FH Vorarlberg
University of Applied Sciences

Integration eines Feature-orientierten Testsystems in den Entwicklungszyklus technischer Systeme

Bachelorarbeit
zum Erlangen des akademischen Grades

Bachelor of Science in Engineering (BSc)

Fachhochschule Vorarlberg
Informatik - Software and Information Engineering

Betreut von
Dipl.-Ing. Dr. techn. Ralph Hoch

Vorgelegt von
Marco Prescher

Dornbirn, am 1. Juli 2023

Kurzreferat

Integration eines Feature-orientierten Testsystems in den Entwicklungszyklus technischer Systeme

Die Entwicklung technischer Systeme ist ein komplexer und kostspieliger Prozess. Daher ist es wichtig, dass die Produkte vor der Auslieferung an den Kunden gezielt und sorgfältig getestet werden. Dies wird durch Zeitdruck und Deadlines oftmals vernachlässigt, oder nur unzureichend durchgeführt. Mit einem gut strukturierten Testplan kann dieses Risiko allerdings minimiert werden, und die Durchführung der Tests mit dem Produktentwicklungszyklus integriert werden. Dadurch kann stabile Hardware sowie effiziente und gut strukturierte Software ohne große Verzögerung an den Kunden ausgeliefert werden.

Durch Methodiken wie zum Beispiel Feature-orientierte Entwicklung ist es möglich, dass bestimmte Features vor dem Release der Produkte abgenommen und getestet werden müssen. Das wiederum ermöglicht, dass neu implementierte Features gründlich getestet werden und somit zu einer hohen Qualität beitragen.

Dies kann erreicht werden, indem Features strukturiert in einer Datenbank eingepflegt werden. Ein auf diesen Featurebeschreibungen basierendes System kann die automatische Testdurchführung unterstützen, gezielt Tests für einzelne Features durchführen und deren Abnahme beschleunigen. Dadurch kann der Testaufwand verringert und den Entwicklern ein fokussiertes Feedback vermittelt werden.

Abstract

Integrating a feature-oriented testing system into the development cycle of technical systems

The development of technical systems is a complex and costly process. Therefore, it is important that products are thoroughly tested before delivery to the customer. However, this is often neglected or done insufficiently due to time pressure and deadlines. A well-structured test plan can minimize this risk and integrate the testing process into the product development cycle. This allows stable hardware and efficient, well-structured software to be delivered to the customer without significant delays.

Using techniques such as feature-oriented development, certain features must be accepted and tested before the product release. This in turn allows newly implemented features to be thoroughly tested and contribute to high quality.

This can be achieved by structuring features in a database. A system based on these feature descriptions can support automatic testing, conduct targeted tests for individual features, and accelerate their acceptance. This reduces testing effort and provides focused feedback to the developers.

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

API Application Programming Interface

JSON JavaScript Object Notation

TCMS Test Case Management System

TDD Test Driven Development

Danksagung

Ich möchte mich aufrichtig bei Ralph Hoch, der Firma Gantner Instruments und allen Mitarbeitenden bedanken, die mir bei der Vollendung dieser Arbeit geholfen haben. Ihre Unterstützung und Expertise waren von unschätzbarem Wert und haben zum erfolgreichen Abschluss dieses Projekts beigetragen. Vielen Dank für Ihre harte Arbeit und Ihr Engagement. Ich schätze Ihre Zusammenarbeit sehr.

1 Einleitung

Proper Einleitung Description

1.1 Motivation

Die Entwicklung technischer Systeme ist ein komplexer Prozess, der eine hohe Qualität erfordert, um den Anforderungen der Kunden gerecht zu werden. Damit diese Qualität auch gewährleistet wird, müssen Fehler sowie Mängel identifiziert und ausgebessert werden. Ein Test Case Management System (TCMS) bietet eine Lösung, um diesen Prozess zu vereinfachen und effektiver zu gestalten. Durch die Verwendung eines TCMS können Angestellte aus verschiedenen Abteilungen, wie z.B. der Software-, Hardware-, Support- oder Marketingabteilung, die Qualität des Produkts gemeinsam verbessern. Diese Arbeit fokussiert sich daher auf die Integration von einem TCMS in einen laufenden Produktentwicklungs-Zyklus und vergleicht verschiedene vorhandene Lösungen.

