LUNDS UNIVERSITET

Föreläsning 2: Test I, git, Design I

Programvaruutveckling - Metodik 2018 | Markus Borg

Agenda F2

Kursformalia

- Kursmål
- Kursombud

Test Del 1

- Verifiering och validering
- Whitebox-testning
- Introduktion till JUnit

Källkodshantering

- Introduktion till git

Programvarudesign Del 1

- Objektorienterad design
- Refaktorisering

LUNDS UNIVERSITET

ETSA02 Formella kursmål

Kunskap och förståelse

- kunna definiera **grundläggande begrepp** inom utveckling av stora programvarusystem
- kunna beskriva de **vanligaste processerna** för utveckling av stora programvarusystem
- kunna förklara de viktigaste momenten i **kravhanteringsprocessen**
- kunna förklara hur **testning** går till
- kunna beskriva vad en **arkitekturdesign** är
- kunna beskriva de viktigaste stegen i **projektplanering och projektuppföljning**
- kunna beskriva hur organisationer planerar och genomför en **serie av projekt**

Färdighet och förmåga

- kunna **utveckla projektplan**, **kravspecifikation** och **testplan** för ett mindre projekt
- kunna **granska** projektplan, kravspecifikation och testplan för ett mindre projekt.
- kunna skriftligen **formulera text i projektdokumentation**

Värderingsförmåga och förhållningssätt

- förstå **komplexiteten** i uppgiften att utveckla ett programvarusystem.
- ha förståelse för **ingenjörens yrkesroll**

Kursombud

2 X C:

TBD

TBD

2 X I:

TBD

TBD

Första möte i veckan?

Kaffe utlovas!

LUNDS UNIVERSITET

Verifiering & validering

Programvaruutveckling - Metodik 2018 | Markus Borg

Verifiering & Validering

Verifiering

Bygger vi produkten rätt?
Uppfyller vi kravspecifikationen?

Validering

Bygger vi rätt produkt?
Kommer beställaren att bli nöjd?
Når beställaren sina affärsmål?

Dynamisk och statisk testning

Statisk testning (Granskning):

Hitta fel utan att exekvera programmet

Verktyg?
Ja tack!

Dynamisk testning:

Exekvera programmet
för att hitta fel

Varför testar man?

- Hitta fel för att man vill förbättra produkten
- Visa att produkten är bra
- Verifiera att man uppfyller standard
- ...
- … men oavsett vilket så kan man aldrig visa att det inte finns fel, bara att det finns fel…

Testprocessen - flera delprocesser

Testplanering - steg ett i processen

Indata

- Projektplan
- Kravspec

...

- Design
- Manual
- Prototyper

...

Testplanering

Testplan

- Testprocess
- Testobjekt
- Testmiljö
- Kravspårning
- Tidplan
- Dokumentation av resultat
- Begränsningar
- Testfall

Acceptanstestning

Det sista steget i programvarutestning.

Görs när: Innan slutrelease på marknad eller till beställare.

Av vem: Beställaren eller tilltänkta användare.

Mål: Utvärderar om systemet uppfyller förväntningarna i den riktiga miljön.

University of Strathclyde (2016)

I projekten kommer dock acceptans-testning göras efter gruppernas slutrelease

Systemtestning

Test av hela systemet. Domineras ofta av manuell testning.

Görs när: Inför releaser, milstolpar, certifiering - eventuellt under utveckling

Av vem: Testare. Eventuellt extern oberoende testorganisation.

Mål: Utvärderar om systemet uppfyller kravspecifikationen.

Air Force photo by Ron Mullan (2008)

Systemtestestfall, två exempel från automat

Test case: Buy cola

Pre-condition: Machine in start state.

Post-condition: Machine in start state. A cola can received.

1. Press the cola selection button.
2. Insert a 10kr coin.
3. Insert a 5kr coin.
4. Receive a cola can.

Flickr: hibino

Test case: Press more than one button

Pre-condition: Machine in start state.

Post-condition: Machine in start state. A water can is received.

