Aufgabe 7:
Testkonzept

Inhaltsverzeichnis:

1. Einleitung

Durch den steigenden Funktionsumfang heutiger Softwareanwendungen, insbesondere Webanwendungen, steigt auch die Fehleranfälligkeit in bestimmten Prozessen.
Deshalb ist es nahezu unmöglich sämtliche Ausnahmefälle zu erkennen und zu behandeln.
Aus diesem Grund ist es wichtig die zu entwickelnde Software während der Entwicklung kontinuierlich zu testen. Für Java-Applikationen hat sich dazu das JUnit-Framework etabliert.

1.1. JUnit

JUnit bietet verschiedene Möglichkeiten zum Testen von Anwendungen an. Um eine Testklasse zu erstellen leitet man diese von *junit.framework.TestCase* ab und fügt ihr Testmethoden, mit dem Präfix '*test*', hinzu. Dieser muss *parameterlos*, *public* und nicht *static* sein. In diesen können dann beispielsweise Methoden aus *junit.framework.Assert*[1] aufgerufen werden. Dabei werden in der Testmethode Testwerte und die zu erwartenden Ergebnisse definiert. Die Methode liefert, wenn das Ergebnis nicht mit der Vorgabe übereinstimmt wird ein Fehler erzeugt.
Sinnvoll ist es pro Paket eine Testsuite (*AllTests.java)* zu erstellen, um nicht jeden Klassentest einzeln auszuführen.

2.Testkonzept

Das Testkonzept lässt sich in vier Phasen aufteilen:

2.1. Komponententests
2.2. Integrationstests
2.3. Systemtests
2.4. Abnahmetest

2.1. Komponententests

Zu Beginn werden die einzelnen Klassen getestet, die so genannten Komponententests. Nach dem Test-First-Prinzip wird von jedem Programmierer bereits vor der Implementierung eine Testklasse mit geeigneten Testszenarien erstellt. Um die in der Spezifikation festgelegte funktionale Korrektheit zu zeigen muss sie alle festgelegten Testszenarien erfüllen.
Hier ist für jedes Paket auch die 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.

2.2 Integrationstests

Anschließend wird mit den Integrationstests begonnen. Hier wird nun das Zusammenspiel verschiedener Klassen überprüft. Hierbei steht das Testen von Schnittstellen im Vordergrund. Dies kann durch verschiedene Strategien realisiert werden.
Inkrementelle Strategie: Die Komponenten werden zuerst in kleine Gruppen integriert und anschließend langsam erweitert.
Testzielorientierte Strategie: Bereits vorher wird ein Testziel formuliert und es werden die zum erreichen dieses Ziels erforderliche Komponenten integriert.
Geschäftsprozessorientierte Strategie: Die Komponenten werden gemäß ihrer Zugehörigkeit zu Geschäftsprozessen integriert.
Funktionsorientierte Strategie: Die Komponenten werden nach funktionalen Merkmalen integriert.

Da wir bereits eindeutig definierte Geschäftsprozesse formuliert haben werden wir uns während der Entwicklung auf die Geschäftsprozessorientierte Strategie konzentrieren.

1 Ein Satz von Methoden, welcher nur Ausgaben liefert wenn die Testmethode scheitert.

2.3. Systemtest

Beim Systemtest wird das Gesamtsystem auf die Erfüllung der im Pflichtenheft festgelegten Anforderungen überprüft. Hierbei wäre es sinnvoll den Test auf einer möglichst realistischen Umgebung zu testen, das heißt auf einer Hardware und Softwareumgebung, welche die des Kunden relativ ähnlich ist. Beim Systemtest wird ein spezielles Augenmerk auf die Richtigkeit des Softwareprodukts gelegt. Es wird aber auch auf Sicherheit, Angemessenheit, Ordnungsmäßigkeit und Interoperabilität sind wichtige funktionale Anforderungen. Hierzu werden folgende Test ausgeführt:

- Funktionstests
- Sicherheitstests
- Interoperabilitätstests

Wichtige nichtfunktionale Anforderungen sind unter anderem Effizienz, Zuverlässigkeit und Benutzbarkeit, dies wird durch

- Leistungstests
- Performanztests
- Tests des Speicherverbrauchs und der Prozessorauslastung
- Recovery Testing
- Benutzbarkeit

getestet.

Es ist wichtig einige gute, aussagekräftige Testfälle zu finden, da man nicht alle möglichen Eingaben testen kann. Der Systemtest sollte von allen Teammitgliedern gemeinsam durchgeführt werden. Gegebenenfalls sollte man auch aussenstehende Personen in den Test einbeziehen um einige neue Testideen zu bekommen.

2.4. Abnahmetest

Der Abnahmetest wird am Ende der Softwareentwicklung durch den Auftraggeber durchgeführt.

Testablauf:
Jedes Zweierteam ist während der Implementierungsphase für die Beseitigung auftretender Fehler verantwortlich. Sollte dies nicht ohne weiteres möglich sein, so sollte ein Fehlerprotokoll erstellt werden und den anderen Teammitgliedern zugänglich gemacht werden oder der Fehler sollte bei einem Treffen mit allen Gruppenmitgliedern angesprocchen werden und wenn möglich auch gleich dort behoben werden.