

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	7
Abkürzungsverzeichnis	8
1 Einleitung	10
1.1 Motivation	10
1.2 Problemstellung	10
1.3 Zielsetzung	11
2 Stand des Wissens	13
2.1 Testarten	13
2.1.1 Unit-Tests	14
2.1.2 Integrationstests	14
2.1.3 Funktionstests	14
2.1.4 Leistungstests	15
2.2 Blackbox/Whitebox Testing	15
2.3 Test Driven Development	15
2.4 Test Case Management System	15
2.4.1 Überblick über Test Case Management Systeme	15
2.4.2 Allgemeine Vorteile von Test Case Management Systemen	18
2.5 Features von Test Case Management Systemen	18
2.5.1 Aspekte von Test Case Management Systemen	18
2.5.2 Aktuelle Trends und Herausforderungen	18
2.5.3 Bewertung von Test Case Management Systemen	18
3 Konzept	19
3.1 Scrum	19
3.2 Architektur	19
3.2.1 Backend	19

3.3	Vorentwicklungsphase	19
3.3.1	Projektstruktur	19
3.3.2	Domain Model	19
3.3.3	ER Model	19
3.3.4	Featurestruktur	19
3.3.5	Featurebedingungen	19
3.3.6	Featureverwaltung	20
3.4	Entwicklungsphase	20
3.4.1	Verwaltung und Struktur von Features	20
3.4.2	Featurestruktur zur Datenbank	20
3.4.3	Visualisierung von Featurestrukturen	20
4	Technische Umsetzung	21
4.1	Verwendete Technologien	21
4.1.1	MariaDB	21
4.1.2	Docker	21
4.1.3	Microsoft .NET Core	21
4.1.4	Microsoft ASP.NET Core	21
4.1.5	Entity Framework Core	21
4.1.6	OpenAPI	21
4.2	Sprints	21
4.3	User Stories	21
4.3.1	Story 1	22
4.3.2	Story 2	22
4.3.3	Story n	22
5	Ergebnisse	23
6	Fazit und Ausblick	24
	Literaturverzeichnis	25

Abbildungsverzeichnis

- 2.1 Liste von aktuell führenden Test Case Management Systemen/Tools
(Quelle: *7 Best Test Management Tools* (2023)) 17
- 6.1 Ein String wird in einem Scope erzeugt und der Variable hello zugewie-
sen. Am Ende des Scopes wird der Speicher freigegeben. 24
- 6.2 Nach 100 Simulationsschritten, einem minimalen Gruppenanteil $B_{min} =$
0.4 und einer Nachbarschaftsgröße von einem Feld sind die Gruppen
bereits sichtlich separiert. 24

Abkürzungsverzeichnis

API Application Programming Interface

JSON JavaScript Object Notation

TCMS Test Case Management System

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 wie ein Test Case Management System verwendet wird und was die empfohlene Vorgehensweise ist. Zusätzlich werden Technologien analysiert, mit denen eine Implementation von einem TCMS gut umsetzbar 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 beschrieben, 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 15 werden wir diese Systeme genauer untersuchen.

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 Continuous-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 Blackbox/Whitebox Testing

Proper Black-box/Whitebox Testing Description

2.3 Test Driven Development

Proper Test Driven Development Description

2.4 Test Case Management System

Ein Test Case Management System 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.

Ein TCMS verfügt über Funktionen wie Testfall-Erstellung, Testfall-Verwaltung, Testfall-Ausführung und Ergebnisberichterstattung. Es kann auch eine integrierte Umgebung für die Zusammenarbeit von Testern und Entwicklern bereitstellen.

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

2.4.1 Überblick über 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.

Kern 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.

Kern 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.

Kern 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.

Kern 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 große Basis Funktionalität, auf diese wir in Abschnitt 2.5 auf Seite 18 genauer eingehen werden, und je nachdem andere oder zusätzliche Funktionen und Integrationsmöglichkeiten. Welches Tool am besten geeignet ist, hängt natürlich von der jeweiligen Anwendung ab.



Abbildung 2.1: Liste von aktuell führenden Test Case Management Systemen/Tools
(Quelle: *7 Best Test Management Tools* (2023))

Die in Abbildung 2.1 ersichtlichen Tools zeigen eine aktuell führende Auswahl von beliebten Test Case Management Systemen.

Proper Vorteile
von Test Case
Management
Systemen Des-
cription

2.4.2 Allgemeine Vorteile von Test Case Management Systemen

2.5 Features von Test Case Management Systemen

Dieses Unterkapitel gibt einen Überblick zu den Features von Test Case Management Systeme. Wie in Abschnitt 2.4 auf Seite 15 schon angedeutet wird verfügt ein TCMS über Basis Funktionen mit denen verschiedene Features realisierbar werden.

Zudem gibt es Die Berichterstattung ist im Allgemeinen eine der wichtigsten Funktion eines TCMSs. Diese generiert einen ausführlichen Testbericht und gibt eine Übersicht von allen implementierten Testfällen nach einem Testlauf. Dadurch wird auch die Fehlerverfolgung und Fehleridentifizierung sehr vereinfacht und effizient. Zusätzlich können diese Testberichte nach jedem Testlauf zu den jeweiligen abonnierten Testern gesendet werden.

