TDP005 - Projekt: Objektorienterat system

Kursupplägg, kravspecifikation och utvecklingsmetoder

Pia Løtvedt & János Dani

Institutionen för datavetenskap

LINKÖPING UNIVERSITY

LINKÖPING UNIVERSITY

Personal

Examinator:	Filip Strömbäck
Kursledare:	Pia Løtvedt János Dani
Assistenter:	August Fundin Bengtsson Jesper Jonsson Martin Clason
Kursadministratör:	Annelie Almquist

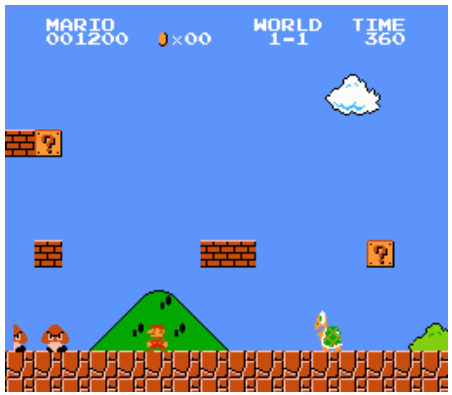
Kursinnehåll

- Introduktion till mjukvaruutveckling
- Objektorientering och UML
- Verktyg
  - IDE - CLion
  - Byggverktyg - Make och CMake
  - Dokumentation - Doxygen

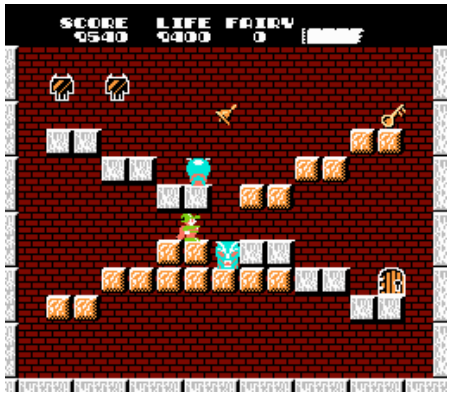
Projekt

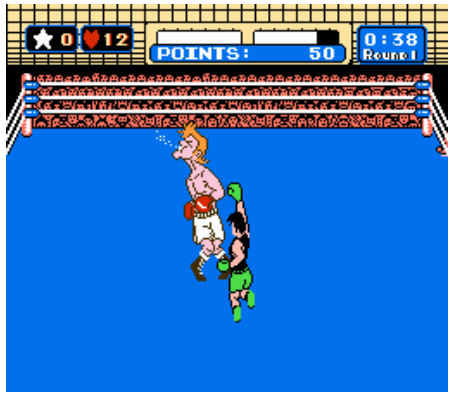
- Design och implementation av ett 2-dimensionellt spel
  - Simulering av en värld
  - Figurer med beteende över tid
  - Spelaren styr minst en av dessa
  - Interaktion mellan figurer
  - (öva på objektorientering)
- Dokumentera spelet och processen

Inspiration



Inspiration





Minimikrav

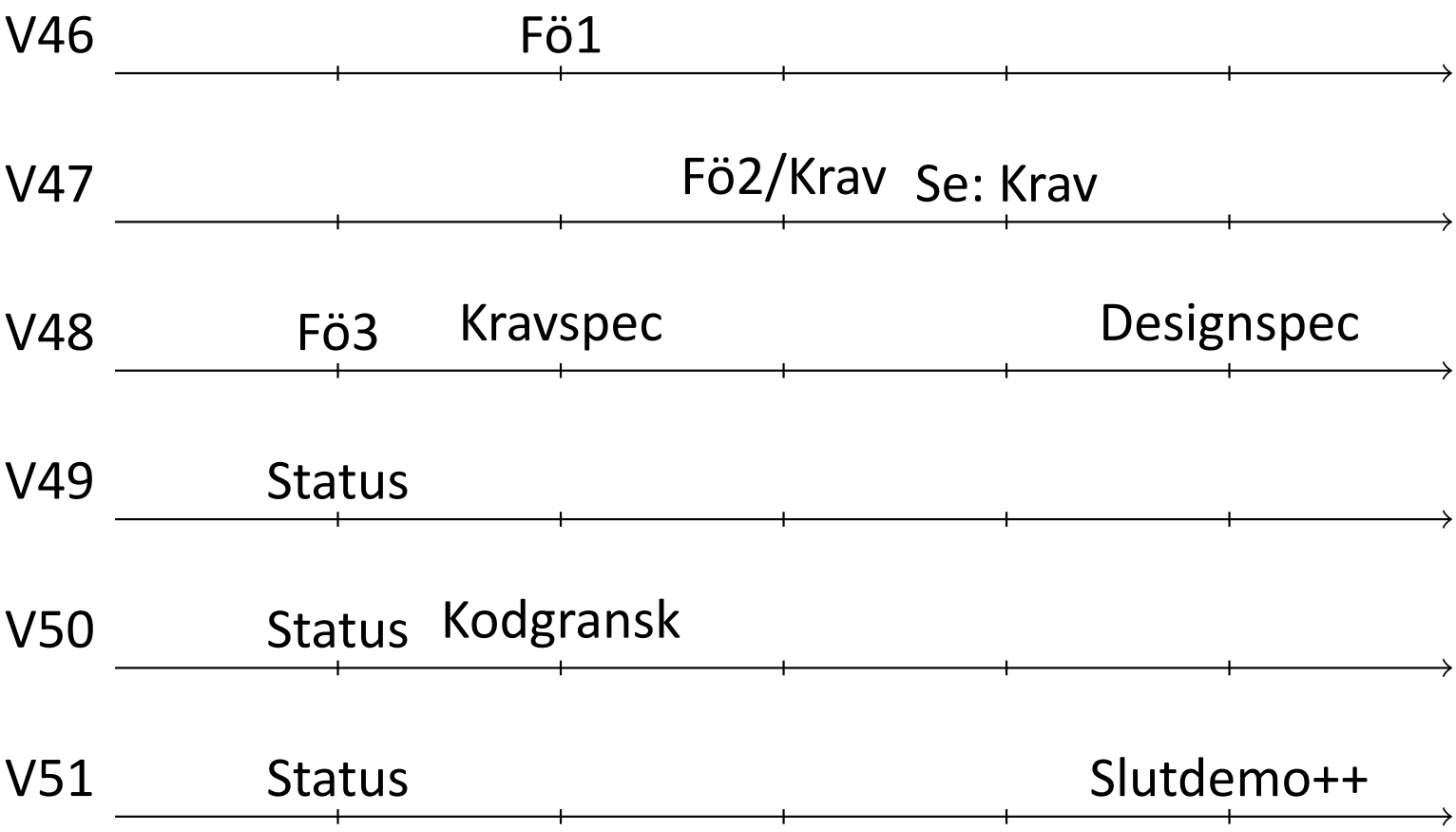
- Spelet ska simulera en 2D-värld i *realtid*.
- Minst 3 typer av objekt.
- Objekten ska röra sig över skärmen.
- Kollisionshantering ska finnas.
- Ska vara enkelt att modifiera banor i spelet.
- Spelet ska upplevas som sammanhängande.