1.2 Problemstellung

Durch die von Abschnitt 1.1 angesprochene Komplexität technischer Systeme ist bekannt, dass Fehler, die erst spät im Entwicklungsprozess entdeckt werden, viel kostspieliger zu beheben sind als Fehler, die frühzeitig identifiziert und behoben werden. Infolgedessen suchen Unternehmen nach Lösungen, um den Testprozess effektiver zu gestalten und Fehler früher im Entwicklungszyklus zu identifizieren. TCMS bietet eine solche Lösung, indem es Entwicklern und Testern ermöglicht, Testfälle effizient zu planen und zu verwalten sowie Testergebnisse zu erfassen, darzustellen und somit auch zu verfolgen. Obwohl Test Case Management Systeme in der Industrie weit verbreitet sind und es einige fertige Lösungen gibt, gibt es jedoch nicht immer die Perfekte Lösung um ein bestehendes TCMS für das eigene Projekt anzuwenden. Insbesondere gibt

es Bedenken hinsichtlich der Anwendbarkeit von TCMS mit anderen Tools und der Skalierbarkeit von TCMS. Diese Probleme stellen Hindernisse dar, die die Einführung von TCMS in einem Unternehmen erschweren können. In dieser Arbeit werden wir uns mit diesen Problemen und Herausforderungen auseinandersetzen, eine Software Lösung entwickeln und untersuchen, wie Test Case Management Systeme effektiv eingesetzt werden können, um den Testprozess zu optimieren und die Qualität eines Produkts zu verbessern.

1.3 Zielsetzung

Das Ziel dieser Arbeit ist es, die Verwendung von Test Case Management Systemen zu untersuchen und zu bewerten. Zudem ein auf unser eigenes Produkt angepasstes TCMS backend zu entwickeln. Hiermit ergeben sich drei relevante fragen:

- Wie kann man Features, Testfälle und Testimplementierungen beschreiben?
- Wie in Datenbank schreiben und lesen?
- Wie mit Testläufen verknüpfen?

Insbesondere möchten wir die folgenden Ziele erreichen:

- Die Vor- und Nachteile der Verwendung von TCMS zu identifizieren und zu analysieren.
- Die Entwicklung und Integration von TCMS in den Entwicklungsprozess zu untersuchen und zu bewerten, einschließlich der Herausforderungen, die bei der Integration von TCMS in bestehende Entwicklungs- und Testprozesse auftreten können.
- Die Konnektivität von TCMS mit anderen Tools und Systemen, die in der Softwareentwicklung verwendet werden, zu untersuchen und zu bewerten.
- Empfehlungen für die erfolgreiche Implementierung von TCMS in der Softwareentwicklung zu geben, einschließlich der Identifizierung bewährter Praktiken.

Durch die Erfüllung dieser Ziele wird diese Arbeit dazu beitragen, das Verständnis für die Verwendung von TCMS in der Produktentwicklung zu verbessern und Unternehmen dabei zu unterstützen, den Testprozess zu optimieren und die Qualität ihrer Produkte zu verbessern.

2 Stand des Wissens

Dieses Kapitel gibt einen Überblick über den aktuellen Stand der Technik von Testarten und Testmanagementsystemen. Darüber hinaus wird untersucht was ein Test Case Management System ist und wie ein solches System verwendet wird. Zusätzlich wird einen Überblick von den Technologien gegeben, mit denen, für diese Arbeit, ein TCMS backend umgesetzt worden ist.

2.1 Testarten

Generell gibt es einige Testarten in der Entwicklung eines Produktes. Es gibt laut Atlassian (2023) mindestens sieben unterschiedliche Testverfahren die je nachdem andere Implementierungen betrachten und testen. In *Software Testing/Qualitätssicherung - Alle Methoden und Tools* (2023) wird beschreiben, dass dabei zwischen Funktionalen und Nicht-Funktionalen Tests unterschieden werden. Zu der funktionalen Testfamilie zählen beispielsweise Unit-Tests und Integrationstests. Wobei hingegen Leistung, Last und Stresstests sowie Usability-Tests zu den nicht funktionalen Testfamilie gehören.

