Testkonzept

Einleitung:

Das Testen von Quellcode ist ein fester Bestandteil von der Implementierungsphase. Jeder neue geschriebene Code soll getestet werden damit im nachhinein keine Fehler auftreten können. Wenn dies nicht durchgeführt wird, könnte das Auffinden von Fehlern sehr schwierig und langwierig werden oder die Fehler werden erst vom Kunden erkannt, was natürlich nicht sein soll. Des weiteren gibt das Testen von Quellcode dem Implementierer die Sicherheit alles richtig gemacht zu haben, womit er dann einen Schritt weiter ist. Man könnte es mit Klettern ohne Seil und Haken vergleichen. Der Kletterer hat nie die Sicherheit das er eine Etappe geschafft hat ohne auf den Anfang zurückzufallen. So könnte es auch dem Implementierer ergehen wenn er sich nicht mit Test seines Quellcodes absichert.

Anwendungsfälle mit J-Unit testen

Zum Testen von Anwendungsklassen gibt es J-Unit. Ein Java-Framework, welches das Schreiben und Ausführen von automatisierten Unit Tests erlaubt. Dies geschieht über eine Bedienungsoberfläche. Der Implementier erhält mit J-Unit ein gebrauchsfähiges Werkzeug um sein Testprozesse konsistent und konsequent zu automatisieren. Da J-Unit mit Quellcode erhältlich ist kann man es auch leicht erweitern. J-Unit ist Opensource und kann zum Beispiel unter http://sourceforge.net/projects/junit/ herunterladen kann.

Aber wie funktioniert das nun mit J-Unit?
Am besten lässt sich dies an einem Beispiel erklären. Wir haben eine Klasse „SummeBerechnen". Es soll nun getestet werden ob diese Klasse auch die Summe berechnet und zwar auch nur auf 2 Stellen hinter dem Komma. Also erstellen wir einen Test der die Werte 2.801 und 2.202 beinhaltet. Es soll ja nun 3.00 heraus bekommen. Wenn dies der Fall ist unsere Klasse „SummeBerechnen" richtig. Tritt dies nicht ein und ein anderer Wert wird zurückgeliefert haben wir einen Fehler in der Implementierung der Klasse „SummeBerechnen" gemacht.

Aufbau von J-Unit Testfälle:

Für jede Anwendungsklasse brauchen wir mindestens eine oder mehre Testfallklassen. Genauso kann es sein das wir nur eine Testfallklasse für mehrer Anwendungsklassen brauchen.
Wie viele Testfallklassen wir erzeugen ist im Moment schwer zusagen. Da sich Testfälle nicht nur auf einzelne Anwendungsklassen sondern auch auf logische Zusammenhänge von Anwendungsklassen aufbauen lassen. Im Allgemeinem gilt für den Aufbau von Testfällen:

1. Objekte erzeugen und in Ausgangszustand bringen
2. Methoden der Objekte heraus kristallisieren
3. Erwartete und tatsächliche Resultate vergleichen

Es kann auch möglich sein das wir auf bestehende Tests zurück greifen, wenn es sich für den zu testenden Anwendungsfall anbietet. Sonst muss natürlich ein neuer Test implementiert werden. Die Benennung von Tests soll so erfolgen, dass man im nachhinein erkennt welche Anwendungsklasse wir mit welchem Test getestet haben. Also zum Beispiel:
Die Anwendungsklasse „Schnittpunkt" wir getestet mit „TestSchnittpunkt".

Konzeptionelle und organisatorische Festlegung der Tests

Story	*Zeitscheibe*	*1. Mitglied PP*	*2.Miglied PP*
1.	1	Ronny Fauth	Michael Freyer
2.	1	Christian Kube	Carsten Enke
3.	1	Conrad Koch	Sören Weißenborn
4.	2	Ronny Fauth	Sören Weißenborn
5.	2	Christian Kube	Michael Freyer
6.	2	Conrad Koch	Carsten Enke
7.	3	Ronny Fauth	Carsten Enke
8.	3	Christian Kube	Sören Weißenborn
9.	3	Conrad Koch	Michael Freyer
10.	4	Christian Kube	Conrad Koch
11.	4	Carsten Enke	Michael Freyer
12.	4	Ronny Fauth	Sören Weißenborn

Jedes PP-Team hat seine Tests sofort nach dem Erstellen an den Verantwortlichen für Tests zu übergeben. Aus den einzelnen Tests soll dann zu einer Release eine Testsuite erstellt werden. Die oben aufgeführten PP-Teams haben sich zu den von ihnen zu erfüllenden Stories geeignete Tests zu überlegen. Die Tests werden vor der Programmierung der Klassen fertig gestellt und dem Verantwortlichen für Test übermittelt.
Die Testsuite zu einem Release ist dann jedem PP-Team zugänglich und es können vorhandene Test weiter verwendet bzw. vorhandener Quellcode in anderen Test verwertet werden.