**Projekt: Mosti**

System - Testdokumentation

[Dokumentstruktur basiert auf RUP „Dokument Test Evaluation Summary“]

# Dokumentinformationen

## Änderungsgeschichte

|  |  |  |  |
| --- | --- | --- | --- |
| Datum | Version | Änderung | Autor |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Inhalt

[1 Dokumentinformationen 2](#_Toc449603931)

[1.1 Änderungsgeschichte 2](#_Toc449603932)

[1.2 Inhalt 2](#_Toc449603933)

[2 Einführung (Introduction) 3](#_Toc449603934)

[2.1 Definitionen und Abkürzungen (Definitions, Acronyms, Abbreviations) 3](#_Toc449603935)

[2.2 Referenzen (References) 3](#_Toc449603936)

[2.3 Übersicht (Overview) 3](#_Toc449603937)

[3 Testvorgehen 3](#_Toc449603938)

[3.1 Funktionale Tests 3](#_Toc449603939)

[3.1.1 Grundtests (Smoke Tests) 3](#_Toc449603940)

[3.1.2 Modul- und Unittests 3](#_Toc449603941)

[3.1.3 Integrationstests 3](#_Toc449603942)

[3.1.4 System Acceptance Test 3](#_Toc449603943)

[3.2 Bedienbarkeit und Nutzerinterface (Usability) 3](#_Toc449603944)

[3.3 Datenschutz, Datensicherheit (Security) 3](#_Toc449603945)

[3.4 Leistungsanforderungen (Performance) 3](#_Toc449603946)

[3.5 Zuverlässigkeit 3](#_Toc449603947)

[3.6 Schnittstellen 3](#_Toc449603948)

[3.7 Wartung und Servicefunktionen 4](#_Toc449603949)

[3.8 Installation 4](#_Toc449603950)

[3.9 Internationalisierung / Lokalisierung 4](#_Toc449603951)

[3.10 Testautomatisierung 4](#_Toc449603952)

[3.11 Verfolgbarkeit (Traceability) 4](#_Toc449603953)

[4 Übersicht der Testpläne 4](#_Toc449603954)

[5 Freigabe von Testergebnissen 4](#_Toc449603955)

# Einführung (Introduction)

## Definitionen und Abkürzungen (Definitions, Acronyms, Abbreviations)

Vgl. separates Glossary-Dokument im Repository, Dateiname: Glossary.docx

## Referenzen (References)

Buch: Craig Larman: UML 2 und Patterns angewendet (2005)

## Übersicht (Overview)

In diesem Dokument werden verschieden Testverfahren erläutert die wir für unsere Software entwickelt haben. Wir gehen darauf ein wie unsere Tests durchgeführt und diese dokumentiert werden.

# Testvorgehen

## Funktionale Tests

### Grundtests (Smoke Tests)

Schon während der Programmierung werden die einzelnen Codestücke immer von den programmierenden Mitgliedern geprüft. Dafür werden z.B. in der Datenbank einige Testkunde angelegt, um die Verbindung zur Datenbank oder die richtige Anzeige zu überprüfen. Des Weiteren werden die Codes vor dem Commit ins Repository auf ihre Lauffähigkeit getestet, d.h. es wird überprüft, ob das Codesegment unabhängig von eventuellen Nutzereingaben überhaupt kompilieren oder zur Laufzeit irgendwelche Exceptions werfen. Da die diese Tests innerhalb der ersten Programmierung ausgeführt werden, können sie nicht dokumentiert werden.

### Modul- und Unittests

Nach der Erstellung einer vorläufigen Version der einzelnen Module werden diese durch ein Teammitglied, das nicht an der Programmierung desselben Moduls beteiligt war, in den Unittests geprüft.

Jede Testfunktion prüft genau eine andere Funktion und ruft diese mit unterschiedlichen Parametern auf, berechnet aber gelichzeitig den richtigen Rückgabewert und vergleicht diesen dann mit der Rückgabe der zu testenden Funktion. Eventuelle Fehler und Unstimmigkeiten werden vom Tester behoben und die Ergebnisse an den verantwortlichen Programmierer weitergegeben.

Zur Überprüfung der Testfunktionen verwenden wir das Tool EclEmma ( EclEmma Java Code Coverage, Version 2.3.3.102602231923, ID: com.mountanminds.eclemma.feature.feature.group, Provider: Mountainminds GmbH & Co. KG), welches überprüft, ob alle möglichen Testfälle der geschriebenen Testfunktion durchlauen wurden. Ziel ist Hierbei eine Codecoverage von mindestens 90%.

### Integrationstests

Um die Mosti-Software möglichst sicher und konsistent zu halten, werden alle Methoden auf ein nötiges Mindestmaß an Sichtbarkeit reduziert. Hierbei soll sichergestellt werden, dass interne Änderungen möglichst wenig Auswirkung nach außen und zu anderen Klassen bzw. Packages hat.

Beim Zusammenfügen von mehreren Modulen muss hierbei allerdings getestet werden, ob die Packages untereinander auf alle Funktionen, die sie von anderen Packages brauchen, auch zugreifen können.

Soll ma no beschriben, wie mia des testen?

