question is how to specify the transient behavior itself. For this, we adopt the approximation of the resilience triangle presented by Zobel [Zob10], that is presented in Chapter 3, and use it to characterize transient behavior. The initial loss caused by an event and the time to recovery of the system create a triangle. Zobel argues that this triangle's area can be used to approximate the loss of resilience caused by an event. Based on this assumption, we use the initial loss, the time to recovery, and the resilience loss as the three dimensions to define the transient behavior introduced by a cause of transient behavior into a service. Thus, the three remaining questions for this use case are: *How high is the initial loss acceptable for transient behavior? What is the maximum duration (time to recovery) for transient behavior? What is the maximum loss of resilience that is still acceptable during transient behavior?* The respective metrics are the acceptable initial loss in percent, the maximum duration in seconds, and the loss of resilience given by Equation (5.1). In it, t_0 is the start of the event, t_1 is the time at which the system has completely recovered, $Q_e(t)$ is the expected quality at time t, and $Q_a(t)$ is the actual quality at time t.

$$(5.1) \quad R = \int_{t_0}^{t_1} [Q_e(t) - Q_a(t)]\, dt$$

The five metrics presented for this use case can be used to specify the transient behavior that is acceptable for a service.

5.1.3 UC 2: Verification of Compliance to Specifications

A specification is only of value if its fulfillment can be verified. For this reason, the second use case's goal is the verification of the compliance to the requirements for transient behavior of a service or service instance from the DevOps and test engineer's viewpoint. Table 5.2 presents this use case's GQM model.

The first thing needed to reach this goal is the transient behavior specification for this service. As defined in Section 5.1.2, the metrics that define such a specification are a quality of service function, a transient behavior cause, the accepted maximum initial loss, the accepted maximum duration, and the maximum loss of resilience during transient behavior. To determine whether transient behavior is complying with this specification, the two of them must be compared. Thus, the answer to the question *Has the transient behavior within a specific period been within the specification?* comprises four metrics. The first is the cause of the actual transient behavior of the service. The second is the difference between the actual initial loss and the specified accepted maximum initial loss. If the difference is greater than zero, this dimension of the transient behavior specification is violated as the actual initial loss exceeds the specified initial loss. The next metric is the difference between the transient behavior's actual duration and the specified maximum duration that is still accepted. Again, if the difference is greater than zero, the transient behavior's duration does not comply with the specification. The final metric is the difference between the actual and the specified loss of resilience while the transient behavior lasts. As in the other cases, the specification is not met if the difference is greater than zero.

Since we specify transient behavior through multiple aspects, the question of whether a service complies with its specification is not trivial. The specification may be only partially violated by a behavior, e.g., if two dimensions are satisfied while the third is not. Whether this constitutes a violation of the entire specification should be defined at the same time as the specification itself.

To verify the compliance of transient behavior with a recorded specification, the DevOps engineer first needs to find and quantify occurrences of transient behavior. Therefore, these challenges are the next two use cases presented in this section.

5.1.4 UC 3: Identifying occurrences of Transient Behavior

The goal of this use case is the identification of transient behavior occurrences in a service or service instance from the DevOps engineer's viewpoint. Its GQM model is shown in Table 5.3.

The first question regarding this use case is: *What are the services that comprise the system architecture?* We need to know the services that comprise the system, how many instances they run, and which API endpoints they have to investigate them for transient behavior. The metrics that answer this question are the service names, the number of instances of each service, and their endpoints' names. The second concern is the definition of the quality of service, given by the quality of service function. The next question is: *At what time does a change introduce transient behavior into the service?* Since we are looking for transient behavior, our points of interest are the start and the end time of transient behaviors. The start of transient behavior should be marked by a degradation of quality that exceeds a threshold, the initial loss. Thus, the metric that answers this question is the time at which the initial loss occurs. The final question concerns itself with the

Goal	Purpose	Find
	Issue	occurrences of transient behavior
	Object	in a service/service instance
	Viewpoint	from the DevOps engineer's viewpoint
Question	Q1	What are the services that comprise the system architecture?
Metrics	M1	Service name
	M2	Number of service instances
	M3	Service endpoints
Question	Q2	How is the quality of service quantified?
Metrics	M4	Quality of service function
Question	Q3	At what time does a change introduce transient behavior into the service?
	M5	Time of initial loss
Question	Q4	What is the duration of the identified transient behavior?
Metrics	M5	Time of initial loss (t_0)
	M6	Time of complete recovery from transient behavior (t_1)
	M7	Difference t_1 and t_0

Table 5.3: GQM model for UC 3.

duration of the transient behavior. The metrics needed to answer it are the time of the initial loss (t_0) and the time at which the service completely recovered (t_1). The behavior's duration is given by Equation (5.2).

(5.2) $T = t_1 - t_0$

5.1.5 UC 4: Quantification of the Extent of Transient Behavior

This use case's goal is to quantify the extent of transient behavior in a service or service instance from the DevOps engineer's viewpoint. Once we have identified an occurrence of transient behavior, we need to quantify it to be able to compare it with the specification.

The first question that needs to be clarified for this use case is how the quality of service is quantified. As in the previous use cases, this is answered by the quality of service function.

As previously defined in Section 5.1.2, we describe transient behavior using the three aspects of initial loss, duration or time to recovery, and loss of resilience. To quantify the initial loss, we need to know the quality of service function and the time when the initial loss occurs (t_0). The initial loss is given by Equation (5.3) where $Q_e(t_0)$ is the expected quality of service at t_0 and $Q_a(t_0)$ is the actual quality of service at t_0.

(5.3) $X = Q_e(t_0) - Q_a(t_1)$

Goal	Purpose	Quantify
	Issue	the extent of transient behavior
	Object	in a service/service instance
	Viewpoint	from the DevOps engineer's viewpoint
Question	Q1	How is the quality of service quantified?
Metrics	M1	Quality of service function
Question	Q2	How high is the initial loss of the transient behavior?
Metrics	M1	Quality of service function
	M2	Time of initial loss (t_0)
	M3	Initial loss (Difference $Q_e(t_0)$ and $Q_a(t_0)$)
Question	Q3	What is the duration of the transient behavior?
Metrics	M2	Time of initial loss (t_0
	M4	Time of complete recovery from transient behavior (t_1)
	M5	Difference t_1 and t_0
Question	Q4	What is the loss of resilience suffered during the transient behavior?
Metrics	M1	Quality of service function
	M6	$\int_{t_0}^{t_1} [Q_e(t) - Q_a(t)]\, dt$

Table 5.4: GQM model for UC 4.

The second aspect, the duration of the transient behavior, was already defined in Section 5.1.4. Finally, the last aspect loss of resilience has been previously defined in Equation (5.1). By computing these three metrics for an occurrence of transient behavior, it can be quantified, and subsequently, compared to a specification that limits for the same features.

5.1.6 UC 5: Analysis of the Effect of Transient Behavior on Dependent Services

A microservice often depends on other services. Transient behavior that manifests in one service can propagate and cause transient behavior in other services with dependencies on the degraded service. One finding from the expert interviews reported in Chapter 4 is that it is of interest to know which services exhibit transient behavior after a change to the system occurs. Hence, analyzing the impact that transient behavior has on the quality of dependent services is another critical use case.

As described in Table 5.5, this use case's goal is to analyze the impact of transient behavior on dependent services from the DevOps engineer's viewpoint.

To reach this goal, we need to know the services comprising the system and their dependencies. The first question that concerns this use case is: *What are the services that comprise the system architecture?* The metrics that answer it are the service names, the number of their instances, and the names of their API endpoints. The next concern are the dependencies between the services: *Which services have dependencies to the impacted service?* To answer it, knowledge about the service interdependencies is needed. Precisely, the names of the services dependent on the service that exhibits transient behavior are needed.

Goal	Purpose	Analyze
	Issue	the impact of transient behavior
	Object	on dependent services
	Viewpoint	from the DevOps engineer's viewpoint
Question	Q1	What are the services that comprise the system architecture?
Metrics	M1	Service name
	M2	Number of service instances
	M3	Service endpoints
Question	Q2	Which services have dependencies to the impacted service?
Metrics	M4	Service dependencies
	M5	Names of dependent services
Question	Q3	At what time does a change introduce transient behavior into the first service?
Metrics	M6	Time of initial loss (t_0)
Question	Q4	Does this transient behavior propagate to dependent services?
Metrics	M5	Names of dependent services
	M6	Time of initial loss in first service (t_0)
	M7	Time of complete recovery from transient behavior of first service (t_1)
	M8	Loss of resilience in dependent service between t_0 and t_1

Table 5.5: GQM model for UC 5.

To investigate whether transient behavior in one service causes transient behavior in a dependent service, we need to analyze how the quality of the dependent service changes after the first service degrades. For this, we first need to find out at what time the original service's transient behavior starts. The metric that answers this question is the time at which the initial loss takes place. Next, we evaluate if this transient behavior propagates to the dependent services. Multiple metrics are required to answer this point. First, we need the name of the dependent service. Then, we must know the time of the initial loss in the first service (t_0) and its time of recovery (t_1). If the dependent service's quality is affected by the behavior, the degradation occurs in this period. The last metric is the loss of resilience between t_0 and t_1, defined in Equation (5.1). If this loss exceeds a certain threshold, the dependent service also exhibits transient behavior during this period. Since the two transient behaviors coincide, it is fair to say that one of them either caused the other or that the same event has caused both.

The next section details how the metrics put forward for the use cases in this section can be visualized, facilitating the exploration of transient behavior and its specification.

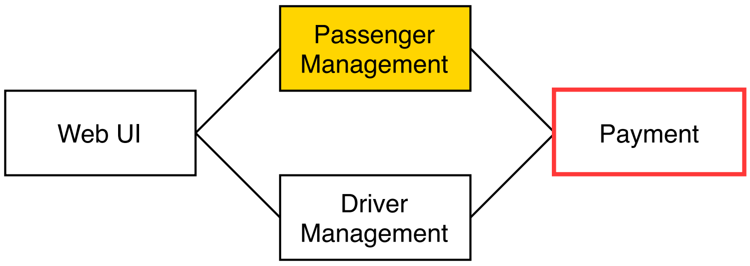


Figure 5.2: Visualization of the services and their dependencies.

5.2 Visualizations

One insight from the expert interviews presented in Chapter 4 was that visualizations of transient behavior would benefit software architects and DevOps engineers in their daily work. In this section, various visualizations of different aspects of transient behavior are proposed. They are designed to support architects and engineers in most of the use cases defined in Section 5.1. The last use case defined in Section 5.1.6 is not taken into account as this concept focuses on the specification and the analysis of transient behavior in a single microservice. Section 5.2.1 presents a visualization of the different services of a system and their dependencies. Subsequently, Section 5.2.2 introduces a visualization of the quality of service of a single service endpoint, and Section 5.2.3 describes a visualization of specifications for transient behavior. Finally, Section 5.2.4 presents a visualization for the loss of resilience of a single service endpoint during an occurrence of transient behavior.

5.2.1 Architecture Visualization

The essential metrics concerning a microservice system's architecture are the services' name, their endpoints, their number of instances, and dependencies. The following approach focuses on the service names and dependencies to keep the architecture visualization simple. If the services are considered vertices and the dependencies edges, the architecture forms a graph [Wes+96]. A common visual representation of graphs is the node-link diagram [SSKB14]. In a node-link diagram, vertices are mapped to nodes (graphical shapes) and edges to links (straight or curved lines). A significant problem of node-link diagrams is visual clutter [RLN07], a state in which the excess of items creates an unintelligible visualization. Force-directed layouts can be used to mitigate this problem. An example of a force-directed layout algorithm is the Fruchterman and Reingold algorithm [FR91]. The algorithm simulates a repulsion force between the nodes while connected nodes simultaneously attract each other, leading to uniform and symmetric graph layouts.

For this concept, the service names are mapped to nodes and the service dependencies to the links that connect them. Figure 5.2 illustrates how this visualization looks like for the examplary microservice system introduced in Chapter 2. The visualization is kept simple to not distract from the main focus: the service names and their dependencies. The red frame around the `Payment` service indicates that this is the service that the user selected to examine more information about this service, such as other visualizations. Whether or not a service meets all its transient behavior specifications is mapped to the node's color. When a specification is violated, the node is highlighted in orange. Using color the approach makes visible which services do not comply with their specifications at a glance.

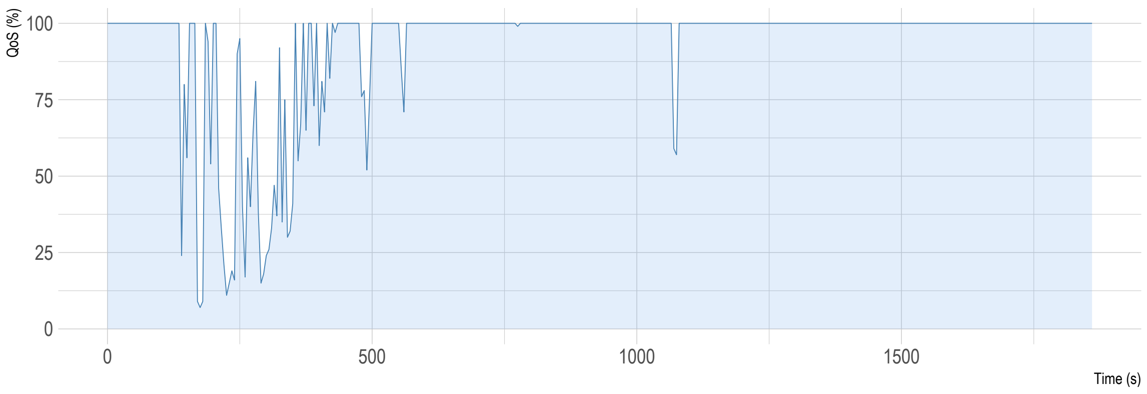


Figure 5.3: Visualization of the quality of service over time.

5.2.2 Quality of Service Visualization

A service endpoint's transient behavior can be perceived in the respective quality of service function over time. The quality of service is represented on a scale from zero to one hundred percent. The initial loss experienced at the beginning of transient behavior is an abnormal significant drop in the quality function. As the system recovers from the event that caused such behavior, the quality of service slowly increases until, finally, again it reaches one hundred percent.

Figure 5.3 shows how the service function's quality is visualized. The x-axis of the chart is the time in seconds; the y-axis is the quality of service in percent. A line plot represents the quality of service over time; the area under the line is filled for better visibility. Thus, this chart visualizes the actual transient behavior of a service endpoint. The metrics initial loss and time to recovery are visible. The loss of resilience is represented in another visualization, discussed in Section 5.2.4.

Since each service endpoint's quality of service can differ, a different quality of service visualization is needed for each endpoint.

5.2.3 Specification Visualization

As defined in Section 5.1.2, a transient behavior specification has three dimensions: initial loss, duration, and loss of resilience. A resilience curve or resilience triangle [Zob10] visualizes the first two, while the area of the formed triangle represents the loss of resilience. Thus, we use resilience triangles to visualize specifications for transient behavior.

Figure 5.4 shows such a specification. It can be seen that the quality of service value drops to 40%, representing the maximum acceptable initial loss. Subsequently, it steadily increases until it returns, after 50 seconds, to the expected value of 100%, which constitutes the specified maximum duration.

The different aspects of transient behavior must be compared to verify transient behavior's compliance with a specification. This is possible by placing the visualization of the specification on top of the visualization of the endpoint's actual quality of service, as demonstrated in Figure 5.5.

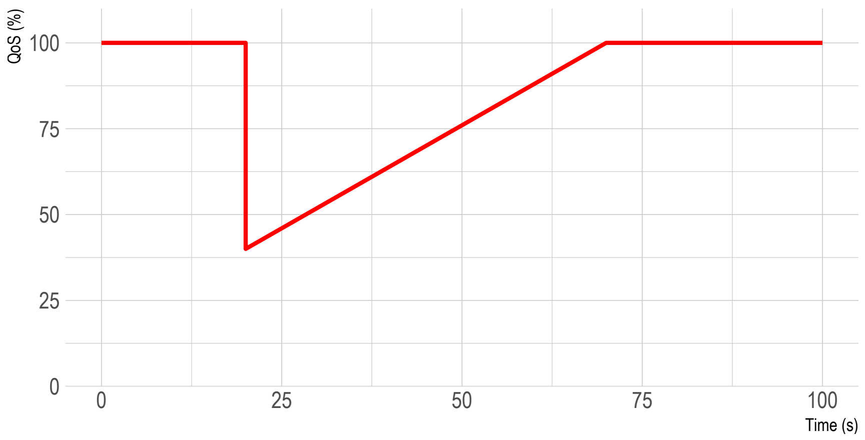


Figure 5.4: Visualization of a specification for transient behavior.

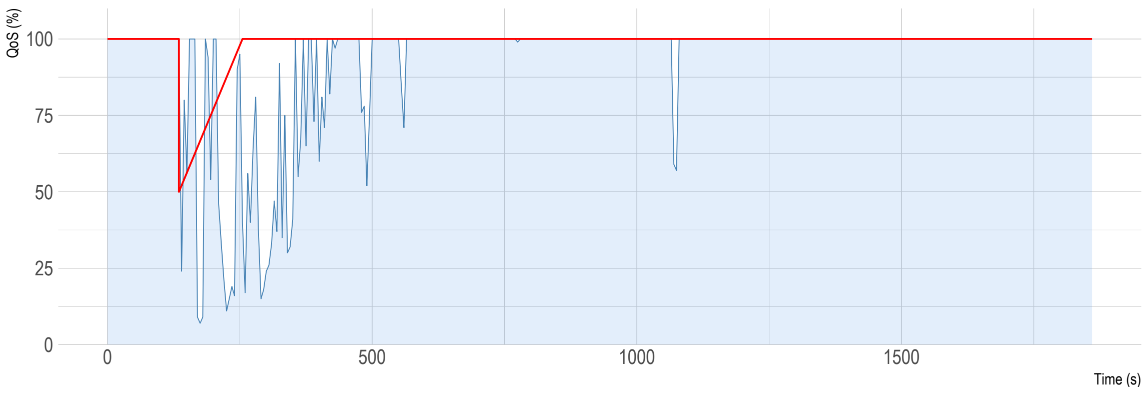


Figure 5.5: Comparison between the transient behavior of a service and its specification.

Comparing the value of the initial loss is straightforward by comparing how far down the lines drop. The transient behavior duration can be compared by examining whether the actual quality of service returns to a value of 100% before the plotted specification does.

This visualization does not explicitly include the specified loss of resilience of transient behavior. The area of the resilience triangle implicitly represents it. However, it is difficult to precisely compare this area with the actual loss of resilience that usually does not form a real triangle. Furthermore, the value of the area cannot be recognized by only looking at it. For these reasons, the loss of resilience specification is visualized in a different visualization, presented in Section 5.2.4.

5.2.4 Loss of Resilience Visualization

The loss of resilience during an event is defined in Equation (5.1) as the area above the quality of service function, highlighted in orange in Figure 5.6. It is difficult to read the loss of resilience in the quality of service plot. Therefore, the accumulated loss of resilience suffered during transient behavior is visualized in a separated plot. There, it is represented by a line and the area below it. The x-axis is again the time in seconds, while the y-axis is the accumulated loss of resilience.

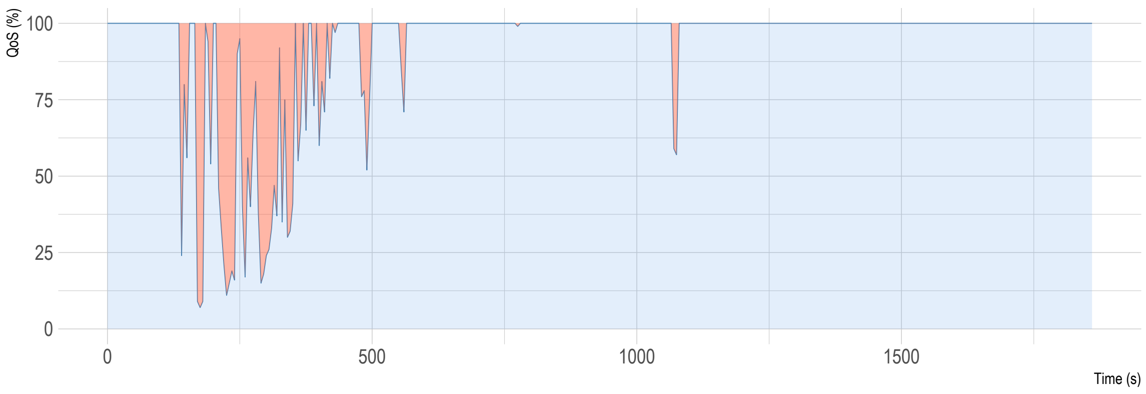


Figure 5.6: The orange area marks the loss of resilience of the service.