Krav på implementationen finns mer utförligt på kurshemsidan

Upplägg

- 3 föreläsningar
  - Introduktion, kravspecifikationen, projektmetoder
  - Projektmetoder, Make/CMake och Git i projekt
  - SFML och UML
- 3 valfria labbar (använd dem om ni kör fast)
  - CLion
  - Make och CMake
  - SFML
- Projekthandledning vid behov

Tidplan


V46
Fö1
V47
Fö2/Krav
Se: Krav
V48
Fö3
Kravspec
Designspec
V49
Status
V50
Status
Kodgransk
V51
Status
Slutdemo++

Statusrapporter

- Kort avstämning, via e-post
- Vad har gjorts under veckan?
- Vad tänker ni göra under kommande vecka?
- Prioriterad *backlog*

Examination

- Dokument:
  - Kravspecifikation
  - Designspecifikation
  - Kodgranskningsprotokoll
  - Individuell reflektionsrapport
- Statusrapporter
- Individuellt portfoliotillägg
- Programredovisning

Examination

- Dokument:
- Statusrapporter
- Individuellt portfoliotillägg
- Programredovisning
  - Demosession - visa upp ert projekt
  - Kod, dokumentation och kompileringsinstruktioner

Examination

Rapporterna bedöms med G eller VG:

- Kravspecifikation
- Kodgranskningsprotokoll
- Designspecifikation
- Individuell reflektionsrapport

Projektet bedöms med 3, 4, 5:
Slutbetyg:

Examination

Rapporterna bedöms med G eller VG:
Projektet bedöms med 3, 4, 5:

- Program och kod
- Se kurshemsidan för krav

Slutbetyg:

Examination

Rapporterna bedöms med G eller VG:
Projektet bedöms med 3, 4, 5:
Slutbetyg:

- 3: Godkänt på alla moment
- 4: 4 på projektet + minst 1 VG
- 5: 5 på projektet + VG på reflektion + ett till VG

Redovisning

- Kod lämnas in via GitLab
  - Maila länk till er assistent
  - Bjud in assistent och oss
- Dokument lämnas in via e-post
  - Till er assistent
  - Dokument i PDF-format
  - Statusrapporter som text i mailet
  - Se till att ämnesraden börjar med ”TDP005:”

1 Kursinformation
2 Mjukvaruprojekt
3 Kravspecifikation
4 Metoder
5 Systemdesign och OOP
6 Testning
7 Kom ihåg

LINKÖPING UNIVERSITY

Vad är ett projekt?

- Ett *definierbart* ändamål
  - Definieras i kravspecifikationen. Funktionalitet, prestanda, etc.
- Ett *unikt* åtagande
  - Inte rutinarbete, avser något som inte gjorts identiskt tidigare.
- En *tillfälligt* aktivitet
  - Det finns en tydlig början och ett tydligt slut.

Mjukvaruprojekt - Software Engineering

Mål:

- Konstruera stora och komplexa mjukvarusystem
- Följa användares och beställares önskemål
- Hålla budget- och tidsramar
- Uppfylla kvalitets- och underhållskrav

Alltså behövs:

- Metod, Verktyg, Riktlinjer,
- med mera

Begrepp

Metodologi:

- Läran om hur metoder konstrueras, värderas och deras generella egenskaper.

Metod:
Principer:

Begrepp

Metodologi:
Metod:

- En samling principer för ett helt projekt
  - Scrum
  - Extreme Programming (XP)