Ein paar der meist vorkommenden Testarten sind:

- Unit-Tests
- Integrationstests
- Funktionstests
- Leistungstests

Um eine Testabdeckung sicherzustellen und den allgemeinen Testprozess zu optimieren, gibt es, wie in Abschnitt 1.1 auf Seite 10 schon besprochen, Test Case Management Systeme. Diese Systeme bieten eine zentrale Plattform zur Verwaltung von Testfällen,

dass alle erforderlichen Tests durchgeführt und dokumentiert werden. In Abschnitt 2.4 auf Seite 16 werden wir diese Systeme genauer untersuchen.

Ammann und Offutt (2016)

2.1.1 Unit-Tests

Im Allgemeinen sind Unit-Tests sehr einfache Tests die beispielsweise Methoden einer Klasse mit unterschiedlichen Parametern testet. Sie sind automatisierbar und können von einer Continuos-Integration-Pipeline sehr schnell durchgeführt werden. Dieser Ansatz ermöglicht eine frühe Entdeckung von Fehlern und eine kontinuierliche Überprüfung der Funktionsfähigkeit des Produkts im Entwicklungsprozess.

2.1.2 Integrationstests

Integrationstests sind Tests, die sicherzustellen, dass verschiedene Module oder Services, problemlos miteinander interagieren können. Durch diese Tests kann die Funktionalität der Anwendung insgesamt überprüft werden, wie beispielsweise die Interaktion mit einer Datenbank oder der Zusammenarbeit von Mikroservices.

2.1.3 Funktionstests

Funktionstests werden integriert, um ausschließlich Ergebnisse einer gegebenen Funktion zu überprüfen. Dabei werden Spezifikation vor der Testausführung festgelegt und diese dann mit den Ergebnissen verglichen.

Im Gegensatz dazu stellt ein Integrationstest sicher, dass verschiedene Komponenten einer Anwendung zusammenarbeiten. Dabei werden Interaktionen zwischen verschiedenen Modulen oder Services, wie z.B. Datenbankabfragen oder die Kommunikation zwischen Mikroservices, überprüft.

Während Integrationstests beispielsweise nur prüfen, ob Datenbankabfragen generell möglich sind, wird bei einem Funktionstest ein bestimmter Wert aus der Datenbank abgerufen und dieser mit den angegebenen Spezifikationen geprüft.

2.1.4 Leistungstests

Leistungstests sind Verfahren, die dazu dienen, das Verhalten eines Systems unter verschiedenen Auslastungsmaximierungen zu überprüfen. Sie werden oft dazu verwendet um Zuverlässigkeit, Geschwindigkeit, Skalierbarkeit und Reaktionsfähigkeit einer Anwendung zu testen. Außerdem können Leistungstests mögliche Engpässe in einer Anwendung identifizieren.

2.2 Black-Box und White-Box Testing

Sowohl Black-Box als auch White-Box Tests werden in der Software Entwicklung häufig verwendet, um Fehler als auch die Qualität des Produktes zu evaluieren. Dabei gibt es zwischen den beiden wichtige Unterschiede.

Bei White-Box Testing handelt es sich grundsätzlich um Unit-Tests die den Code und die Struktur des zu testeten Produkts überprüft. Wobei der Inhalt des Codes für den Tester einsehbar ist. White-Box Testing bezieht sich dabei nur auf die interne Funktion des Produkts.

Bei Black-Box Testing wird die interne Struktur, das Design und die Implementierung nicht berücksichtigt. Hier werden nur die Ausgaben oder Reaktionen von dem System geprüft. Black-Box Testing bezieht sich hiermit nur auf die externe Funktion des Produkts.

