Testen

Testen und Softwarequalität (1)

- Testfälle können nicht alles abdecken
- gefundene Fehler $\approx$ fixer Anteil an vorhandenen Fehlern
- 1 Fehler ausgebessert $\rightarrow$ 10 Fehler eingebaut
- Fehler in jeder Softwareentwicklungsphase
- nicht alle Fehler werden sichtbar (aber für wen?)
- seltene Fehler sind großes Sicherheitsrisiko (Eindringen)
- „Fehler“ können Absicht sein (Eindringen ermöglichen)

Testen und Softwarequalität (2)

- Testen kann Problemstellen aufdecken
- aber Qualitätssteigerung nur durch statisches Verstehen (z.B. Code Reviews), nicht Korrektur gefundener Fehler
- Testfälle zur Spezifikation verwendbar

Teststufen

- Unittest
- Integrationstest
- Systemtest
- Abnahmetest

Testmethoden

- Black-Box-Test (Testen am Ende)
- White-Box-Test (Programm und Testfälle gleichzeitig)
- Grey-Box-Test (Testfälle zuerst)

Testarten

- Funktionaler Test
- Nichtfunktionaler Test
- Schnittstellentest
- Oberflächentest
- Stresstest (z.B. Crasch- oder Lasttest)
- Sicherheitstest
- Regressionstest

Laufzeitmessungen

Einflüsse auf Laufzeiten

- Latenz (Einweglatenz, Round-Trip-Time)
- Bandbreite
- feingranularer Parallelismus
- Thread-Parallelismus
- Datenmengen, Rechnerbelastung, Betriebssystem, Compiler, Interpreter, Hardware, . . .

Häufiger Fehler:

- Hochrechnen auf andere Datenmenge

Echtzeitsysteme

- Zeitüberschreitung = Fehler
- harte versus weiche Echtzeitsysteme
- harte Echtzeitsysteme häufig sicherheitskritisch
- Laufzeitmessungen + Beweisverfahren, oft mit Redundanz
- Java dafür kaum geeignet, eher C

Nachvollziehen des Programmablaufs

Probleme beim Debuggen

- wissen nicht, worauf wir achten müssen
- Änderungen von Programmen (Debug-Anweisungen) bzw. Variablenwerten wirken sich oft anders aus als erwartet
- Verwendung des Debuggers ändert Programmverhalten
- Fehler zeigen sich nicht dort, wo sie passieren („Nadel im Heuhaufen“)
- betroffene Programmstellen nur statisch zu verstehen

Eingrenzen von Fehlern

- Orientierung
- Hypothese
- Planung
- Durchführung
- Auswertung

Fehlermeldungen

- Compiler erkennt Inkonsistenzen, aber keine Fehler
- daher können Fehlermeldungen irreführend sein
- Fehlermeldung nicht als Handlungsanweisung verstehen, sondern

  Fehlerursache finden!