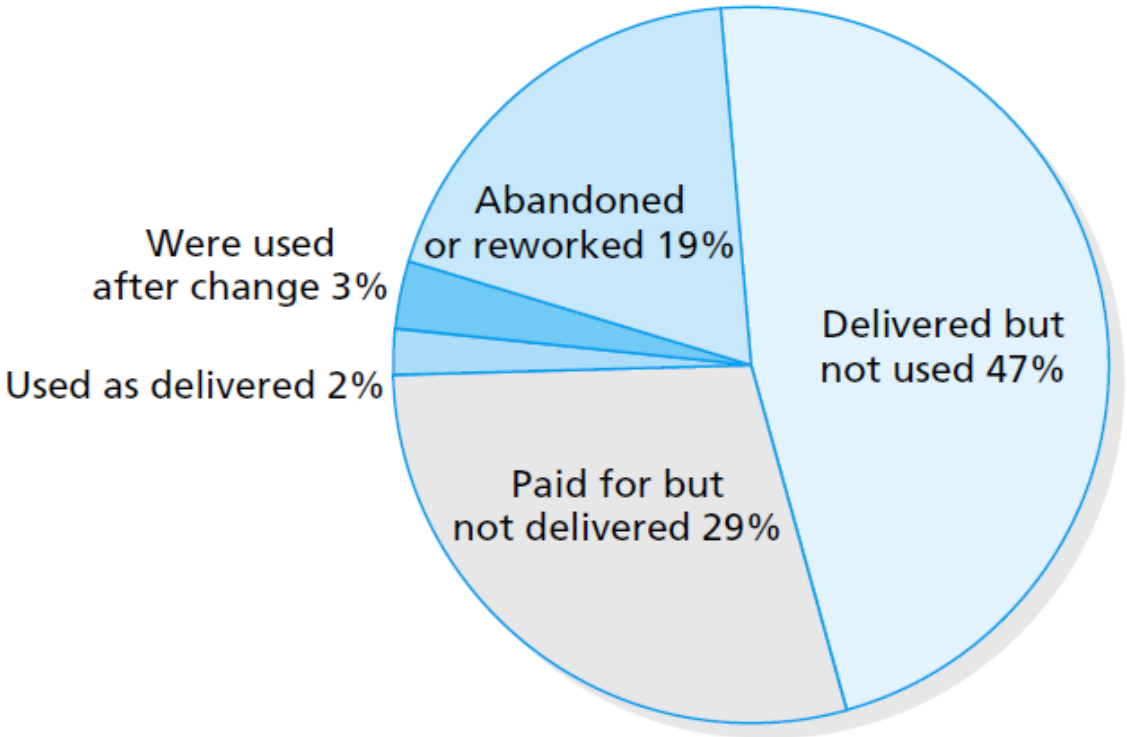
Principer:

Begrepp

Metodologi:
Metod:
Principer:

- Ett sätt att uppnå ett visst mål
  - Parprogrammering
  - Planning poker

Varför behövs detta?



Software Engineering for Students: A Programming Approach, D. Bell

Projekt

Ett projekt löper allmänt i ordningen:

	Fas	**Resultat**
1	Förstå problemet	Kravspecifikation
2	Planlägg lösningen	Projektplan
3	Genomför planen	Designspecifikation, kod
4	Utvärdera resultat	Ny kod, testdokument

1 Kursinformation
2 Mjukvaruprojekt
3 Kravspecifikation
4 Metoder
5 Systemdesign och OOP
6 Testning
7 Kom ihåg

LINKÖPING UNIVERSITY

Kravspecifikationen beskriver

- Vad ska byggas?
  - Spelidé
  - Målgrupp
  - med mera
- Hur ska det fungera?
  - Hur interagerar spelaren med spelet?
  - Hur beter sig saker på skärmen?
  - Hur saker med varandra?
- Funktionella och ickefunktionella krav
- Men inte kodstruktur eller liknande

Kravspecifikationen beskriver

- Vad ska byggas?
  - Spelidé
  - Målgrupp
  - med mera
- Hur ska det fungera?
  - Hur interagerar spelaren med spelet?
  - Hur beter sig saker på skärmen?
  - Hur saker med varandra?
- Funktionella och ickefunktionella krav
- Men inte kodstruktur eller liknande

$\Rightarrow$ betraktar produkten som en svart låda

Designnivåer i kravspecifikationen

- Vision - vad är den bärande tanken bakom systemet?
- Mål - vad är det mer konkreta målet med systemet?
- Målgrupp - vilka ska använda systemet?
- Tjänster
  - Vad ska man kunna göra med systemet?
  - Vad erbjuder systemet för tjänster?
- Användbarhetsmål - hur ska tjänsterna upplevas?

Designnivåer i kravspecifikationen

- Funktionalitet och innehåll
  - Går det att beskriva mer konkret vilken funktionalitet och vilket innehåll som ska finnas i systemet?
  - Interaktionsstruktur - Hur ser användargränssnittet ut?
  - Interaktionstekniker - Hur interagerar man?
  - Fysisk form - Är det en fysisk produkt? Mjukvara för en dator?
- Säkerhetskrav
- Eventuella hårdvaru- och prestandakrav

Krav

- Ska-krav
  - minimikrav för att produkten ska accepteras
- Bör-krav
  - implementeras i mån av tid

Krav

- Ska-krav
  - minimikrav för att produkten ska accepteras
- Bör-krav
  - implementeras i mån av tid
- Tänk på:
  - Tydliga krav - ska kunna genomföras av andra
  - Mätbara krav - tydligt när kravet är klart

En bra kravspecifikation ska producera likvärdiga resultat oavsett vem man ger den till. (Prova gärna om ni vill!)

Exempel på krav

- Spelaren ska kunna ta skada

Exempel på krav

- Spelaren ska kunna ta skada (dåligt)
- När spelarens figur kolliderar med något farligt ska spelaren ta skada

Exempel på krav

- Spelaren ska kunna ta skada (dåligt)
- När spelarens figur kolliderar med något farligt ska spelaren ta skada (bättre)
- När spelarens figur kolliderar med något farligt ska spelarens hälsa minska med 10 enheter

Exempel på krav

- Spelaren ska kunna ta skada (dåligt)
- När spelarens figur kolliderar med något farligt ska spelaren ta skada (bättre)
- När spelarens figur kolliderar med något farligt ska spelarens hälsa minska med 10 enheter (ännu bättre)
- När spelarens hälsa har nått noll ska spelet avslutas och “Game Over” visas.

Exempel på ska- och bör-krav

Ska-krav:

1. Spelaren ska kunna förflytta sin figur på skärmen med hjälp av piltangenterna.
2. Fiendefigurerna ska flytta sig i förutbestämda banor på skärmen.
3. När spelarens figur kolliderar med en fiendefigur så ska spelet avslutas.

Exempel på ska- och bör-krav

Ska-krav:

1. Spelaren ska kunna förflytta sin figur på skärmen med hjälp av piltangenterna.
2. Fiendefigurerna ska flytta sig i förutbestämda banor på skärmen.
3. När spelarens figur kolliderar med en fiendefigur så ska spelet avslutas.

