Testkonzept

1. Einleitung

Es ist wichtig, die auszuführenden Testreihen genau zu planen, um die Gewähr zu haben, dass bei späteren Modifikationen am Design bzw. Quelltext alle explizit und implizit bestehenden Abhängigkeiten und Voraussetzungen auch weiterhin berücksichtigt sind. Um die Zuverlässigkeit und die Qualität der Software und des gesamten Systems zu verbessern,sind Tests durchzuführen.

2.Überblick über Junit

JUnit bietet dem Programmierer die Möglichkeit sogenannte Unit Tests automatisch durchführen zu lassen. Eine solche Unit kann einzelne Methoden über ganze Klassen bis hin zu ganzen Komponenten darstellen. Dabei laufen diese Einheiten völlig getrennt voneinander ab, sodass es zu keinen ungewollten Einstreueffekten kommt. JUnit stellt dazu einen Rahmen in Form von Java Klassen zur Verfügung. Mit diesen Klassen werden Unit Tests entwickelt. Es wird geprüft ob für bestimmte Eingaben auch das erwartete Ergebnis vorliegt. Mit den Test Runners ist es möglich einzelne Test Units oder gar ganze Test Suites vollautomatisch zu testen. Da man diese Test Units und Suites immer wieder ausführen kann ohne jedesmal von Hand ein Testszenario zu schaffen ist es möglich konsequent und konsistent für alle Teammitglieder zu testen. Zu jeder Klasse wird eine Testklasse erzeugt, die von junit.framework.TestCase abgeleitet wird.
Die wichtigen Schritte zum Konstruieren des JUnit-Frameworks:
• Importieren aller Klassen unter junit.framework.* mit der import-Anweisung
• Vererbung von Testcase mit extends-Anweisung
• Aufruf der Methode super(string) in Konstruktor
Zu jeder zu testenden Methode XXX wird eine testXXX()-Methode in der Testklasse erstellt. In dieser erfolgt der Aufruf der zu testenden Methode zusammen mit den jeweils passenden Parametern. Danach werden in diversen assertXXX()-Anweisungen die zu erfüllenden Erwartungen formuliert.
Das Entwurfsmuster von JUnit ist als Kompositum (engl. composite) bekannt. Es erlaubt genau, dass wir beliebig viele TestCase und TestSuite Objekte zu einer umfassenden Testsuite-Hierarchie kombinieren können.

3.Komponententest

Komponententest erfolgen einzeln für jede implementierte Klasse und jedes Package. Diese Test werden von den Programmierern der entsprechenden Story durchgeführt. Dazu wird zu jeder Klasse eine Testklasse erzeugt. Diese beinhaltet die Testmethoden. Der Name der Testmethoden setzt sich aus “test“ und dem Namen der zu testenden Methode zusammen.
Beispiel Testmethoden: testgetBenutzername(), testsetBenutzername(), ...
Beispiel einer Testklasse:
import java.util.*;

```
import junit.framework.*;
public class TestExample extends TestCase{ public TestExample(String testname)
{ super(testname);
}
public testsetBenutzername(Benutzername name)
{ this.setBenutzername (name);
assertEquals(name, this.getBenutzername ());
}...
Public static void main(String[] args)
{ junit.swingui.TestRunner.run(TestExample.class);
}
}
```

In diesem vereinfachtem Beispiel wird überprüft, ob eine Benutzername korrekt mittels setBenutzername(Benutzername name) gesetzt wird. Liefert assertEquals(Benutzername, this.getBenutzername()) ein false,wird der Test abgebrochen.(als Voraussetzung wird angenommen, dass getProperty() bereits erfolgreich getestet wurde).

4.Integrationstest

Der Integrationstest ist die erste Generalprobe für die in einem Team entwickelten Softwarekomponenten.Da alle Systemkomponenten und deren Beziehungen untereinander während des Tests sichtbar sind , kann man den Integrationstest auch als „white box“ Test bezeichnen. Aufgabe des Integrationstests ist es, das fehlerfreie Zusammenwirken von Systemkomponenten und Teilmodulen zu überprüfen. Voraussetzung für den Integrationstest ist, dass jede einzelne Systemkomponente in einem Komponententest (Entwicklertest) für sich bereits geprüft ist. Stück für Stück werden die Komponenten zusammengeführt (integriert) und das Zusammenwirken getestet. Komponenten, die im Test noch nicht integriert sind, werden durch Testtreiber bzw. Platzhalter ersetzt.

