Ausgabe 2012 / 3
Erscheinungsart: ca. 4x jährlich in elektronischer Form

Unit-Tests

Warum Unit-Tests schreiben?

Wir alle kennen das: Der Zeitpunkt für das Release der neuesten Version unserer Software ist gekommen. Der Product Owner fragt, ob wir wirklich ausliefern können, ob auch wirklich alles getestet ist und ob auch sicher keine Fehler durch Seiteneffekte in nicht veränderten Systemteilen vorhanden sind.

In vielen Softwareprojekten können diese Fragen nicht mit Sicherheit beantwortet werden. Das ganze Team hat nach bestem Wissen und Gewissen so viel wie möglich noch einmal durchgetestet, aber alles zu testen war auch dieses Mal aus zeitlichen Gründen nicht möglich.

Da viele Kunden schon dringend auf das Release warten, wird die neue Version trotz Bauchweh aller Beteiligten freigegeben. Glücklicherweise sind diesmal keine katastrophalen Fehler mit ausgeliefert worden, dennoch treffen täglich neue Fehlermeldungen vom Kunden ein.

So, oder so ähnlich laufen leider auch heute noch viele Softwareprojekte. Die Ursachen dafür sind schnell gefunden:

- Die Zeit reicht nicht aus, um vor jedem Release alles manuell durchzutesten.
- Manuelle Tests, meist über die GUI ausgeführt, können nicht alle Fehler finden.

Unit-Tests können helfen, beide Ursachen zu bekämpfen. Sie ermöglichen es, große Teile der Software bereits während der Entwicklung zu testen. Indem sie automatisiert und sehr schnell laufen, kann jeder Entwickler sofort prüfen, ob die geänderte Komponente nach der gerade durchgeführten Anpassung am Quellcode noch ordnungsgemäß funktioniert.

So können viele Fehler bereits sehr früh entdeckt und behoben werden. Dadurch wiederum sinkt der Aufwand in der abschließenden Releasetestphase und die Qualität der ausgelieferten Software steigt.

Was ist ein Unit-Test?

Ein Unit-Test testet die kleinstmöglichen Bausteine einer Software, also meist einzelne Methoden und Klassen. Jede Methode wird für sich getestet. Sämtliche Schnittstellen nach außen, also zu anderen Methoden, zur Datenbank etc. werden durch Test Doubles[1] (Dummies, Fakes, Stubs, Spys, Mocks) ersetzt.

In der Testpyramide bilden die Unit-Tests die breite und stabile Basis (siehe Abbildung 1).

1 http://xunitpatterns.com/Mocks,%20Fakes,%20Stubs%20and%20Dummies.html

Viele unserer Kunden, die Software-Produkte entwickeln, verwenden Unit-Tests als maßgebliche Testtechnik. Wir haben jedoch immer wieder beobachtet, dass die verantwortlichen Personen sich oft zu sehr darauf verlassen und dadurch andere Testarten (Integrationstests, unabhängige Systemtests, prozessbezogene Tests, Usability-Tests) vernachlässigen.

Leider wird auch häufig zu gutgläubig den „grünen Lämpchen“ vertraut, die vom Build-Server nach Ausführung der Unit-Tests angezeigt werden. Als Qualitätsverantwortlicher sollte man sich auf jeden Fall folgende kritischen Fragen stellen:

- Kann man ausschließen, dass nur ein einziger Standard-Testfall erstellt wurde, um ein „grünes“ Testergebnis zu erhalten?
- Reichen die erstellten Unit-Tests aus, um die gewünschte Qualität sicherzustellen?
- Welche Arten von Tests benötigt man bei unterschiedlichen Risikoklassen?
- Sind Unit-Tests-Entwickler und Entwickler des zu testenden Codes dieselbe Person oder werden die Tests von einer unabhängigen Person erstellt?
- Was sagt eine 100%-ige Code-Coverage der Unit-Tests aus?

Wichtig ist, dass der Qualitätsverantwortliche sich nicht nur auf automatisch generierte Metriken oder Ampel-Darstellungen am Dashboard verlässt, sondern z.B. durch Test- und Code-Reviews auch sicherstellt, dass ein passendes Qualitätsniveau bei den Unit-Tests erreicht und auf Dauer eingehalten wird.

Dipl.-Ing. Johannes Bergsmann
Staatl. befugter und beeideter Ingenieurkonsulent für Informatik

Der Quality-Knowledgeletter ist ein periodisches Informationsmedium von Software Quality Lab und dessen Partnern mit den Schwerpunkten Projekt-, Prozess- und IT-Qualitäts-Management. Inhalt: fachliche Beiträge und Schwerpunktthemen, Vorstellung neuer Produkte und Leistungen, neue wissenschaftliche Erkenntnisse und andere Fachbeiträge aus unseren Themenbereichen. Aktuelle Fach- und Forschungsbeiträge sind willkommen.