Bör-krav:

4. Spelaren ska kunna välja vilka tangenter som ska användas för att styra spelet via inställningsmenyn.
5. Fiendefigurerna ska röra sig mot spelarens figur.

Kravspecifikation i kursen

- Ska beskriva ert mål med spelet
- Ska innehålla minst 6 ska-krav och 3 bör-krav
- Kraven ska tillsammans täcka hela spelidén

Vid kursens slut ska ni uppfylla alla ska-krav för att få betyg 3.

Kravspecifikationen fungerar som ett kontrakt mellan er och kursledningen om vad som ska utvecklas.

Backlog

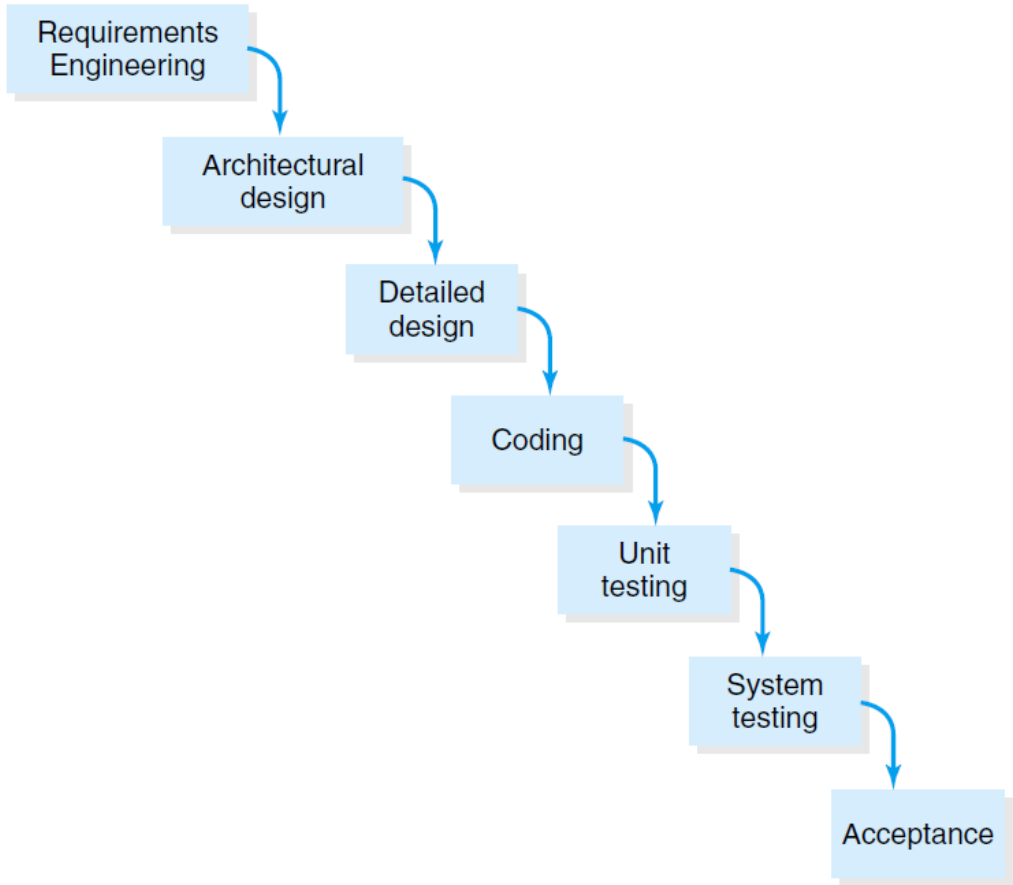
Prioriterad lista över era krav med veckomål utmarkerade.

2.	Fiender ska röra på sig
1.	Styra spelarens figur
3.	Kollision
5.	Svårare fiender
4.	Välja tangenter

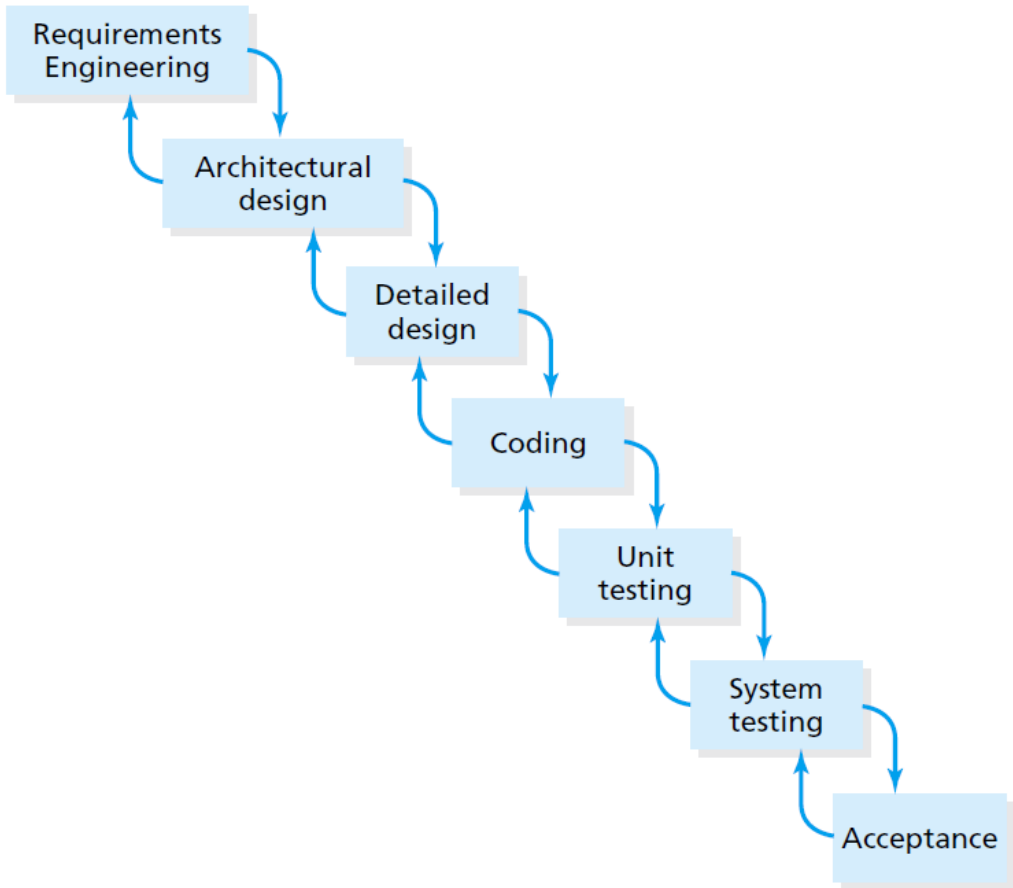
1 Kursinformation
2 Mjukvaruprojekt
3 Kravspecifikation
4 Metoder
5 Systemdesign och OOP
6 Testning
7 Kom ihåg