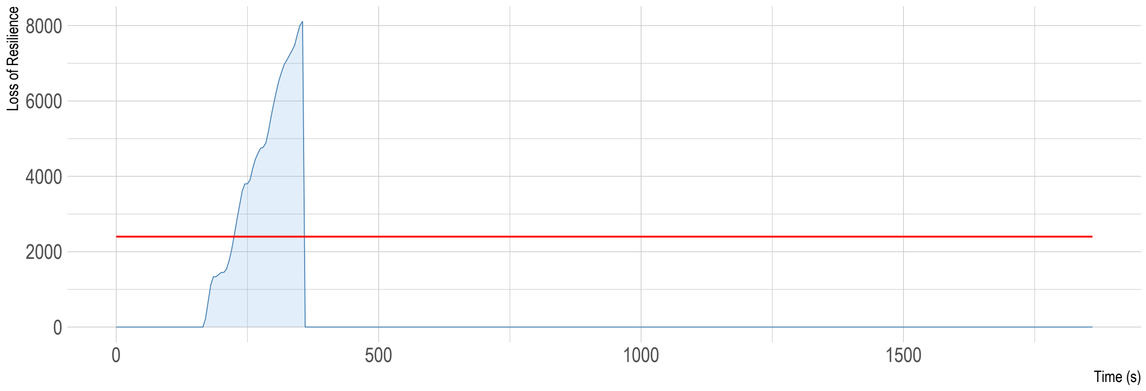


Figure 5.7: Visualization of the actual loss of resilience of a service and the specified loss of resilience. The specified maximum loss of resilience is represented by a red line.

Figure 5.7 shows how such a chart looks like. In it, only the loss during transient behavior is plotted. The loss of resilience is accumulated throughout the transient behavior. Thus, the value at the time of recovery is the total loss of resilience suffered during the transient behavior. The specified maximum loss of resilience is represented in the plot by a red line. The specification is violated as soon as the line representing the actual loss of resilience exceeds the red line.

5.3 Conversational Visualization Approach

This section presents considerations taken for a conversational software visualization that supports software architects and DevOps engineers when creating and verifying specifications for transient behavior through visualizations. It supports UC 1, defined in Section 5.1.2, and UC 2, defined in Section 5.1.3. To fulfill these two use cases, transient behavior must first be identified and quantified. Therefore, the UC 3 and UC 4, discussed in Section 5.1.4 and Section 5.1.5 respectively, are also implemented by this approach.

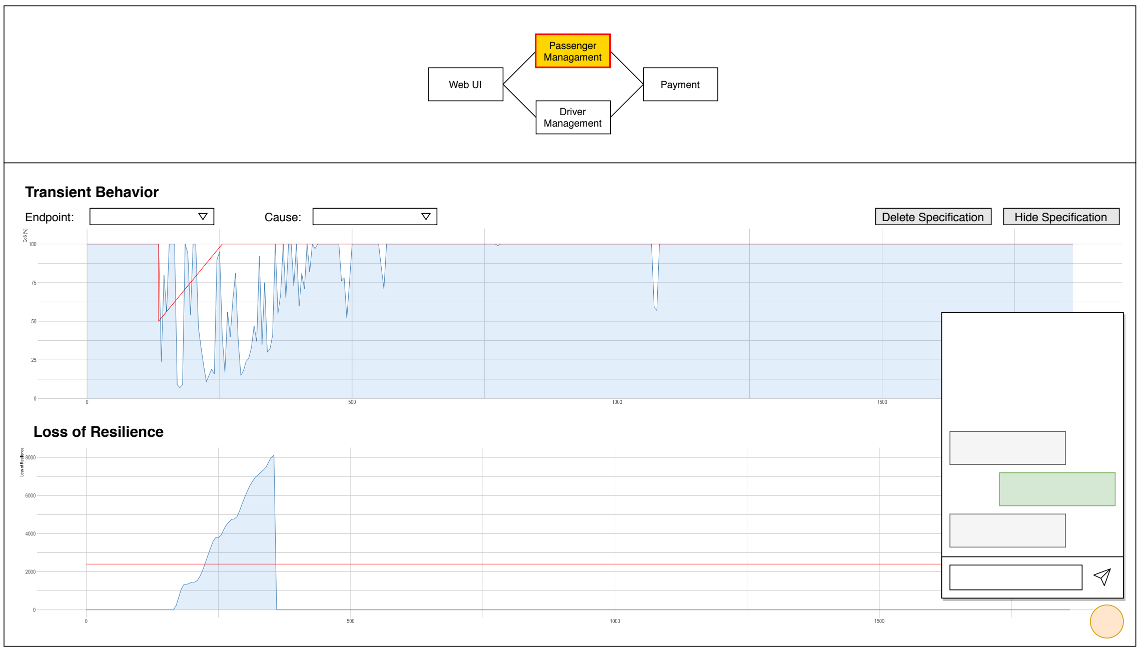


Figure 5.8: Sketch of the conversational visualization.

The design of this approach combines the visualizations presented in Section 5.2 into a dashboard with multiple coordinated views [Sch08]. It provides users an overview of the system's quality. Services can be selected to unveil more information about the transient behavior of their endpoints. In this respect, the concept follows Shneiderman's visual information-seeking mantra: "overview first, zoom and filter, then details-on-demand" [Shn96].

The dashboard comprises multiple visualizations with various interaction possibilities. The expert interviews presented in Chapter 4 revealed that engineers want to gain insights quickly without dealing with the unnecessary complexity of tooling. They often have a specific question in mind that they want to solve with the help of a visualization but might not be familiar with encodings, visualizations, configurations, interactions, or other possibilities of a visualization tool. Bieliauskas and Schreiber [BS17] proposed taking advantage of conversational interfaces to facilitate the interaction with visualizations. As two benefits of such an approach, they list accessibility to new users and a more natural way of interacting with software visualizations. For this reason, the proposed concept includes a conversational interface in the form of a chatbot that can be used to interact with visualizations. Another advantage of a chatbot is that developers already interact frequently with different kinds of bots [SZ16], a notion that was reinforced in one of the expert interviews.

Figure 5.8 shows a wireframe of the user interface of the envisioned design. Section 5.3.1 presents details of the dashboard and how it incorporates the proposed visualizations. The integration and functionality of the chatbot are described in Section 5.3.2.

5.3.1 Dashboard

The proposed dashboard consists of two main areas, as shown in Figure 5.8. The architecture visualization, presented in Section 5.2.1, gives an overview of the system's services and their dependencies at the top of the dashboard. Furthermore, it illustrates which services might exhibit transient behavior that exceeded the specifications. The user can interact with the visualization by selecting a service. Once selected, the service is marked in the visualization. When a user selects a service, the transient behavior visualization, proposed in Section 5.2.2, and the loss of resilience visualization from Section 5.2.4 are displayed. These visualizations give detailed information about transient behavior that might have occurred in a service endpoint and its specifications. As described in Section 5.2.3, they allow examining whether a service endpoint adheres to its specifications for transient behavior. User interface elements above the transient behavior visualization allow the user to configure the visualizations and create or delete transient behavior specifications. The displayed visualizations show the data for one service endpoint at a time. The user can change the endpoint that should be visualized.

When creating a new specification, the cause of the transient behavior to be specified is selected before the acceptable initial loss, duration, and loss of resilience are entered. Once created, the specification is visualized in the same charts as the actual service behavior. To achieve that, the tool analyses the quality data for transient behavior. The algorithm detects at what time the behavior is introduced into the service and which time it recovers. Subsequently, the specification is plotted at the matching position. If multiple instances of transient behavior are identified, the specification is plotted on top of each instance. For periods where no transient behavior has been detected, the specification line is plotted at 100% quality of service to indicate the expected service quality.

The loss of resilience chart visualizes the loss of resilience of the selected service endpoint for each detected occurrence of transient behavior. For periods without transient behavior, the loss of resilience is zero, even if the quality of service deviates slightly from 100% during such a period. The loss of resilience is exclusively plotted for transient behavior instances. The acceptable maximum loss of resilience is visualized by a red line.

5.3.2 Chatbot

Instead of directly interacting with the visualizations, the user can interact with them through the integrated chatbot. This enables the manipulation of the dashboard through natural text inputs. Additionally to controlling the visualizations, the chatbot can be used to create, edit, or delete specifications for transient behavior. This is realized through a collection of intents that the chatbot recognizes. Each intent has multiple parameters, called entities, such as the name of a service or the acceptable initial loss that is specified for specific transient behavior. When the chatbot recognizes an intent but the input is missing one or multiple entities, the chatbot requests the missing entities from the user.

The user interface includes a button at the bottom right corner that opens the chatbot window. The window is displayed on top of the dashboard and can be opened and closed upon need. This way, the chatbot is not in the way if the user wants to directly explore the visualizations but is still readily available directly in the tool itself. The user interacts with the chatbot by entering natural text messages into a chat field. The message history displays messages and replies above the input field.

In the following, the intents through which the chatbot can be used to interact with the dashboard are explained.

Select Service

The `Select Service` intent makes it possible to select a service in the architecture visualization to show its endpoints and their transient behavior visualizations. Its only entity is the name of the service that should be selected. The intent is invoked by inputs like “Please select the Payment service” or “Show me the transient behavior of the Passenger Management service”.

Show Specification

The `Show Specification` intent enables the user to ask the chatbot to show the specification for the transient behavior of a specific service caused by a particular event. These two aspects are defined by two entities: the service name and the cause of the specified behavior. The input for this intent are phrases like “Show me the specification for the Web UI service for transient behavior caused by failures”.

Add Specification

Through the `Add Specification` intent, the user can add a specification for transient behavior through the chatbot. It has four entities: the service for which the specification should be created, the cause of the behavior that should be specified, the maximum initial loss that is accepted, and the maximum duration that is accepted for transient behavior. The accepted loss of resilience is automatically calculated based on these parameters, in accordance with the resilience triangle model [Zob10]. The intent allows the addition of a specification through an input like “During a deployment, the initial loss in the Driver Management service cannot exceed 30% and the duration should not be longer than three minutes”.

Edit Specification

An existing specification can be edited via the `Edit Specification` intent. Attributes that can be changed are the maximum initial loss and the maximum duration of transient behavior. The accepted loss of resilience is adapted accordingly. the entities of this intent are the name of the service that concerns the specification, the cause of the behavior that is specified, and either the new accepted initial loss or duration, depending on which attribute should be changed. This makes it possible to change a specification through a phrase like “Change the accepted initial loss for transient behavior caused by a failure in the Web UI service to 20%”.

Delete Specification

The `Delete Specification` intent can be used to delete an existing transient behavior specification. To identify the specification that should be deleted, the entities service name and transient behavior cause must be defined in the input. The inputs that are mapped to this intent are phrases such as "Remove the specification for changing load created for Payment." When a specification is deleted, it is removed from the visualizations.

Help

The `Help` intent was added to support users who need orientation to learn how to interact with the chatbot. It does not have any entities and can be invoked by different phrases like "I need help", "What can I ask you?", or "How does this work?". Thereupon, the chatbot replies with a short explanation of its functionality and a list of the intents that it recognizes.

6 Prototype

This chapter describes the prototypical implementation of the approach that was proposed in Chapter 6. Section 6.1 gives an overview over the technologies that have been used to create the prototype. Afterward, Section 6.2 presents the design of the prototype before Section 6.3 details the data structure and the API of the backend. Subsequently, Section 6.4 presents the Dashboard component of the prototype. Section 6.5 explains the algorithm that is used to identify transient behavior in the quality of service data. Finally, Section 6.6 describes the implementation of the chatbot.

6.1 Technologies

The dashboard of the prototype is implemented as a website. The website's structure is created with the Hypertext Markup Language (HTML) and styled using Cascading Style Sheets (CSS). Additionally, Bootstrap [BTS], a popular CSS framework for developing responsive websites, was used to quickly design a responsive website. JavaScript is used as well to make the website dynamic.

The visualizations are created with the JavaScript visualization library D3.js [Bos20]. D3.js stands for "Data-driven documents". The library is the most popular open-source JavaScript library for generating interactive data visualizations in web browsers. It works with web standards such as HTML, CSS, and Scalable Vector Graphics (SVG) and allows to directly manipulate the Document Object Model (DOM) of a website.

The chatbot is implemented with Dialogflow [DFC], a chatbot framework by Google. We already gained some prior experience with Dialogflow in another project [OBM+20], where no issues arose with the framework. It has a free version, provides a web interface for training the chatbot that is easy to use, and allows webhooks that process intents. Furthermore, it provides a chat interface that can easily be integrated into an existing website. For this reason, we choose to use the provided chat interface instead of integrating the chatbot into a messenger platform like Slack or implementing our own chat interface.

The backend of the prototype is implemented with Django [Dja20], a Python web framework. Django was designed to help developers to build web applications quickly and without having to reinvent the wheel. Originally it is designed to handle Hypertext Transfer Protocol (HTTP) requests, but with the addition of the Django Channels library [Dja18] it extends is capabilities beyond that and can also handle WebSockets. WebSockets allow interactive bi-directional communication sessions between a browser and a server [Moz19]. We take advantage of this technology to enable communication between the chatbot and the dashboard.

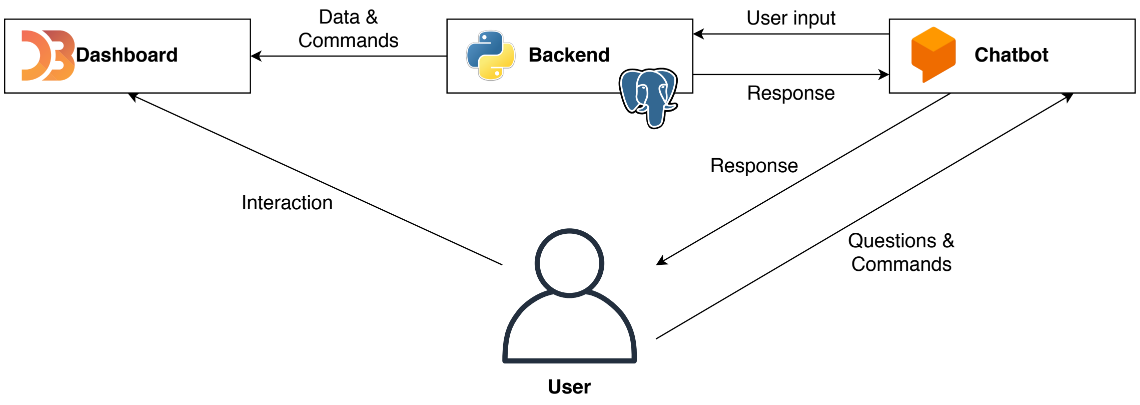


Figure 6.1: The architecture of the prototype.

6.2 System Design

The architecture of the prototype follows a modular approach. It is illustrated in Figure 6.1.

The Django backend stores the data in a Postgresql database and provides a Representational State Transfer (REST) API for the dashboard. Postgresql is a popular open-source relational database management system [Pos20] that is well integrated with Django. Section 6.3 details how the data and the backend API are structured. The dashboard requests data via HTTP from the backend and creates interactive visualizations for the data with D3.js. Furthermore, the dashboard establishes a WebSocket connection with the backend as soon as it is opened. This connection is used to receive messages from the backend. The user can interact with the dashboard to explore the data, create transient behavior specifications, and analyze transient behavior. When a specification is added in the dashboard, a HTTP POST request that includes the parameters of the specification is sent to the backend, which stores the specification in the database. The user can also interact with the chatbot by asking questions or giving commands like creating a specification or selecting a particular service. If Dialogflow recognizes an intent, it sends information about the input through a webhook to the backend. The backend analyses the intent and performs specific actions depending on the intent. For example, if the intent is to create a specification, the backend stores the specification in the database. The backend also sends a command to the dashboard via the WebSocket connection for each intent that it processes. The structure and content of these messages are explained in Section 6.3. For an intent that creates a specification, this would be the specification parameters so that the dashboard can visualize it. After the backend processed an intent, it creates a response that is returned to Dialogflow. Dialogflow then outputs this response in the chat interface. For example, if the user asks the chatbot for a specific specification, the backend retrieves the specification parameters from the database and embeds them into a response that the chatbot presents in the chat interface. The backend is the central component of the prototype, connecting the dashboard and the chatbot.

```
{
    "id": 14,
    "name": "Cart",
    "endpoints": [
        "getCart",
        "addToCart"
    ],
    "violation_detected": false
}
```

Listing 6.1: JSON representation of a service object.

```
{
    "source": 14,
    "target": 11
}
```

Listing 6.2: JSON representation of a dependency object.

6.3 Data Structure and REST API

The Postgresql database stores data about the system in four different tables. The first table contains the system's services. Each service has a name, a list of endpoints, and a boolean variable that stores whether a violation of one of the specifications defined for this service was automatically detected.

The second table contains the service dependencies. Each dependency consists of a source service and a target service.

The third table contains the complete measurement data for all services and endpoints. A measurement is comprised of (i) a timestamp, (ii) the service from which it was taken, (iii) the endpoint from which it was taken, (iv) the URI of that endpoint, (v) its quality of service value, (vi) the loss of resilience in case of failure, (vii) the loss of resilience in case of deployment, (viii) the loss of resilience in case of changing load. The last three attributes are instantiated with a value of zero and calculated anew each time the specification for the respective transient behavior cause is changed.

The final table contains transient behavior specifications. Each specification has the following attributes: (i) the service for which it was created, (ii) the cause for which it was specified, (iii) the maximum initial loss that is acceptable, (iv) the maximum time to recovery that is acceptable, (v) the maximum loss of resilience that is acceptable.

The backend provides access to all of these data objects through a REST API. The API allows Create, Read, Update, Delete (CRUD) operations on the objects. Furthermore, it accepts query parameters. These allow filtering the measurements that are returned by service and endpoint. The specifications can be filtered by service and cause. The data is returned in the JavaScript Object Notation (JSON) format.

```
{
    "service": 14,
    "time": 215.0000,
    "callId": 0,
    "uri": "getCart",
    "qos": 47,
    "failureLoss": 1757.5000,
    "deploymentLoss": 0.0000,
    "loadBalancingLoss": 0.0000
    }
```

Listing 6.3: JSON representation of a measurement object.

```
{
    "id": 184,
    "service": 14,
    "cause": "failure",
    "max_initial_loss": 40,
    "max_recovery_time": 120,
    "max_lor": 2400
    }
```

Listing 6.4: JSON representation of a specification object.

Listing 6.1 demonstrates how the JSON representation of a service looks like. Each service object has a unique identifier. Listing 6.2 is the JSON representation of a dependency as it is returned by the API. The `source` and `target` attribute reference the identifier of the respective service. The measurement object, as shown in Listing 6.3, contains the quality of service data as well as the loss of resilience data. The endpoint from which this measurement was taken is encoded in the `callId` attribute. The specification endpoint encodes the attributes of a specification into a JSON object, as demonstrated in Listing 6.4.

6.4 Dashboard

The dashboard of the prototype is shown in Figure 6.2. The design is based on the sketch that was presented in Section 5.3.1.

The top third of the dashboard contains the architecture visualization. The user can select a service by clicking on it or by asking the chatbot to select a service. A red border marks the selected service. If the `violation_detected` attribute of a service object is `true`, the service is highlighted in an orange color.

