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

SPA Single Page Application

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

Einleitung

Test Driven Development (TDD) TDD Harman (2012) Abb. 6.2

Harman 2012

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, die in der Software/Hardware Entwicklung, Support, Marketing etc., die Qualität des Produkts verbessern, Fehler frühzeitig erkennen, beheben und den Entwicklungsprozess effizienter gestalten. Diese Arbeit fokussiert sich daher auf die Integration von einem TCMS in einen laufenden Produktentwicklungs-Zyklus und vergleicht verschiedene vorhandene Lösungen.

1.2 Problemstellung

Die Komplexität von Projekten steigt kontinuierlich an und es ist eine Herausforderung, die Qualität des entwickelten Systems sicherzustellen. Es 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

In diesem Kapitel wird eine systematische Überprüfung der angewandten Technologien vorgenommen. Dabei werden die verschiedenen Typen von Tests, die Verwendung von Test Case Management Systeme sowie die Organisation und Strukturierung von Funktionalitäten dieser gründlich analysiert.

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 gute Testabdeckung sicherzustellen und den Testprozess zu optimieren, gibt es sogenannte TCMS. Diese Systeme bieten eine zentrale Plattform zur Verwaltung von Testfällen und Testpläne, dass alle erforderlichen Tests durchgeführt und dokumentiert werden. In Kapitel Abschnitt 2.3 auf Seite 15 werden wir diese Systeme genauer untersuchen.

2.1.1 Unit-Tests

Im Allgemeinen sind Unit-Tests automatisierbar und können von einem Continuos-Integration-Server sehr schnell durchgeführt werden. Dieser Ansatz ermöglicht eine frühe Entdeckung von Fehlern und eine kontinuierliche Überprüfung der Funktionsfähigkeit des Produkts im Entwicklungsprozess. Durch die Integration von Unit-Tests kann die Qualität des Produkts verbessert werden, indem Fehler frühzeitig erkannt und behoben werden, bevor sie sich auf andere Teile der Anwendung auswirken.

2.1.2 Integrationstests

Integrationstests sind Verfahren, um sicherzustellen, dass verschiedene Module oder Services, die von einer Anwendung genutzt werden, problemlos miteinander interagieren können. Durch diese Tests kann die Funktionalität der Anwendung insgesamt überprüft werden, insbesondere hinsichtlich des Zusammenspiels von einzelnen Komponenten, wie beispielsweise der Interaktion mit einer Datenbank oder der Zusammenarbeit von Mikroservices.

2.1.3 Funktionstests

Funktionstests sind Verfahren, die sich auf die Erfüllung der Anforderungen einer Anwendung konzentriert. Im Fokus steht die Überprüfung der Ausgabe einer Aktion, während die Zwischenzustände des Systems bei der Durchführung dieser Aktion nicht im Detail untersucht werden.

Im Gegensatz dazu zielt ein Integrationstest darauf ab, sicherzustellen, dass verschiedene Komponenten einer Anwendung reibungslos zusammenarbeiten. Dabei werden Interaktionen zwischen verschiedenen Modulen oder Services, wie z.B. Datenbankabfragen oder die Kommunikation zwischen Mikroservices, überprüft. Integrationstests erfordern mehr Aufwand, da mehrere Teile der Anwendung zusammenarbeiten müssen, um die Funktionalität der Anwendung zu gewährleisten.

Obwohl Funktionstests und Integrationstests beide mehrere Komponenten miteinander interagieren lassen, gibt es Unterschiede in der Art und Weise, wie sie durchgeführt werden. Während Integrationstests beispielsweise nur prüfen können, ob Datenbankabfragen generell möglich sind, wird bei einem Funktionstest ein bestimmter, von

den Produktanforderungen vorgegebener Wert aus der Datenbank abgerufen und auf Übereinstimmung mit den Anforderungen geprüft.

2.1.4 Leistungstests

Leistungstests sind Verfahren, die dazu dienen, Verhalten eines Systems unter verschiedenen Auslastungsbedingungen zu überprüfen. Sie sind ein geeignetes Mittel, um die Zuverlässigkeit, Geschwindigkeit, Skalierbarkeit und Reaktionsfähigkeit einer Anwendung zu messen. Ein Leistungstest kann beispielsweise die Antwortzeiten einer Anwendung bei der Ausführung einer hohen Anzahl von Anfragen zu überprüfen. Dabei werden oft auch Aspekte wie das Verhalten eines Systems bei großen Datenmengen untersucht.