5.Systemtest

Jetzt soll es bei dem schrittweise einzelne Komponenten des Systems hinzugenommen und das resultierende Gebilde getestet werden. Das Ziel des Systemtests ist die Überprüfung des Gesamtsystems (d.h. der Test des komplett integrierten Systems)auf Erfüllung der aus den Benutzeranforderungen abgeleiteten Entwickleranforderungen. Beim Systemtest wird die Erfüllung sowohl von funktionalen wie auch nicht-funktionalen Anforderungen geprüft. Die wichtigste funktionale Anforderung ist sicherlich die Richtigkeit der errechneten Ergebnisse, aber es existieren noch weitere funktionale Kriterien wie z.B. Sicherheit. Daneben muss das System auch nicht-funktionale Anforderungen einhalten. Dies sind Qualitätskriterien wie z.B. Effizienz, Benutzbarkeit und Zuverlässigkeit.

Testsuite

Das XP-Paradigma „Tests first“ verlangt, dass zu jeder Story bereits vor Beginn der Implementierungsarbeiten ein Satz von Testbeispielen angelegt wird, die vom zu erstellenden Code ohne Beanstandungen abgearbeitet werden. Der Satz von Testbeispielen wird dann in die Testsuite integriert, gegen die alle folgenden Release-Versionen des Systems stabil laufen müssen.
Im folgenden werden die funktionalen Tests genau beschrieben:

1.Testsuite(gemäß Storie 1)

- Kontrolle, ob die Anzeige der Daten unter verschiedenen Browsern und mit verschiedenen Auflösungen funktioniert.
- Kontrolle der Funktionsfähigkeit der Buttons und Links gemäß der bis dahin umgesetzten Teile der Software.

2.Testsuite(gemäß Storie 2)

- die Tests aus der 1. Testsuite müssen mit den neuen Implementierungen weiterhin erfolgreich verlaufen.
- Test, ob nach erfolgreicher Anmeldung am System jeder Benutzer die richtigen Zugriffsrechte erhalten hat, d.h. z.B. dass nur der Administrator das System administrieren kann.
- Test, ob der Administrator erfolgreich einen Benutzer anlegen, speichern und löschen kann.

3.Testsuite(gemäß Storie 3)

- die Tests aus der 1. und 2. Testsuite müssen mit den neuen Implementierungen weiterhin erfolgreich verlaufen.
- Test, ob der Nutzer die Möglichkeit aus den hinterlegten Ontologien in einer Drop-Down-Liste die gewünschte auszuwählen hat.
- Test, ob die Klassen der gewählten Ontologie tabellarisch angezeigt werden können.
- Kontrolle des Klassenauswahl.
- Test, ob die Instanzen nach den Filterkriterien angezeigt und in einer Tabelle dargestellt werden können.

4.Testsuite(gemäß Storie 4)

- die Tests aus der 1., 2. und 3. Testsuite müssen mit den neuen Implementierungen weiterhin erfolgreich verlaufen.
- Test, ob der Nutzer die Möglichkeit sein hinterlegtes Profil abzurufen und sich

dieses anzeigen zu lassen hat.

- Test, ob die Benutzerdaten erfolgreich geändert werden können, diese Änderungen gespeichert werden und dass der Benutzer anschließend eine Bestätigung erhält.
- Test, ob neues Passwort erfolgreich erstellt werden konnte, d.h. ob der Benutzer sich am System mit Hilfe des neuen Passwortes einloggen kann, sonst muss er eine Fehlermeldung erhalten.
- Test, ob die Spalten gemäß den Nutzerangaben angezeit und ausgeblendet werden können.

5.Testsuite(gemäß Storie 5)

- die Tests aus der 1., 2., 3.und 4. Testsuite müssen mit den neuen Implementierungen weiterhin erfolgreich verlaufen.
- Test, ob anhand der tabellarisch dargestellten Instanzeigenschaften sortiert werden können.
- Test, ob die Editierbarkeit geprüft werden kann.
- Test, ob die Editierfunktion von den Instanzen funktioniert.
- Test, ob bei der Ausführung der einzelnen übersichten bei einem Aufruf die Nutzerrechte überprüft werden kann.
- Kontrolle der Ablaufreihenfolge der gesamten Programme.

Die Tests des Laufzeitverhaltens werden durchgeführt, nachdem die Funktionalität erfolgreich getestet wurde.