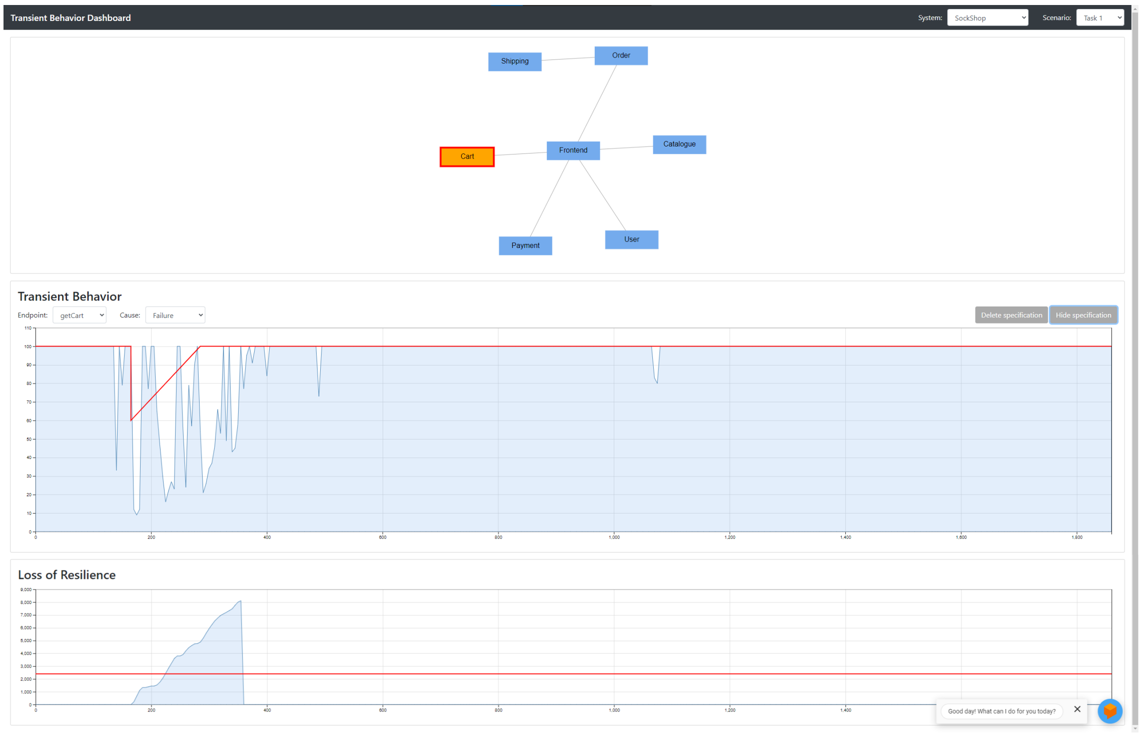


Figure 6.2: The dashboard of the prototype.

Below the architecture visualization are user interface elements through which the user can change the visualization. The `endpoint` selection box allows selecting the endpoint for which the data should be visualized. The `cause` selection box makes it possible to select the specification that should be plotted. On the right, multiple buttons are shown. These buttons can be used to show or hide the specification in the charts, add a new specification, or delete a specification.

Below these elements is the transient behavior specification, which corresponds to the visualization proposed in Section 5.2.3. The blue area represents the endpoint's quality of service, and the red line represents the transient behavior specification. The loss of resilience visualization at the bottom of the dashboard corresponds to the visualization proposed in Section 5.2.4 and shows the accumulated loss of resilience for each instance of transient behavior. The maximum acceptable loss of resilience is plotted as the red line.

When a service is selected in the architecture visualization, the dashboard sends an HTTP GET request to the backend, requesting the measurements for the first endpoint of the first service. When the endpoint is changed, the new endpoints measurement data is requested likewise. When the user clicks the `Show Specification` button, the dashboard requests the specification for the selected service and the selected transient behavior cause. The loss of resilience that is plotted in the bottom chart is computed in the backend whenever a new specification is created. To achieve that, the backend needs to detect instances of transient behavior in the quality of service data, compute its start and end time, and then compute the interval above the quality of service function.

A transient behavior specification can be added for the cause that is selected in the cause selection box by clicking the `Add Specification` button. As a result, a dialog windows is opened, which is depicted in Figure 6.3. The user can specify the maximum acceptable initial loss and the maximum

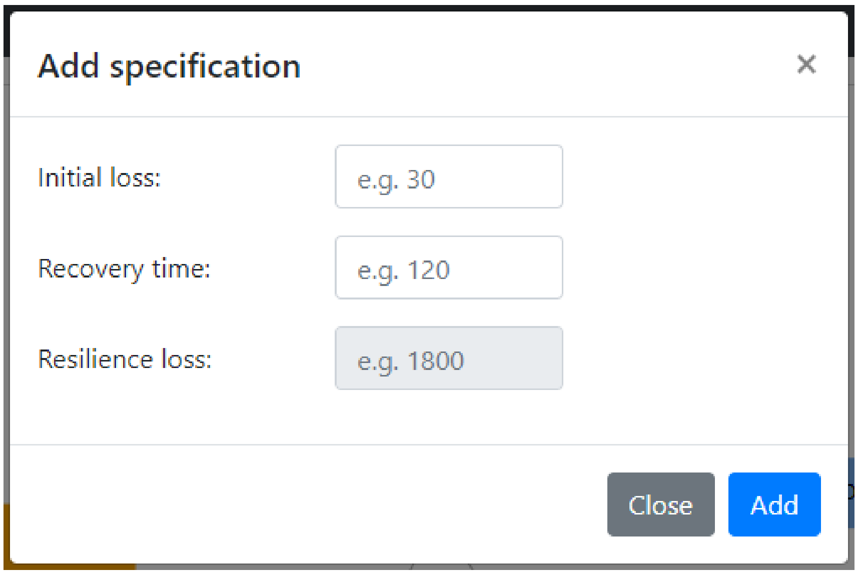


Figure 6.3: Add specification dialog.

time to recovery in the window. the maximum loss of resilience is computed based on these two metrics. When the user clicks the Add button, the specification is send to the backend through an HTTP POST request, where it is stored in the database.

6.5 Transient Behavior Identification

The backend detects transient behavior in the quality of service data of an endpoint by iterating over each measurement and checking if it could be the start point of an instance of transient behavior. First, the algorithm checks if the measurement is within a period that has already been identified as transient behavior. If this is the case, this measurement is skipped as an endpoint can only exhibit one occurrence of transient behavior at a time. Otherwise, the algorithm checks if the measurement's quality of service is below the expected quality of service level of 100%. If this is the case, the algorithm checks if the measurement is not just a slight fluctuation of the quality of service but the start of an instance of transient behavior. This is done by checking the median quality of service during the next five seconds. If the measurements' sample rate is too low to get quality of service values for the next five seconds, the value is interpolated between the current and the next measurement. If the median quality of service during the next five seconds is below the threshold of 90%, the algorithm assumes that the degradation of quality is not just a fluctuation but actually the start of transient behavior. In that case, the timestamp of the measurement is saved as the start point of transient behavior. To compute the endpoint of the transient behavior phase, the algorithm iterates over all remaining measurements to examine when the quality of service returns to the expected quality of 100%. Again, the algorithm looks at the median of the next five seconds for each measurement to ensure that the quality of service recovered. When the algorithm finds a measurement that exhibits a quality of service of 100% again, the timestamp is saved as the endpoint of the transient behavior. Last, the algorithm checks if the duration of the transient behavior that it found is shorter than the specified maximum duration of transient behavior. If this is the case, the algorithm sets the time at which the endpoint must have recovered as the new endpoint to avoid plotting multiple specifications on top of each other in the transient behavior visualization. The algorithm keeps looking for potential start points of transient behavior in the rest of the measurement until it iterated over all of them. In the end, it returns a list that contains all start and endpoints of transient behavior. This list is used to plot the transient behavior specifications at the correct position in the transient behavior chart and get the limits for the computation of the loss of resilience.

6.6 Chatbot

The chatbot was trained to recognize the intents that have been defined in Section 5.3.2. In Dialogflow, intents are trained by adding training phrases to them. For the chatbot to recognize the entities included in a phrase, they have to be marked in the web interface. Dialogflow provides a range of standard entities like the duration or artists' names. However, it is also possible to train custom entities. For our chatbot, we added the entities `service name` and `tb cause` to enable the chatbot to recognize service names and transient behavior causes in the user input.

When the chatbot recognizes an intent in the user input, it sends a webhook request to the backend. An example of such a request can be seen in Listing 6.5. It contains the text entered by the user (here `queryText`), the detected entities (here called `parameters`), the detected intent, the confidence for the detected intent, and many other pieces of information.

The backend listens for such requests and performs specific actions, depending on the intent that was recognized. This action might be to create, edit, or delete a specification, or it might just be to forward a command to the visualization dashboard. The backend forwards all intents to the dashboard in any circumstance. This is implemented via WebSockets. The dashboard establishes a WebSocket connection with the backend as soon as it is opened. Afterward, the backend can send messages to the dashboard. When an intent is recognized, the backend sends a command message to the dashboard that contains the intent that was recognized and its parameters. Listing 6.6 shows an example of such a command message. It contains the command to show a particular specification in the visualization.

The WebSockt connection between the dashboard and the backend enables the chatbot to interact with the visualization and is at the center of the prototype.

```
{
  "responseId": "5a193cdd-83b3-46ed-8e4d-eade12c8d15d-ce5e18e2",
  "queryResult": {
    "queryText": "Select the Cart service",
    "parameters": {
      "service_name": "Cart"
    },
    "allRequiredParamsPresent": true,
    "fulfillmentMessages": [
      {
        "text": {
          "text": [
            ""
          ]
        }
      }
    ],
    "outputContexts": [
      {
        "name": "projects/transientbehaviorbot-itoo/agent/sessions/518797bb-e3bc-2e3f-
e8a1-357b17ee4fcb/contexts/__system_counters__",
        "parameters": {
          "no-input": 0,
          "no-match": 0,
          "service_name": "Cart",
          "service_name.original": "Cart service"
        }
      }
    ],
    "intent": {
      "name": "projects/transientbehaviorbot-itoo/agent/intents/fd0b050c-b21a-4542-9
a33-3910bf048d1c",
      "displayName": "Select Service"
    },
    "intentDetectionConfidence": 1,
    "languageCode": "en"
  },
  "originalDetectIntentRequest": {
    "source": "DIALOGFLOW_CONSOLE",
    "payload": {}
  },
  "session": "projects/transientbehaviorbot-itoo/agent/sessions/518797bb-e3bc-2e3f-
e8a1-357b17ee4fcb"
}
```

Listing 6.5: Dialogflow request that shares information about a recognized intent with the backend.

```
{
    "type": "interaction",
    "intent": "Show Specification",
    "params": {
        "service_name": "Cart",
        "tb_cause": "failure"
    }
}
```

Listing 6.6: WebSocket message containing the command to show the specification for the Cart service for transient behavior caused by failure.

7 Evaluation

This chapter presents an evaluation of the conversational software visualization approach proposed in Chapter 5. The evaluation was conducted on the prototype that was introduced in Chapter 6. In the following, Section 7.1 states the goals of the evaluation and Section 7.2 outlines the methodology of the evaluation. After that, Section 7.3 describes the two test systems and the data that were used in the evaluation. In Section 7.4, the experimental setting is introduced. The results of the experiments are summarized in Section 7.5 and subsequently discussed in Section 7.6.

7.1 Evaluation Goals

This section presents the goals of this evaluation of the proposed approach. Three research questions are investigated in the evaluation. The first two research question aim at the overall feasibility of the approach. Research question one investigates whether it supports software architects and DevOps engineers in the specification of transient behavior and research question two investigates whether the approach supports them in the verification of such specifications. The third research question is concerned with the usability of the approach with regard to the proposed visualizations and user interactions. The last research examines how helpful the integration of the chatbot is in the designed approach.

Using the template by Wohlin et al. [WRH+12], the goals of this evaluation can be formulated in the following summary:

> Analyze *a conversational visualization approach* for the purpose of *supporting software architects and DevOps engineers in the specification and verification of transient behavior* with respect to *its functionality and usability* from the point of view of the *software architects and DevOps engineers*.

The research questions are formulated as follows:

RQ1: Is the proposed approach able to support architects and engineers in the specification of transient behavior?

RQ2: Is the proposed approach able to support architects and engineers in the verification of specifications for transient behavior?

RQ3: How good is the usability of the proposed approach?

RQ4: How helpful is the integration of a chatbot in the proposed approach?

7.2 Evaluation Methodology

The stated research questions all require the participation of software architects or DevOps engineers in the evaluation to judge how well the approach supports them in the stated tasks. The evaluation of the usability of the approach and the helpfulness of the chatbot also necessitates the judgement of this group of stakeholders. However, transient behavior is a complex topic that requires a certain amount of expertise. Thus, the number of suitable participants is limited to experts with preexisting knowledge of this domain. As there are no established approaches for using novel human-computer interfaces to support stakeholders in the specification and analysis of transient behavior, it is difficult to compare the proposed solution to a benchmark solution. These factors impede the design of a quantitative study for the evaluation of the approach, which usually requires a large number of participants.

For these reasons, a qualitative approach is more sensible for this evaluation than a quantitative one. Greenberg and Buxton [GB08] even warn that evaluations that do not fit the circumstance of a solution can be ineffective and even harmful to a novel idea. They especially remark that quantitative evaluations can mute innovative visions that may not conform to norms or utilize technologies that are still immature. Furthermore, they note that quantitative evaluations lack meaningful critique of a solution, which is especially important for an early academic prototype. Similarly, Isenberg et al. [IIC+13] identified a trend of more visualization researchers choosing evaluation methods that include analyzing the performance and subjective feedback of study participants. Additionally, they found out that a variety of publications at renowned visualization conferences evaluate ideas by examining the value that is created for domain experts. This is investigated through case studies where experts have to solve common problems that they face through the application of a prototype. A small number of participants is sufficient for such an expert study, three to five participants can already produce meaningful results. Xu et al. [XWY+19] for example successfully conducted a case study with only two domain experts to evaluate an approach for interactive visual analysis of anomalous performances in cloud computing systems. Thus, we are confident that an expert study is the right evaluation method for our approach and creates meaningful qualitative insights on the defined research questions.

The case study is conducted on data collected from two microservice systems. The first system is a mockup payroll accounting system that Kesim et al. [KWK+20a] implemented for their experience report on scenario-based resilience evaluation and improvement of microservice architectures. As part of their work they created a dataset of performance and reliability measurements during a series of chaos experiment that they conducted. The dataset is publicly available online [KWK+20b]. Section 7.3 describes the system and the dataset in more detail. The second test system used in the evaluation is SockShop [WC17], a demo microservice application that has been found to be a realistic representation of a real microservice system [AMPJ17]. Avritzer et al. [AFJ+20] assessed the scalability of microservice architecture deployment configurations and conducted a range of experiments under different conditions on the SockShop system. Notably, they also investigated the impact of security attacks on the performance of different architecture deployment configurations. The dataset of the performance measurements they collected was made available to us for use in this evaluation. During the case study, participants had to complete two tasks with the help of the prototype. The tasks were designed around the specification and verification of requirements for transient behavior. Each participant was familiar with the test system for which they had to solve the tasks and has domain knowledge about transient behavior. After the completion of the task, the

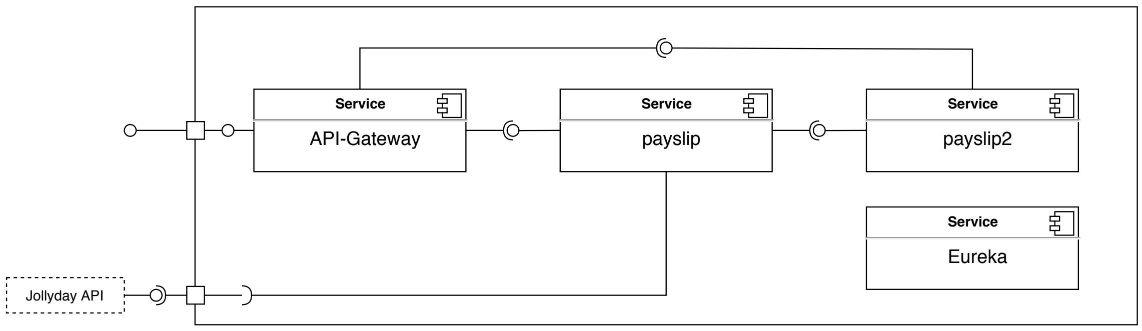


Figure 7.1: Architecture of the mockup payroll accounting system, adopted from [KWK+20a].

participants had to complete a questionnaire regarding their experience with the prototype to allow answering the research questions. The questionnaire was accompanied with a discussion about the experience, the approach, and potential improvements.

7.3 Evaluation Setup

This section describes the setup of the evaluation. Section 7.3.1 describes the mockup payroll accounting system and Section 7.3.2 describes the SockShop system, which are both used as test systems in the expert study.

The expert study was conducted remotely. The user interface of the prototype was hosted online in order for the participants to be able to access it. The study sessions were held through Cisco WebEx Meetings[1], a video conferencing solution from Cisco. All documents that the participants needed for the study were shared with them online. While they worked on the tasks, they shared their screen for us to observe their interaction with the prototype. The shared screen was recorded for later analysis with the integrated recording feature of WebEx Meetings. The participants received a copy of the questionnaire but orally informed the research investigator of their answers for him to document them.

7.3.1 Mockup Payroll Accounting System

The first test system is a mockup payroll accounting system created by Kesim et al. [KWK+20a] for the conduction of chaos experiments. The data collected in those experiments is the basis for the quality of service data used in the evaluation. The architecture of the system is depicted in Figure 7.1. It consists of four microservices. The `API-Gateway` service manages all incoming requests and forwards them to instances of `payslip` and `payslip2`. The `Eureka` service provides service discovery for the system. The `payslip` service utilizes an in-memory database and a third-party API. Furthermore, it can forward requests to `payslip2`. Requests can also be directly sent to `payslip2`.

The endpoints of `Payslip` and `payslip2` are described in Table 7.1.

[1]https://www.webex.com/

Name	Endpoint	Description
Internal dependency	/ergebnisdaten	Calls payslip2 via payslip
Database read	/abwesenheiten	Reads entry from db of payslip
External dependency	/service/api/v1/holidays	Calls Jollyday API via payslip
Database write	/abwesenheiten	Writes entry into db of payslip
Gateway ping	-	Checks if API-Gateway is alive
Unaffected service	/abwesenheiten/abwesenheitenId	Send request directly to payslip2

Table 7.1: Endpoints of `payslip` and `payslip2`.

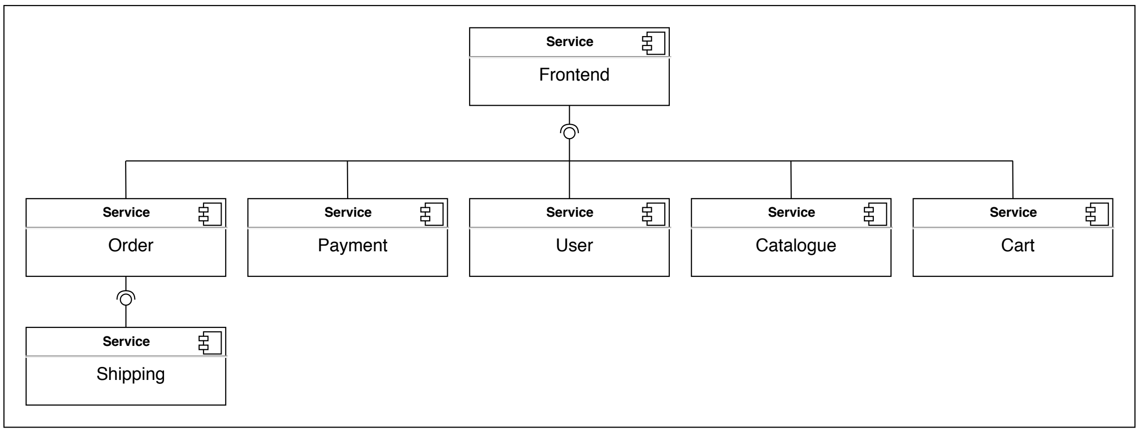


Figure 7.2: Architecture of the SockShop demo microservice system.

The following two scenarios were used in the evaluation: In the first scenario, one instance of the `payslip` service is terminated to simulate a service failure. In the second scenario, the `payslip` service implements the *retry* resilience pattern. One of its instances is again terminated. However, thanks to the pattern, the impact on the performance of the service is reduced.

7.3.2 SockShop

The second test system is SockShop [WC17], a realistic microservice demo application. Avritzer et al. [AFJ+20] conducted a range of experiments on SockShop to evaluate different microservice architecture deployment configurations. The created dataset is especially interesting for us because they also assessed the impact of security impacts on the deployment configurations. These attacks created transient behavior in the system, which the participants of our expert study will analyze. Avritzer et al. used a modified version of the Mirai malware to attack the system under test, a botnet used in a DDoS attack in 2016.

The architecture of the SockShop system is depicted in Figure 7.2. It consists of seven services: `Payment`, `Orders`, `Carts`, `Catalogue`, `Users`, and `Shipping`. The `Frontend` service connects all the other services. The `Catalogue` service provides information about the catalog and products. The `Order` service provides ordering capabilities to the system and communicates with the `Shipping`

Service	Endpoint
Frontend	home
	catalogue
	showDetails
	basket
	viewOrdersPage
Catalogue	getCatalogue
	catalogueSize
	cataloguePage
	getItem
	getRelated
	tags
Order	createOrder
	getOrders
Cart	getCart
	addToCart
User	login
	getCustomer
	getCard
	getAddress

Table 7.2: Endpoints of the services of SockShop.

service that provides shipping capabilities. The `Cart` service provides shopping carts for users. The `User` service is responsible for user account storage, including credit cards and addresses. Finally, the `Payment` service provides payment functionality.