Durch Leistungstests können mögliche Engpässe in einer Anwendung identifiziert werden. Außerdem können Leistungstests dazu beitragen, die Stabilität einer Anwendung bei maximaler Belastung zu beurteilen.

2.2 Einführung in das Test Case Management

Proper Einführung in das Test Case Management Description

2.2.1 Methoden und Techniken im Test Case Management

Proper Methoden und Techniken im Test Case Management Description

2.3 Test Case Management System

Wie in Kapitel Abschnitt 2.2 auf Seite 15 besprochen worden ist, gibt es zur Unterstützung von Test Case Management-Prozessen sogenannte Test Case Management Systeme. 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 TCMS ein wichtiges Werkzeug, um einen strukturierten und effektiven Testprozess zu gewährleisten und den Qualitätsstandard einer Anwendung zu verbessern.

2.3.1 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 der beliebtesten Tools sind:

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

  Eine webbasierte Testmanagement-Software, die eine einfache und intuitive Oberfläche bietet. TestRail ermöglicht es, Testfälle zu planen, durchzuführen und zu verwalten und bietet auch umfangreiche Berichtsfunktionen.

- *Tricentis Test Management for Jira* (2023)

  Ein beliebtes Projektmanagement-Tool, das auch Funktionen für das Testmanagement bietet. JIRA ermöglicht es, Testfälle zu planen, durchzuführen und zu verwalten und bietet auch umfangreiche Berichtsfunktionen.

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

  Ein webbasiertes Testmanagement-Tool, das in JIRA integriert werden kann. Zephyr ermöglicht es, Testfälle zu planen, durchzuführen und zu verwalten und bietet auch Funktionen zur Automatisierung von Tests.

- *qTest Manager – Test Case Management* (2023)

  Eine webbasierte Testmanagement-Software, die eine einfache und intuitive Oberfläche bietet. QTest ermöglicht es, Testfälle zu planen, durchzuführen und zu verwalten und bietet auch umfangreiche Berichtsfunktionen.

Diese TCMS-Tools bieten eine große Basis Funktionalität und je nachdem andere oder zusätzliche Funktionen und Integrationsmöglichkeiten. Welches Tool am besten geeignet ist, hängt natürlich von der jeweiligen Anwendung ab.

2.3.2 Funktionen und Eigenschaften von Test Case Management Systemen

2.3.3 Aktuelle Trends und Herausforderungen

2.3.4 Bewertung von Test Case Management Systemen

2.4 Relationale Datenbank

2.4.1 MariaDB

2.5 Docker

Proper Funktionen und Eigenschaften von Test Case Management Systemen Description

Proper Aktuelle Trends und Herausforderungen Description

Proper Bewertung von Test Case Management Systemen Description

Proper Relationale Datenbank Description Description

Proper MariaDB Description

Proper Docker Description

3 Lösungsansatz

Proper Lösungsansatz

3.1 Scrum

Proper Scrum Description

3.2 Architektur

Proper Architektur Description

3.2.1 Backend

Proper Backend Description

3.3 Vorentwicklungsphase

Proper Vorentwicklungsphase Description

3.3.1 Projektstruktur

Proper Projektstruktur Description

3.3.2 Domain Model

Proper Domain Model Description

3.3.3 ER Model

Proper ER Model Description

3.3.4 Featurestruktur

Proper Featurestruktur Description

3.3.5 Featurebedingungen

Proper Featurebedingungen Description

3.3.6 Featureverwaltung

Proper Featureverwaltung Description

3.4 Entwicklungsphase

Proper Entwicklungsphase Description

3.4.1 Verwaltung und Struktur von Features

Proper Verwaltung und Struktur von Features Description

3.4.2 Featurestruktur zur Datenbank

Proper Featurestruktur zur Datenbank Description

3.4.3 Visualisierung von Featurestrukturen

Proper Visualisierung von Featurestrukturen Description

4 Implementierung

Proper Implementierung

4.1 Sprints

Proper Sprints Description

4.2 User Stories

Proper User Stories Description

4.2.1 Story 1

Proper Story 1 Description

4.2.2 Story 2

Proper Story 2 Description

4.2.3 Story n

Proper Story n Description

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.

FH Vorarlberg
University of Applied Sciences

Abbildung 6.2: Nach 100 Simulationsschritten, einem minimalen Gruppenanteil $B\text{min} = 0.4$ und einer Nachbarschaftsgröße von einem Feld sind die Gruppen bereits sichtlich separiert.

Literatur

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).

qTest Manager – Test Case Management (2023). Tricentis. URL: https://www.tricentis.com/products/unified-test-management-qtest/test-case-manager/ (besucht am 17. 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 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