Objektorienterede metoder

Gang 9: Afprøvning
Måder at teste på
Automatiseret afprøvning med JUnit

projektopgaven i OOM

- 5 minutters præsentation af din idé
  - Marcelo Bombaci - holdbarhedsdokumentationen
  - Peter - DIMS (softwaredistributionssystem)
  - Ilsemarie
  - Brian - baseline tool (tjekke versioner af filer)
  - Jan Bjerregaard Madsen - udenlandske vareforsendelser
  - Torben
  - Henrik Klarskov
  - Morten
  - Jens
  - Lisbeth Friis - Luftforureningsvarslingssystem
  - Anders Jørgen - et PoS system
  - Nasser Mobaraki - Andedammens Regnskab
  - Klaus Elmquist - design af procedurale mønstre
  - Anders - Grænseflade mellem ERP-system SAPS og Axapta

Afprøvning (test)

- Vigtigt hvis man ønsker programmer af høj kvalitet
- Formål: At finde (logiske) fejl
  - Psykologisk modstrid
    - Andre end programmøren bør (også) stå for afprøvningen!
- Jo tidligere fejl findes, desto lettere (og dermed billigere) er det at finde dem og rette dem.
- Afprøvning bør foregå på alle stadier af udviklingsprocessen.
- Regression
  - Tidligere test bør gentages hver gang der er foretaget ændringer!

Vigtigste måder at afprøve på

- Modultest (eng.: Unit testing)
  - Hver enhed (klasse eller gruppe af samShørende klasser) afprøves uafhængigt før enhederne integreres i hele systemet.
- Integrations- og systemtest:
  - Afprøve delene og hele systemet samlet.
- Accepttest/slutbrugertest:
  - Sikrer at systemet opfører sig som forventet af brugeren.

Modultest

- Afprøve de enkelte moduler
  - Formål: Verificere at modulet er i overensstemmelse med moduldesignet
  - Modul = klasse eller gruppe af samhørende klasser
- JUnit
  - En ramme for modultest (kan dog også bruges til integrationstest)

JUnit i praksis

- Demonstration

Integrationstest

- Afprøve samspillet mellem de enkelte moduler
  - Formål: Verificere at de er i overensstemmelse med programdesignet
- Fremgangsmåder
  - Inkrementaltest
    - Et modul adderes ad gangen
      - Nedefra og op
        - Først integreres de grundlæggende moduler og deres samspil testes
      - Oppefra og ned
        - Fordel: Man har tidligt en "skal" af systemet kørende
        - Ulempe: Man må skrive 'test-stubbe' der opfører sig som moduler
  - Totaltest ('big bang'-test)
    - Alle moduler samles og hele systemet testes på én gang
      - Nemt at konstatere fejl
      - Sværere at finde grundene til fejlene
    - Kan *ikke* anbefales i større systemer

Afprøvningsstrategier

- Blackbox testing - grænsefladetest
  - Afprøvning af delens grænseflade til resten af systemet.
  - Kontrollerer at parameteroverførsler er som specificeret
- Whitebox testing - test
  - Afprøvning sker i forhold til at man ved hvordan delen er implementeret (den indre logik og kodestruktur).

Blackbox Testing

- Ækvivalensinddeling:
  - Mængden af det mulige input inddeles i ækvivalensklasser.
  - Grænseværdier bør afprøves.
  - Mængden af det mulige input bør indeholde både gyldigt og ugyldige input.
- Positiv afprøvning:
  - Testtilfælde som forventes at gå godt.
- Negativ afprøvning:
  - Testtilfælde som forventes at gå galt.

Udvidelser til JUnit

- HttpUnit
  - Til webbaserede systemer
- Cactus
  - Til EJB
  - Til webbaserede systemer
- Borland JDBCFixture
  - Tjek af databaseændringer
- Borland ComparisonFixture
  - Sammenligning af output med en fil