The different endpoints of the services are described in Table 7.2.

Both scenarios used in the evaluation included a DDoS attack on the infrastructure on which the services were hosted. Due to different deployment configurations, the attack's impact on the system is much higher in the first scenario than in the second scenario. However, the attacks created transient behavior in multiple services and API endpoints.

7.4 Experiment Settings

This section provides a detailed description about the experiment settings. Section 7.4.1 introduces the five experts that were selected for the expert study. Then, Section 7.4.2 described the procedure of the study. Following, Section 7.4.3 outlines the tasks that the participants had to solve. Afterward, Section 7.4.4 discusses the questionnaire that was used in the study.

7.4.1 Selection of Participants

Five experts were selected to take part in the user study. These experts have previous knowledge of transient behavior and are each familiar with one of the test systems and the related dataset.

Participant A was also one of the interviewees interviewed for Chapter 4 where we called him Torsten. He is a software architect and holds a PhD in computer science. He is deeply familiar with the accounting system and the accompanying dataset.

Participant B also holds a PhD in computer science and conducts research in the field of software engineering. He is familiar with the SockShop system and created the related dataset.

Participant C also holds a PhD in computer science. His research focus is software engineering and formal methods. He knows the SockShop system but is not very familiar with the dataset that is used in the evaluation.

Participant D and Participant E are students that have previously researched topics related to transient behavior. They have implemented the mockup accounting system and created the dataset that is used in the evaluation.

7.4.2 Procedure

Each study session was scheduled to take approximately an hour. The session started with a short introduction to the topic. Afterward, the participant was informed about his rights during the study. The participants already returned the signed consent form before the appointment. Next, information about the respective test system, transient behavior, and the prototype was provided to the participant. After the participant read the information, he had ten minutes to familiarize himself with the prototype. During this time, the participant could ask questions about the tool and the approach. Once potential questions were resolved, the participant would start to work on two tasks. He had ten minutes for each task. The tasks consisted of one or multiple specification and analysis tasks. The participant's interaction with the prototype was observed and recorded. After each task, he was presented with the correct solution. After the second task, the participant was given a questionnaire with questions about his experience. The participant would state his answers to the research investigator, who documented them. Once the questionnaire was completed, a short discussion about the answers was led.

7.4.3 Tasks

Independent of the test system on which the study was conducted, each participant had to complete two tasks. They had ten minutes to solve each task. For the tasks involving the accounting system, the participants had to specify transient behavior for the `payslip` service and subsequently analyze whether all endpoints of the service meet the specification. The scenario of the first task is an instance failure of the `payslip` service that introduces transient behavior into the system. After entering the specification, the participant had to analyze if it is fulfilled. The correct result was that all endpoints violate the specification. In the second task, a resilience pattern was implemented by the `payslip` service. The participants had to add the same specification as in the first task. When they verified the specification's fulfillment, the expected solution was that the specification is now met in all service endpoints.

For the tasks involving the SockShop system, the participants had to create specifications for the transient behavior of the `Cart` and the `User` services and analyze whether all endpoints of the services meet the specifications. The scenario of the first task is a DDoS attack, which causes the load on the system to increase drastically and thus introduces transient behavior into the system. The participants had to create two specifications and analyze if they are fulfilled. The correct answer to this task was that the endpoint `getCart` of the `Cart` service and the endpoint `login` of the `User` service violate the specifications. In the second task, the system's protection against security attacks was improved. The participants had to add the same specifications again as in the first task. When the specification was verified, the expected solution was that all endpoints of the two services now met the specifications.

7.4.4 Questionnaire

The questionnaire is mostly based on the System Usability Scale from Brooke [Bro96] and NASA TLX [HS88] developed by the National Aeronautics and Space Administration (NASA). It can be found in Appendix B.4. The questionnaire starts with twelve statements about the participants' experience with the prototype to investigate the prototype's usability. The participants were asked to indicate their agreement with the statements on a scale from one (strongly disagree) to five (strongly agree). The questionnaire continues with four questions designed to assess the cognitive load during the tasks. These questions were adapted from NASA TLX. The participants were asked to answer on a scale from one (very low) to five (very high). The last four questions of the questionnaire are open questions. The first asks which features of the prototype the participant perceived as most effective. The second one asks which features were perceived as least effective. Question three regards the chatbot and is concerned with its helpfulness. The final question asks the participants how they would have solved the tasks without the prototype.

7.5 Description of Results

This section describes the results of the evaluation. First, we present how long the participants worked on the tasks, before giving an overview over the results structured according to the research questions.

The time that it took the participants to solve each task was measured. However, no distinction between specification and verification of transient behavior was made, so no statement about each of these subtasks' duration can be formed. Furthermore, some participants tried different prototype features while working on a task or took the time to describe what they were doing. Thus, these measurements are not an entirely realistic assessment of the time it takes to complete the given tasks. Nonetheless, they reveal some interesting observations.

Figure 7.3 shows how much time the individual participants took to complete the given tasks. It should be noted that the SockShop scenario is more extensive than the accounting system scenario; thus, it took longer to complete those tasks. The completion times of the three participants working

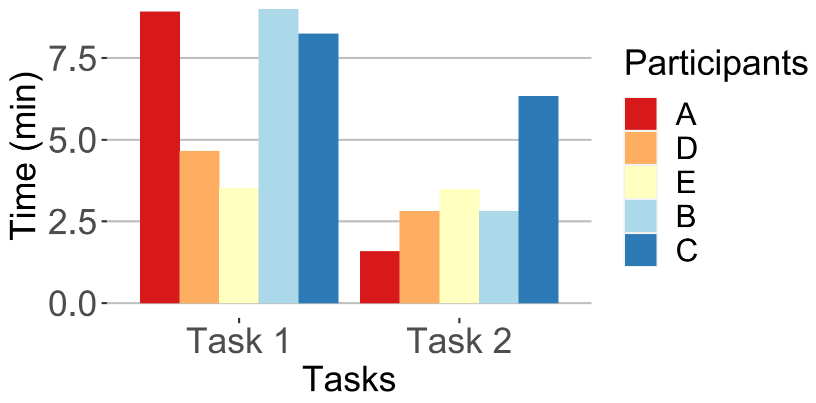


Figure 7.3: The time that it took the participants to solve the respective task.

on the accounting system scenario are displayed first, followed by the two participants' completion times working on the SockShop scenario. It can be observed that it took the participants five minutes and 42 seconds on average to complete the first task of the accounting system scenario and eight minutes and 37 seconds to complete the first task of the SockShop scenario. Except for participant E, it took less time to solve the second task. The second task of the accounting system scenario took, on average, two minutes and 39 seconds, a reduction by three minutes and three seconds. Solving the second task of the SockShop scenario took the participants four minutes and 35 seconds on average, four minutes and two seconds faster than the first task. Here, Participant C stands out because he used the chatbot to create the specifications in the second task, which took considerably longer than the specification creation through the graphical user interface.

7.5.1 RQ1: Is the proposed approach able to support architects and engineers in the specification of transient behavior?

All participants were able to successfully create the specifications for transient behavior given in the task description. Except for Participant C, the participants specified the transient behavior exclusively through the user interface. They started the tasks by examining the architecture visualization and selecting the service for which they wanted to create a specification directly in the visualization. Only Participant C created two specifications with the chatbot. He stated he wanted to try out the chatbot to compare the workflow with the user interface's direct interaction.

While participants B and C struggled with creating specifications through the dialog form at first, they quickly figured out how it works. The other participants did not have any difficulties identifying and entering the specification parameters. Participant B found the selection box for the cause of the transient behavior confusing at first because it was not apparent to him what its purpose is. Furthermore, he commented that it was not clear for him in the beginning if the initial loss is the percentage by which the quality degrades or if it is a threshold above which the quality of service must stay. It was not clear for Participant C, which cause he should specify for transient behavior that is introduced by a security attack. At first, it was also not apparent to him if a separate specification has to be created for each service endpoint. Participant D stated that it is not apparent what the loss of resilience parameter represents at the time of the specification. However, its meaning becomes apparent in the visualization.

Participant A stated that he is confused by the specification's resilience triangle representation because its meaning is not intuitive. He believes resilience curves are an excellent method for visualizing transient behavior; however, the representation of specifications as resilience triangles should be improved to depict the individual attributes more distinctly.

Participant A wanted to look up the parameters of the specification that he created. However, this is not possible in the user interface. Instead, he had to ask the chatbot for the parameters. He commented that he would prefer it if the parameters would be displayed directly in the visualization, a feature that Participant B also missed in the prototype.

When Participant C added a specification through the chatbot, he added it to the wrong service at first. However, he quickly realized his mistake, was able to delete the wrong specification, and create another specification for the correct service.

Participant D created a specification for transient behavior introduced for deployments instead of specifying the cause as service failure. Since this has no other consequences for the representation of the specification or the visualized data, this had no further impact on his performance in the respective task.

When asked which features of the prototype were most effective, Participant E listed the fact that fields of the dialog, through which a specification is added, store the parameters of the previously created specification. He perceived this as effective because he did not need to look up the parameters a second time working on task two.

A feature that is missing, according to Participant E, is a list that gives an overview of all specifications that have been created for a system. Without such a list, the user has to look through every service when searching for a particular specification.

Participants A and D commented that the visualization of specifications is a useful addition to existing specifications, such as resilience scenarios. However, Participant D voiced a valid concern that transient behavior specifications are often not elicitated in practice. The expert interviews described in Chapter 4 confirmed this concern. The prototype presents no solution to this problem and would suffer from a lack of transient behavior specifications in practice.

7.5.2 RQ2: Is the proposed approach able to support architects and engineers in the verification of specifications for transient behavior?

Except for two instances, the participants were able to verify the specifications for transient behavior correctly. In one case, an occurrence of transient behavior was mistakenly labeled as violating the specification. In another case, the participant argued that it is impossible to decide whether the behavior meets the specification.

The loss of resilience and transient behavior visualizations were named the most effective prototype features for this task by three participants. Two participants especially perceived the visualization of the specifications on top of actual transient behavior as the most effective feature. Participant C noted that it is straightforward to detect specification violations in the loss of resilience visualization. The visualization clearly shows if the loss of resilience is higher than accepted.

However, for Participant D the loss of resilience metric was too abstract to be helpful. He would have also preferred to specify the quality of service function himself or at least to choose between different options. Likewise, Participant A thought that the prototype was too abstract because of the quality of service function. Instead, he would prefer to directly examine different concrete performance metrics like response times, throughput, and the number of failed requests. He also thought that the use of real performance metrics would make the approach easier to understand.

Generally, the participants used the loss of resilience chart to look for first indications that transient behavior exceeds the specification and subsequently examined the transient behavior chart to confirm this suspicion. Participant D additionally stated that the ability to examine each individual microservice and endpoint is helpful and Participant A noted that the visualization of transient behavior as resilience curves adds value. He can imagine the adoption of such a representation in practice.

Participant B remarked that it would be more convenient to give an overview of the data of all endpoints of a service at once. This could be achieved either in one visualization or in different visualizations that are displayed simultaneously. Having to select different endpoints to examine them separately is time-consuming and requires more effort. Similarly, he stated that it is not feasible in production to explore each service and endpoint for specification violations. Instead, he would prefer a list of instances in which a specification was breached or to receive a notification when a violation occurs.

Participant A expected that the resilience triangle would be placed where the actual loss exceeds the specified loss for the first time. Instead, the initial loss of the specification is positioned where the quality of service begins to degrade. He stated that this complicates the analysis of the time to recovery. He believes that the duration should be measured from when the observed initial loss exceeds the specified initial loss for the first time. In fact, Participant E misjudged the observance of a specification in the second task due to this inaccuracy and argued that an instance of transient behavior violated the specification, although the behavior was still within the requirements. Similarly, Participant B commented that it is not clearly recognizable in all instances, whether the duration of transient behavior is longer than specified. Participant A also noted that it would be useful if the visualization would highlight the period during which a specification is violated in the visualization.

At first, Participant A stated that the specification had been met in the first scenario when the expected answer was that the specification is violated. However, he later revised that statement and said that it is impossible to decide whether the specification is met or not because the formulation of the requirements is ambiguous. He explained that no indication is given if the specification is met if the system fulfills only a part of it. He stated that it would be interesting to see the duration for which the maximum loss is exceeded highlighted in the visualization.

Participant A concluded that it would be useful to support defined resilience scenarios with the proposed visualizations to be able to analyze if the scenarios are met in production. Participant D agreed and stated that he could imagine that the presented approach is extremely helpful for analyzing transient behavior if the necessary data and the specifications are already available. Participant B thought that the approach is especially suited for exploring transient behavior and communicating problems that it causes to business stakeholders and managers. However, he believes that the current prototype is not suited for reporting transient behavior in production. The reason for that is that the manual examination of transient behavior requires too much effort for such a use case. Instead, specification violations should be recognized automatically and directly reported to the engineers. Furthermore, he stated that the tool would create real value if the definition of search queries were possible, for example, in the chatbot. He envisioned that such queries could show all instances of transient behavior that meet a particular condition, similar to filtering the presented data. Participant A also expected that the chatbot would provide some sort of filter functionality and thinks that such a feature is essential for large systems with many services. Participant C commented that the prototype was well implemented.

When asked how they would have approached the given tasks without the prototype, four participants answered that they are not aware of other tools that could be used to solve the tasks. Three participants explained that they would write a custom script to analyze the data. Two participants stated that they would attempt to create a visualization of the data and then attempt to judge whether the specifications are fulfilled or not. Participant D commented that it would be difficult to compare resilience scenarios, which are usually available in text form, with the data — even if the data was plotted in a chart.

7.5.3 RQ3: How good is the usability of the proposed approach?

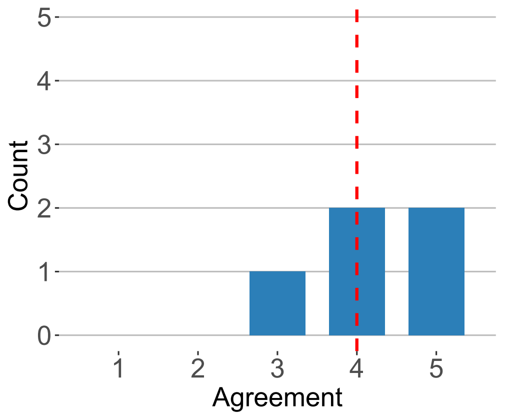
The participants filled out a questionnaire about their experience with the prototype to investigate its usability. While the number of participants is too small to generalize the results or warrant quantitative analysis, this section presents the questionnaire results and the participants' impressions about the prototype's usability. The first part of the questionnaire presented eleven statements about the prototype. The participants had to indicate on a scale from one (strongly disagree) to five (strongly agree) how much they agree with each statement. Figure 7.4 and Figure 7.5 visualize the distribution of the participants' answers. The dotted red line indicates the median agreement.

The participants agreed with a median of four that they would use a tool like the prototype for their work. Participant A gave the lowest answer with a value of three. He believes that the approach has potential but must be improved to generate value in practice. Participant E answered four and noted that the prototype's use case is not relevant for his personal work, but he can imagine that the tool is useful for engineers concerned with transient behavior analysis. Next, the participants mostly strongly agreed that the tool was easy to use; the median answer for this question is five. Participants B and D remarked that the visualizations of the transient behavior and the loss of resilience are missing axis labels. They think that it would be easier to understand the visualizations if those were added. The tool's complexity was judged with a median of one, meaning they disagree with the statement that the prototype is unnecessarily complicated. Furthermore, they believed that the tool's various functions were mostly well-integrated, with a median of five. Participant E remarked that there are some small inconsistencies in the prototype. For instance, the user interface elements that control the transient behavior and loss of resilience visualizations are located in a position that gives the impression that they only control the transient behavior visualization. Furthermore, he missed a way to see an overview of all specifications that have been created. All participants strongly disagreed with the statement that there was too much inconsistency in the tool. Regarding the time that it takes to learn to use the prototype, four of the participants strongly agreed that most people would learn to use the tool very quickly. Only Participant A answered the question with a value of two. He explained that the tool is easy to learn, but the theory of transient behavior is complex, and it is time-consuming to understand the concepts behind the term. Furthermore, he claimed that the specification functionality of the tool is difficult to understand at first. Similarly, Participant D remarked that ten minutes of training are enough time to get acquainted with the tool if the user is already familiar with the topic of transient behavior. Participant E stated that the visualizations are easy to understand and to use.

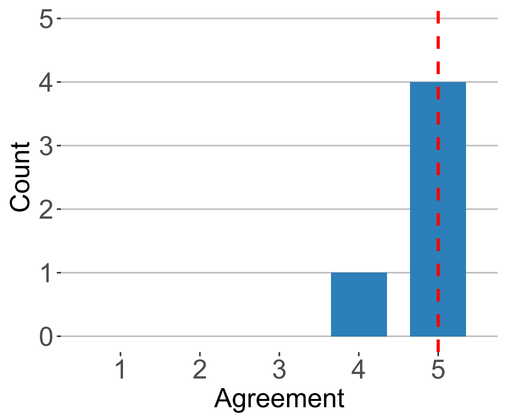
All participants strongly disagreed with the statement that the tool was very cumbersome to use. Next, they felt mostly confident using the tool. The answers for this statement ranged between three and five, with a median of four. Participant C remarked that he first needed some time to get used to the tool. However, after he finished the second task, he felt confident while using the tool. Also, Participant E stated that he felt good using the prototype even though he did not have much experience with it. The statement *I needed to learn a lot of things before I could get going with the tool* was answered with a value of two by three participants, two participants answered with a value of three. Participant E, who answered with a value of two, explained that he had to learn a few things before working with the tool. However, he was already previously familiar with the terms and concepts used by the tool. The participants mostly thought that the included visualizations were easy to understand; the median answer was “strongly agree” with one Participant each judging the statement with three and four. Finally, the participants stated that the included visualizations helped them to solve the tasks. The median answer for this statement was a value of five.

Four further questions were targeted to gain insight into the participants cognitive load while they completed the tasks. The participants answered on a scale from one (very low) to five (very high). Figure 7.6 depicts the distribution of the answers for these questions.

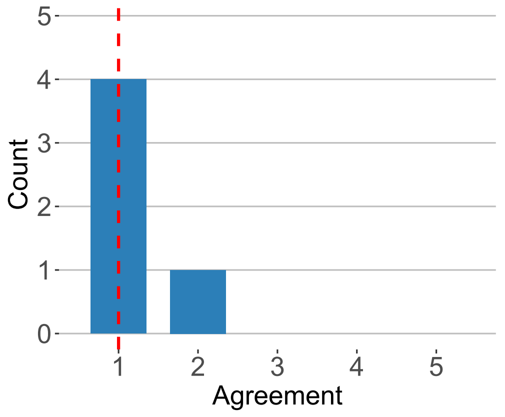
The first question was *How mentally demanding was it to solve the tasks using the prototype?* All participants answered the question with a value of two, which indicates that the tasks were not very mentally demanding for them. The next question was *How successful were you in accomplishing what you were asked to do?* In this question, the scale of the answers was from one (perfect) to five (failure). The median is a value of one, indicating that the participants were successful for the most part. Participant C answered with a value of two because he had mistakenly created a specification for the wrong service in one task. However, he quickly corrected his mistake. Participant E answered with a value of three because he erroneously stated for the second task that the transient behavior of the endpoints violated the specification. The third question asked the participants how hard they had to work to accomplish their level of performance. Three participants answered with a value of two, which is the median. Two participants answered with “very low”. This indicates that the participants did not have to work hard to perform well in the given tasks. Participant B merely lamented that it takes much effort to switch between the individual endpoints. He would have preferred to examine multiple endpoints at once. The final question was *How insecure, discouraged, irritated, stressed, and annoyed were you?* All of the participants answered with “very low”.



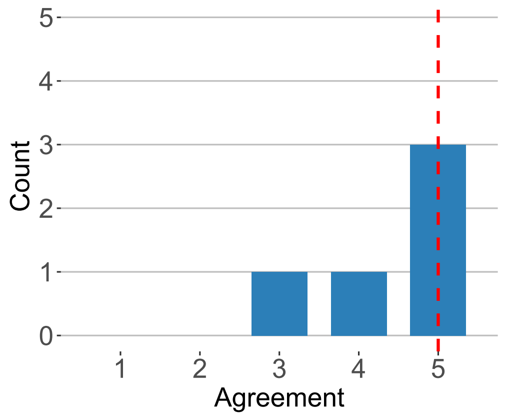
(a) I think that I would use a tool like that for my work.