1. Press the water selection button.
2. Press the beer selection button.
3. Insert a 5kr coin.
4. Receive a water can.

Spårbarhet mellan krav och testfall

Alla krav ska testas av minst ett testfall

Alla testfall ska testa minst ett krav

Spårbarhetsmatris

	Test case												
Req	1	2	3	4	5	6	7	8	9	10	11	12	13
1													
2													
3													
4													
5													
6													
7													
8													
9													
10													
11													
12													
13													

LUNDS UNIVERSITET

Testprotokoll

Redogör för resultatet av test

- t.ex. en tabell med testfall, testresultat datum, testare, ...

TESTPROTOKOLL

Testfallsnr.	Korrekt ja/nej	Position i testfall	Feltyp	Grad	PR nr.

Integrationtestning

Testa att integrera komponenter.

Görs när: Så ofta som möjligt, gärna automatiskt.

Av vem: Utvecklare eller testare.

Mål: Säkerställa att komponenterna fungerar ihop.

@randal_olson (2016)

Enhetstestning

Automatisk tester av systemets minsta ingående delar. Oftast enskilda klasser.

Görs när: Kontinuerligt. Helst varje gång något ändras i källkoden.

Av vem: Testfall skapas av utvecklare och exekveras automatiskt.

Mål: Testa all produktionskod.

Testtekniker: Black-box vs. White-box

Black-box

- Programmet ses som en ”svart låda”. Kunskap om koden används ej när testfall skapas.
- Kravspecifikationen används för att ta fram testfall
- Jämför utfall med förväntat resultat

White-box

- Kräver tillgång till koden
- Testar både utfall och programmets kontrollflöde

White-box testning

- Tanken är att testfallen ska täcka all kod
  - Men vad menas med ”all kod”?
- Betrakta kodens kontrollflödesgraf

Täcka rader (statement coverage)

- Exekvera alla rader minst en gång
  - Alla noder i kontrollflödesgrafen

Täcka grenar (branch coverage)

- Exekvera alla grenar minst en gång
  - Alla bågar i kontrollflödesgrafen
  - Innefattar även komplett radtäckning

Täcka vägar (path coverage)

- Exekvera samtliga vägar från startnod till slutnod
  - Innefattar även komplett grentäckning
  - Antalet testfall exploderar med loopar!

4 vägar

Hur många vägar?

Täcka enkla vägar (simple path coverage)

- Exekvera samtliga linjärt oberoende vägar från startnod till slutnod
  - Innefattar komplett grentäckning
  - Hanterar problematiken med loopar
  - Ofta en rimlig kompromiss

Vägar:
(1,1)
(0,0)
(1,0)
(0,1)

Ej linjärt oberoende!

McCabe's Cyclomatic Complexity (CC)

- Används för att beräkna antalet enkla vägar i en kontrollflödesgraf

- Antal linjärt oberoende vägar:

$$\boxed{CC = \#bågar - \#noder + 2}$$

- Krav
  - En enda komponent i grafen
  - En unik startnod samt en unik slutnod

Exempel (CC = #bågar - #noder +2)

If:
CC = 3-3+2 = 2

If-else:
CC = 4-4+2 = 2

3 linjärt oberoende vägar

2 linjärt oberoende vägar

CC = 8-7+2 = 3

Ett något större exempel

CC = #bågar - #noder + 2 =

16 - 14 + 2 = 4

CC används också som ett mått på komplexitet

Färdigtestat?

När vi gjort vad vi lovat i projekt- och testplan:

t ex visat att alla kraven på
alla abstraktionsnivåer är uppfyllda:

- ...traverserat alla grenar i koden...
- ...med alla upptänkliga kombinationer av variabler...
- ...på alla plattformar...
- ...med >maxbelastning

Sammanfattning - Test

Testning kan påvisa fel, men inte bevisa att det inte finns fel

Testprocessen består av flera delprocesser: enhetstest, integrationstest, systemtest, acceptanstest

Testmetoder kan vara antingen ”black box” eller ”white box”

Dokumentgranskning är en testform där man kritiskt läser ett dokument på ett systematiskt sätt.

