Qualitätssicherung von Software

Prof. Dr. Holger Schlingloff

Humboldt-Universität zu Berlin
und
Fraunhofer FIRST

Hinweise

- Übungen: Teilnahme wird *dringend empfohlen*
- Exkursion: **17.12.** (nicht heute oder so)
- versprochener Nachtrag über Entscheidungstabellen folgt

Kapitel 2. Testverfahren

2.1 Testen im SW-Lebenszyklus
2.2 funktionsorientierter Test
- Modul- oder Komponententest
- Integrations- und Systemtests

2.3 strukturelle Tests, Überdeckungsmaße
2.4 Test spezieller Systemklassen
- Test objektorientierter Software
- Test graphischer Oberflächen
- Test eingebetteter Realzeitsysteme

2.5 automatische Testfallgenerierung
2.6 Testmanagement und -administration

Integrationstest

- Ziel: Systematische Erprobung des korrekten Zusammenspiels von Modulen
- „Modultest auf höherer Abstraktionsebene“
- White-Box auf Modulebene
  - Komponenten und Verbindungen sind sichtbar
- Vorbedingung: die elementaren Module sind gut getestet; man kann annehmen, dass sie weitgehend fehlerfrei sind (neu auftretende Fehler sind auf Inkompatibilitäten zwischen den Modulschnittstellen zurückzuführen)
- Methode: Platzhalter und Treiber

Platzhalter (stubs, Stümpfe)

- Ein Platzhalter (stub) ist ein Pseudo-Modul, der die Funktionalität einer noch nicht geschriebenen oder integrierten Komponente (partiell) emuliert
- Implementierungsmöglichkeiten
  - z.B. Rückgabe eines konstanten Wertes statt eines berechneten Funktionswertes
  - z.B. Anzeigen der Eingabewerte und Zurückgeben von vom Benutzer eingegebener Werte
  - z.B. Erzeugung von schnellen Prototypen oder schlecht synthetisierten Funktionen, die mit wenig Aufwand erstellt werden kann („Wegwerfsoftware“)

Treiber und Orakel

- Ein Treiber ist ein Modul welches ein zu testendes Modul (IUT, Implementation under Test) aufruft und mit Eingabedaten versorgt
- Orakelproblem: Entscheidung ob die von der IUT zurück gelieferten Werte korrekt sind
  - lösbar: automatische Testauswertung im Treiber
  - unlösbar: manuelle Bewertung der Tests
- Beispiele für lösbare bzw. unlösbare Orakelprobleme
  - Treiber zum Test einer Additionsfunktion
  - sende „i“, prüfe ob Turingmaschine „i“ terminiert
  - wird ein Programmtext korrekt übersetzt?
  - Erfolgt die Berechnung innerhalb einer Millisekunde?

Ergebnisse Integrationstest

- Was kann bei der Integration schiefgehen?
  - undokumentierte Seiteneffekte
  - Eigenschaften des Betriebssystems, Speicherlecks
  - Parallelität, Verklemmungen
  - Kommunikationsverzögerungen
- Oft wird für die Integration zusätzliche „glueware“ benötigt, die nicht im Modultest getestet wurde

Beispiel: Taschenrechner

- Funktionalität: wissenschaftlich-technischer Rechner, textuelle Eingabe, mehrere Instanzen, Rückgriff auf frühere Berechnungen

(a) Entwerfen sie eine Systemdekomposition

(b) Entwerfen Sie ein Konzept für die Integrationstests

- jetzt!

Pause!

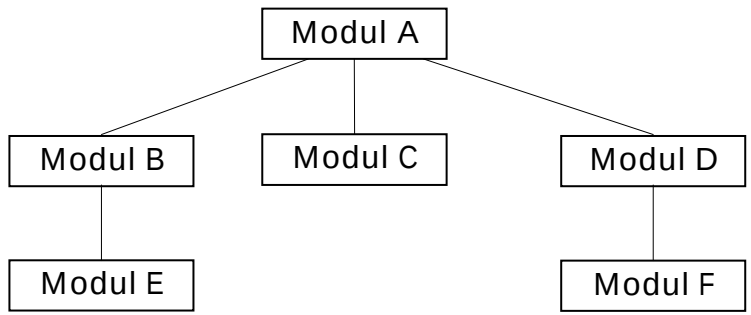
Endbenutzer-Abnahmetest

Durchführung (1)

- Möglichkeiten: Big-Bang, Top-down, Bottom-Up, Sandwich
- Big-Bang
  - Alle individuellen Komponenten werden an einem Tag zusammengesetzt und getestet
  - Klingt verwegen, ist aber manchmal nicht anders machbar (z.B. wegen Verfügbarkeit spezieller Ressourcen, organisatorische Trennung zwischen Testphasen nicht möglich o.ä.)
- Top-Down
  - Platzhalter für alle Komponenten vorbereitet
  - Übergeordnetes Modul wird mit Platzhaltern getestet
  - diese werden einer nach dem anderen durch untergeordnete Module ersetzt
    - breadth-first oder depth-first
  - während die Module integriert werden, müssen einige Tests erneut durchgeführt werden (Regressionstest)

Durchführung (2)

- Bottom-Up
  - Treiber für alle Komponenten vorbereitet
  - Basismodule werden in sogenannte „Builds“ gruppiert und integriert
  - idealerweise wertet der Treiber die Ergebnisse der Testläufe aus
  - Treiber werden nach und nach ersetzt, Funktionsumfang wächst ständig
- Sandwichmethode
  - Zusammenwachsen des Systems von oben und unten
  - Stümpfe für übergeordnete Module, Treiber für Basismodule
  - Platzhalter bzw. Treiber werden bedarfsgerecht (in Abhängigkeit der Testphase) ersetzt



Vor- und Nachteile (1)

- Big-bang
  - ☺ keinerlei Zusatzaufwand für Treiber/Platzhalter
  - ☹ unsystematische Methode, Test problematisch
  - ☹ Fehlerlokalisation evtl. schwierig
- inkrementell
  - ☺ Integration bei Fertigstellung der Komponente möglich
  - ☺ Testfälle können vergleichsweise einfach konstruiert werden, Testüberdeckung kann gewährleistet werden
  - ☹ Gegebenenfalls viele Testtreiber/Platzhalter nötig

Vor- und Nachteile (2)

- Top-down
  - ☺ frühe Verfügbarkeit des Systems aus Benutzersicht („Prototyp“)
  - ☺ Verzahnung von Entwurf und Implementierung
  - ☹ schwierige Implementierung der Platzhalter
  - ☹ mit zunehmender Integrationstiefe Schwierigkeiten bei der Konstruktion von Testfällen für tiefer liegende Komponenten
  - ☹ Zusammenspiel der Systemsoftware, Hardware und Anwendungsebene wird erst sehr spät getestet
- Bottom-up
  - ☺ keine Platzhalter nötig
  - ☺ Testbedingungen leicht herstellbar
  - ☺ Testergebnisse einfach zu interpretieren
  - ☺ Fehleingaben zur Prüfung der Ausnahmebehandlung
  - ☹ lauffähiges Gesamtsystem erst sehr spät verfügbar
  - ☹ Fehler in der Produktdefinition werden spät erkannt, können zu umfangreichen Änderungen führen