# TDP005 - Projekt: Objektorienterat system

## Kursupplägg, kravspecifikation och utvecklingsmetoder

Pontus Haglund & Simon Ahrenstedt

Institutionen för datavetenskap

LINKÖPING UNIVERSITY

LINKÖPING UNIVERSITY

# Personal

| | |
|---|---|
| Examinator: | Filip Strömbäck |
| Kursledare: | Pontus Haglund<br>Simon Ahrenstedt |
| Assistenter: | Tobias Elfstrand<br>Simon Ahrenstedt<br>Malte Nilsson |
| Kursadministratör: | Helene Pers |

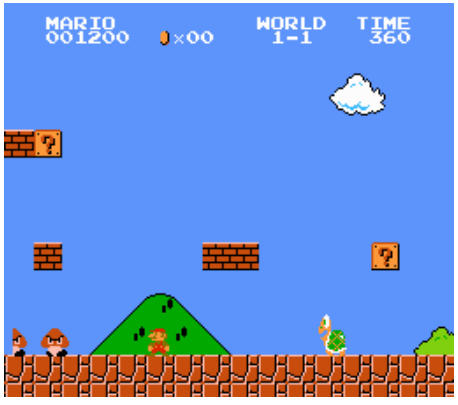
# Kursinnehåll

- 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

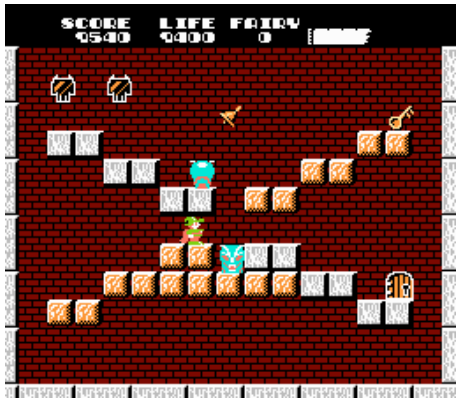


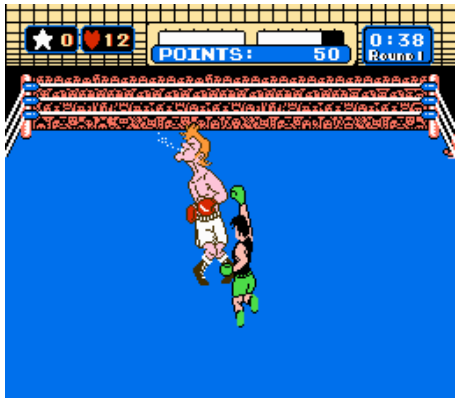




# Inspiration



# Inspiration

Om man inte är förtrogen med 8-bitars eran (NES) så kan man istället tänka på indie-titlar från Steam:

- Binding of Isaac
- Vampire Survivours
- Broforce
- FEZ
- Gauntlet
- Braid

# Minimikrav

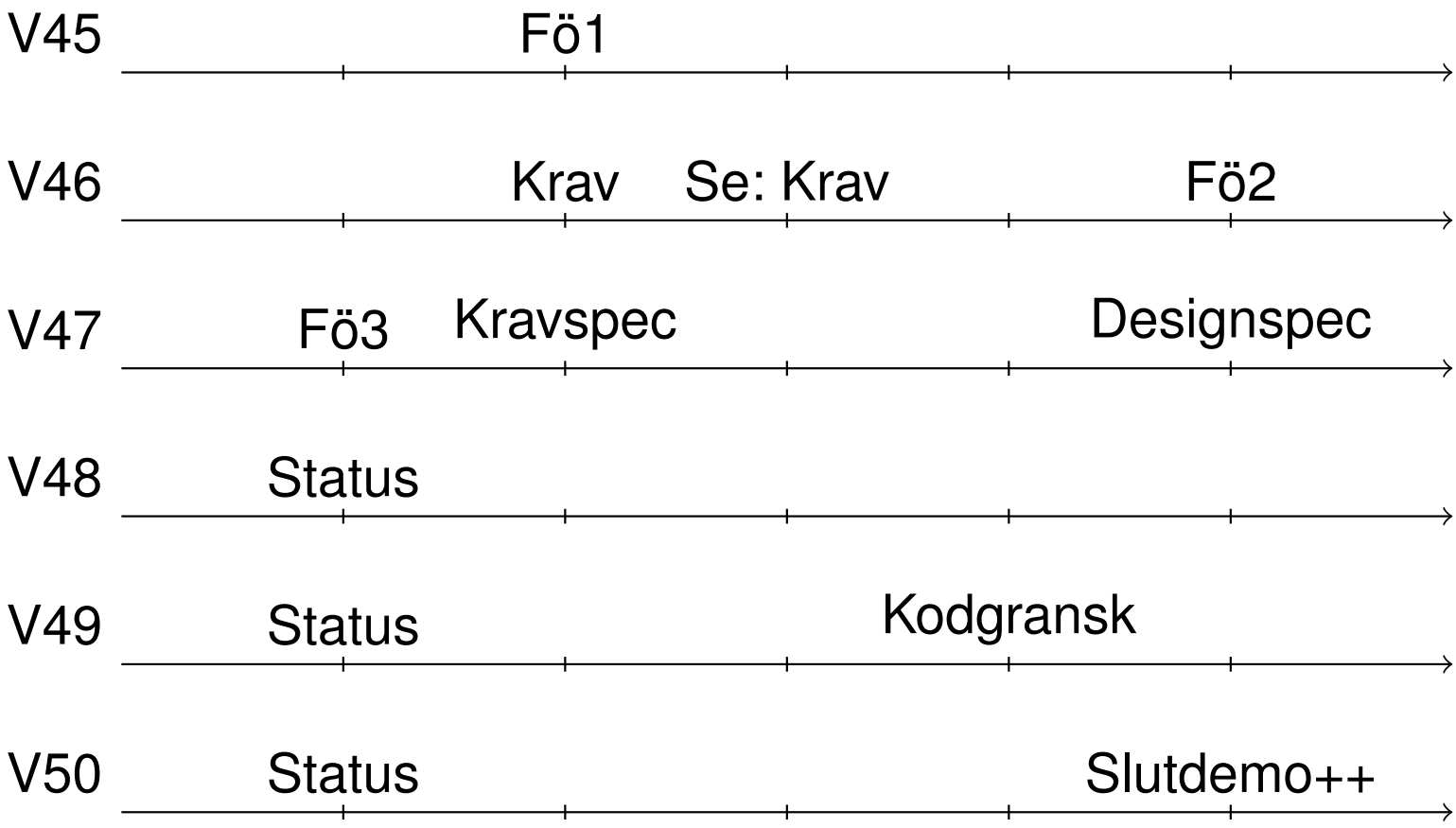
- Spelet ska simulera en 2D-värld i *realtid*.
- Minst 3 typer av objekt.
- Objekten ska röra sig över skärmen.
- Kollisionshantering ska finnas.
- Ska vara enkelt att modifiera banor i spelet.
- Spelet ska upplevas som sammanhängande.

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
  - CLion
  - Make och CMake
  - SFML
- Projekthandledning vid behov

# Tidplan


V45
Fö1
V46
Krav
Se: Krav
Fö2
V47
Fö3
Kravspec
Designspec
V48
Status
V49
Status
Kodgransk
V50
Status
Slutdemo++

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

LINKÖPING UNIVERSITY

# Examination

Alla momenten för betyg i kursen:

- Kravspecifikation
- Designspecifikation
- Statusrapporter
- Kodgranskningsprotokoll
- Individuellt portfoliotillägg
- Implementation av objektorienterat system
- Programredovisning

# Kravspecifikation

- Presenteras i detalj senare

# Designspecifikation

Full beskrivning på kurssidan (**läs den**). Specifikation för hur ni tänker implementera spelet. Var delvis given i TDP003.