Nidhra und Dondeti (2012, Seite 12)

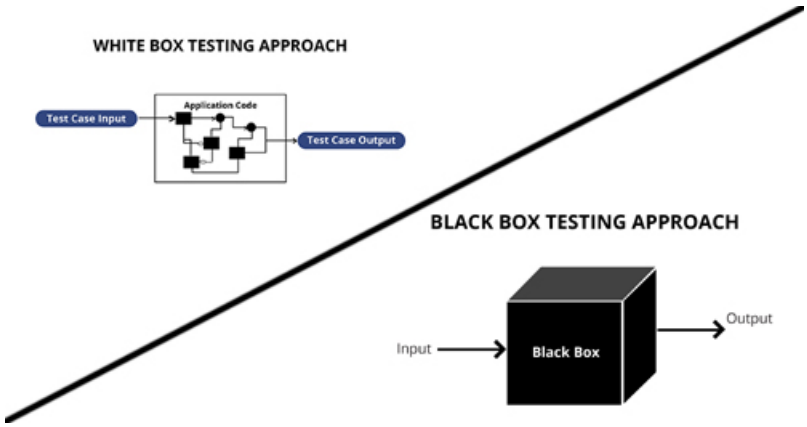

Abbildung 2.1: Black-Box/White-Box Testing (Quelle: Khandelwal (2019))

2.3 Test Driven Devlopment

Test Driven Development (TDD) ist ein Konzept bei dem zuerst ein Test geschrieben wird der fehlschlagt und dann genau soviel Code geschrieben wird, dass der Test erfolgreich durchgeführt werden kann. TDD bietet daher mehrere Vorteile:

- Geschriebener Code kann überarbeitet oder verschoben werden, ohne dass die Gefahr besteht, wesentliche Teile des Codes zu beschädigen.
- Die Tests selbst wurden getestet.
- Die Anforderungen werden mit fast keinem Aufwand umgesetzt, da nur die benötigte Funktion geschrieben wird.

Ammann und Offutt (2016)

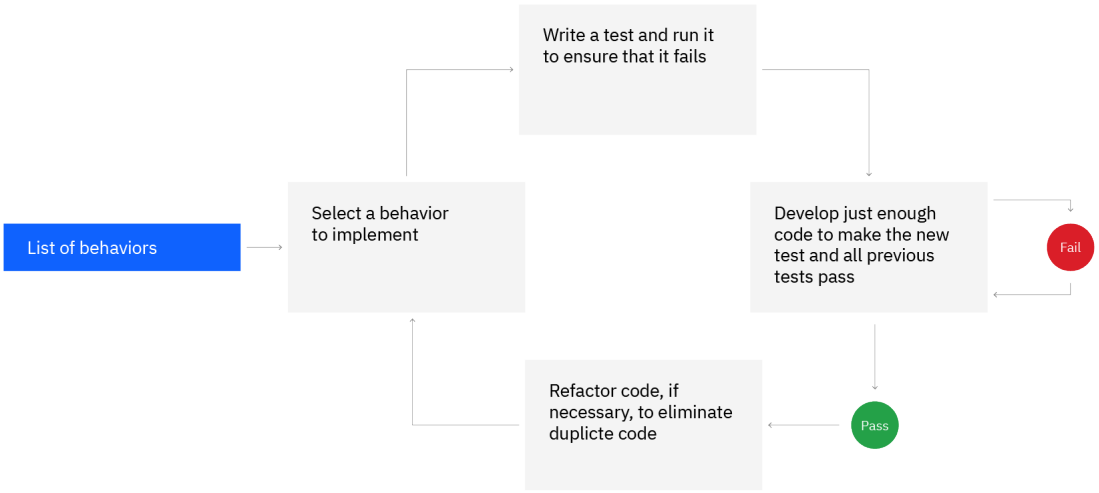

Abbildung 2.2: TDD cycle (Quelle: *Test-driven development - IBM Garage Practices* (2023))

2.4 Test Case Management System

Ein Test Case Management System (TCMS) ist eine Software, mit dem Test-Teams für ein bestimmtes Projekt oder eine Anwendung Testfälle verwalten, organisieren

und verfolgen können. Es hilft bei der Planung, Überwachung und Dokumentation von Tests und ermöglicht es, Testfälle sicher und effizient zu verwalten.

