Testen von Softwaresystemen **SS 2000**

C. Steindl **Feedback 1**

Nehmen Sie sich etwas Zeit und beantworten Sie die folgenden Fragen:

1. Wieviel Zeit verbringe ich (prozentuell) mit Testen?

40-50% (2x)
10% (2x)
40% Design, 20% Code, 40% Testen und Fehlersuche
5% testen, 35% Fehlersuche
40%
70-80%
Das Testen ist die Basis für die Qualität, ~50%
10-20% (2x)
<10%
5%
5-10%
20%
1%
zu wenig
50%
10%, 40-50% inkl. Bug-Fixing (Änderungen von Programmen, übersetzen, testen)

2. Teste ich systematisch?

Nein (5x)
Kaum (2x)
Ja
Teilweise, erst wenn alles implementiert ist.
Ja, alle neu entwickelten Komponenten und die Komponenten, die mit ihnen in Beziehung stehen.
Ja, zuerst wird ein Plan erstellt. Dann die Testfälle definiert => Ausführung => Auswertung
Teilweise: Abtesten der Randbereiche und wenige Standardfälle
wahrscheinlich nicht
Nicht immer
so gut wie gar nicht
Ja (ich glaube schon)
eigentlich weniger
Nein (vielleicht im Ansatz)
Je nach Fehler, ja; die meisten Fehler werden jedoch zufällig bzw. von anderen entdeckt, nicht jedoch wenn ich gerade im Zyklus (Änderungen an Programmen,...) bin.

3. Welche Modelle für den Software-Lebenszyklus kenne ich?

Life-Cycle-Modell (3x)
Wasserfallmodell (12x)
Spiralmodell (11x)
V-Modell (6x)
Prototyping (1x)
Anforderungsanalyse => Spezifikation => Entwurf => Implementierung => Testen => zurück zu Entwurf oder Implementierung oder Ende

Siehe Software-Engineering 1
Normaler Kreis

4. Wo siedle ich Testen im Software-Lebenszyklus an?

Sollte in jeder Phase stattfinden, spätestens an deren Ende
Theoretisch da, wo ich es praktisch selten mache ☺
überall, aber in verschiedenen Formen
Vor der Auslieferung der Software
Nach jedem entwickelten Teil (Spezifikation, Source Code,...)
Testen ist nicht die letzte Phase der Softwareentwicklung (in Realität leider anders). Es soll ein Teil jeder Phase werden.
eher am Ende
~Mitte
in jedem Zyklus, fortlaufend
Nach der Implementierung
Entwicklung, Wartung
In jedem Meilenstein (Phase)
fortlaufendes Testen während der Entwicklung des Programms (Implementierungsphase); Übernahmephase, bzw. auch vor Meilensteinen
Bei der Implementierung (danach) und nach der Abnahme
Nach Fertigstellung einer Komponente, nach Integration ins System, nach Rückmeldung vom Kunden => Coding bis Wartunsphase

5. Was ist Black-Box-Testen? Was ist White-Box-Testen?

Black-Box: Ausgaben des Systems mit Erwartungswerten vergleichen
- Man testet nur von außen die gegebene Schnittstelle
- Chiptesten
- Ich weiß nicht, wie die Software arbeitet, und teste die Funktionalität
- Man weiß nichts über Programmaufbau
- Programmiertest (Test von innen), Funktionen getestet
- Ich kenne die Implementierung nicht => nur Input/Output-Test
- Nur Eingabe-Ausgabe Test, zu testendes System nicht verändern/inspizieren
- ? (2x)
- Eingabe-Ausgabe Test
- Details nicht bekannt (Implementierung); gibt Input ein,
  - sieht nach, ob Output richtig ist
- werden Schnittstellen geteststst
- nur Funktionalität wird überprüft (Input, Output)
- Der zu testende Teil wird als Black-Box angesehen und
  - eine spezielle Eingabe übergeben => Ausgabe überprüfen
- Nur prüfen, ob das I/O-Verhalten stimmt
- Code ist nicht bekannt (nur Schnittstelle)