LUNDS UNIVERSITET

Automatisk enhetstestning

Fundamental idé

- Utvecklarna skriver testkod som exercerar produktionskod
  - tester konstrueras genom assertions, dvs. villkor som förväntas vara sanna
  - om någon assertion inte håller => test case failed!
- Organisera testfall i sviter
- Ramverk används för enkel exekvering och översikt

Användningsområden

- Regressionstestning
- Stöd vid refaktorisering (Lab 2)
- Grunden i “test first development”

JUnit

- Open source-ramverk för enhetstestning av Java
  https://github.com/junit-team
- Integrerat i Eclipse
- Exempel

```
public class MyFirstClassTest {

    @Test
    public void myFirstMethod(){
        String str = "JUnit is working fine";
        assertEquals("JUnit is working fine", str);

    }
}
```

JUnit

- Open source-ramverk för enhetstestning av Java
  https://github.com/junit-team
- Integrerat i Eclipse
- Exempel

```
public class MyFirstClassTest {

    @Test
    public void myFirstMethod(){
        String str = "JUnit is working fine";
        assertEquals("JUnit is working fine", str);

    }
}
```

JUnit

Package Explorer | JUnit

Finished after 0.059 seconds

Runs: 6/6 Errors: 0 Failures: 1

etsa02_lab2.test.MovementSystemTest [Runner: JUnit 4] (0.011 s)

- testTwoEnemies (0.001 s)
- testOneEnemyAtBearing180 (0.000 s)
- testOneEnemyAtBearingNegativeGreaterThan90 (0.005 s)
- testOneEnemyAtBearing0 (0.000 s)
- testOneEnemyAtBearingNegativeLessThan90 (0.001 s)
- testOneEnemyAtBearingPositiveLessThan90 (0.004 s)

Exempel på JUnit assertions

```
void assertEquals(boolean expected, boolean actual)

void assertTrue(boolean condition)

void assertNotNull(Object object)

void assertSame(object1, object2)

void assertArrayEquals(expectedArray, resultArray)
```

Mockobjekt

- I praktiken fungerar vissa komponenter inte självstående
- Mockobjekt kan användas för att förenkla komponentens interaktion med övriga delar av systemet
  - Returnerar fejkad data för att möjliggöra enhetstest

Jordan Schaenzle (2012)

Getting started with Lab2 and JUnit

Videos on
https://drive.google.com/open?id=1qEvYQYHSZVP30EASvfrgXmtpCVkTH5Lj

LUNDS UNIVERSITET

git

Programvaruutveckling - Metodik 2018 | Patrik Persson

LUNDS UNIVERSITET

Programvarudesign I

Programvaruutveckling - Metodik 2018 | Markus Borg

Objektorienterad design

- Implementationsnära design
- Beskrivning av hur komponenter implementeras på klassnivå
- Klasser beskriver meningsfulla entiteter i problemdomänen
  - Substantiv i beskrivningen blir klasser
  - Operationer implementeras i metoder

LUNDS UNIVERSITET

Objektorientering - Fundamentala koncept

1. Inkapsling

2. Abstraktion

3. Arv

4. Polymorfism

by Sinipull for codecall.net

Unified Modeling Language - UML

- Generellt språk för att modellera programvarusystem
  - Standardiserat av ISO
  - Tre skapare, varav en svensk: Ivar Jacobson
- Statisk modell av systemet
  - Klassdiagram
- Dynamisk modell av systemet
  - Sekvensdiagram

Mer i kursen ”EDAF60 Objektorienterad modellering och design”

LUNDS UNIVERSITET

Klassdiagram

+ public
- private

relationstyp och multiplicitet

parametrar och returtyper

Lab 2 - refaktorisera enligt klassdiagram

Sekvensdiagram

(c-sharpcorner)

Designmål: Låg koppling (coupling)

- I hur stor utsträckning är klasser i programmet kopplade till varandra?
- Låg koppling underlättar underhåll och evolution