Zusammenfassend ist das TCMS ein wichtiges Werkzeug, um einen strukturierten und effektiven Testprozess zu gewährleisten und den Qualitätsstandard einer Anwendung zu verbessern.

2.5 Merkmale von Test Case Management Systemen

Dieser Abschnitt gibt einen Überblick zu den Hauptmerkmalen von Test Case Management Systeme. Ein TCMS verfügt normalerweise über Basisfunktionen wie Testfall-Erstellung, Testfall-Verwaltung, Testfall-Ausführung und Ergebnisberichterstattung. Es kann auch eine integrierte Umgebung für die Zusammenarbeit von Testern und Entwicklern bereitstellen.

Zusätzlich zu den Basisfunktionen sind das die 6 wichtigsten Merkmale, die eine gutes TCMS ausmachen:

- Attraktive Benutzeroberfläche und benutzerfreundliches Design
- Zulassung von mehreren Projekten und Benutzerberechtigungen
- Nachvollziehbarkeit
- einfache Zeitplanung und Organisation
- Überwachung und Metriken
- Flexibilität

Mehr zu den einzelnen punkte schreiben?

Lead (2023)

2.6 Allgemeine Vorteile von Test Case Management Systemen

Der Hauptvorteil des Einsatzes von einem TCMS besteht darin, dass sie den Testprozess verbessern und so schneller zu einem besseren Produkt führen. Sie sind auch unerlässlich, um die Gesamtkosten des Testens zu kontrollieren, indem sie die Testautomatisierung nutzen, um einen reibungslosen Ablauf zu gewährleisten.

Im Folgenden werden die spezifischen Vorteile von Testmanagement-Tools für die Testausführungsabläufe erläutert:

- Sie geben einen besseren Überblick über das zu testende System, halten den gesamten Prozess auf Kurs und koordinieren die Testaktivitäten.
- Sie helfen bei der Feinabstimmung des Testprozesses, indem sie die Zusammenarbeit, Kommunikation und Datenanalyse unterstützen.
- Sie verfolgen Aufgaben, Fehler und Testergebnisse und vereinfachen den Prozess, indem sie alles in einer einzigen Anwendung erledigen.
- Sie sind skalierbar und können eingesetzt werden, wenn die Testaktivitäten umfangreicher und komplexer werden.

Lead (2023)

2.7 Überblick von bestehenden Test Case Management Systeme

Heutzutage gibt es natürlich schon eine Vielzahl von fertigen und direkt verwendbaren Test Case Management Systemen auf dem Markt, die die Verwaltung von Tests vereinfacht.

Einige bekannte Tools sind:

- *TestRail – Orchestrate Testing. Elevate Quality.* (2023)

  Webbasiertes Test Case Management System, ist zentralisiert und hat eine einfache und intuitive Oberfläche.

Features:

  - Test Planung, Verwaltung und Ausführung
  - Echtzeit Berichterstattung und Analysen
  - Rückverfolgbarkeits- und Abdeckungsberichte für Anforderungen, Tests und Fehler

- *Tricentis Test Management for Jira* (2023)

  Webbasiertes Test Case Management System und hat moderne und intuitive Oberfläche.

  Features:

  - Test Planung, Verwaltung und Ausführung
  - Echtzeit Test Status Update
  - Berichterstattung

- *Zephyr Test Management Products | SmartBear* (2023)

  Webbasiertes Test Case Management System, moderne Oberfläche und kann in JIRA integriert werden.

  Features:

  - Test Planung, Verwaltung und Ausführung
  - Test Automatisierung
  - Echtzeit Visualisierung von Projektstatus

- *Tricentis qTest for Unified Test Management* (2023)

  Webbasiertes Test Case Management System und hat moderne und intuitive Oberfläche.