Lead (2023)

Proper Aspekte
von Test Case
Management
Systemen Des-
cription

2.5.1 Aspekte von Test Case Management Systemen

2.5.2 Aktuelle Trends und Herausforderungen

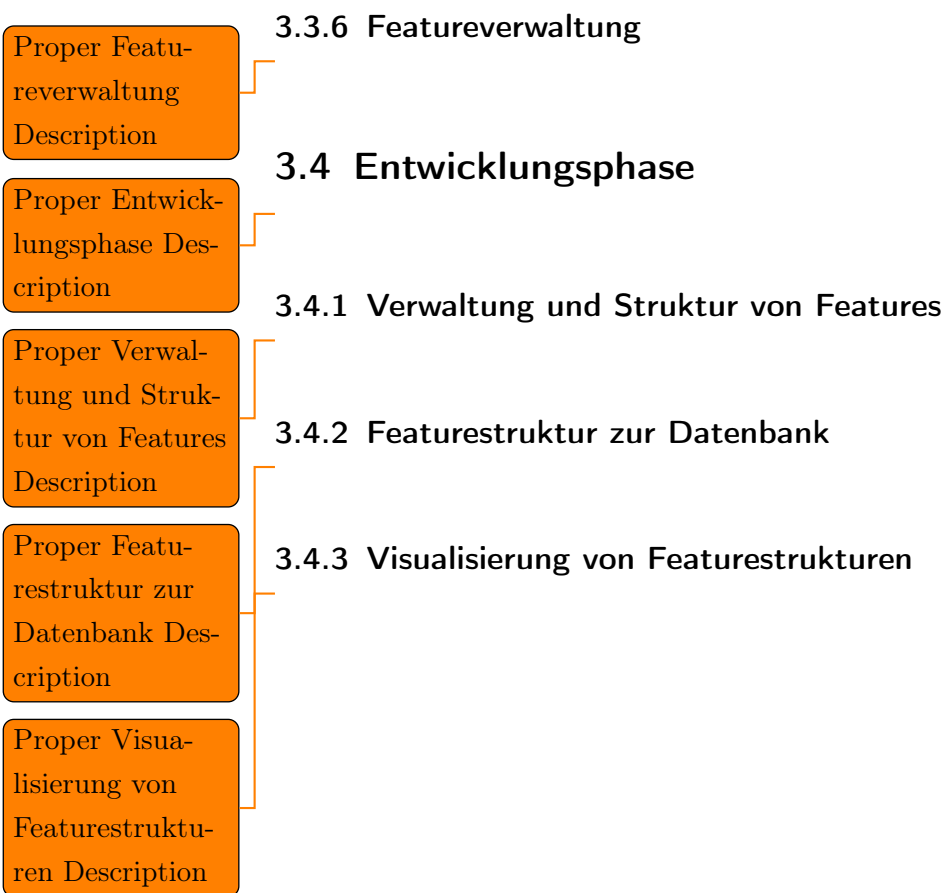
Proper Aktuelle
Trends und Her-
ausforderungen
Description

2.5.3 Bewertung von Test Case Management Systemen

Proper Bewer-
tung von Test
Case Manage-
ment Systemen
Description

3 Konzept





4 Technische Umsetzung

	Proper Implementierung
4.1 Verwendete Technologien	Proper Verwendete Technologien Description
4.1.1 MariaDB	Proper MariaDB Description
4.1.2 Docker	Proper Docker Description
4.1.3 Microsoft .NET Core	Proper .NET Core Description
4.1.4 Microsoft ASP.NET Core	Proper ASP.NET Core Description
4.1.5 Entity Framework Core	Proper Entity Framework Core Description
4.1.6 OpenAPI	Proper Open-API Description
4.2 Sprints	Proper Sprints Description
4.3 User Stories	Proper User Stories Description

Proper Story 1
Description

4.3.1 Story 1

Proper Story 2
Description

4.3.2 Story 2

Proper Story n
Description

4.3.3 Story n

5 Ergebnisse

Proper Ergebnisse
Description

6 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 6.1: Ein String wird in einem Scope erzeugt und der Variable hello zugewiesen. Am Ende des Scopes wird der Speicher freigegeben.



Abbildung 6.2: Nach 100 Simulationsschritten, einem minimalen Gruppenanteil $B_{\min} = 0.4$ 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).
- 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).
- Harman, Graham (2012). „The Well-Wrought Broken Hammer: Object-Oriented Literary Criticism“. In: *New Literary History* 43.2. Publisher: Johns Hopkins University Press, S. 183–203. ISSN: 1080-661X. DOI: 10.1353/nlh.2012.0016. URL: <https://muse.jhu.edu/pub/1/article/483016> (besucht am 10.02.2023).
- Lead, The QA (2023). *Articles Archives*. The QA Lead. URL: <https://theqalead.com/topics/> (besucht am 19.04.2023).
- 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).
- 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).
- 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