Designmål: Hög samhörighet (cohesion)

Hur väl innehållet i en klass ”hänger samman”

Exempel:

- **Logisk** – t.ex. en klass som sköter all utmatning av data
- **Temporal** – t.ex. en klass som sköter all ”uppstart” eller ”avslut”
- **Procedurbaserad** – aktiviteter som utförs efter varandra slås ihop
- **Kommunikationsbaserad** – delar som behandlar samma data slås ihop
- **Sekventiell** – kedja av aktiviteter som hänger ihop i sekvens
- **Funktionell** – innehållet står för en enstaka funktion, t ex ”sortera vektor”

LUNDS UNIVERSITET

Designmönster

Beprövade lösningar på återkommande problem

- Ursprungligen från arkitektur
- Inom programvara används objektorienterade koncept

Några exempel:

- **Singleton** garanterar ett unikt objekt av en klass
- **Iterator** traverserar en samling objekt
- **Visitor** gör en operation på samtliga objekt i en samling

Exempelmönster: Observer

- Separera ett objekts tillstånd från presentation
- Möjliggör flera olika vyer

LUNDS UNIVERSITET

Tentafrågor - exempel

Programvaruutveckling - Metodik 2018 | Markus Borg

Tentafråga VT 2012

U1 Test hela vägen (15 p)

Beskriv de olika nivåerna av dynamisk testning under ett mjukvaruutvecklingsprojekt från enhetstestning till acceptanstest. För varje nivå bör beskrivningen åtminstone svara på följande frågor: Vem? Vad? När? Hur? Hur länge? Varför? Använd åtminstone följande begrepp i rätt sammanhang:

Acceptanstest	Kravtäckningsmatris	Testprotokoll
Black-box-test	Parvis testning	Validering
Branch coverage	Regressionstest	Verifiering
Enhetstest	Stopkriterium	White-box-test
Gränsvärdestestning	Stresstest	
Integrationstest	Systemtest	

Tentafråga VT 2012

Förslag

U1 Test hela vägen (15 p)

- Förstå skillnad på **verifiering/validering** samt **statisk/dynamisk** testning

Enhetstestning
- Av utvecklarna medan källkod skrivs, whitebox på klassnivå
- Testsvit används som regressionstest

Integrationstest
- Av utvecklarna efter hand som klasser blir klara, whitebox med kodtäckning, stoppkriterium?

Systemtest
- Av testarna, testar helheten, parvis testning, gränsvärden
- blackbox baserat på kraven mot kravtäckningsmatris
- Stresstest?

Acceptanstest
- Av testarna med kundrepresentant, blackbox

LUNDS UNIVERSITET

Tentafråga VT 2012

U3 Samma sak på olika sätt (15p)

Ett *användarfall (use-case)*, ett *acceptanstestfall* och ett *scenario i en användarmanual* kan mycket väl beskriva i princip samma tänkta händelseförlopp.

a) Börja med att formulera ett eget *användarfall* på kanske 5-10 punkter för vilket system som helst som **inte** har att göra med cykelgarageprojektet eller de andra exemplen som använts i kursen. Systemet kan finnas på riktigt eller vara påhittat.

   Formulera därefter ett (1) *acceptanstestfall* och en (1) *manualltext* som relaterar till användarfallet.

   Inled med en kort beskrivning av ditt tänkta system så man förstår användarfallets sammanhang. Presentera sedan gärna användarfallet, testfallet och manualtexten i en tabell med tre kolumner så att de blir lättare att jämföra med varandra.

b) Jämför de tre olika beskrivningarna och diskutera varför de skiljer sig åt.

