Softwaretechnik-Praktikum SS 2007

GR-07-2
Projektleiter: Michael Hütel Autor: DS,JN

1. Einleitung

Ein fehlerfreies System kann bei komplexen Softwareprodukten in der Regel nicht garantiert werden. Um dennoch korrekte Funktionalitäten zu gewährleisten, muss schon von Beginn an der Entwicklungsphase kontinuierlich getestet werden. Durch das Vorbeugen, Erkennen, Aufspüren und Beseitigen von Fehlern beim Testen, werden später Herstellungs- und Qualitätskosten gespart und Folgekosten vorgebeugt.
Zur Sicherstellung der Zuverlässigkeit und Qualität der Prüfungsanmeldung werden bereits während der Implementierungsarbeiten Stories und Klassen in Komponententests geprüft. Zur Automatisierung der Testumgebung ziehen wir das JUnit-Framework heran. Zusätzlich finden eine Reihe von Systemtests statt, die speziell auf die Benutzeroberfläche aus Sicht des Anwenders ausgelegt sind.

2. JUnit

Mit dem JUnit-Framework können kleine Einheiten (Units) wie Klassen oder Methoden Java-basierter Programme automatisiert getestet werden. Dazu erstellt man Testklassen die von junit.framework.TestCase abgeleitet und mit Testmethoden ausgestattet werden. Die Methoden haben den Präfix test. Anschließend kann der Test mittels run() aufgerufen werden.
Im Falle eines Fehlschlages gibt JUnit die nötigen Exceptions zur Bestimmung der Ursache aus. Die einzelnen Testreihen können in eine Testsuite integriert werden, da so nicht jeder Klassentest einzeln ausgeführt werden muss. Für jedes Paket wird eine Testsuite erstellt, die alle Tests des Paketes und der Unterpakete enthält. So werden dann alle eingebetteten Tests automatisch durchgeführt.

3. Tests

3.1 Komponententest

Komponententests beziehen sich auf einzelne Klassen, Methoden und deren zusammengesetzte Strukturen, die aufsteigend nach Komplexität geprüft werden, bis hin zum Test der ganzen Story (Integrationstest für das Zusammenspiel der einzelnen Komponenten). Nach dem XP-Paradigma „Tests first“ erstellt der Implementierer dazu von vornherein die benötigten Testklassen mit aussagekräftigen, repräsentierenden Instanzen. Die Testklassen werden dann paketweise in Testsuiten integriert.
Zur Übersicht verwenden wir folgende Notationen:

- Die Testklassen heißen „Test“ + Klassenname
- Die Testsuites heißen „Suite“ + Paketname
- Zu jedem Paket muss eine entsprechend erfolgreich durchgeführte Testsuite mit den beinhaltenden Testklassen vorliegen

JUnit stellt einige Prüf-Methoden zur Verfügung , mit denen Bedingungen für einen erfolgreichen Test definiert werden können:

- **void assertTrue**(boolean condition): Fehler wenn boolscher Ausdruck false ist
- **void assertEquals**(Object expected, Objekt actual): Prüfung auf Gleichheit
- **void assertSame**(Object expected, Object actual): Prüfung ob zwei Objekte identisch sind

- **void assertNull**(Object obj), **void assertNotNull**(Object obj): Prüfung ob Objekt „null“ ist oder nicht

3.2 Systemtest

Der Systemtest überprüft das gesamte Softwareprojekt; im ersten Schritt auf seine funktionalen Anforderungen bezüglich des Pflichtenheftes. Der Test wird dabei aus Sicht des Anwenders durchgeführt, wobei nicht der innere Aufbau sondern nur die Benutzeroberfläche des Systems sichtbar ist. Wichtig sind hier insbesondere der Funktionstest, der Sicherheitstest und der Interoperabilitätstest. Im zweiten Schritt werden nichtfunktionale Anforderungen des Gesamtsystems kontrolliert. Dazu gehören unter Anderem:

- Vollständigkeit
- Leistung
- Performanz
- Zuverlässigkeit
- Fehlertoleranz/Robustheit
- Benutzbarkeit
- Dokumentation

Eine Automatisierung des Systemtests ist in der Regel nicht möglich. Deshalb sollte dieser in Gegenwart der kompletten Entwicklergruppe durchgeführt oder sehr sorgfältig protokolliert werden. Besonders wichtig ist auch hier die Wahl aussagekräftiger Testszenarien, da ein alles umfassender Test unmöglich ist.