LINKÖPING UNIVERSITY

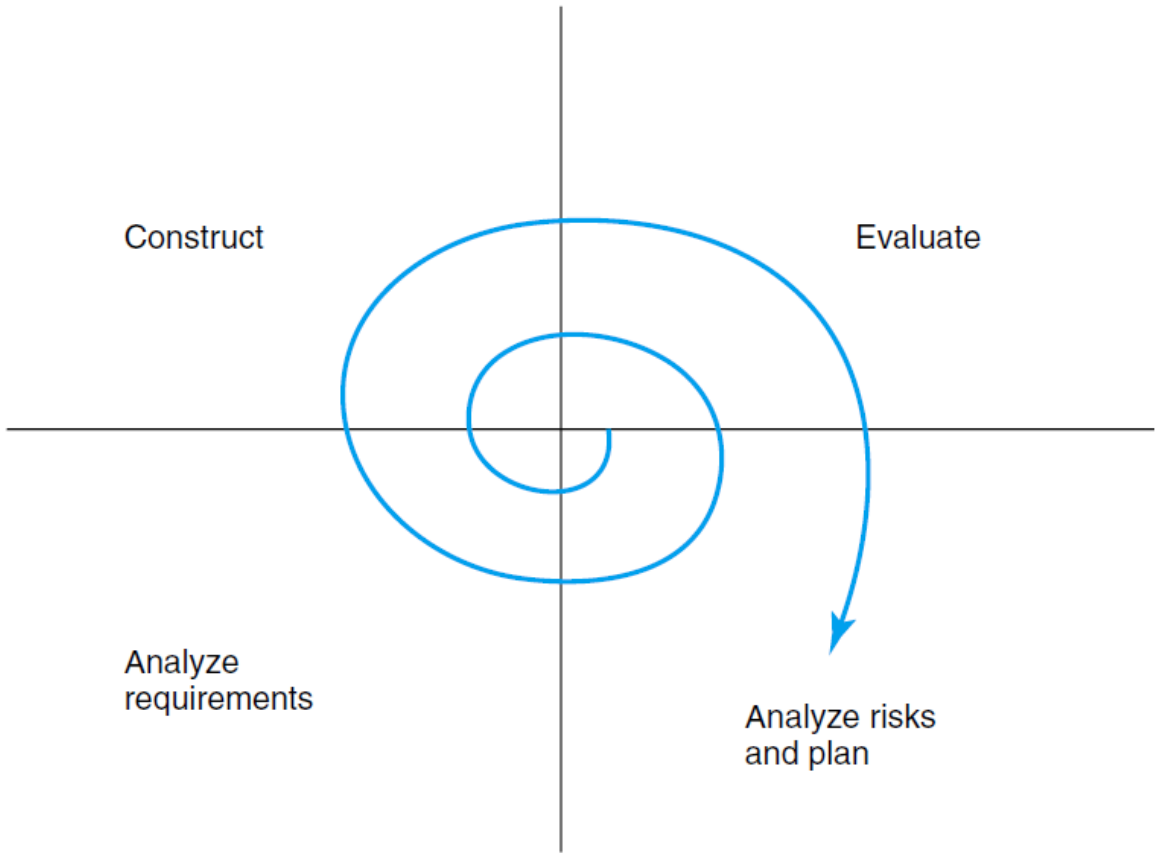
Vattenfallsmodellen



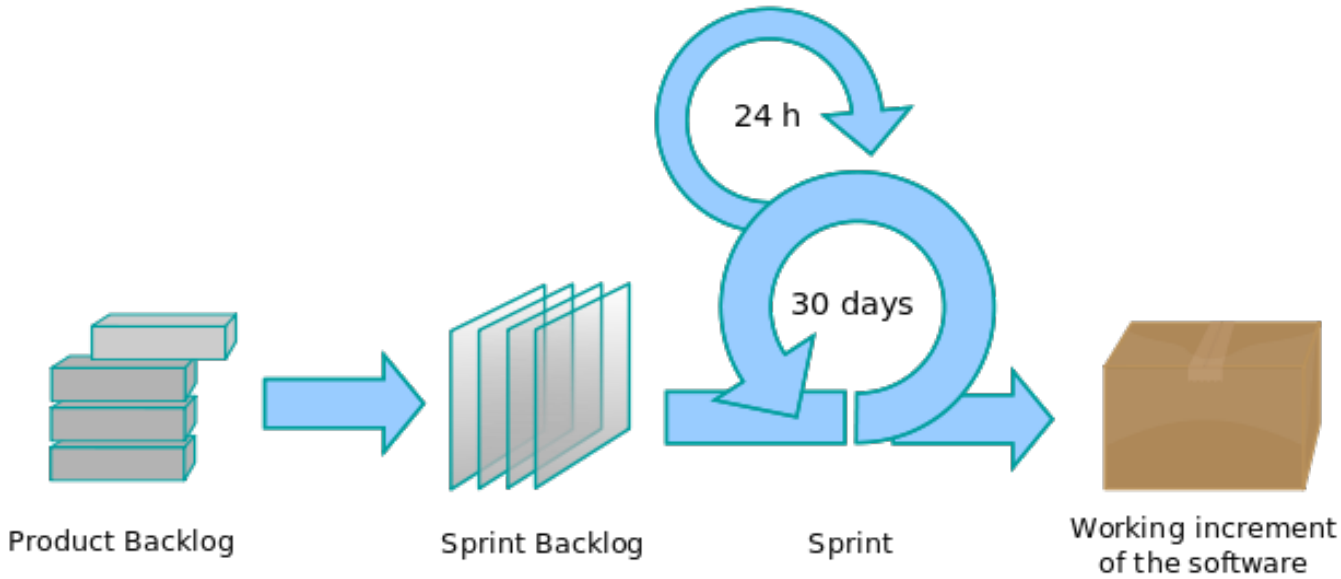
Vattenfallsmodellen med återhopp



Spiralmodellen

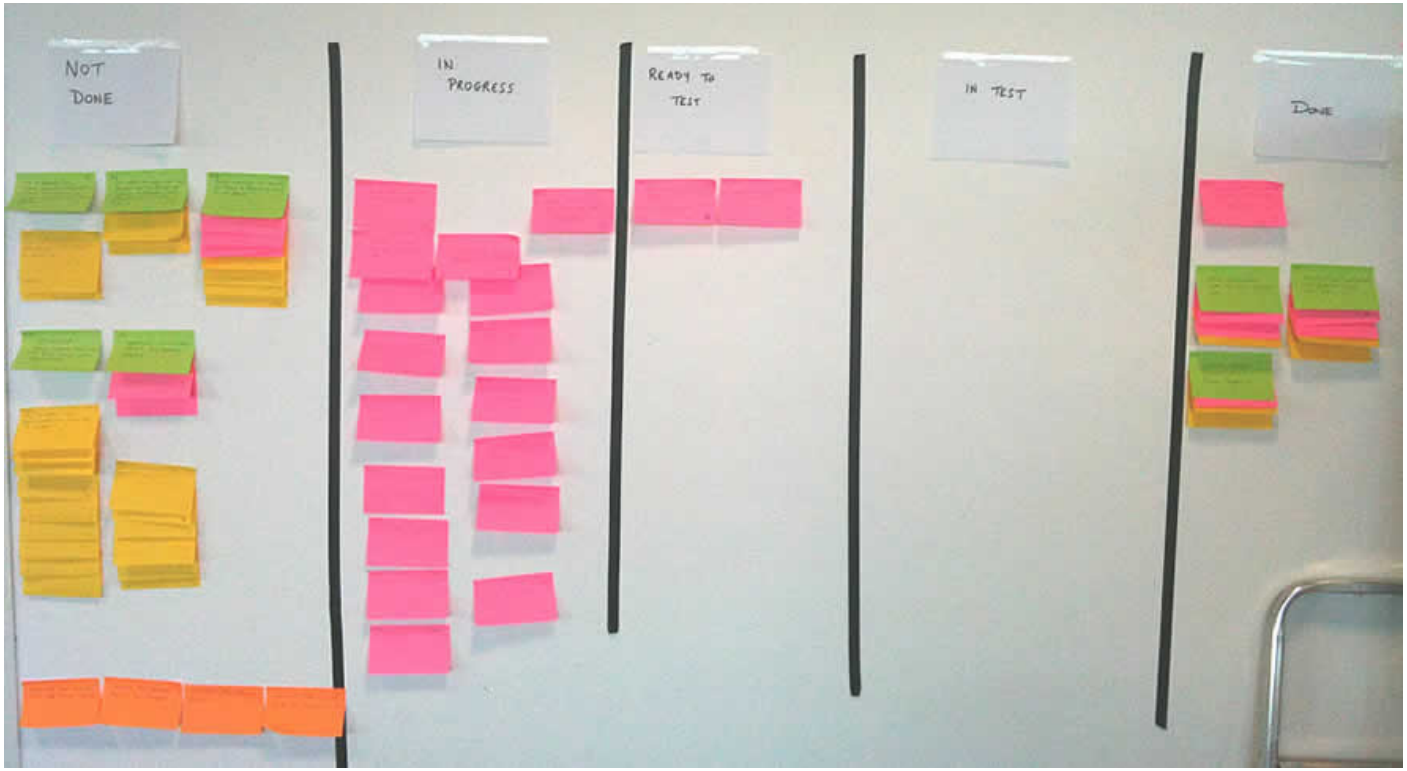


Scrum

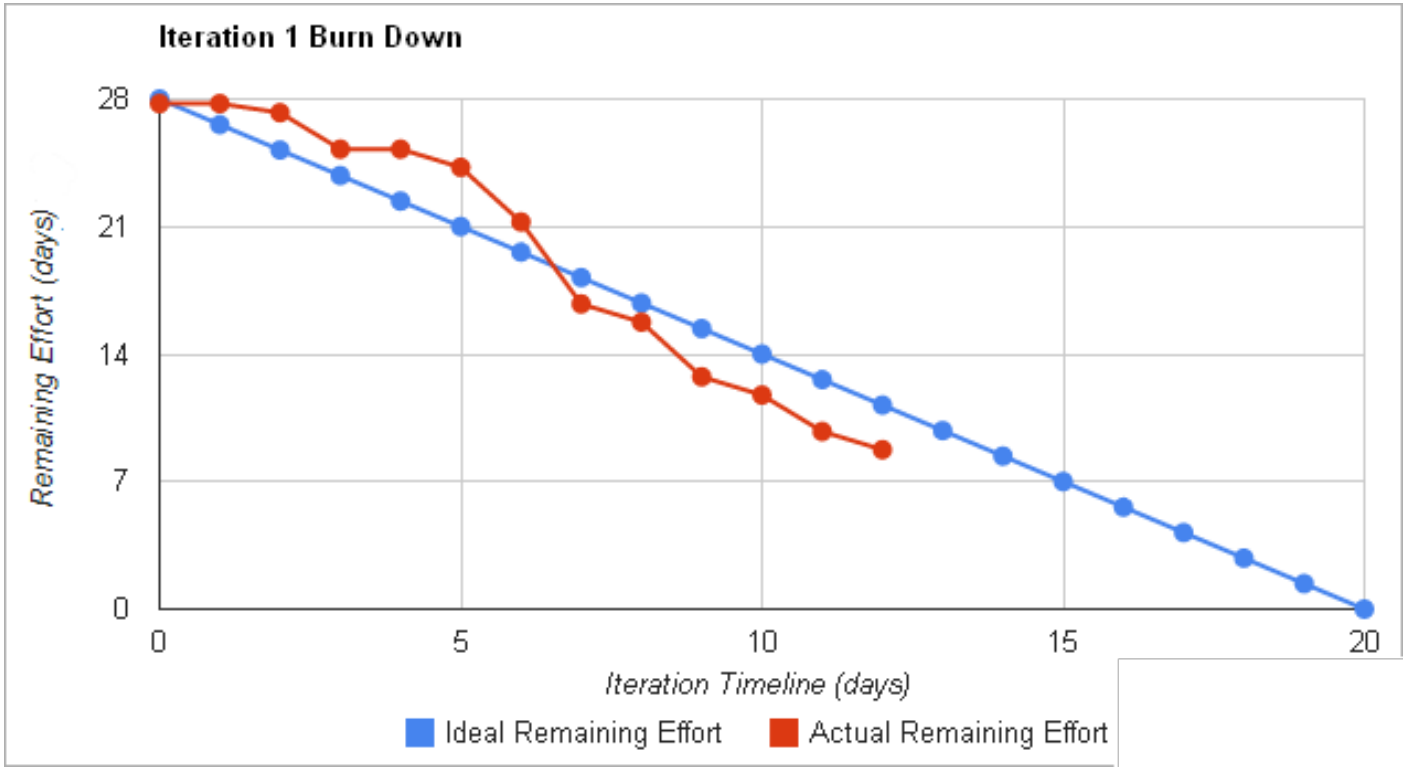


By Lakeworks - Own work, GFDL

Scrum - TODO board



Scrum - Burndown chart



Prototyping

- Bygg en enkel prototyp
- Visa den för beställaren
- Använd den feedback ni fått för att förbättre kravspecifikationen
- Fortsätt tills beställaren är nöjd!

Prototyping passar ofta bra tillsammans med agila metoder!