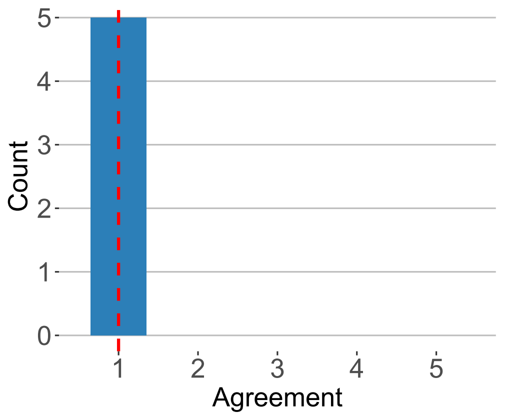
(b) I thought the tool was easy to use.



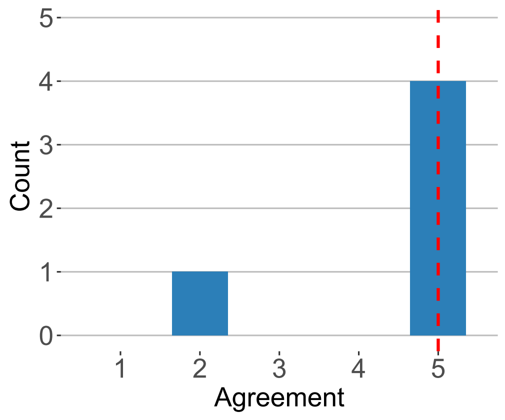
(c) I found the tool unnecessarily complex.



(d) I found the various functions of the tool were well integrated.

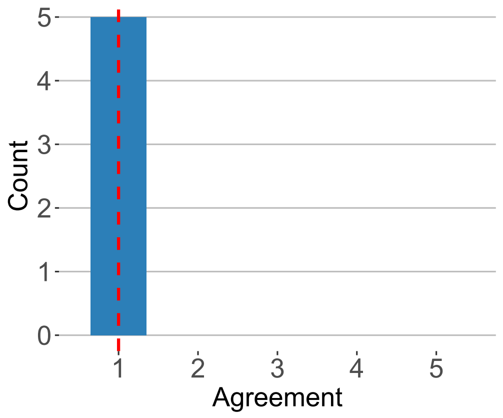


(e) I thought there was too much inconsistency in the tool.

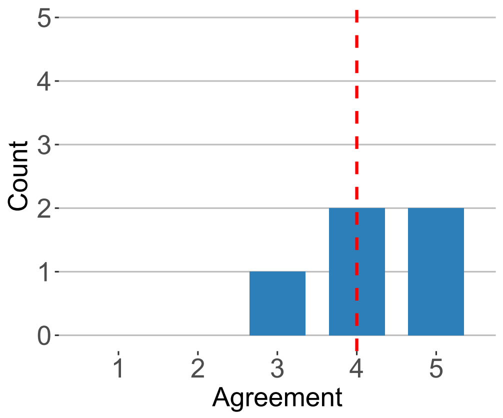


(f) I would imagine that most people would learn to use the tool very quickly.

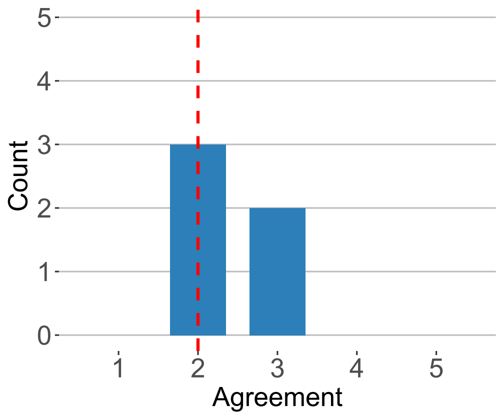
Figure 7.4: Distribution of answers for the first six usability statements.



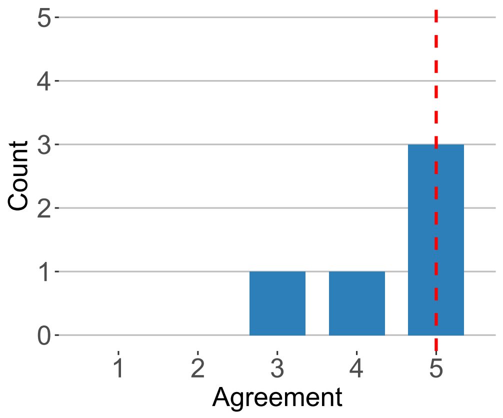
(a) I found the tool very cumbersome (awkward) to use.



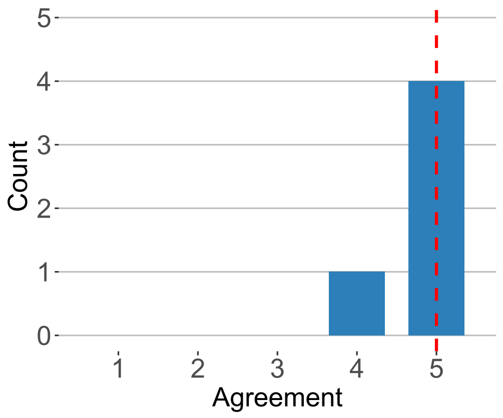
(b) I felt very confident using the tool.



(c) I needed to learn a lot of things before I could get going with the tool.

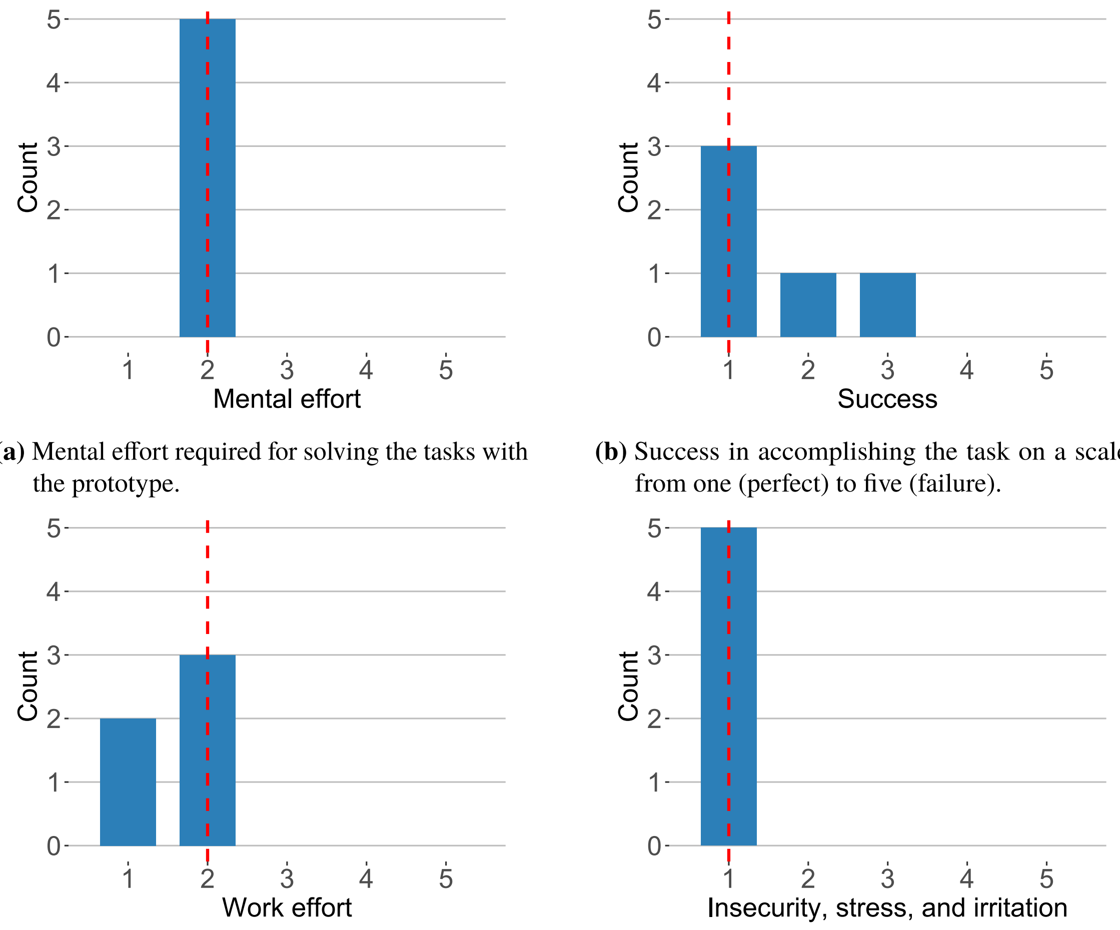


(d) I think the included visualizations were easy to understand.



(e) I think the included visualizations helped me to solve the task.

Figure 7.5: Distribution of answers for the last five usability statements.



(a) Mental effort required for solving the tasks with the prototype.

(b) Success in accomplishing the task on a scale from one (perfect) to five (failure).

(c) Work effort required to accomplish level of performance.

(d) Insecurity, discouragement, irritation, and stress during the tasks.

Figure 7.6: Distribution of answers for the four questions concerning cognitive load.

7.5.4 RQ4: How helpful is the integration of a chatbot in the proposed approach?

One statement of the questionnaire concerned the helpfulness of the chatbot that is integrated into the prototype. Figure 7.7 shows the answers for that statement. Four participants answered with a three, indicating that they neither perceived the chatbot as helpful nor unhelpful. Participant A found the chatbot unhelpful and answered with a value of two.

When asked about the least efficient features of the prototype, four participants named the chatbot. There are two main reasons for this. Four participants stated that the chatbot was not helpful because it was not needed to solve the tasks. It did not add additional value to the visualizations. Participant A specified that the visualizations are good enough to render the chatbot redundant. Also, three participants raised the issue that the chatbot interaction takes longer than the direct interaction with the visualizations and the graphical user interface of the prototype. Two participants further explained that the input that is accepted by the chatbot is unclear at first. Some knowledge of the chatbot is required to interact with it efficiently. The chatbot should support the user by explaining which input is needed. Participants A and B expected to be able to filter the data and the created specifications with the chatbot's help. However, this functionality is not provided by the chatbot. Furthermore, Participant A asked the chatbot different questions about the dataset, which the chatbot could not answer. He asked the chatbot to show him all endpoints that violate the specification, but the chatbot did not understand the question. Next, he asked the chatbot to list all endpoints where the quality degraded by more than 40%. However, the chatbot identified this as a request to change the loss of the transient behavior specification. Similarly, the chatbot recognized the wrong intent on different occasions when participants asked questions that the chatbot can not answer. Another question that was asked multiple times was whether the current transient behavior violated the specification. Nevertheless, the chatbot cannot answer this question either. Participant A noted that he would have expected to be able to change the endpoint that is displayed using the chatbot. However, the chatbot can only be used to change the displayed service and the displayed specification. Participants A and D perceived the chatbot rather as annoying than as helpful. The reason for that is that they had to follow a trial-and-error approach to identify the supported functionality of the chatbot. However, the chatbot did not provide additional functionality that was worth the effort of understanding the features it provides.

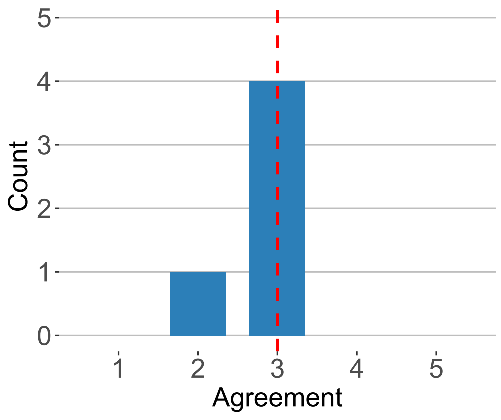


Figure 7.7: Level of agreement with the statement *The integration of a chatbot in the tool was helpful.*

Three of the participants stated that they think the chatbot could be more helpful when interacting with larger systems or working on more complex tasks. The stated reason was that the visualizations might become cluttered if too much data is visualized. If that is the case, the direct interaction with the visualizations could take longer than the interaction with the chatbot. Participant E stated that the given tasks were too simple to require the use of the chatbot. Participants A and E mentioned that the chatbot was well integrated with the rest of the prototype. It was not in the way when interacting with the visualizations but could be accessed at any time if needed.

7.6 Discussion of Results

This section presents a discussion of the results and answers for the research questions. Section 7.6.1 focuses on the suitability of the approach for the specification of transient behavior. Afterward, Section 7.6.2 discusses if the approach supports architects and engineers in the verification of transient behavior specifications. Subsquently, Section 7.6.3 investigates the usability of the proposed approach. The helpfulness of the chatbot is discussed in Section 7.6.4 and Section 7.6.5 identifies potential threats to the validity of the results.

7.6.1 RQ1: Is the proposed approach able to support architects and engineers in the specification of transient behavior?

All participants were able to create specifications for transient behavior in the prototype without major difficulties. This provides evidence that the proposed approach can support software architects and DevOps engineers in the definition and documentation of requirements for transient behavior. However, the parameters for the specification were stated in the task in an obvious manner. In real projects, these requirements have to be elicitated from the customer which is the greater challenge. The findings of the expert interviews confirmed that this is the biggest problem that they face regarding the specification of transient behavior. The prototype is not able to support stakeholders in this task.

The proposed approach can support stakeholders in the definition and documentation of specifications for transient behavior. However, some improvements can be made to increase its effectiveness.

Additionally, the resilient triangle representation of a specification should be improved to depict the different dimensions of the specification more clearly. The addition of labels that indicate the value of the attributes could already help. This would also resolve the issue that the parameters of a specification can only be checked through the chatbot. The specification of transient behavior through the three dimensions initial loss, duration, and loss of resilience is not intuitive and has to be learned before it can be applied. Nonetheless, it does not take much time to become familiar with the concept.

Another addition that would improve the proposed approach is an overview over all specifications that have been created. In the current version of the prototype, such an overview is missing and only one specification can be examined at a time. Furthermore, it is not possible to see which specification have been added to the system.

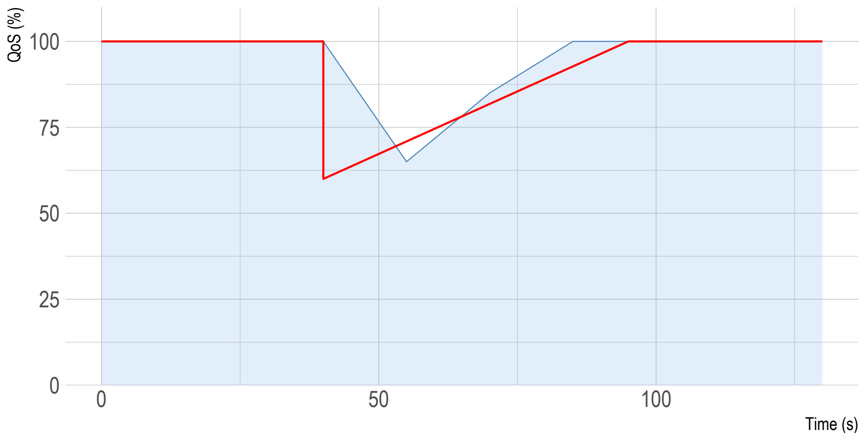


Figure 7.8: Deviation between the plotted specification and transient behavior.

7.6.2 RQ2: Is the proposed approach able to support architects and engineers in the verification of specifications for transient behavior?

The participants were mostly able to correctly identify whether the occurrences of transient behavior fulfilled the specifications. Especially the fact that the specification was plotted in the same chart as the actual behavior was helpful for this task. While the approach is not flawless, the participants agreed that it is helpful for this kind of task. Furthermore, they stated that they are not aware of any other solution that facilitates the analysis of transient behavior specifications.

The participants used the combination of the transient behavior and loss of resilience visualizations to verify whether a specification was met. The loss of resilience chart was beneficial for identifying instances of transient behavior. This is because the chart makes it obvious when the loss of resilience exceeds the accepted value. This was often used as the first indication that a specification is violated.

The proposed approach supports stakeholders in the exploration of transient behavior. However, future improvements are needed to make the tool a viable option for production use.

The transient behavior visualization gives more details about the behavior of an endpoint. It allows examining the quality degradation and duration of a transient behavior instance in more detail. However, it is not always obvious what the duration of an instance of transient behavior is. An improvement to the visualization would be to highlight the period during which a specification is not fulfilled in the visualization.

Furthermore, as can be seen in Figure 7.8, the specification is not always correctly plotted on top of the transient behavior visualization. The quality of service seems to degrade slowly in the example because a measurement is only sampled every thirty seconds. The quality was still at 100% at 40 seconds. At 55 seconds, however, it already degraded to 60%. The prototype recognizes this correctly as transient behavior. However, the initial loss of the specification is plotted when the quality starts to degrade. Because the initial loss of the specification happens instantly, the two curves differ from each other. This makes it difficult to compare the duration of the transient behavior with the duration that is specified. Additionally, the misalignment leads to the quality function erroneously being plotted below the specification's recovering line. This led to the incorrect assessment of a specification violation by Participant E in the study.

Another point that the evaluation revealed was that engineers would prefer to choose their own quality of service function. The ambiguity of the abstract quality of service function unknown to the user made the scenarios more difficult to understand. Moreover, engineers would like to switch between different metrics to investigate different performance aspects of a system.

The fact that the participants completed the second task faster than the first task indicates that they quickly learned how to use the tool.

To results indicated that the approach is suited for the exploration of transient behavior and the analysis of individual specifications. However, engineers miss features that let them filter or query the data for specification violations. In production, the reporting of violations is essential. Nevertheless, the presented approach's current state is not suited for such a task but is limited to exploration and communication tasks. Future work should provide a way to get a quick overview of transient behavior that exceeds the specifications without searching for it throughout the whole dataset.

7.6.3 RQ3: How good is the usability of the proposed approach?

The participants rated the prototype mostly as easy to use. Furthermore, they stated that it is not difficult to learn how to use the tool. The only obstacle is the complicated topic of transient behavior, which requires some expertise.

The proposed approach is easy to learn and to use. The provided visualization are helpful for the stated tasks.

The provided visualizations are helpful and support the users in their tasks. Also, the mental effort required to successfully apply the prototype to the specification and analysis of transient behavior is low despite the difficult domain. All these facts speak for the high usability of the tool. However, it should be remarked that the sample size of five experts is not large enough to generalize the findings to the whole population of software architects and DevOps engineers.

7.6.4 RQ4: How helpful is the integration of a chatbot in the proposed approach?

Except for one exception, the participants did not use the chatbot to solve the tasks. The only reason why they interacted with the chatbot at all was out of curiosity. The chatbot interactions take too long to be a viable alternative to the direct interaction with the visualizations. Furthermore, the chatbot does not provide any value beyond controlling the visualizations and creating specifications. Both tasks can be completed faster without the chatbot. To be helpful, the chatbot needs to provide other features that distinguish him from the functionality provided by the dashboard. The participants wanted to use the chatbot to filter the presented data or directly search for specification violations. If it were to be extended with features such as those, the chatbot might become helpful for analyzing transient behavior.

The chatbot is not helpful in its current state because it adds no additional value and interactions take longer than with the visualization.

Another point is the manageable size of the two test systems and the simplicity of the given tasks. For more complex tasks or a larger system, the chatbot might be a faster method for finding services than the visualization due to the limited screen space.

The participants also complained that the input which the chatbot understands is not apparent. This leads to trial-and-error approaches of trying to figure out what the chatbot can do. That the bot occasionally mistakenly recognizes intents aggravates this issue. The chatbot needs to communicate more clearly what he can do and what input he expects.

7.6.5 Threats to Validity

This subsection discusses potential threats to the validity of the results of the evaluation.

Regarding the applicability of the results to the real world, the specified tasks were relatively simple and well defined. This is often not the case in actual software projects, where non-functional requirements are often not sufficiently specified. The expert interviews reinforced that this holds especially true for requirements for transient behavior.

Furthermore, the used datasets were created during load tests and chaos experiments and not measurements from real microservice applications. For this reason, the data might not be representative of the real performance data of a microservice system. Also, the recorded scenarios contain data for only several minutes. The production performance monitoring data from a real system is far more extensive. This would increase the difficulty of analyzing transient behavior substantially. Therefore, the given tasks might not be representative of problems that are faced in the real world. However, there is no abundance of performance datasets that cover scenarios that involve transient behavior. In particular, it has to be noted that the study participants need to be familiar with the test systems and the presented data. For this reason, the choice of test systems and datasets was constricted.