  Features:

  - Test Planung, Verwaltung und Ausführung
  - Migrationsmöglichkeit von alten Test Management Lösungen
  - Integrationsmöglichkeit mit Jenkins, Azure Pipelines, Bamboo oder jedem anderen CI/CD-Tool

- Anpassbare Dashboards, um über alle Releases, Projekte oder Programme im gesamten Unternehmen zu berichten
- Berichte per E-Mail oder URL teilen

Diese Test Case Management Systeme bieten eine Grundfunktionalität, auf diese wir in Abschnitt 2.5 auf Seite 17 eingegangen sind. Zu den Grundfunktionalitäten bieten diese Systeme jeweils andere oder zusätzliche Funktionen und Integrationsmöglichkeiten. Welches Tool am besten geeignet ist, hängt natürlich von der jeweiligen Anwendung ab.

Abbildung 2.3: Liste von aktuell führenden Test Case Management Systemen/Tools (Quelle: *7 Best Test Management Tools* (2023))

Die in Abbildung 2.3 ersichtlichen Tools zeigen eine aktuell führende Auswahl von beliebten Test Case Management Systemen.

2.8 Verwendete Technologien

Dieser Abschnitt gibt eine Übersicht über die verwendeten Technologien, die für die implementieren, von einem backend eines Test Case Management Systems eingesetzt

worden sind.

2.8.1 MariaDB

Proper MariaDB Description

MariaDB documentation (2023)

2.8.2 Docker

Proper Docker Description

Ghosh (2020)

2.8.3 Microsoft .NET Core

Proper .NET Core Description

BillWagner (2023)

2.8.4 Microsoft ASP.NET Core

Proper ASP.NET Core Description

BillWagner (2023)

2.8.5 Entity Framework Core

Proper Entity Framework Core Description

BillWagner (2023)

2.8.6 OpenAPI

Proper OpenAPI Description

OpenAPI (2023) *OpenAPI Generator* (2023)

3 Anforderungen

Proper Anforderungen

3.1 Test

Test

4 Konzept

Proper Lösungsansatz

4.1 Scrum

Proper Scrum Description

Rubin (2012)

4.2 Architektur

Proper Architektur Description

Richards und Ford (2020)

4.2.1 Backend

Proper Backend Description

4.2.2 Domain Driven Design

Proper Domain Driven Design Description

Vernon (2013)

4.3 Vorentwicklungsphase

Proper Vorentwicklungsphase Description

4.3.1 Projektstruktur

Proper Projektstruktur Description

4.3.2 Domain Model

Proper Domain Model Description

4.3.3 ER Model

Proper ER Model Description

4.3.4 Featurestruktur

Proper Featurestruktur Description

4.3.5 Featurebedingungen

Proper Featurebedingungen Description

4.3.6 Featureverwaltung

Proper Featureverwaltung Description

4.4 Entwicklungsphase

Proper Entwicklungsphase Description

4.4.1 Verwaltung und Struktur von Features

Proper Verwaltung und Struktur von Features Description

4.4.2 Featurestruktur zur Datenbank

Proper Featurestruktur zur Datenbank Description

4.4.3 Visualisierung von Featurestrukturen

Proper Visualisierung von Featurestrukturen Description

5 Technische Umsetzung

Proper Implementierung

5.1 User Stories

Proper User Stories Description

5.1.1 Story 1

Proper Story 1 Description

5.1.2 Story 2

Proper Story 2 Description

5.1.3 Story n

Proper Story n Description

6 Ergebnisse

Proper Ergebnisse Description

7 Fazit und Ausblick

Proper Fazit und Ausblick

```
{
    // Start eines neuen Scopes

    let hello = String::from("hello");
    // Ende des Scopes. hello wird an dieser Stelle freigegeben
}
// hello kann hier nicht mehr verwendet werden
```

Abbildung 7.1: Ein String wird in einem Scope erzeugt und der Variable hello zugewiesen. Am Ende des Scopes wird der Speicher freigegeben.

Abbildung 7.2: Nach 100 Simulationsschritten, einem minimalen Gruppenanteil $B\text{min} = 0.4$ und einer Nachbarschaftsgröße von einem Feld sind die Gruppen bereits sichtlich separiert.

