

# **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's 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

<b>Abbildungsverzeichnis</b>	<b>7</b>
<b>Abkürzungsverzeichnis</b>	<b>8</b>
<b>1 Einleitung</b>	<b>10</b>
1.1 Motivation . . . . .	10
1.2 Problemstellung . . . . .	10
1.3 Zielsetzung . . . . .	10
<b>2 Stand des Wissens</b>	<b>11</b>
2.1 Testarten . . . . .	11
2.1.1 Unit Test . . . . .	11
2.1.2 Integration Test . . . . .	11
2.2 Test Case Management System . . . . .	11
2.3 Verwaltung und Struktur von Features . . . . .	12
2.3.1 Featurestruktur zur Datenbank . . . . .	12
2.3.2 Visualisierung von Featurestrukturen . . . . .	12
2.4 Relationale Datenbank . . . . .	12
2.4.1 MySQL . . . . .	12
2.4.2 PostgreSQL . . . . .	12
2.4.3 MariaDB . . . . .	12
2.5 Single Page Application (SPA) . . . . .	12
2.6 Docker . . . . .	12
<b>3 Lösungsansatz</b>	<b>13</b>
3.1 Scrum . . . . .	13
3.2 Architektur . . . . .	13
3.2.1 Backend . . . . .	13
3.2.2 Frontend . . . . .	13

3.3	Vorentwicklungsphase . . . . .	13
3.3.1	Projektstruktur . . . . .	13
3.3.2	Domain Model . . . . .	13
3.3.3	ER Model . . . . .	13
3.3.4	Featurestruktur . . . . .	13
3.3.5	Featurebedingungen . . . . .	14
3.3.6	Featureverwaltung . . . . .	14
3.4	Entwicklungsphase . . . . .	14
3.5	Nachentwicklungsphase . . . . .	14
<b>4</b>	<b>Implementierung</b>	<b>15</b>
4.1	Sprints . . . . .	15
4.2	User Stories . . . . .	15
4.2.1	Story 1 . . . . .	15
4.2.2	Story 2 . . . . .	15
4.2.3	Story n . . . . .	15
<b>5</b>	<b>Ergebnisse</b>	<b>16</b>
<b>6</b>	<b>Fazit und Ausblick</b>	<b>17</b>
	<b>Literaturverzeichnis</b>	<b>18</b>

# Abbildungsverzeichnis

6.1	Ein String wird in einem Scope erzeugt und der Variable hello zugewiesen. Am Ende des Scopes wird der Speicher freigegeben. . . . .	17
6.2	Nach 100 Simulationsschritten, einem minimalen Gruppenanteil $B_{\min} = 0.4$ und einer Nachbarschaftsgröße von einem Feld sind die Gruppen bereits sichtlich separiert. . . . .	17

## **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\_well-wrought\_2012 Abb. 6.2

## 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 deshalb auf die Integration eines TCMS Systems in einen laufenden Produktentwicklungs-Zyklus und vergleicht verschiedene vorhandene Lösungen.

## 1.2 Problemstellung

Proper Problemstellung

## 1.3 Zielsetzung

Proper Zielsetzung

## 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 Testmanagement-Systemen sowie die Organisation und Strukturierung von Funktionalitäten gründlich analysiert.

### 2.1 Testarten

Proper Testing  
Description

#### 2.1.1 Unit Test

Proper Unit  
Test Descripti-  
on

#### 2.1.2 Integration Test

Proper Integrati-  
on Test Descrip-  
tion

### 2.2 Test Case Management System

Ein TCMS ist eine Software, die verwendet wird, um Testfälle für ein bestimmtes Projekt oder eine Anwendung zu verwalten und zu organisieren. 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 Verwaltung und Struktur von Features

Proper Verwaltung und Struktur von Features Description

### 2.3.1 Featurestruktur zur Datenbank

Proper Featurestruktur zur Datenbank Description

### 2.3.2 Visualisierung von Featurestrukturen

Proper Featurestruktur zur Datenbank Description

## 2.4 Relationale Datenbank

### 2.4.1 MySQL

Proper Relationale Datenbank Description

### 2.4.2 PostgreSQL

Proper MySQL Description

### 2.4.3 MariaDB

Proper PostgreSQL Description

## 2.5 Single Page Application (SPA)

Proper MariaDB Description

## 2.6 Docker

Proper Single-Page-Webanwendung 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
3.2.2 Frontend	Proper Frontend 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

Proper Featurebedingungen  
Description

### 3.3.5 Featurebedingungen

### 3.3.6 Featureverwaltung

Proper Featureverwaltung  
Description

## 3.4 Entwicklungsphase

Proper Entwicklungsphase  
Description

## 3.5 Nachentwicklungsphase

Proper Nachentwicklungsphase  
Description

## 4 Implementierung



## 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.

## 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