The experts that have been selected for the expert study introduce a large amount of uncertainty into the evaluation. There is a possibility that they are not a representative sample of the population of software architects and DevOps Engineers, the user group that the proposed approach is addressing. They could be more or less skilled than the average software architect or DevOps engineer. Also, their willingness to participate in a user study could indicate a higher than average motivation that could influence their performance in the tasks. This thread to validity could be reduced by recruiting a bigger sample size for the evaluation. However, as explained in Section 7.2, it is difficult to find many domain experts that are willing to participate in a time-consuming user study. Furthermore, there is evidence that five experts are a sufficient number of participants for a qualitative evaluation [IIC+13]. Additionally, a qualitative evaluation approach yielded better results than a quantitative approach would have been able to for this prototype [GB08].

Another potential threat to validity is the method of the expert study. The questionnaire might not be extensive enough to capture all aspects of the participants' experience. Many insights were also derived from comments that the participants made about the prototype. Those comments could be subjective. Moreover, the participants might not have discussed all aspects of the prototype and the task with the same thoroughness. Also, the study was conducted remotely via video conferencing. Technological obstacles might have influenced the participants' performance or their perception of the prototype.

8 Conclusion

The conclusion summarizes the thesis and discusses its benefits and limitations.

8.1 Summary

This work examined how HCI and visualization approaches can be employed to support software architects and DevOps engineers in the specification and analysis of transient behavior. Therefore, it created a link between the domains of reliability engineering, transient behavior analysis, HCI, and visualization. An introduction to the concepts of transient behavior and resilience was given. Furthermore, novel methods from the field of HCI and the basics of visualization were presented. The thesis reported a series of expert interviews conducted to learn how transient behavior is treated in the industry. The findings of those interviews showed that transient behavior is not yet commonly specified in software projects. Instead, architects are struggling to convince business stakeholders of the necessity and importance of non-functional requirements. The interviews also revealed that practitioners want tools that can be integrated into their existing technology stack without much effort and that use commonly adopted technologies. The experts stated that basic visualizations for transient behavior would be needed before approaches beyond that with the application of novel human-computer interfaces would pay off. Based on the knowledge from the literature research and the expert interviews, a concept was proposed that uses visualizations in combination with a conversational interface to support architects and engineers. The envisioned solution is based on several use cases defined to capture the functional requirements for such a solution. The approach proposes to visualize transient behavior and specifications thereof as resilience curves. The concept was implemented as a prototype, based on D3.js, Django, and Dialogflow, and evaluated in an expert study. The evaluation results showed that his approach has merit for the specification and exploration of transient behavior. Furthermore, it is easy to use, and the proposed visualizations are useful. Nonetheless, a variety of improvements have been identified that would make the prototype more helpful, especially for production. Finally, the chatbot's integration in the prototype was not helpful because it did not provide additional value over the visualizations.

8.2 Discussion

The evaluation showed that the prototype is well suited for interactively exploring transient behavior. However, it misses querying and filtering functionalities that would allow finding violations of specifications quickly. One such improvement is adding a reporting feature for transient behavior specification violations that do not require the user to explore the data of all services manually to find a violation. Another result of the evaluation was that the chatbot's integration in the prototype was not helpful because it did not provide additional value over the visualizations. On the contrary, the interaction with the chatbot took more time and was less intuitive than the interaction with the visualizations. To be useful, the chatbot must clearly explain the input that he understands and distinguish himself from the functionality provided by the visualizations. It would add value if the chatbot could answer questions about the data and find violations of specifications. In its current state, the interaction with the visualizations is so much more intuitive and faster than the chatbot that the chatbot's integration in the tool is unnecessary.

Another interesting issue that the expert interviews revealed is that many business stakeholders are oblivious to transient behavior and the importance of non-functional requirements in general. The proposed approach does not present a solution to this challenge. The specifications that can be defined within the prototype still need to be elicited together with the customer. Without the necessary data or the required input for the transient behavior specifications, the implemented prototype is of little use. However, if specifications, e.g., in the form of resilience scenarios, already exist, it requires little effort to visualize and analyze them with the proposed approach.

8.3 Future Work

The proposed approach has much potential for improvements. Future work should aim at extending the functionality of the prototype. The first improvement should be to provide an overview over the created specifications. In the current state a specification is only visible when the respective service and transient behavior cause are selected. For a large number of specifications the overview is quickly lost and engineers waist time searching for the correct specification. A list of specified behavior would remedy this issue.

Another starting point for future work is the addition of a filter or search feature. Multiple study participants stated that they missed such a feature in the prototype. Such a filter could be used to show only instances of transient behavior that violate a given specification. The chatbot could act as input field for search queries and return a set of visualizations that fits the entered query.

Further work is required to make the approach also feasible for production use. To achieve that, the prototype should be able to identify transient behavior specifications automatically and report them to the responsible engineers. The automatic identification of transient behavior implemented in the prototype is already a good start but needs to be extend to provide more accurate results. The estimation of a specification violation based on a exceeding of the maximum loss of resilience value needs to be refined further, too, to be viable for such a use case. Also, the prototype works only with static data at the moment. In order to be used as part of a continuous integration pipeline, the approach needs to be able to handle dynamic data.

The chatbot has potential for future work as well. The evaluation showed that it is not helpful in its current state. The interaction with the chatbot needs to be simplified and better explained by the bot itself. When users are not aware of the abilities of the bot they refrain from using it if they have an easier to use alternative like the visualizations. Furthermore, more intents could be added to the chatbot that increase its usefulness. Participants expected that the bot would be able to answer questions about the data or highlight transient behavior that is not within a specification. Future work should aim at adding such functionality to the bot.

Additionally, this thesis did not explore solutions that employ other human-computer interfaces such as virtual or augmented reality, voice interfaces, or multi-touch tables. Future work should aim at adapting the proposed concept for the use with other interfaces. The visualizations that have been created could be transferred into virtual reality without major problems. The datasets that have been used in the evaluation could serve as a benchmark to compare this approach with similar solutions.

Another direction that was not in the scope of the proposed approach is supporting stakeholders in the elicitation of requirements for transient behavior. The interviewees from the expert interviews stated this as the biggest challenge in regard to transient behavior. A solution for such a problem would probably differ widely from our solution as the use cases are entirely different.

Bibliography

[96] *New International Webster's Comprehensive Dictionary of the English Language*. Trident Press International, Naples, FL, 1996 (cit. on p. 8).

[AFJ+20] A. Avritzer, V. Ferme, A. Janes, B. Russo, A. van Hoorn, H. Schulz, D. Menasché, V. Rufino. “Scalability Assessment of Microservice Architecture Deployment Configurations: A Domain-based Approach Leveraging Operational Profiles and Load Tests”. In: *Journal of Systems and Software* (2020), p. 110564 (cit. on pp. 64, 66).

[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 p. 7).

[Ama20] Amazon Web Services, Inc. *Amazon Lex*. 2020. URL: https://aws.amazon.com/en/lex/ (visited on 11/15/2020) (cit. on p. 13).

[AMPJ17] C. M. Aderaldo, N. C. Mendonça, C. Pahl, P. Jamshidi. “Benchmark requirements for microservices architecture research”. In: *2017 IEEE/ACM 1st International Workshop on Establishing the Community-Wide Infrastructure for Architecture-Based Software Engineering (ECASE)*. IEEE. 2017, pp. 8–13 (cit. on p. 64).

[AS] K. B. Ahmad Abdellatif, E. Shihab. *A Repository of Research Articles on Software Bots*. http://papers.botse.org (cit. on p. 11).

[BBV+01] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, et al. “Manifesto for agile software development”. In: (2001) (cit. on p. 7).

[BCE+03] M. Bruneau, S. E. Chang, R. T. Eguchi, G. C. Lee, T. D. O'Rourke, A. M. Reinhorn, M. Shinozuka, K. Tierney, W. A. Wallace, D. Von Winterfeldt. “A framework to quantitatively assess and enhance the seismic resilience of communities”. In: *Earthquake spectra* 19.4 (2003), pp. 733–752 (cit. on pp. 8, 9, 20).

[BCK03] L. Bass, P. Clements, R. Kazman. *Software architecture in practice*. Addison-Wesley Professional, 2003 (cit. on p. 17).

[BE96] T. Ball, S. G. Eick. “Software visualization in the large”. In: *Computer* 29.4 (1996), pp. 33–43 (cit. on p. 17).

[Bec] S. Beck. *Conversational Transient Behavior Visualization*. URL: https://github.com/Cambio-Project (cit. on p. 3).

[Bec20] S. Beck. *Evaluating Human-Computer Interfaces for Specification and Comprehension of Transient Behavior in Microservice-based Software Systems (Supplementary Material)*. 2020. DOI: 10.5281/zenodo.4288378. URL: https://doi.org/10.5281/zenodo.4288378 (cit. on p. 3).

[Ber83] J. Bertin. *Semiology of graphics; diagrams networks maps*. Tech. rep. 1983 (cit. on p. 15).

[BHJ16] A. Balalaie, A. Heydarnoori, P. Jamshidi. "Microservices architecture enables devops: Migration to a cloud-native architecture". In: *Ieee Software* 33.3 (2016), pp. 42–52 (cit. on p. 5).

[BJPM16] B. Beyer, C. Jones, J. Petoff, N. R. Murphy. *Site Reliability Engineering: How Google Runs Production Systems*. Ö'Reilly Media, Inc.", 2016 (cit. on p. 37).

[BM13] M. Brehmer, T. Munzner. "A multi-level typology of abstract visualization tasks". In: *IEEE transactions on visualization and computer graphics* 19.12 (2013), pp. 2376–2385 (cit. on p. 16).

[Bol80] R. A. Bolt. ""Put-that-there" Voice and gesture at the graphics interface". In: *Proceedings of the 7th annual conference on Computer graphics and interactive techniques*. 1980, pp. 262–270 (cit. on p. 10).

[Bos20] M. Bostock. *D3: Data-driven documents*. 2020. URL: https://d3js.org/ (visited on 11/25/2020) (cit. on p. 53).

[Bro96] J. Brooke. "SUS: a "quick and dirty'usability". In: *Usability evaluation in industry* (1996), p. 189 (cit. on p. 69).

[BS17] S. Bieliauskas, A. Schreiber. "A conversational user interface for software visualization". In: *2017 ieee working conference on software visualization (vissoft)*. IEEE. 2017, pp. 139–143 (cit. on pp. 19, 49).

[BTP] L. Bedu, O. Tinh, F. Petrillo. "A tertiary systematic literature review on Software Visualization". In: *2019 Working Conference on Software Visualization (VISSOFT)*. IEEE, pp. 33–44 (cit. on p. 17).

[BTS] *Bootstrap*. URL: https://getbootstrap.com/ (visited on 11/25/2020) (cit. on p. 53).

[BWZ15] L. Bass, I. Weber, L. Zhu. *DevOps: A software architect's perspective*. Addison-Wesley Professional, 2015 (cit. on pp. 1, 6, 7).

[Car99] M. Card. *Readings in information visualization: using vision to think*. Morgan Kaufmann, 1999 (cit. on pp. 13, 14, 16).

[Clo20] Cloudflare, Inc. *Famous DDoS attacks | The largest DDoS attacks of all time*. 2020. URL: https://www.cloudflare.com/learning/ddos/famous-ddos-attacks/ (visited on 11/26/2020) (cit. on p. 8).

[CM84] W. S. Cleveland, R. McGill. "Graphical perception: Theory, experimentation, and application to the development of graphical methods". In: *Journal of the American statistical association* 79.387 (1984), pp. 531–554 (cit. on p. 15).

[CRB10] G. P. Cimellaro, A. M. Reinhorn, M. Bruneau. "Framework for analytical quantification of disaster resilience". In: *Engineering structures* 32.11 (2010), pp. 3639–3649 (cit. on p. 8).

[CU2020] *30+ Chatbot Usecases / Applications in Business in 2020*. 2020. URL: https://research.aimultiple.com/business-chatbot/ (visited on 11/15/2020) (cit. on p. 11).

[CZ19] C. Czepa, U. Zdun. “How Understandable Are Pattern-based Behavioral Constraints for Novice Software Designers?” In: *ACM Transactions on Software Engineering and Methodology (TOSEM)* 28.2 (2019), pp. 1–38 (cit. on p. 1).

[DDF+03] A. Dix, A. J. Dix, J. Finlay, G. D. Abowd, R. Beale. *Human-computer interaction*. Pearson Education, 2003 (cit. on p. 16).

[DE98] A. Dix, G. Ellis. “Starting simple: adding value to static visualisation through simple interaction”. In: *Proceedings of the working conference on Advanced visual interfaces*. 1998, pp. 124–134 (cit. on p. 16).

[DFC] *Dialogflow Concepts*. URL: https://cloud.google.com/dialogflow/docs/concepts (visited on 11/15/2020) (cit. on pp. 12, 53).

[Die07] S. Diehl. *Software visualization: visualizing the structure, behaviour, and evolution of software*. Springer Science & Business Media, 2007 (cit. on p. 17).

[Dja18] Django Software Foundation. *Django Channels*. 2018. URL: https://channels.readthedocs.io/ (visited on 11/25/2020) (cit. on p. 53).

[Dja20] Django Software Foundation. *Django: The web framework for perfectionists with deadlines*. 2020. URL: https://www.djangoproject.com/ (visited on 11/25/2020) (cit. on p. 53).

[DRB16] D. G. Dessavre, J. E. Ramirez-Marquez, K. Barker. “Multidimensional approach to complex system resilience analysis”. In: *Reliability Engineering & System Safety* 149 (2016), pp. 34–43 (cit. on p. 21).

[Few09] S. Few. *Now you see it: simple visualization techniques for quantitative analysis*. Sirsi) i9780970601988. 2009 (cit. on p. 15).

[FKH15] F. Fittkau, A. Krause, W. Hasselbring. “Exploring software cities in virtual reality”. In: *2015 ieee 3rd working conference on software visualization (vissoft)*. IEEE. 2015, pp. 130–134 (cit. on p. 19).

[FL14] M. Fowler, J. Lewis. *Microservices: A definition of this new architectural term*. Visited: 2020-02-02. 2014. URL: https://martinfowler.com/articles/microservices.html (cit. on p. 5).

[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. 8).

[FR91] T. M. Fruchterman, E. M. Reingold. “Graph drawing by force-directed placement”. In: *Software: Practice and experience* 21.11 (1991), pp. 1129–1164 (cit. on p. 45).

[Fri16] U. Friedrichsen. *Resilient Software Design - Robuste Software Entwickeln*. 2016. URL: https://www.informatik-aktuell.de/entwicklung/methoden/resilient-software-design-robuste-software-entwickeln.html (visited on 11/14/2020) (cit. on p. 7).

[GB08] S. Greenberg, B. Buxton. “Usability evaluation considered harmful (some of the time)”. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. 2008, pp. 111–120 (cit. on pp. 64, 82).

[GC08] Y. Ghanam, S. Carpendale. “A survey paper on software architecture visualization”. In: *University of Calgary, Tech. Rep* (2008) (cit. on p. 17).

[GME05] D. Gračanin, K. Matković, M. Eltoweissy. “Software visualization”. In: *Innovations in Systems and Software Engineering* 1.2 (2005), pp. 221–230 (cit. on p. 17).

[Goa19] L. Goasduff. *Chatbots Will Appeal to Modern Workers*. 2019. URL: https://www.gartner.com/smarterwithgartner/chatbots-will-appeal-to-modern-workers/ (visited on 11/15/2020) (cit. on p. 11).

[Goo20] Google. *Dialogflow*. 2020. URL: https://cloud.google.com/dialogflow (visited on 11/15/2020) (cit. on p. 13).

[HBC+92] T. T. Hewett, R. Baecker, S. Card, T. Carey, J. Gasen, M. Mantei, G. Perlman, G. Strong, W. Verplank. *ACM SIGCHI curricula for human-computer interaction*. ACM, 1992 (cit. on pp. 9, 10).

[HBE96] C. G. Healey, K. S. Booth, J. T. Enns. “High-speed visual estimation using preattentive processing”. In: *ACM Transactions on Computer-Human Interaction (TOCHI)* 3.2 (1996), pp. 107–135 (cit. on p. 15).

[HBR16] S. Hosseini, K. Barker, J. E. Ramirez-Marquez. “A review of definitions and measures of system resilience”. In: *Reliability Engineering & System Safety* 145 (2016), pp. 47–61 (cit. on p. 8).

[HF10] J. Humble, D. Farley. *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Pearson Education, 2010 (cit. on pp. 6, 7).

[Hol73] C. S. Holling. “Resilience and stability of ecological systems”. In: *Annual review of ecology and systematics* 4.1 (1973), pp. 1–23 (cit. on p. 8).

[HS88] S. G. Hart, L. E. Staveland. “Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research”. In: *Advances in psychology*. Vol. 52. Elsevier, 1988, pp. 139–183 (cit. on p. 69).

[IBM20] IBM. *IBM Watson Assistant*. 2020. URL: https://www.ibm.com/cloud/watson-assistant (visited on 11/15/2020) (cit. on p. 13).

[IIC+13] T. Isenberg, P. Isenberg, J. Chen, M. Sedlmair, T. Möller. “A systematic review on the practice of evaluating visualization”. In: *IEEE Transactions on Visualization and Computer Graphics* 19.12 (2013), pp. 2818–2827 (cit. on pp. 64, 82).

[JLW11] J. Jain, A. Lund, D. Wixon. “The future of natural user interfaces”. In: *CHI’11 Extended Abstracts on Human Factors in Computing Systems*. 2011, pp. 211–214 (cit. on p. 10).

[KDMB17] L. C. Klopfenstein, S. Delpriori, S. Malatini, A. Bogliolo. “The rise of bots: A survey of conversational interfaces, patterns, and paradigms”. In: *Proceedings of the 2017 conference on designing interactive systems*. 2017, pp. 555–565 (cit. on p. 11).

[Kri10] M. Krigsman. *Cloud-based IT failure halts Virgin flights*. 2010. URL: https://www.zdnet.com/article/cloud-based-it-failure-halts-virgin-flights/ (visited on 11/23/2020) (cit. on p. 1).

[KWK+20a] D. Kesim, L. Wagner, J. von Kistowsiki, S. Frank, A. Hakamian, A. van Hoorn. “Scenario-based Resilience Evaluation and Improvement of Microservice Architectures: An Experience Report”. In: Under review. 2020 (cit. on pp. 64, 65).

[KWK+20b] D. Kesim, L. Wagner, J. von Kistowsiki, S. Frank, A. Hakamian, A. van Hoorn. *Scenario-based Resilience Evaluation and Improvement of Microservice Architectures: An Experience Report (Supplementary Material)*. 2020. URL: https://codeocean.com/capsule/9354020/ (visited on 11/24/2020) (cit. on p. 64).

[Lyu07] M. R. Lyu. “Software reliability engineering: A roadmap”. In: *Future of Software Engineering (FOSE’07)*. IEEE. 2007, pp. 153–170 (cit. on p. 7).

[Mac86] J. Mackinlay. “Automating the design of graphical presentations of relational information”. In: *Acm Transactions On Graphics (Tog)* 5.2 (1986), pp. 110–141 (cit. on pp. 14, 15).

[Mar20] B. Marr. *Tech Trends in Practice: The 25 Technologies that are Driving the 4th Industrial Revolution*. John Wiley Sons, 2020 (cit. on p. 13).

[MBN18] L. Merino, A. Bergel, O. Nierstrasz. “Overcoming issues of 3D software visualization through immersive augmented reality”. In: *2018 IEEE Working Conference on Software Visualization (VISSOFT)*. IEEE. 2018, pp. 54–64 (cit. on p. 19).

[MGAN17] L. Merino, M. Ghafari, C. Anslow, O. Nierstrasz. “CityVR: Gameful software visualization”. In: *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE. 2017, pp. 633–637 (cit. on pp. 1, 9).

[MGAN18] L. Merino, M. Ghafari, C. Anslow, O. Nierstrasz. “A systematic literature review of software visualization evaluation”. In: *Journal of systems and software* 144 (2018), pp. 165–180 (cit. on p. 18).

[Mic19] Microsoft. *Microsoft Bot Framework*. 2019. URL: https://dev.botframework.com/ (visited on 11/15/2020) (cit. on p. 13).