White-Box: Innere Abläufe des Systems testen (z.B. Pfadtesten)
- tiefe Einblicke, kann mehr Testen
- ? (2x)
- Ich führe gezielte Zustände in der Software her.
- Man kennt Programmaufbau
- Support-Test (Test von außen), ob es der Spezifikation entspricht,...
- Kenne Implementierung und kann daher bessere Testfälle entwickeln
- in System eingreifen, einzelne Komponenten testen
- Codeinspektion + speziellere Tests

einzelne Teile sichtbar
werden Codestücke getestet
Innenleben wird getestet (an verschiedenen Stellen,...)
Algorithmen Schritt für Schritt durchtesten
Codestücke testen, Komponenten testen, in die Software „hineinschauen“
Implementierung ist bekannt

Vielleicht Testen ohne/mit Einblick in den Sourcecode

6. Was sind die Unterschiede zwischen „unit testing“, „integration testing“ und „system testing“?

Unit: Test eines einzelnen Subsystems
einzelne Teile
partielles Testen der Bestandteile
Einheiten einzeln testen
Es wird jede Komponente einzeln getestet
Testen der einzelnen Prozeduren,...
einzelne Teilsysteme testen
einzelne Module werden getestet
Komponenten einzeln getestet
einzelne Codestücke (Algorithmen) getrennt testen
Unit ist ein Teil der ohne Integration in System passiert
Testen eines abgeschlossenen Systems, unabhängig vom Gesamtsystem
Software-Teilbereiche testen
Nach Schreiben (Kodieren) einer Komponente

Integration: Test mehrerer Subsysteme bzw. deren Zusammenarbeit
wie die Teile zusammenspielen
Zusammenspiel der Einheiten testen
Zusammenspiel der Klassen / Komponenten wird getestet
Spielen die einzelnen Teile richtig zusammen?
das gesamte Softwaresystem wird in der BS-Umgebung getestet
Tests beim Zusammenführen
Test in abgeschlossenem System (Versuchsaufbau)
Wie passt ein konkreter Teil in das Gesamtsystem, ein auswechselbarer Teil
Nach Integration ins System

System: Test des Gesamtsystems (5x)
alles zusammen
Das System als Ganzes testen
Das gesamte System wird getestet, vor allem Schnittstellen zwischen Packages
Funktioniert auch im Gesamtsystem (mit realen Daten)
Das Softwaresystem wird auf Kompatibilität zu anderen Applikationen getestet
Test des Gesamtsystems (im Betrieb)
bei Betrieb
das System selbst

Deckt die verschiedenen Bereiche ab, welche was ist, lernen wir hoffentlich
Schätzungsweise die Detailebene des Testens
?

7. Was erwarte ich mir von der Test-LVA?

Schein
Schein (positiv!!)

Behebung des in 2. angemerkten Schadens
Leitfaden zum Testen
Die Erleuchtung
mehr Wissen beim Testen und vor allem, dass ich mich dazu aufringen kann
Lernen, Software zu testen
Hilfestellung zu systematischem Testen, Automatisierungsmöglichkeiten zum Testen
Nützliche Ideen, um die Software-Qualität zusätzlich zu verbessern.
Fehler, die jetzt gemacht werden. Verbesserungsvorschläge (günstig und nicht zeitraubend), Automatisierung (Tools,...)
Lernen von zusätzlichen Teststrategien, wie dokumentiere ich Test (Nachvollziehbarkeit)
Hintergrundwissen gewinnen, um systematisch/zielgerichtet testen zu können
Grundlagenwissen über Testmöglichkeiten, -verfahren; Tools als Basis für weitere praktische Anwendungen => Vertiefung
Hilfen für „größere“ Projekte; (systematische) Testmethoden kennenlernen
Mehr Plan und Motivation zum Testen.
Bezug auf Java
positive Note
guten Gesamtüberblick über das Gebiet und einige konkrete (Programm-)Beispiele
Testmöglichkeiten kennenlernen (klingt jetzt vielleicht banal, aber auf dem Gebiet bin ich noch unerfahren)
Dass ich Techniken lerne, die mir helfen, meine Programme sicherer, korrekter und zuverlässiger zu machen, ohne dabei den Testaufwand zu erhöhen.
Neue Testtechniken kennenlernen / vertiefen. Wie macht man es richtig? Testprogramme kennenlernen. Terminologien.