Metod i kursen

Mer agilt än i förra kursen:

- En ”sprint” i veckan
- Måndagsmöten motsvarar summering av förra veckan och planering för kommande
- Statusrapporten innehåller er backlog
- Större möjlighet att experimentera

1 Kursinformation
2 Mjukvaruprojekt
3 Kravspecifikation
4 Metoder
5 Systemdesign och OOP
6 Testning
7 Kom ihåg

LINKÖPING UNIVERSITY

Systemdesign

- Mål: omsätta kravspecifikationen till en lösning
- Konceptuell design:
  - Vad systemet gör
  - Skrivet i beställarens språk utan teknisk jargong
  - Kopplat till kravspecifikationen

Systemdesign

- Mål: omsätta kravspecifikationen till en lösning
- Konceptuell design:
  - Vad systemet gör
  - Skrivet i beställarens språk utan teknisk jargong
  - Kopplat till kravspecifikationen
- Teknisk design:
  - Hur systemet gör saker
  - Platform
  - Hierarki och funktion hos programkomponenter
  - Datastrukturer och dataflöde

Kort historia

- I början: Ingen struktur, gör som du vill
- Strukturerad programmering
  - Problem: skalar inte till stora system
- Objektorientering
  - Java, C++
- Multiparadigm?

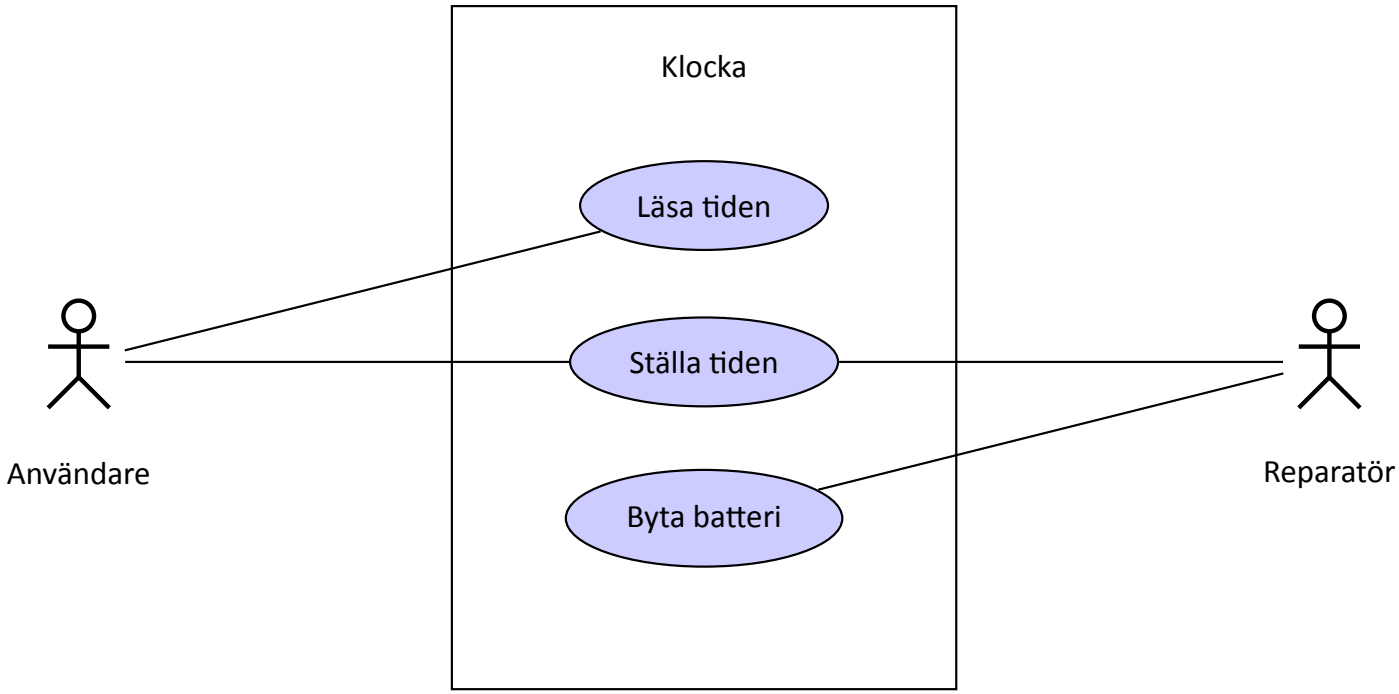
Vad är ett objekt?