[Mon15] A. Montalenti. *Kafkapocalypse: a postmortem on our service outage*. 2015. URL: https://blog.parse.ly/post/1738/kafkapocalypse/ (visited on 11/23/2020) (cit. on p. 37).

[Moz19] Mozilla. *WebSockets*. 2019. URL: https://developer.mozilla.org/de/docs/WebSockets (visited on 11/25/2020) (cit. on p. 53).

[NC93] L. Nigay, J. Coutaz. “A design space for multimodal systems: concurrent processing and data fusion”. In: *Proceedings of the INTERACT’93 and CHI’93 conference on Human factors in computing systems*. 1993, pp. 172–178 (cit. on p. 11).

[New15] S. Newman. *Building microservices: designing fine-grained systems*. Ö’Reilly Media, Inc.”, 2015 (cit. on pp. 1, 5, 6).

[Nyg18] M. T. Nygard. *Release it!: design and deploy production-ready software*. Pragmatic Bookshelf, 2018 (cit. on pp. 6, 7).

[OBM+20] D. Okanović, S. Beck, L. Merz, C. Zorn, L. Merino, A. van Hoorn, F. Beck. “Can a Chatbot Support Software Engineers with Load Testing? Approach and Experiences”. In: *Proceedings of the ACM/SPEC International Conference on Performance Engineering*. 2020, pp. 120–129 (cit. on p. 53).

[Pos20] PostgreSQL Global Development Group. *PostgreSQL: The World’s Most Advanced Open Source Relational Database*. 2020. URL: https://www.postgresql.org/ (visited on 11/25/2020) (cit. on p. 54).

[PRD07] R. Poppe, R. Rienks, B. van Dijk. “Evaluating the future of HCI: challenges for the evaluation of emerging applications”. In: *Artifical Intelligence for Human Computing*. Springer, 2007, pp. 234–250 (cit. on p. 18).

[PV18] E. Paikari, A. Van Der Hoek. “A framework for understanding chatbots and their future”. In: *2018 IEEE/ACM 11th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. IEEE. 2018, pp. 13–16 (cit. on p. 11).

[R4J] *CircuitBreaker*. URL: https://resilience4j.readme.io/docs/circuitbreaker (visited on 11/14/2020) (cit. on p. 7).

[RLL+04] L. M. Reeves, J. Lai, J. A. Larson, S. Oviatt, T. Balaji, S. Buisine, P. Collings, P. Cohen, B. Kraal, J.-C. Martin, et al. “Guidelines for multimodal user interface design”. In: *Communications of the ACM* 47.1 (2004), pp. 57–59 (cit. on p. 11).

[RLN07] R. Rosenholtz, Y. Li, L. Nakano. “Measuring visual clutter”. In: *Journal of vision* 7.2 (2007), pp. 17–17 (cit. on p. 45).

[Sch08] M. Scherr. “Multiple and coordinated views in information visualization”. In: *Trends in Information Visualization* 38 (2008), pp. 1–33 (cit. on p. 49).

[SDG+16] T. Savor, M. Douglas, M. Gentili, L. Williams, K. Beck, M. Stumm. “Continuous deployment at Facebook and OANDA”. In: *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*. IEEE. 2016, pp. 21–30 (cit. on p. 7).

[SH16] D. Schmalstieg, T. Hollerer. *Augmented reality: principles and practice*. Addison-Wesley Professional, 2016 (cit. on p. 13).

[Shn96] B. Shneiderman. “The eyes have it: A task by data type taxonomy for information visualizations”. In: *Proceedings 1996 IEEE symposium on visual languages*. IEEE. 1996, pp. 336–343 (cit. on pp. 16, 49).

[SLW19] A. Shakil, C. Lutteroth, G. Weber. “CodeGazer: Making Code Navigation Easy and Natural with Gaze Input”. In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 2019, pp. 1–12 (cit. on pp. 1, 9).

[SPC+16] B. Shneiderman, C. Plaisant, M. Cohen, S. Jacobs, N. Elmqvist, N. Diakopoulos. *Designing the user interface: strategies for effective human-computer interaction*. Pearson, 2016 (cit. on p. 9).

[SSKB14] B. Saket, P. Simonetto, S. Kobourov, K. Börner. “Node, node-link, and node-link-group diagrams: An evaluation”. In: *IEEE Transactions on Visualization and Computer Graphics* 20.12 (2014), pp. 2231–2240 (cit. on p. 45).

[SSS+19] P. Seipel, A. Stock, S. Santhanam, A. Baranowski, N. Hochgeschwender, A. Schreiber. “Speak to your software visualization—exploring component-based software architectures in augmented reality with a conversational interface”. In: *2019 Working Conference on Software Visualization (VISSOFT)*. IEEE. 2019, pp. 78–82 (cit. on p. 19).

[Ste+46] S. S. Stevens et al. “On the theory of scales of measurement”. In: (1946) (cit. on p. 14).

[SZ16] M.-A. Storey, A. Zagalsky. “Disrupting developer productivity one bot at a time”. In: *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. 2016, pp. 928–931 (cit. on pp. 11, 49).

[Tho08] R. Thomson. *British Airways reveals what went wrong with Terminal 5*. 2008. URL: https://www.computerweekly.com/news/2240086013/British-Airways-reveals-what-went-wrong-with-Terminal-5 (visited on 11/23/2020) (cit. on p. 1).

[TMC18] C. Toxtli, A. Monroy-Hernández, J. Cranshaw. “Understanding chatbot-mediated task management”. In: *Proceedings of the 2018 CHI conference on human factors in computing systems*. 2018, pp. 1–6 (cit. on pp. 2, 9).

[Tre85] A. Treisman. “Preattentive processing in vision”. In: *Computer vision, graphics, and image processing* 31.2 (1985), pp. 156–177 (cit. on p. 15).

[Tur14] M. Turk. “Multimodal interaction: A review”. In: *Pattern Recognition Letters* 36 (2014), pp. 189–195 (cit. on pp. 10, 11).

[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. 38).

[VBCR02] R. Van Solingen, V. Basili, G. Caldiera, H. D. Rombach. “Goal question metric (gqm) approach”. In: *Encyclopedia of software engineering* (2002) (cit. on pp. 37, 38).

[Ven14] V. A. Venikov. *Transient Phenomena in Electrical Power Systems: International Series of Monographs on Electronics and Instrumentation, Vol. 24*. Vol. 24. Elsevier, 2014 (cit. on p. 5).

[Vog14] W. Vogels. *The story of Apollo - Amazon’s deployment engine*. Visited: 2020-02-02. 2014. URL: https://www.allthingsdistributed.com/2014/11/apollo-amazon-deployment-engine.html (cit. on p. 7).

[VP10] D. Van Krevelen, R. Poelman. “A survey of augmented reality technologies, applications and limitations”. In: *International journal of virtual reality* 9.2 (2010), pp. 1–20 (cit. on p. 13).

[Wal09] R. S. Wallace. “The anatomy of ALICE”. In: *Parsing the Turing Test*. Springer, 2009, pp. 181–210 (cit. on p. 11).

[WC17] Weaveworks, Container Solutions. *SockShop: A Microservices Demo Application*. 2017. URL: https://microservices-demo.github.io/ (visited on 11/24/2020) (cit. on pp. 64, 66).

[Wei66] J. Weizenbaum. “ELIZA—a computer program for the study of natural language communication between man and machine”. In: *Communications of the ACM* 9.1 (1966), pp. 36–45 (cit. on p. 11).

[Wes+96] D. B. West et al. *Introduction to graph theory*. Vol. 2. Prentice hall Upper Saddle River, NJ, 1996 (cit. on p. 45).

[WL07] R. Wettel, M. Lanza. “Visualizing software systems as cities”. In: *2007 4th IEEE International Workshop on Visualizing Software for Understanding and Analysis*. IEEE. 2007, pp. 92–99 (cit. on pp. 18, 19).

[Wol18] S. Wolfe. *Amazon's one hour of downtime on Prime Day may have cost it up to $100 million in lost sales*. 2018. URL: https://www.businessinsider.com/amazon-prime-day-website-issues-cost-it-millions-in-lost-sales-2018-7 (visited on 11/23/2020) (cit. on p. 1).

[WRH+12] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, A. Wesslén. *Experimentation in software engineering*. Springer Science & Business Media, 2012 (cit. on pp. 18, 63).

[WW11] D. Wigdor, D. Wixon. *Brave NUI world: designing natural user interfaces for touch and gesture*. Elsevier, 2011 (cit. on p. 10).

[XWY+19] K. Xu, Y. Wang, L. Yang, Y. Wang, B. Qiao, S. Qin, Y. Xu, H. Zhang, H. Qu. "Cloud-Det: Interactive Visual Analysis of Anomalous Performances in Cloud Computing Systems". In: *IEEE transactions on visualization and computer graphics* 26.1 (2019), pp. 1107–1117 (cit. on p. 64).

[YDW+19] K. Yin, Q. Du, W. Wang, J. Qiu, J. Xu. "On Representing and Eliciting Resilience Requirements of Microservice Architecture Systems". In: *arXiv preprint arXiv:1909.13096* (2019) (cit. on pp. 2, 6, 8, 9, 16, 22).

[YW16] N. Yodo, P. Wang. "Engineering resilience quantification and system design implications: A literature survey". In: *Journal of Mechanical Design* 138.11 (2016) (cit. on p. 8).

[ZNR+18] H. Züllighoven, J. Nitschke, E. Reiswich, M. Kowalczyk, H. Schwentner. "Vom Design digitaler Arbeitsgegenstände: Planung des Schiffverkehrs im Hamburger Hafen mit dem Peiltisch". In: *Mensch und Computer 2018-Usability Professionals* (2018) (cit. on pp. 1, 9, 19).

[Zob10] C. W. Zobel. "Comparative visualization of predicted disaster resilience". In: *Proceedings of the 7th International ISCRAM Conference*. ISCRAM. 2010, pp. 1–5 (cit. on pp. 20, 40, 46, 51).

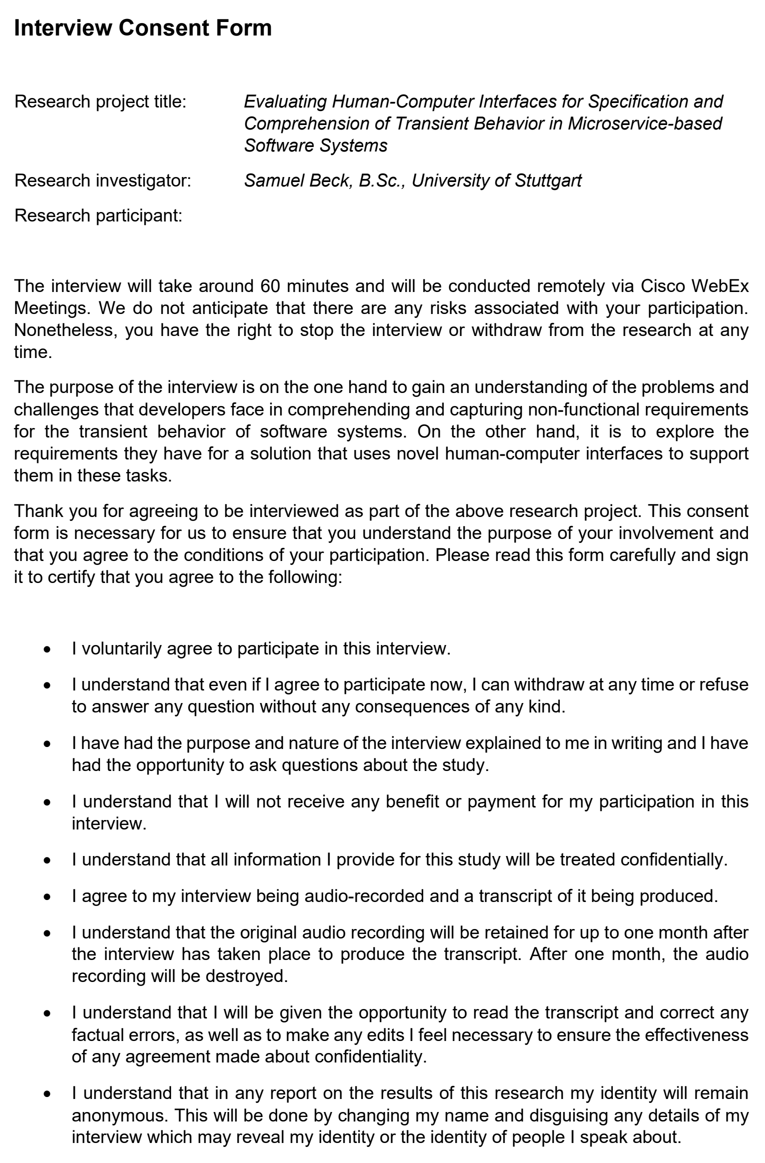
All links were last followed on November 26, 2020.

A Interview Resources

This chapter presents resources that have been used for the expert interviews. It is structured as follows.

Appendix A.1 – Consent Form: The consent form that was used for the interviews.

A.1 Consent Form



Interview Consent Form

Research project title:	*Evaluating Human-Computer Interfaces for Specification and Comprehension of Transient Behavior in Microservice-based Software Systems*
Research investigator:	*Samuel Beck, B.Sc., University of Stuttgart*
Research participant:	

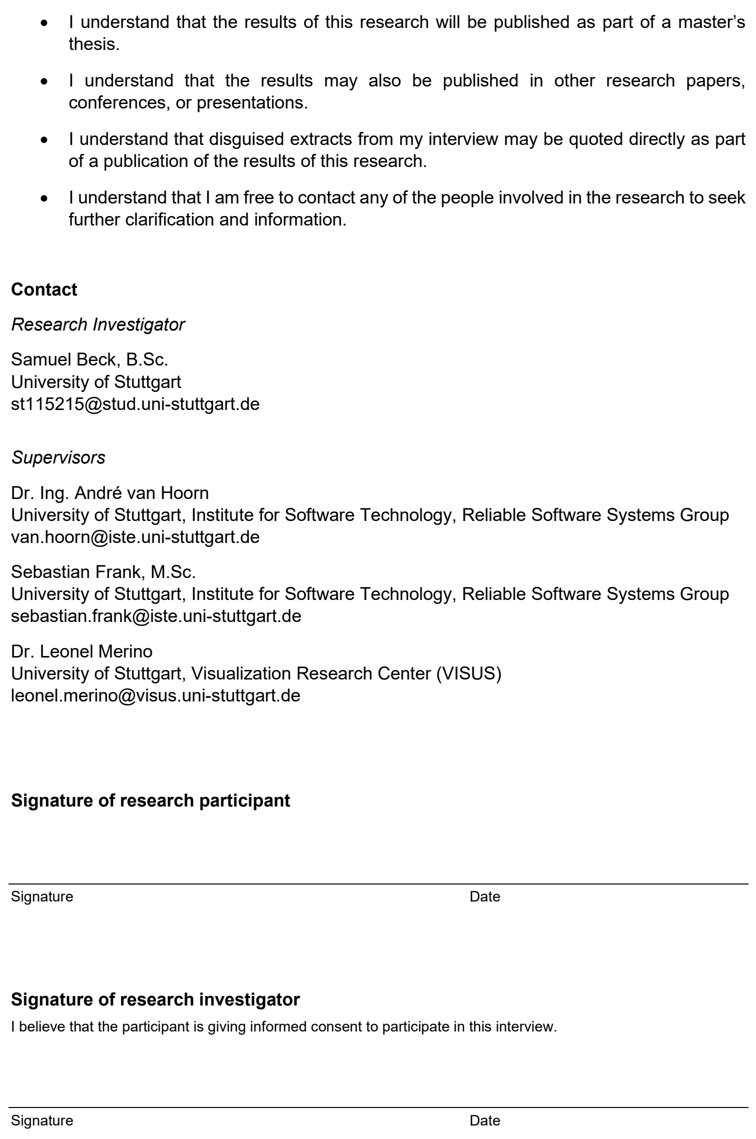
The interview will take around 60 minutes and will be conducted remotely via Cisco WebEx Meetings. We do not anticipate that there are any risks associated with your participation. Nonetheless, you have the right to stop the interview or withdraw from the research at any time.

The purpose of the interview is on the one hand to gain an understanding of the problems and challenges that developers face in comprehending and capturing non-functional requirements for the transient behavior of software systems. On the other hand, it is to explore the requirements they have for a solution that uses novel human-computer interfaces to support them in these tasks.

Thank you for agreeing to be interviewed as part of the above research project. This consent form is necessary for us to ensure that you understand the purpose of your involvement and that you agree to the conditions of your participation. Please read this form carefully and sign it to certify that you agree to the following:

- I voluntarily agree to participate in this interview.
- I understand that even if I agree to participate now, I can withdraw at any time or refuse to answer any question without any consequences of any kind.
- I have had the purpose and nature of the interview explained to me in writing and I have had the opportunity to ask questions about the study.
- I understand that I will not receive any benefit or payment for my participation in this interview.
- I understand that all information I provide for this study will be treated confidentially.
- I agree to my interview being audio-recorded and a transcript of it being produced.
- I understand that the original audio recording will be retained for up to one month after the interview has taken place to produce the transcript. After one month, the audio recording will be destroyed.
- I understand that I will be given the opportunity to read the transcript and correct any factual errors, as well as to make any edits I feel necessary to ensure the effectiveness of any agreement made about confidentiality.
- I understand that in any report on the results of this research my identity will remain anonymous. This will be done by changing my name and disguising any details of my interview which may reveal my identity or the identity of people I speak about.

Figure A.1: Interview consent form page 1.



- I understand that the results of this research will be published as part of a master's thesis.
- I understand that the results may also be published in other research papers, conferences, or presentations.
- I understand that disguised extracts from my interview may be quoted directly as part of a publication of the results of this research.
- I understand that I am free to contact any of the people involved in the research to seek further clarification and information.

Contact

Research Investigator

Samuel Beck, B.Sc.
University of Stuttgart
st115215@stud.uni-stuttgart.de

Supervisors

Dr. Ing. André van Hoorn
University of Stuttgart, Institute for Software Technology, Reliable Software Systems Group
van.hoorn@iste.uni-stuttgart.de

Sebastian Frank, M.Sc.
University of Stuttgart, Institute for Software Technology, Reliable Software Systems Group
sebastian.frank@iste.uni-stuttgart.de

Dr. Leonel Merino
University of Stuttgart, Visualization Research Center (VISUS)
leonel.merino@visus.uni-stuttgart.de

Signature of research participant

Signature Date

Signature of research investigator

I believe that the participant is giving informed consent to participate in this interview.

Signature Date

Figure A.2: Interview consent form page 2.

B Evaluation Resources

This chapter presents resources that have been used for the evaluation. It is structured as follows.

Appendix B.1 – Consent Form: The consent form that was used for the user study.

Appendix B.2 – Information: Information that was handed out to the participants at the beginning of the session.

Appendix B.3 – Task Descriptions: The tasks that the participants had to work on.

Appendix B.4 – Questionnaire: The questionnaire that was used for the study.

B.1 Consent Form

B.2 Information

At the beginning of each study session, the participant read the following information about the respective test system, transient behavior, and the prototype.

B.2.1 Payroll Accounting System

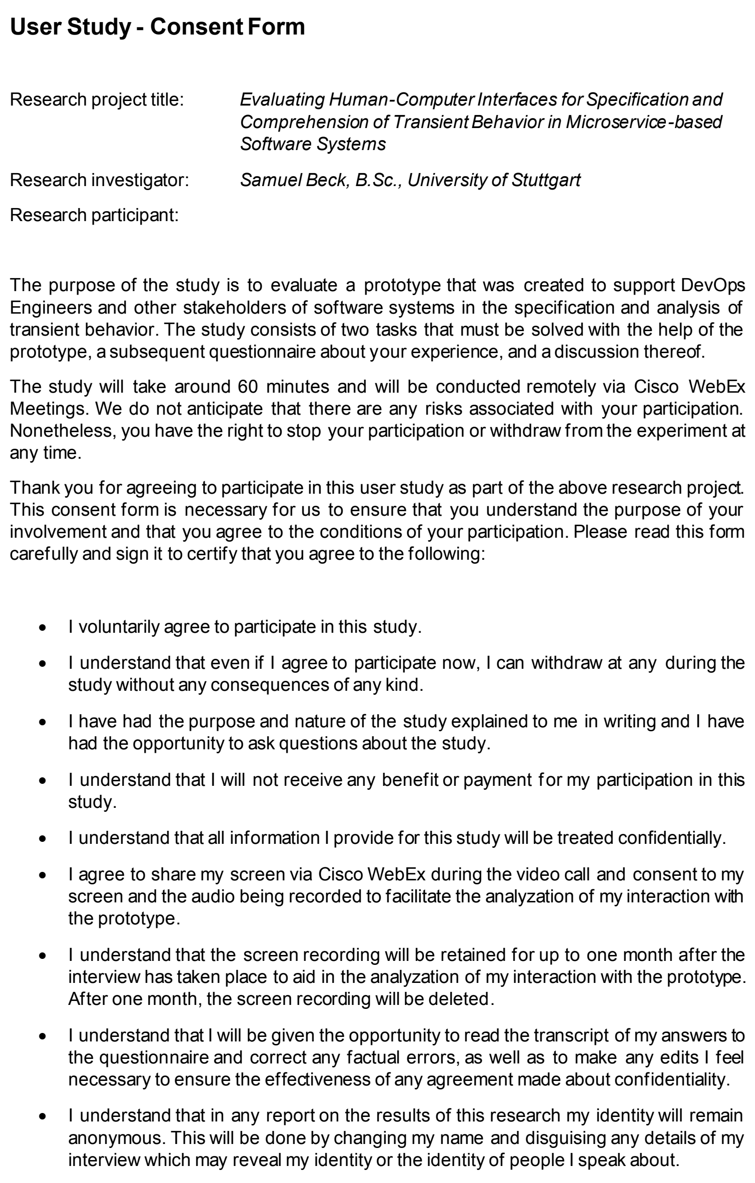
The mocked payroll accounting system consists of four services: `API-Gateway`, `payslip`, `payslip2`, and `Eureka`. Its architecture is depicted in Figure B.3.

