Softwaretest
- også af "ikke testbar" software

DAPUG erfamøde 7. marts 2012

Thomas Vedel, *Thomas Vedel Consult*
email: thomas@veco.dk

Hvorfor softwaretest?

Software er sjældent fejlfri

- Test sikrer at softwaren virker som forventet

Men hvordan tester man?

- Manuel test / automatiseret test
- Testdrejebog (repeterbarhed)
- Funktionstest
- Modultest / Unit test
- Regressionstest
- Data drevet test
- Load test / Stress test

Testdrejebog (repeterbarhed)

"Drejebog"

- Beskriver formålet med den enkelte test
- Beskriver testen trin for trin
- Beskriver hvilket resultat der forventes
- Test skal kunne repeteres for at tjekke at programmet "gør som forventet" efter ændringer i programkoden
- En god test afspejler en typisk brug af programmet, men skal gerne komme "ud i hjørnerne" for at fange fejlsituationer

Funktionstest

Test af en given funktion

- I forhold til brugeren
- I forhold til resten af softwaren
- Funktionstest tester HVAD programmet gør, ikke HVORDAN
  - Man tester "på overfladen" og ikke de interne tilstande i programmet
- "Brugerens test" af om programmet kan udføre de opgaver som brugeren forventer

Modultest / Unit test

Test af de procedurer, funktioner og metoder som et programmodul stiller til rådighed

- Et "modul" kan være en enkelt funktion, en klasse eller et helt funktionslibrary
- Unit test tester detaljeret funktionaliteten i en lille, afgrænset del af programmet
- For at kunne lave effektive unit tests skal "testability" være tænkt ind i programdesignet, så de enkelte kodestumper er afkoblet fra hinanden uden for store indbyrdes afhængigheder.
  - Problematisk: Globale variable, databaseadgang,...

Regressionstest

Gentage tidligere udført test og sammenligne resultatet med det forventede

- Sikrer at skidtet stadig virker efter programændringer
- Regressionstest kan begynde med det samme man har fungerende kode
- Test-suiten vokser og vokser hen over tid, efterhånden som programmet vokser
- Er i praksis umulig at gennemføre uden automatisering af testopgaven

Datadrevet test

Test gennemføres mange gange med forskellige sæt af input data

- Sikrer at programmet fungerer som forventet med forskellige data i stedet for kun med ét sæt data
- Får både "afprøvet grænser" og "typiske scenarier" hvis data udvælges med omhu
- Er i praksis tidskrævende at gennemføre uden automatiseret test

Load test / stress test

Test af at tingene også fungerer med mange brugere / i spidsbelastninger

- Svartid på 2 sek. med én bruger
  - Hvad er så svartiden med 20 samtidige brugere?
    - 2 sek.?
    - Eller 40 sek.?
    - Eller...?
  - Kan systemet overhovedet håndtere 100 samtidige brugere?
  - Hvor "hurtige" må brugerne være inden systemet overbelastes? (transaktioner pr. sek. eller lign.)

Hvordan effektiviseres testen?

- Hvis først testen gennemføres til sidst
  - er fejlene svære at rette -> dyr udvikling
  - skal softwaren gerne ud på markedet hurtigst muligt -> tidspres -> interessekonflikter
  - bliver testen "lemfeldig" fordi den ikke er tænkt ind i processen
- Løsningen?
  - Tænk testen ind i udviklingsforløbet og se den som et middel til at mindske fejlene
  - Lav automatiseret test igen og igen og igen og...
  - På gamle projekter: Indfør test gradvist
  - På nye projekter: Test det der giver mening

Værktøjer til automatiseret test

Freeware: AutoIt (http://www.autoitscript.com)

- Generelt scriptsprog til automatisering af Windows GUI og generel scripting
  - Indtastninger
  - Musebevægelser
  - Manipulering af vinduer og (windows)kontroller
  - (Avanceret) notepadlignende editor
  - Scripts kan kompileres
  - Har sine begrænsninger - men det er GRATIS! - og det er et fremragende scriptsprog

Værktøjer til automatiseret test (2)

Billigt: FinalBuilder (http://www.finalbuilder.com)

- MEGET mere avanceret end AutoIt

  (og meget mere effektivt og lettere at bruge)

  - Kender "alt" hvad der findes af værktøjer og udviklingsmiljøer (Delphi, Subversion, SQL Server...)
  - Visuelt miljø, både til redigering af scripts og afvikling af dem (man kan SE hvad man laver)
  - Scripts kan afvikles på stand alone server som kan tilgås via webinterface
  - Medfølger Delphi i en begrænset version

Værktøjer til automatiseret test (3)
(nej, logning er ikke test, men alligevel...)

CodeSite (http://www.raize.com/DevTools/CodeSite)

- Asynkron logning / debugging
- Medfølger Delphi XE og XE2 i light version
  - Købeversion virker både med Delphi og .NET
  - Logning indbygges i programmet
  - Fuld kontrol med hvad der logges
  - Ekstremt nemt at bruge
  - Belaster det kørende program meget lidt
  - Mere pålidelig end den indbyggede debugger (og KAN være meget mere effektiv i fejlfindingen)

Værktøjer til automatiseret test (4)

SmartBear TestComplete (http://smartbear.com/products/qa-tools/automated-testing-tools)

Dyrt - og godt! - købegrej

Understøtter alle former for automatiseret test:

- Funktionstest
- Modultest / Unit test
- Regressionstest
- Data drevet test
- Load test / Stress test

Giver adgang til interne variable / objekter i eksisterende programmer man ikke tør pille i...

Sammenfatning
(- eller hvordan man overlever i praksis...)

- Brug versionsstyring!!!
- Indfør unit test i nye programmer
  - Og så vidt muligt når du laver ændringer i gamle...
- Brug logning til at finde mystiske fejl
  - CodeSite både under udvikling og drift
  - EurekaLog til at rapportere "unhandled exceptions"
- Automatiser build og test
  - Brug FinalBuilder til automatisering
  - Brug FinalBuilder eller noget mere avanceret (f. eks. TestComplete) til test
  - Og benyt dig så af, at det er automatiseret (kør igen og igen og igen og igen og...)