- Abstraktion av världen
- Kan *utföra* saker som svar på *meddelanden*
- Har ansvar för sitt eget tillstånd
- Oberoende enheter
- Delad data undviks
- Systemfunktionalitet uttrycks som samarbete av flera olika objekt

Objektorienterad analys och design (OOA/OOD)

- Börja med användningsfall (eng. *use cases*)
- Kategorisera efter aktörer, dvs. *vem* som utför saker
- Hitta beroenden mellan olika fall
- Hitta klasser exempelvis med hjälp av CRC-kort
- Formalisera med hjälp av klassdiagram (i UML)

Användningsfall



CRC-kort

Class	
Responsibilities	*Collaborators*

CRC-kort

Class	
Responsibilities	*Collaborators*

Exempel:

Klocka	
Visa tiden	LCD-display
Ändra tiden	LCD-display, knappar
Byta batteri	Batteri

Implementation

- Standarder
- Kodningsriktlinjer - stilguide

Implementation

- Standarder
- Kodningsriktlinjer - stilguide
- Dokumentation
  - Intern dokumentation
    - Header-kommentarer
    - Kodkommentarer
    - Namngivning
    - Data-dokumentation
  - Extern dokumentation
    - Översikt av systemets komponenter
    - Ingår i systemdokumentationen
  - Använd Doxygen!

LINKÖPING UNIVERSITY

Testning - översikt

- Enhetstestning
  - Fokuserar på enskilda klasser eller moduler
- Integrationstestning
  - Fokuserar på interaktion mellan moduler
- Systemtestning
  - Testar systemet i sin helhet
- Användbarhetstestning
  - Hur enkelt är det att använda det vi byggt?
- Regressionstestning - fungerar allt fortfarande?
- Acceptanstestning - har kraven uppfyllts?

Design och testning

- Designen bör ta hänsyn till testning
  - Bra design uppmuntrar till testning av moduler snarare än hela systemet
- Enkla designer är ofta enkla att testa
  - Tydliga abstraktionsbarriärer
  - Väldefinierad modulfunktionalitet
- Objektorienterad design
  - Respektera abstraktionsbarriärer
  - Moduler behandlas som enheter med ett väldefinierat gränssnitt mot omvärlden

Automatisera testning!

- Tråkigt att testa
- Skapa testsvit som byggs ut i takt med programmet
- Snabbt och smidigt att kompilera och köra testerna
- Integrera med Git - förhindra incheckning om testerna misslyckas
- Bibliotek: Många att välja på, Catch är bra!

Testfall

- Black-box
  - Tittar bara på specifikationen
  - Försök glömma koden
  - Be någon annan att skriva testfallen
  - Skriv testfall efter specifikationen först, implementera sen

Testfall

- Black-box
  - Tittar bara på specifikationen
  - Försök glömma koden
  - Be någon annan att skriva testfallen
  - Skriv testfall efter specifikationen först, implementera sen
- White-box
  - Tar hänsyn till koden
  - Skriv testfall som knäcker koden, testa gränsfall etc.
  - Skrivs av programmerare som kan koden
  - Se till att varje kodrad exekeveras minst en gång

Ekvivalensklasser för black-boxtestning

- Utgå från förväntat resultat
- Dela upp alla testfall i grupper
- Varje grupp innehåller testfall som producerar samma typ av resultat
- Testa gränserna för varje klass
- Glöm inte testfall som producerar fel!

Exempel

`getDaysOfMonth(year, month)` ger 6 klasser:

(30 dagar, 31 dagar, februari) $\times$ (skottår, ej skottår)

```
REQUIRE(getDaysOfMonth(1, 1200) == 31);
REQUIRE(getDaysOfMonth(7, 1300) == 31);
REQUIRE(getDaysOfMonth(4, 1996) == 30);
REQUIRE(getDaysOfMonth(9, 2001) == 30);
REQUIRE(getDaysOfMonth(2, 2000) == 29);
REQUIRE(getDaysOfMonth(2, 2100) == 28);
```

Testdriven utveckling (TDD)

- Börja med att skriva test
- Skriv kod tills testerna lyckas
- Repetera tills allt är implementerat

1 Kursinformation
2 Mjukvaruprojekt
3 Kravspecifikation
4 Metoder
5 Systemdesign och OOP
6 Testning
7 Kom ihåg

LINKÖPING UNIVERSITY

Kom ihåg

- Webreg - se till att ni är anmälda (deadline 15 november)
- GitLab - skapa repository, bjud in assistent och oss (pialo23 & janda98)
- Deadline för första inlämning av kravspecifikationen den 20 november

www.liu.se

LINKÖPING UNIVERSITY