Einsendungen: **info@software-quality-lab.at**

Weitere Infos zu diesem und anderen Themen:
www.software-quality-lab.com

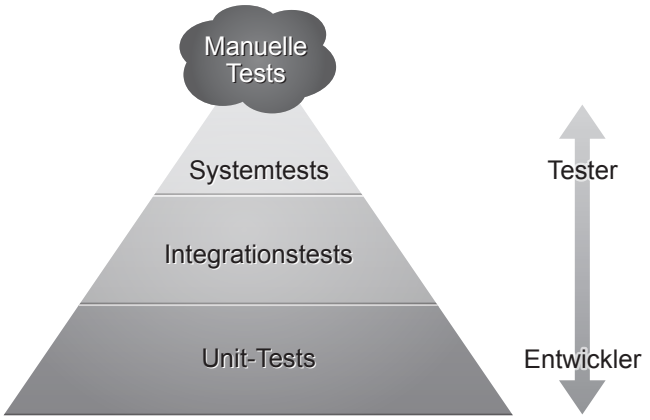

Abb. 1: Unit-Tests bilden die Basis der Testpyramide

Unit-Tests sind immer automatisierte Tests. Sie werden während oder sogar vor der Programmierung des Quellcodes erstellt.

Unit-Tests werden in derselben Programmiersprache und Umgebung wie die Software selbst implementiert. Meist sind Entwickler für die Implementierung von Unit-Tests verantwortlich. Sie sind es auch, die die Unit-Tests immer wieder ausführen und dadurch häufig Feedback zum gerade implementierten Code erhalten.

Was ist kein Unit-Test?

Wichtig ist, beim Schreiben von Unit-Tests immer die Zielsetzung vor Augen zu haben, nämlich eine einzelne Komponente (Klasse, Methode) – losgelöst und unabhängig von allen anderen Komponenten – testen zu können. Hier einige Beispiele für Tests, bei denen es sich nicht um Unit-Tests handelt:

- Der Test greift auf eine Datenbank zu.
- Der Test greift über das Netzwerk auf einen Webservice zu.
- Der Test analysiert Log-Daten.

All dies sind Beispiele für Integrationstests, welche das Zusammenspiel von Komponenten innerhalb eines Softwaresystems prüfen.

Eine klare Trennung von Unit-Tests und Integrationstests ist deshalb wichtig, da die Ausführung der Integrationstests lange – in der Regel Minuten – dauern kann (Zugriff auf eine Datenbank, HW-Interaktion). Dies ist zu lange, um während der Entwicklung oft ausgeführt zu werden und sofortiges Feedback zum Code zu erhalten. Ein Unit-Test soll nur einen Bruchteil einer Sekunde für die Durchführung benötigen.

Welche Rolle spielen Unit-Tests in agilen Methoden?

Unit-Tests bilden in vielen agilen Methoden (z. B. SCRUM, eXtreme Programming) die Basis der Qualitätssicherung. In Scrum beispielsweise gehört das Vorhandensein von ausreichenden Unit-Tests in jede gute Definition-of-Done. In XP (eXtreme Programming) ist Test-driven Development, also das Schreiben von Unit-Tests vor dem eigentlichen Applikationscode, eine der Kernpraktiken.

Wann schreibt man Unit-Tests?

Es gibt drei Möglichkeiten:

1. **Während der Entwicklung der zu testenden Komponente:**
   Ist eine Komponente implementiert und durch Debugging und erste Tests stabilisiert, werden die Unit-Tests dafür entwickelt. Dies hat den Vorteil, dass der Aufwand für mehrmaliges Refactoring der Tests aufgrund von Änderungen am Quellcode gering ist. Nachteil ist, dass die Tests spät geschrieben werden und dadurch später Feedback liefern.

2. **Vor der Implementierung der zu testenden Komponente („Test-driven Development - TDD“):**
   In der testgetriebenen Entwicklung werden zuerst sämtliche Unit-Tests implementiert. Diese schlagen natürlich alle fehl, da noch kein Code existiert, der sie erfolgreich erfüllt. Danach wird in kleinen Schritten solange Code geschrieben, bis alle Tests erfolgreich durchlaufen. Die Vorteile dieser Methode liegen darin, dass zum einen der entwickelte Code von Anfang an vollständig getestet ist und dass unnötiger Code („Gold Plating“) verhindert wird. Vielfach fehlt Entwicklern die nötige Konsequenz, um durchgängig testgetrieben zu entwickeln.

3. **In der Testphase nach Abschluss der Implementierungsphase:**
   Die Unit-Tests werden erst implementiert, wenn sämtliche Implementierungsarbeiten abgeschlossen sind. Feedback während der Entwicklung ist dadurch nicht möglich. Ein weiterer Nachteil bei diesem Ansatz ist, dass die Testphase oft aus Zeitdruck gekürzt wird und somit auch die Unit-Tests nicht mehr ausreichend erstellt werden.

In jedem Projekt muss festgelegt werden, was der beste Zeitpunkt für die Entwicklung der Unit-Tests ist. Manche agilen Vorgehensmodelle schreiben den Zeitpunkt vor (z.B. XP mit dem TDD-Ansatz), manche überlassen es dem Team (z.B. SCRUM).

Vollständiger Knowledge Letter Zugang

Wir freuen uns, dass Sie an diesem Thema Interesse haben und den Knowledge Letter von Software Quality Lab bis hierher gelesen haben.

Dieser Knowledge Letter ist eine Vorschau (gekürzte Version des gesamten Artikels).

Wenn Sie den ungekürzten Knowledge Letter lesen möchten, registrieren Sie sich bitte unter
http://www.software-quality-lab.com/download/knowledge-letter/anfrage-knowledge-letter/

Sie erhalten nach der Registrierung vollen Zugang zu allen bisherigen Knowledge Lettern von Software Quality Lab und erhalten automatisch künftige Knowledge Letter per E-Mail.