### System Acceptance Test

Vgl. separates Dokument: Testplan.odt

## Bedienbarkeit und Nutzerinterface (Usability)

Um die Bedienbarkeit und Nutzerfreundlichkeit des Systems noch vor dem Release der Software zu testen, stehen uns verschieden Mitarbeiter der Mosterei Hemau zur Verfügung. Nach einem vorgefertigten Template werden regelmäßig User Acceptance Tests mit verschiedenen Probanden (unterschiedliches Alter, unterschiedliche Erfahrungen mit Computern usw.) durchgeführt, die Ergebnisse ausgewertet und bei Bedarf die Software geändert bzw. angepasst. Geänderte oder neu entstandene Anforderungen werden laut dem Requirement Management Plan behandelt.

## Datenschutz, Datensicherheit (Security)

## Leistungsanforderungen (Performance)

Die Leistungsanforderungen der Mosti-Software werden einerseits in den Unittest von den einzelnen Mitgliedern nach den Festlegungen in der Anforderungsspezifikation geprüft. Des Weiteren können die Probanden des User Acceptance Tests die Leistung der Software in ihren Testbögen bewerten. Ein Härtefalltest, für den in der Datenbank bis zu 1000 Datensätze angelegt werden, ist geplant. Bei Nichterfüllen der Leistungsanforderungen müssen im Rahmen von Codereviews ineffiziente Codestrukturen gefunden und beseitigt werden.

## Zuverlässigkeit

Die Zuverlässigkeit der Mosti-Software wird innerhalb der Unittests geprüft. Dabei wird jede nach außen hin sichtbare Methode durch eine Testfunktion abgedeckt, die der zu testenden Methode jeweils verschiedene Parameter übergibt und das errechnete Ergebnis überprüft. Die Unittests werden sowohl schon während der Programmierung einzelner Module als auch zum Abschluss desselben durchgeführt. Alle Änderungen im Code, die eine Schnittstelle nach außen besitzen, müssen vor dem Release getestet worden sein und die Testdokumentationen den Mitgliedern zugänglich sein.

## Schnittstellen

Die geplante Mosti-Software besitzt verschiedene Schnittstellen nach außen. Die wichtigste hierbei ist die Verbindung zu einer Datenbank, die z.B. Kunden- oder Mitarbeiterdaten speichert. Die Verbindung zu diese Datenbank kann und wird innerhalb den Unittests abgedeckt. Eine weitere Schnittstelle ist die Verbindung zu einem Drucker, der eine Übersicht über die gekauften Produkte und Dienstleistungen eines Kunden druckt. Auch diese Schnittstelle kann im Rahmen von Unit- oder Funktionalitätstests geprüft werden.

## Wartung und Servicefunktionen

## Installation

Um die Installationsanforderungen der Software zu testen, ist ein Funktionalitätstest vorgesehen, innerhalb welchem die Software auf einem Computer neu installiert wird. Für die Kunden ist eine Bedienungsanleitung vorgesehen, die auch die Installationsanleitung enthält.

## Internationalisierung / Lokalisierung

Unsere Software soll nach ihrer Fertigstellung zum Download auf einer entsprechenden Website zur Verfügung stehen. Auch hier sind Tests mit Anwendern geplant, um zu gewährleisten, dass die Software schnell und einfach heruntergeladen werden kann. Wenn die Testergebnisse es implizieren, wird eine Anleitung zum Download erstellt. Möglichkeiten zur Internationalisierung (z.B. verschiedene Sprachen, verschiedene Währungen innerhalb der Mosti-Software) sind nicht geplant.

## Testautomatisierung

*<Welche Tests werden mit welchen Werkzeugen automatisiert? Wer erstellt / prüft die automatisierten Tests?>*

*Es wird im Team entschieden welche Mitglieder die Tests erstellen und welche sie prüfen.*

Die geplanten Unittests werden innerhalb von Eclipse ausgeführt. Dafür wird von Mavenprojekten ein eigener Testordner vorgegeben. Unittests werden mithilfe des Eclipsetools EclEmma durchgeführt.

User Acceptance Tests orientieren sich an den im Testplan spezifizierten Template und werden ohne zusätzliche Werkzeuge ausgeführt.

## Verfolgbarkeit (Traceability)

In der Mitte der Construction-Phase ist eine Sitzung mit allen Teammitgliedern geplant, in welcher über die geplanten und die bisher tatsächlich umgesetzten Anforderungen gesprochen und diskutiert wird.

*Wie wird sichergestellt, dass alle erfassten Anforderungen verifiziert werden?>*

# Übersicht der Testpläne

Bisher dokumentierte Tests:

Kassenfunktion:

Funktionalitätstest Kasse-1.odt

Kundenverwaltung:

Funktionalitätstest KundenVer-1.odt

*<Hier werden alle Dokumente, in denen Testfälle beschrieben werden, identifizierbar erfasst.>*

# Freigabe von Testergebnissen

Es wird im Team entschieden auf Grund welcher Kriterien die Testergebnisse akzeptiert werden.

Für Klassifikationsschema minor – moderate – major für Fehler. Für Alpha-Release werden moderate und minor Fehler, für Final Release nur minor Fehler akzeptiert.