Testkonzept

Gruppe: Geo05

Erstellt durch Verantwortlichen für Tests: Karsten Kunze

1. Allgemeines:

Der Verantwortliche für Tests übernimmt die Koordination der Tests. Des weiteren implementiert er die Testklassen.

Die Tests werden wöchentlich durchgeführt, wobei der aktuelle Quellcode per CVS den Tester erreicht. Wenn beim Testen mit Junit Fehler festgestellt werden, wird der Implementierer benachrichtigt. Dies geschieht mittels der Textdatei „Testlog“, welche per E-Mail übermittelt wird. Der Implementierer versucht dann den Fehler zu beheben, **bevor** weitere Neuerungen am Quellcode bzw. Programm vorgenommen werden. Mittels weiterer Kommunikation über CVS (für den Quellcode) und E-Mail (Übermittlung des Testlog) werden nach und nach alle Fehler behoben bis das aktuelle Programm fehlerfrei läuft. Somit ist gewährleistet, das Fehler relativ frühzeitig erkannt werden und des wegen eine Anhäufung von mehreren Fehlern weitestgehend verhindert wird. Alle Unit Tests werden in der Klasse „Tests“ zusammengefasst. Wir benutzen folgendes Prinzip:

2. Konzept „Test first“:

Die Testmethoden werden **vor** dem eigentlichen Start der jeweiligen Implementierungsphase im gemeinsamen Gruppengespräch festgelegt. Der Verantwortliche für Tests schreibt dann die Unit Tests, noch bevor wir den zu testenden Programmcode schreiben. Idealerweise wird jede funktionale Programmänderung zuvor durch das Schreiben eines weiteren Tests motiviert. Wir entwerfen diesen Test so, dass er zunächst fehlschlägt, weil das Programm die gewünschte Funktionalität noch nicht besitzt. Erst anschließend schreiben wir den Code, der den Test zum Laufen bringt. Auf diese Weise wird die gesamte Programmentwicklung durch das unmittelbare Feedback konkreter Tests angetrieben.

Frage: Was wollen wir denn testen, wenn noch kein Quellcode geschrieben wurde?
Antwort: Zuerst die Tests zu schreiben, ist eine Möglichkeit, um herauszufinden, was wir programmieren müssen und was nicht, und wie wir auch sicherstellen können, dass wir tatsächlich programmieren werden, was wir programmieren wollten.

Um das so genannte „Testgetrieben Programmieren“ zu realisieren, analysieren wir zunächst, wie die erforderliche Klasse funktionieren und sich verhalten sollte. Wir dokumentieren dann Ihr Verständnis in einem ausführbaren Unit Test. Man kann sich dabei vorstellen, der zu testende Code wäre einfach schon geschrieben. Wir entwerfen die Tests aus der Verwendungsperspektive, so, wie Sie sich die Schnittstelle der zu testenden Klasse wünschen.

Beispiel: Man hat eine Klasse „Euro“, die es zu testen gilt. Uns interessiert noch vor dem Entwickeln dieser Klasse, ob zum Beispiel die Eurobeträge korrekt in einer Variable vom Typ double gespeichert werden und ob eine Funktion „GibBetrag“ korrekt funktioniert. Also wird die Testklasse „EuroTest“ unter anderem mit der Methode „TestGibBetrag“ ausgestattet sein, welche testet, ob beim Erstellen von „2.00 Euro“ auch wirklich der korrekte Betrag von der Methode „GibBetrag“ zurückgegeben wird.

Das frühe Testen hat eine ganze Reihe positiver Auswirkungen. Ein Vorteil dieser Technik wird hier schon unmittelbar deutlich. Wenn man die Tests zuerst schreibt, wird der Quellcode auch testbar sein. Den Test haben wir ja schließlich gerade schon geschrieben. Quellcode dagegen, der nicht unter Gesichtspunkten der einfachen Testbarkeit entworfen wurde, lässt sich auch nachträglich meist nur noch schwer testen.
Wir führen nach jedem Schritt alle gesammelten Tests aus, um sicherzugehen, dass wir nicht ungewollt das Verhalten des Programms verändert haben.

3. Vorgehensweise:

Der Zyklus des Testens und Programmierens lässt sich wie folgt darstellen:

1. Besprechung der Gruppe vor der jeweiligen Implementierungsphase
2. Der Verantwortliche für Tests schreibt zunächst die Tests, welche vorher abgesprochen wurden. Dieser schlägt zunächst fehl, da noch kein Quellcode vorhanden ist.
3. Der Implementierer schreibt (bzw. korrigiert) den Quellcode und übermittelt ihn per CVS dem Verantwortlichen für Tests.
4. Der Verantwortliche für Tests testet den Quellcode und dokumentiert auftretende (bzw. behobene) Fehler in der Textdatei „Testlog“, welche er nach Beendigung seiner Arbeit dem Implementierer per E-Mail zukommen lässt.
5. Schritte 3 und 4 werden solange wiederholt, bis keine Fehler mehr auftreten bzw. alle Fehler behoben wurden.

4. Aufbau „Testlog.txt“:

Es sollte folgende Notation verwendet werden:

1. *Bei festgestellten Fehlern:* Datum : Testnummer : Fehlertext von Junit
2. *Bei behobenen Fehlern:* Datum : Testnummer : „behoben“

Die Testnummer ist fortlaufend zu wählen beginnend mit „Test001“.

Beispiel „Testlog.txt“:

```
22.10.03 : Test001 : Error in class K1

23.10.03 : Test002 : Error in class K2

24.10.03 : Test001 : behoben

…
```