The `API-Gateway` service manages all incoming requests and forwards them to instances of `payslip` and `payslip2`. The `Eureka` service provides service discovery for the system. The `payslip` service utilizes an in-memory database and a third-party API. Furthermore, it can forward requests to `payslip2`. Requests can also be directly sent to `payslip2`.

The endpoints of `payslip` and `payslip2` are recorded in Table B.1.

B.2.2 SockShop

The Sock Shop microservice system consists of seven services: `Payment`, `Orders`, `Carts`, `Catalogue`, `Users`, and `Shipping`. Its architecture is depicted in Figure B.4.



User Study - Consent Form

Research project title:	*Evaluating Human-Computer Interfaces for Specification and Comprehension of Transient Behavior in Microservice-based Software Systems*
Research investigator:	*Samuel Beck, B.Sc., University of Stuttgart*
Research participant:	

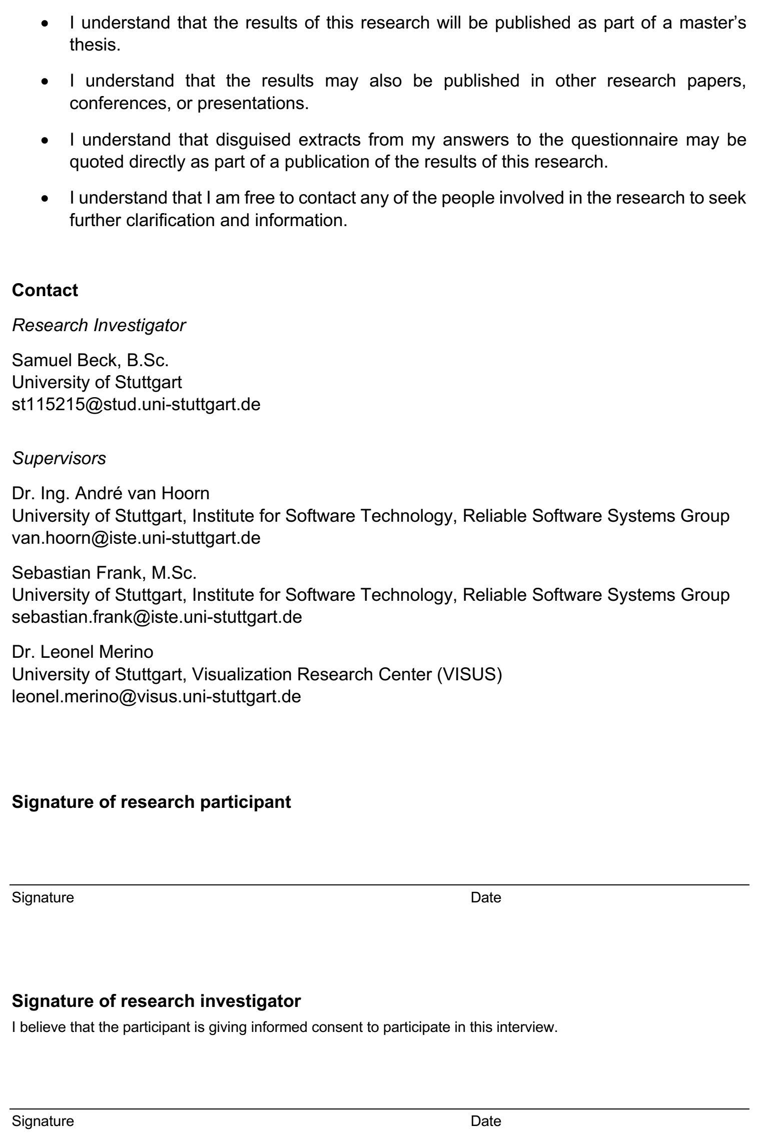
The purpose of the study is to evaluate a prototype that was created to support DevOps Engineers and other stakeholders of software systems in the specification and analysis of transient behavior. The study consists of two tasks that must be solved with the help of the prototype, a subsequent questionnaire about your experience, and a discussion thereof.

The study will take around 60 minutes and will be conducted remotely via Cisco WebEx Meetings. We do not anticipate that there are any risks associated with your participation. Nonetheless, you have the right to stop your participation or withdraw from the experiment at any time.

Thank you for agreeing to participate in this user study as part of the above research project. This consent form is necessary for us to ensure that you understand the purpose of your involvement and that you agree to the conditions of your participation. Please read this form carefully and sign it to certify that you agree to the following:

- I voluntarily agree to participate in this study.
- I understand that even if I agree to participate now, I can withdraw at any during the study without any consequences of any kind.
- I have had the purpose and nature of the study explained to me in writing and I have had the opportunity to ask questions about the study.
- I understand that I will not receive any benefit or payment for my participation in this study.
- I understand that all information I provide for this study will be treated confidentially.
- I agree to share my screen via Cisco WebEx during the video call and consent to my screen and the audio being recorded to facilitate the analyzation of my interaction with the prototype.
- I understand that the screen recording will be retained for up to one month after the interview has taken place to aid in the analyzation of my interaction with the prototype. After one month, the screen recording will be deleted.
- I understand that I will be given the opportunity to read the transcript of my answers to the questionnaire and correct any factual errors, as well as to make any edits I feel necessary to ensure the effectiveness of any agreement made about confidentiality.
- I understand that in any report on the results of this research my identity will remain anonymous. This will be done by changing my name and disguising any details of my interview which may reveal my identity or the identity of people I speak about.

Figure B.1: User study consent form page 1.



- I understand that the results of this research will be published as part of a master's thesis.
- I understand that the results may also be published in other research papers, conferences, or presentations.
- I understand that disguised extracts from my answers to the questionnaire may be quoted directly as part of a publication of the results of this research.
- I understand that I am free to contact any of the people involved in the research to seek further clarification and information.

Contact

Research Investigator

Samuel Beck, B.Sc.
University of Stuttgart
st115215@stud.uni-stuttgart.de

Supervisors

Dr. Ing. André van Hoorn
University of Stuttgart, Institute for Software Technology, Reliable Software Systems Group
van.hoorn@iste.uni-stuttgart.de

Sebastian Frank, M.Sc.
University of Stuttgart, Institute for Software Technology, Reliable Software Systems Group
sebastian.frank@iste.uni-stuttgart.de

Dr. Leonel Merino
University of Stuttgart, Visualization Research Center (VISUS)
leonel.merino@visus.uni-stuttgart.de

Signature of research participant

Signature	Date

Signature of research investigator

I believe that the participant is giving informed consent to participate in this interview.

Signature	Date

Figure B.2: User study consent form page 2.

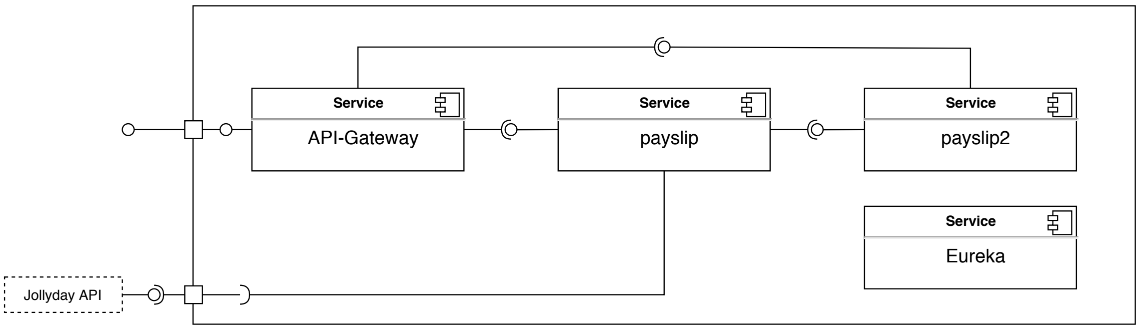


Figure B.3: Architecture of the mocked payroll accounting system.

Name	Endpoint	Description
Internal dependency	/ergebnisdaten	Calls payslip2 via payslip
Database read	/abwesenheiten	Reads entry from db of payslip
External dependency	/service/api/v1/holidays	Calls Jollyday API via payslip
Database write	/abwesenheiten	Writes entry into db of payslip
Gateway ping	-	Checks if API-Gateway is alive
Unaffected service	/abwesenheiten/abwesenheitenId	Send request directly to payslip2

Table B.1: Endpoints of payslip and payslip2.

The `Frontend` service connects all the other services. The `Catalogue` service provides information about the catalogue and products. The `Order` service provides ordering capabilities to the system and communicates with the `Shipping` service that provides shipping capabilities. The `Cart` service provides shopping carts for users. The `User` service is responsible for user account storage, including credit cards and addresses. Finally, the `Payment` service provides payment functionality.

The endpoints of the services are recorded in Table B.2.

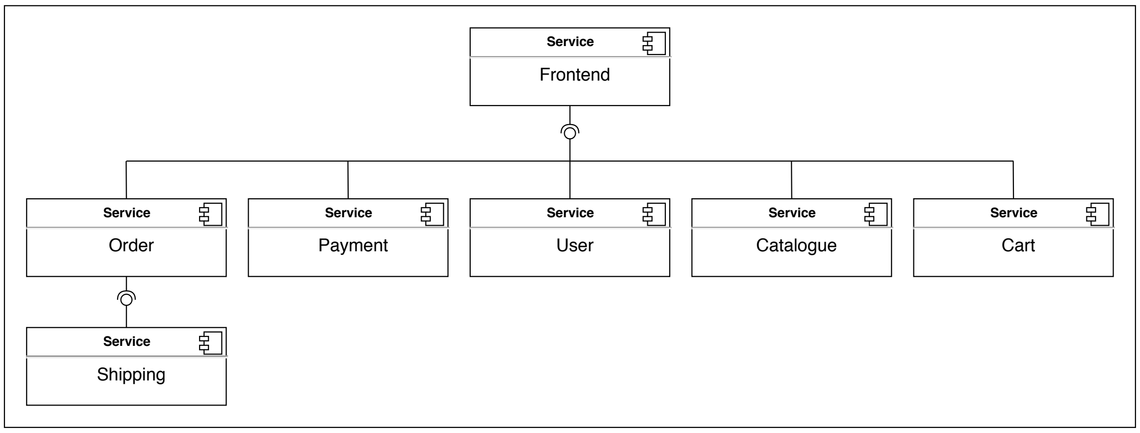


Figure B.4: Architecture of the SockShop system.

Service	Endpoint
Frontend	home
	catalogue
	showDetails
	basket
	viewOrdersPage
Catalogue	getCatalogue
	catalogueSize
	cataloguePage
	getItem
	getRelated
	tags
Order	createOrder
	getOrders
Cart	getCart
	addToCart
User	login
	getCustomer
	getCard
	getAddress

Table B.2: Endpoints of the services of SockShop.

B.2.3 Transient Behavior

Transient behavior occurs in a software system while the system is changing from one steady state to another. Such changes can for example be failures, deployments or a load balancer starting or stopping service instances. Modern software systems are constantly subjected to such changes. This means that they are repeatedly in this state of transient behavior in which the quality of the system temporarily deviates from its defined steady state.

A popular visualization for the impact that such changes have on a system are so called resilience curves. They visualize the quality of a system over time. An instance of a resilience curve can be found in Figure B.5

Ordinarily, the quality of the system is at 100%. When an event occurs that impacts the quality of the system, the quality usually declines drastically at the beginning of this event. This decline of quality is also called initial loss. It indicates by how much the quality of the system is degraded by this event.

Eventually, the system starts to recover from the impact of the event and the quality of the system increases again until it finally returns to 100%. The duration from the start of the event until the complete recovery of the system is called recovery time. Together, the initial loss and the recovery time can be used to characterize how resilient a system is to an event by indicating how much the quality of the system degrades due to the event and how fast the system can recover from the event.

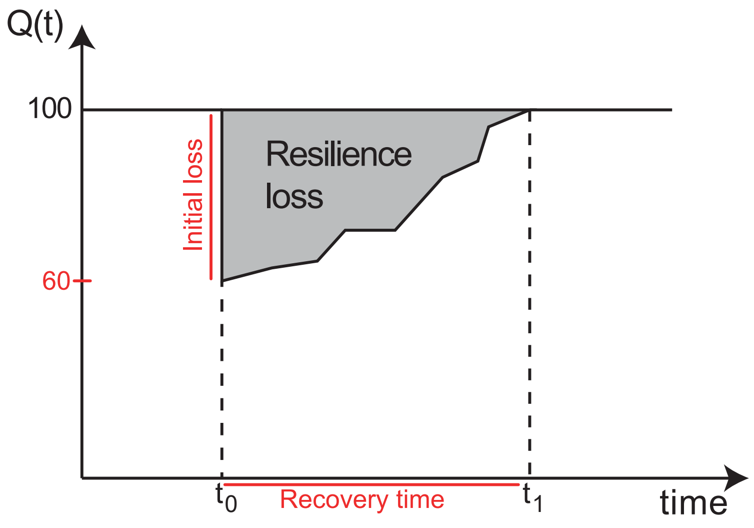


Figure B.5: Resilience curve with indication of initial loss, recovery time, and resilience loss.

The resilience curve can be reduced to a triangle formed by the initial loss and the recovery time.The area of this triangle (or the area above the resilience curve) can be used to measure the loss of resilience caused by an event to a system. The smaller this loss of resilience is, the more resilient the system.

B.2.4 Prototype

The prototype consists of four main components.

1. Architecture visualization
2. Transient behavior visualization
3. Loss of resilience visualization
4. Chatbot

The architecture visualization at the top shows the different services that comprise the system and their dependencies to each other. A service can be selected by clicking on it or by asking the chatbot to do so. When a service might be violating a specification for transient behavior it is highlighted in orange.

When a service is selected, the transient behavior visualization below it shows the monitoring data for the selected endpoint of that service. The endpoint can be selected in a selection above the visualization. The chart shows a quality of service metric which is a utility function. In this case it simply corresponds to the rate of successful requests processed by the service.

On the right side above the transient behavior visualization is a button that toggles the visualization of specifications that have been created for this service. If specifications are shown, they can be added or deleted with a button. Next to the endpoint selection is another selection box that allows to select the cause for transient behavior for which a specification should be created or visualized.Specifications are shown directly in the transient behavior visualization as a red line.

When specifications are toggled, a loss of resilience visualization chart is shown at the bottom that illustrates the loss of resilience of each transient behavior occurrence at the selected endpoint of the service. A red line shows the maximum loss of resilience that was specified for the selected cause for the service.

The chatbot on the bottom right can be used to interact with the visualizations and to show, create, delete, and edit specifications for different transient behaviors. You simply have to write your command or request in the chat field.

B.3 Task Descriptions

B.3.1 Payroll Accounting System

Task 1

You are a DevOps-Engineer whose task it is to monitor the transient behavior of the payroll accounting system.

During your shift, an internal error occurs in one of the instances of the payslip service while the system is running. This error leads to the termination of the service instance. Consequently, a new instance of paysip is automatically deployed and started.

You agreed with your customer that the system should be able to uphold quality of service during such a failure. It was specified that the service quality of the payslip service should never drop below 60%. Furthermore, the service must have recovered from the incident within two minutes and return to a quality of service of 100% after that time. This applies to all API endpoints of the service.

After the incident, your task is now to evaluate whether the system was able to fulfill these specifications for all the service's endpoints.

Use the prototype to analyze the monitoring data of the different endpoints, enter the mentioned specification for transient behavior after a failure, and examine for each service endpoint whether the specification is met.

Task 2

After the previous incident, your team decided to implement a resilience pattern in the payslip service, which should make the system more resilient to failures. The pattern you have chosen is the retry pattern. If a request fails, the system will attempt to retry it multiple times after a short time interval. This gives the system some time to start a new service instance that might be able to process the request.

Your assumption is that you should be able to meet the specification for the payslip service in case of a failure with the addition of the retry pattern. The specification that you agreed to with your customer is the same as before. Even in the case of an instance failure, the quality of service of payslip should never fall below 60% and the service must have recovered back to 100% after two minutes.

You already conducted a chaos experiment in which an instance of the payslip service was terminated during runtime to investigate whether the system now actually meets the requirements of this specification.

Use the prototype to again analyze the monitoring data of the different endpoints. Again, enter the specification for transient behavior after a failure and examine for each service endpoint whether the specification is now met.

B.3.2 SockShop

Task 1

You are a DevOps-Engineer whose task it is to monitor the transient behavior of the Sock Shop system.

During your shift, your system was attacked with a DDoS attack. The increased load led to a degradation of the quality of service of some services.

You agreed with your customer that the system should uphold quality of service during such an attack. It was specified that the service quality of the Cart service should never drop below 40%. Furthermore, the service must have recovered from the incident within three minutes and return to a quality of service of 100% after that time The service quality of the User service should equally never drop below 40%. It must fully recover from an incident within 90 seconds. These specifications apply to each endpoint of the respective service.

After the incident, your task is now to evaluate whether the system was able to fulfill these specifications.

Use the prototype to analyze the monitoring data of the different endpoints for each service, enter the mentioned specifications for transient behavior caused by increased load, and examine for each service endpoint whether the specification is met.

Task 2

After the previous incident, your team worked hard to improve the system's protection against DDoS attacks. As it turns out, this was a good precaution. Another DDoS attack was just launched against the system this morning.

Your task is to evaluate whether the improvements are effective and if the system was able to stay within the specifications this time.

The specifications are the same as in the first task. The service quality of the Cart service should never drop below 40%. The service must have recovered back to a quality of service of 100% after three minutes. The service quality of the User service should equally never drop below 40%. It must fully recover from an incident within 90 seconds. Again, these specifications apply to each endpoint of the respective service.

Use the prototype again to analyze the monitoring data of the different endpoints for each service. Again, enter the specifications for transient behavior caused by increased load, and examine for each service endpoint whether the specification was met during this incident.

B.4 Questionnaire

For each statement, indicate on a scale from one (strongly disagree) to five (strongly agree) which best describes your reactions to the prototype that you used today.

1. I think that I would use a tool like this for my work.
2. I thought the tool was easy to use.
3. I found the tool unnecessarily complex.
4. I found the various functions of the tool were well integrated.
5. I thought there was too much inconsistency in the tool.
6. I would imagine that most people would learn to use the tool very quickly.
7. I found the tool very cumbersome (awkward) to use.
8. I felt very confident using the tool.
9. I needed to learn a lot of things before I could get going with the tool.
10. I think the included visualizations were easy to understand.
11. I think the included visualizations helped me to solve the task.
12. the integration of a chatbot in the tool was helpful.

For each question, give an answer on a scale from one (very low) to five (very high).

13. How mentally demanding was it to solve the tasks using the prototype?
14. How successful were you in accomplishing what you were asked to do? (From one (perfect) to five (Failure))

15. How hard did you have to work to accomplish your level of performance?
16. How insecure, discouraged, irritated, stressed, and annoyed were you?

Open questions:

17. What features of the prototype were most effective and why?
18. What features of the prototype were least effective and why?
19. Why did you think the chatbot was helpful/NOT helpful?
20. How would you have addressed such tasks with your existing toolbox?

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

place, date, signature