Um fehlerfreien Quellcode zu gewährleisten, ist es erforderlich Komponenten des Programms und das Programm selbst ausführlich zu testen. Das hilft Schwachstellen aufzudecken und diese noch während der Implementierungsphase zu verbessern.

Es müssen Komponenten-Tests und System-Tests durchgeführt werden.

Komponenten-Tests

Die Komponenten-Tests, welche die Funktionstüchtigkeit von Methoden gewährleisten, sind mit JUnit (www.junit.org) durchzuführen. JUnit ist in Java geschrieben und bietet so beste Voraussetzungen, um Plattformunabhängig Java-Klassen zu testen. Zudem ist JUnit bereits in der Programmierumgebung Eclipse bzw. Websphere integriert.

Diese Testumgebung nutzt Black-Box-Test's. Das heißt, Code-Einheiten werden über spezielle Testfälle auf korrekte Funktionsweise getestet. Dabei steht die richtige Verarbeitung von Eingaben im Mittelpunkt.

Vor der Implementierung der Story sind geeignete Testbeispiele von demjenigen zu erstellen, der den Code schreibt. Diese sollten alle Klassenübergreifenden Schnittstellen beinhalten, um sicherzustellen, dass diese auch korrekt funktionieren. Dazu gehören auch Konstruktoren und, falls als Schnittstelle definiert, get-, set- und die toString-Methoden. Es sollen vor allem ausführliche Tests bei zu erwartenden Problem-Fällen erstellt werden.

Nachdem die Story fertig und fehlerfrei abgeschlossen wurde wird die dazugehörige Testklasse in einer Testsuite integriert. Diese ruft mehrere Testklassen auf und prüft diese auf Fehlerfreiheit. Die Testsuite muss auch in allen späteren Versionen fehlerfrei funktionieren, um Probleme durch Abhängigkeiten anderer Module zu vermeiden.

Eine Testklasse kann folgende Methoden haben:

- »setUp()«
  Hier gehören die Methoden und Aufrufe hinein, die vor dem eigentlichen Test aufgerufen werden müssen. Dazu kann z.B. das Instanzieren einer Klasse gehören.
- »runTest()«
  Ruft den eigentlichen Test auf.
  Durch Reflektion - eine Java-Technik, die es ermöglicht, zur Laufzeit die Methoden einer Klasse abzufragen - wird die Sache noch einfacher. Statt »runTest()« zu überschreiben, muss die Testklasse nur eine Reihe von Methoden nach dem Schema »void testXXX()« enthalten. Durch den Konstruktor »TestSuite(MyTestClass.class)« werden dann automatisch alle Testmethoden innerhalb der Testsuite durchgeführt.
- »tearDown()«
  Hier gehören Methoden hinein, die nach deinem durchgeführten Test ausgeführt werden müssen. Dazu gehört z.B. dass Dateien wieder geschlossen werden oder dass reservierter Speicher wieder freigegeben wird.

Alle drei Methoden sind leer implementiert und können vom Entwickler überschrieben werden.

Mögliche Test-Befehle für die Testklasse sind zum Beispiel: assertTrue(boolean t);

- assertFalse(boolean t);
- assertEquals(Object o1, Object o2);
- assertNull(Object o);

- assertNotNull(Object o);
- assertSame(Object o1, Object o2);

Wobei Object hier für eine beliebige Datenstruktur/Rückgabewert steht.

Weitere Infos und Beispiele liefert www.junit.org und der Test-Verantwortliche.

Systemtests

Da JUnit nur Komponenten testen kann, ist es für System-Tests unbrauchbar. Die Systemtests müssen durch überprüfen der einzelnen Punkte durchgeführt werden. Zu diesen Tests gehören die folgenden Punkte:

- die Überprüfung der Login-Funktionalität:
  Ein Benutzer darf sich nur unter seinem Login und Passwort anmelden können. Ist das Passwort falsch muss der Benutzer abgewiesen werden. Ist es korrekt, so muss getestet werden, ob er die korrekten Benutzerrechte besitzt (Student, Tutor/Kontrolleur, Dozent)
- Test der Passwortsicherheit:
  Es muss überprüft werden, dass Unbefugte keinen Zugriff auf die Passwörter erhalten können. Also dass sie weder die XML-Datei mit Passwörtern einsehen können, noch dass sie bei irgendwelchen Vorgängen die Passwörter von anderen Benutzern mitbekommen
- Test des HTML-Code:
  Ein weiterer Systemtest ist die Überprüfung der korrekten Darstellung der Webseiten in den üblichen Browsern (Internet Explorer, Netscape/Mozilla, Opera). Dabei muss geprüft werden, ob auch alle im Pflichtenheft niedergeschriebenen Funktionen in den Browsern nutzbar sind.
- Klausuranmeldung / Übungsanmeldung
  Es muss getestet werden, ob die Anmeldungen richtig in die XML-Datei geschrieben werden und so nach einem Neustart des Programms keine Daten verloren gehen.
- Punkte eingeben
  Es muss sichergestellt sein, dass die Punkte von den berechtigten Personen eingegeben werden können. Studenten, dürfen hier keine Änderungen vornehmen können.
- Punkte abrufen
  Hier ist zu testen, ob auch wirklich die Punkte des eingeloggten Benutzers erscheinen und keine weiteren.
- kommunizieren: Nachrichten, die hinterlassen wurden sollen auch nur von demjenigen lesbar sein, an den sie gerichtet ist.
- Weitere einfache Tests:
  - Lösungen herunterladen
  - Übungen hochladen

All diese Tests sind, sofern die Funktionalitäten schon implementiert sind, in jeder Iterationsstufe erfolgreich durchzuführen. Die Testläufe am Ende jeder Story sind vom Verantwortlichen für Tests in einer Textdatei zu protokollieren.