LUNDS UNIVERSITET

Tentafråga VT 2012

Förslag

U3 Samma sak på olika sätt (15p)

System för att administrera **badmintontävlingar**. Exempel: **registrera match**

Användningsfall

1. Skanna QR-kod på rapport
2. Mata in set-siffror
3. Mata in start-/sluttid
4. Mata in antal använda bollar
5. Klicka på ’Bekräfta’

Testfall

1. Skanna QR-kod på rapport TEST_A
2. Mata in set-siffrorna ”15-21, 21-17, 21-8”
3. Mata in starttid ”10:25” och sluttid ”11:30”
4. Mata in antal använda bollar: 12
5. Klicka på ’Bekräfta’
6. Verifiera att ”Match 22” presenteras i lista 2

Scenario i manual

1. Använd QR skannern för att registrera din match
2. Fyll i matchens resultat, uppdelat per set
3. Fyll i klockslag då matchen började resp. slutade
4. Mata in hur många fjäderbollar som användes
5. Klicka på ’Bekräfta’ för att lagra matchresultatet

LUNDS UNIVERSITET

Tentafråga VT 2012

U4 Spårbarhet (10p)

Raja and Kamran genomförde 2008 en systematisk kartläggningsstudie som en del av ett examensarbete i software engineering vid Blekinge Tekniska Högskola. En av författarnas forskningsfrågor rörde vilken definition av spårbarhet som det oftast refereras till i vetenskapliga artiklar. Studenterna fann att den definition som vanligast använts är:

> *"(Requirements) traceability refers to the ability to describe and follow the life of a requirement, in both a forward and backward direction (i.e. from its origin, through its development and specification, to its subsequent deployment and use, and through periods of on-going refinement and iteration in any of these phases)" (Gotel and Finkelstein, 1994*

Frågor:

a) I cykelgarageprojektet underhölls spårbarhet mellan systemkrav och systemtester. Ge exempel på aktiviteter i ett programvaruutvecklingsprojekt som kan underlättas av att denna typ av spårbarhet finns tillgänglig.

b) Spårbarhet kan även existera mellan andra typer av utvecklingsartefakter, t.ex. krav-källkod, källkod-testfall, källkod-manual och defektrapport-krav. Att bestämma vilken typ av spårbarhet som ska underhållas innebär ett avvägande. Redogör för nackdelar med en hög nivå av spårbarhet. Föreslå även exempel på typer av projekt där spårbarhet trots detta brukar anses nödvändigt.

Tentafråga VT 2012

Förslag

U4 Spårbarhet (10p)

Underlättas av spårbarhet:

- Buggrättning
- Ändringshantering
- Vidareutveckling
- Regressionstestning
- Generell systemförståelse
- ...

Nackdelar med spårbarhet. När nödvändigt?

- dyrt att upprätta, särskilt i legacy-system
- krävs mycket arbete att underhålla vid förändringar, svårt för agil utveckling
- lätt att peka ut fel kopplingar, t.ex. för många
- ingår i standarder för säkerhetskritisk utveckling

LUNDS UNIVERSITET

Tentafråga VT 2012

U5 Statisk testning (10 p)

Beskriv hur processen för dokumentgranskning bör gå till. Ange t. ex. inblandade roller, delaktiviteter och delresultat. Diskutera kostnader i form av arbetstid och ledtid, d.v.s. hur lång tid man kan tänkas behöva mellan de olika aktivitetera för att det ska vara realistiskt att genomföra hela granskningsprocessen.

Ge exempel på typer av fel och problem som är lättare att identifiera vid dokumentgranskning än vid dynamisk testning. Motivera dina svar.

LUNDS UNIVERSITET

Tentafråga VT 2012

Förslag

U5 Statisk testning (10 p)

- Roller vid mötet: ordförande, sekreterare, författare, granskare
- Rimligt med inbjudan 1-2 veckor innan mötet
- Individuell granskning: 5-15 sidor/h beroende på komplexitet
- Granskningsmöte: max 2 h
- Uppföljningsmöte en vecka senare

Kursombud

2 X C:

TBD

TBD

2 X I:

TBD

TBD

Första möte i veckan?

Kaffe utlovas!

Att göra inför Lab 2

- Läs labinstruktionen, inkl. läsanvisningarna
- Ladda hem källkoden som ska användas
  - finns på GitHub
- Prova gärna att köra JUnit
  - annars löser vi det i datorsalen!

LUNDS UNIVERSITET