Testkonzept

Inhaltsverzeichnis

1 Einleitung

Durch die steigende Komplexität der heutzutage entwickelten Software steigt auch die Anzahl möglicher Fehler. Es ist jedoch nahezu unmöglich sämtliche Ausnahmefälle abzudecken. Deshalb ist kontinuierliches Testen der Software ein wichtiger und notwendiger Bestandteil der Softwareentwicklung.
Bei teamInstance handelt es sich um eine Webapplikation. Da es schwieriger ist GUI-lastige Anwendungen zu testen ist eine Einhaltung des MVC-Konzeptes erforderlich. Dies ermöglicht einen automatisierten Test des Programmmodells. Für Java hat sich hier das JUnit-Framework etabliert.

Als Maxime sollte hier gelten: *Test until fear turns to boredom.*[1]

1.1 JUnit

JUnit bietet verschiedene Methoden zum Testen an. Um eine Testklasse zu erstellen leitet man diese von `junit.framework.TestCase` ab und fügt ihr Testmethoden hinzu. Diese haben das Präfix `test`. Anschließend kann der Test mittels `run()` aufgerufen werden. Um nicht jeden Klassentest einzeln auszuführen, empfiehlt es sich pro Paket eine *TestSuite* zu erstellen welche alle Tests des Paketes sowie der Unterpakete enthält. So muss zum Schluss nur noch eine TestSuite gestartet werden und alle darunterliegenden Tests werden automatisch mit durchgeführt.

2 Testkonzept

Das Testkonzept teilt sich in vier Phasen auf, nämlich

1. Komponententests
2. Integrationstests,
3. Systemtests und den
4. Abnahmetest.

2.1 Komponententests

Am Anfang erfolgen die Komponententests, also die Tests der einzelnen Klassen. Dazu ist nach dem „Tests-First“-Prinzip von jedem Programmierer bereits vor der Implementierung eine Testklasse mit geeigneten Testszenarien zu erstellen. Um die in der Spezifikation festgelegte funktionale Korrektheit zu zeigen muss sie alle festgelegten Testszenarien erfüllen. Weiterhin ist für jedes Paket auch eine TestSuite `AllTests.java` zu erstellen. In ihr sind alle Testklassen des Paketes sowie seiner Unterpakete zu sammeln um eine leichte Testbarkeit aller Klassen zu garantieren.

[1] Zitat von J. B. Rainsberger, JUnit-FAQ

2.2 Integrationstests

Nach dem erfolgreichen Abschluss der Komponententests kann mit dem Integrationstest begonnen werden. In diesem wird das Zusammenspiel mehrerer Klassen getestet. Hierbei steht vor allem das Testen von Schnittstellen im Vordergrund. Es bieten sich verschiedene Strategien an:

Inkrementelle Strategie Hierbei werden Komponenten zuerst in kleinen Gruppen integriert und dann Stück für Stück erweitert.

Testzieloriente Strategie Es wird vorher ein Testziel festgelegt und dabei die für dieses Testziel erforderlichen Komponenten integriert.

Geschäftsprozessorientierte Strategie Gemäß ihrer Zugehörigkeit zu den einzelnen Geschäftsprozessen findet die Integration der einzelnen Komponenten statt.

Funktionsorientierte Strategie Die Komponenten werden nach funktionalen Merkmalen integriert.

Im Rahmen des Softwaretechnikpraktikums verwenden wir die geschäftsprozessorientierte Strategie.

2.3 Systemtest

Das Ziel des Systemtests ist die Überprüfung des Gesamtsystems auf Erfüllung der im Pflichtenheft spezifizierten Anforderungen. Dazu muss der Test in einer möglichst realistischen Umgebung, d.h. in einer Hardware- und Softwareumgebung, die der des Kundens möglichst ähnlich ist, durchgeführt werden.
Beim Systemtest werden sowohl funktionale als auch nichtfunktionale Anforderungen getestet. Die wichtigste funktionale Anforderung ist hierbei die Richtigkeit des Produktes. Aber auch Sicherheit, Angemessenheit, Ordnungsmäßigkeit und Interoperabilität sind wichtige funktionale Anforderungen. Zur Erfüllung der funktionalen Anforderungen werden dazu folgende Tests durchgeführt:

- Funktionstest,
- Sicherheitstest und
- Interoperabilitätstest.

Wichtige nichtfunktionale Anforderungen sind z.B. Effizienz, Zuverlässigkeit und Benutzbarkeit. Um diese zu testen führt man folgende Tests durch:

- Leistungstest,
- Performanztest,
- Test des Speicherverbrauchs und der Auslastung des Prozessors,
- Recovery Testing und einen
- Benutzbarkeitstest (Usability-Test)

Da es in der Praxis nicht möglich ist sämtliche Eingaben zu testen, ist es wichtig ein gutes Maß an aussagekräftigen Testfällen zu finden. Der Systemtest wird unter Teilnahme aller Teammitglieder durchgeführt und vom Testverantwortlichen geleitet.

2.4 Abnahmetest

Der Abnahmetest steht am Ende des Softwareentwicklungsprozesses und wird vom Auftraggeber durchgeführt.

3 Testablauf

Jedes Zweieream ist während der Implementierungsphase für die Beseitigung aufgetretener Fehler verantwortlich. Treten Fehler auf, die nicht ohne weiteres behoben werden können, so ist ein detailliertes Fehlerprotokoll zu erstellen und den anderen Teammitgliedern zugänglich zu machen, um Lösungen zu entwickeln.