Literatur

7 Best Test Management Tools (2023). URL: https://www.practitest.com/test-management-tools/ (besucht am 18. 04. 2023).

Ammann, Paul und Jeff Offutt (13. Dez. 2016). *Introduction to Software Testing*. Google-Books-ID: 58LeDQAAQBAJ. Cambridge University Press. 367 S. ISBN: 978-1-316-77312-3.

Atlassian (2023). *Die unterschiedlichen Arten von Softwaretests*. Atlassian. URL: https://www.atlassian.com/de/continuous-delivery/software-testing/types-of-software-testing (besucht am 15. 04. 2023).

BillWagner (2023). *.NET documentation*. URL: https://learn.microsoft.com/en-us/dotnet/ (besucht am 19. 04. 2023).

Ghosh, Saibal (3. Okt. 2020). *Docker Demystified: Learn How to Develop and Deploy Applications Using Docker*. Google-Books-ID: 650AEAAAQBAJ. BPB Publications. 243 S. ISBN: 978-93-89845-87-7.

Khandelwal, Abhik (12. Okt. 2019). *Difference between Black Box and White Box Testing | Testing Types*. TestingGenez. URL: https://testinggenez.com/black-box-and-white-box-testing/ (besucht am 01. 05. 2023).

Lead, The QA (2023). *Articles Archives*. The QA Lead. URL: https://theqalead.com/topics/ (besucht am 19. 04. 2023).

MariaDB documentation (2023). MariaDB.org. URL: https://mariadb.org/documentation/ (besucht am 19. 04. 2023).

Nidhra, Srinivas und Jagruthi Dondeti (30. Juni 2012). „BLACK BOX AND WHITE BOX TESTING TECHNIQUES -A LITERATURE REVIEW“. In: *International journal of embedded systems and applications*. DOI: 10.5121/ijesa.2012.2204. URL: https://www.scinapse.io/papers/2334860424 (besucht am 01. 05. 2023).

OpenAPI (2023). OpenAPI Initiative. URL: https://www.openapis.org/ (besucht am 19. 04. 2023).

OpenAPI Generator (2023). URL: https://openapi-generator.tech/ (besucht am 19. 04. 2023).

Richards, Mark und Neal Ford (28. Jan. 2020). *Fundamentals of Software Architecture: An Engineering Approach*. Google-Books-ID: xa7MDwAAQBAJ. Ö'Reilly Media, Inc." 422 S. ISBN: 978-1-4920-4342-3.

Rubin, Kenneth S. (2012). *Essential Scrum: A Practical Guide to the Most Popular Agile Process*. Google-Books-ID: HkXX65VCZU4C. Addison-Wesley Professional. 501 S. ISBN: 978-0-13-704329-3.

Software Testing/Qualitätssicherung - Alle Methoden und Tools (2023). Software Testing/Qualitätssicherung - Alle Methoden und Tools. URL: https://q-centric.com/ (besucht am 15. 04. 2023).

Test-driven development - IBM Garage Practices (2023). URL: https://www.ibm.com/garage/method/practices/code/practice_test_driven_development/ (besucht am 01. 05. 2023).

TestRail – Orchestrate Testing. Elevate Quality. (25. Jan. 2023). URL: https://www.testrail.com/ (besucht am 17. 04. 2023).

Tricentis qTest for Unified Test Management (2023). Tricentis. URL: https://www.tricentis.com/products/unified-test-management-qtest/ (besucht am 18. 04. 2023).

Tricentis Test Management for Jira (2023). Atlassian Marketplace. URL: https://marketplace.atlassian.com/apps/1228672/tricentis-test-management-for-jira (besucht am 17. 04. 2023).

Vernon, Vaughn (6. Feb. 2013). *Implementing Domain-Driven Design*. Google-Books-ID: X7DpD5g3VP8C. Addison-Wesley. 656 S. ISBN: 978-0-13-303988-7.

Zephyr Test Management Products | SmartBear (2023). URL: https://smartbear.com/test-management/zephyr/ (besucht am 17. 04. 2023).

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich vorliegende Bachelorarbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Stellen sind als solche kenntlich gemacht. Die Arbeit wurde bisher weder in gleicher noch in ähnlicher Form einer anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Dornbirn, am 1. Juli 2023 Marco Prescher