- Klassdiagram över hela systemet
- Detaljbeskrivning av 2 centrala klasser
- Kort diskussion (1/2-1 sida) med motivation av designen
- ...

# Statusrapporter

3 gånger under implementationsfasen. En kommunikationsväg.

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

# Kodgranskningsprotokoll

Granska varandras kod efter instruktionen, ha ett möte, skicka in dokumentet som innehåller:

- Hur mötet gick till
- Granskningen ni gjorde av en annan grupps kod
- Granskningen en annan grupp gjorde av er kod

# Individuellt portfolioinlägg

Uppdatera er portfolio med ert senaste projekt

# Implementation av objektorienterade systemet

Implementationen av projektet (koden som levereras)

# Programredovisning

Konferens där ni presenterar ert projekt framför halva klassen.

- Kort redovisning av ditt projekt framför klassen
- Alla går runt och testar varandras spel (som i TDP003)
- Du presenterar också projektet för din assistent som samtidigt tittar igenom er kravspecifikation

# Individuell reflektionsrapport

Lämnas in i slutet av kursen

- Viss frihet att välja innehåll.
- Lite mindre omfattning (2 sidor) än TDP003 (lite svårare att skriva)
- Detta ska innehålla **reflektioner** inte bara erfarenheter

1 Kursinformation
2 Examinerande delmoment
3 Mjukvaruprojekt
4 Kravspecifikation
5 Metoder
6 Systemdesign och OOP
7 Testning
8 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

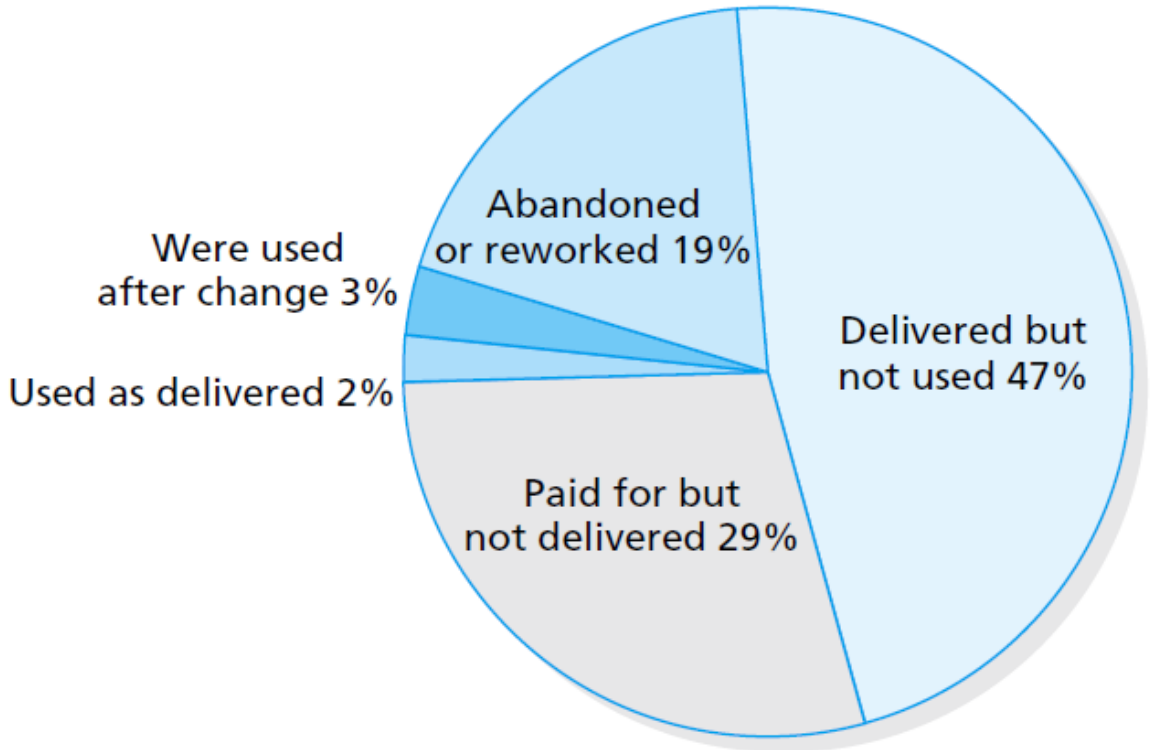
Mål:

- 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

# 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 Examinerande delmoment
3 Mjukvaruprojekt
4 Kravspecifikation
5 Metoder
6 Systemdesign och OOP
7 Testning
8 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 interagerar 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 interagerar 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?
- Vad ska man kunna göra med systemet?
- 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?

# 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.

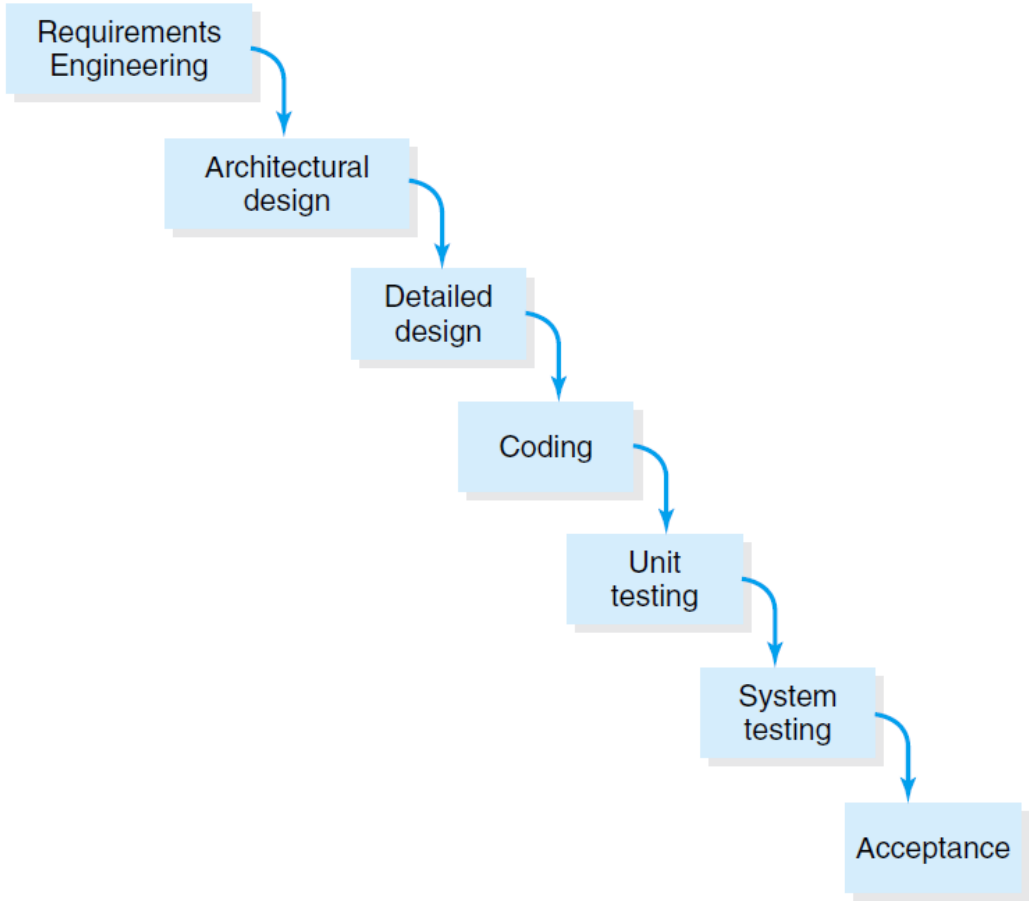
# Backlog

Prioriterad lista över era krav med veckomål utmarkerade.
Relaterar sedan till statusrapporterna.

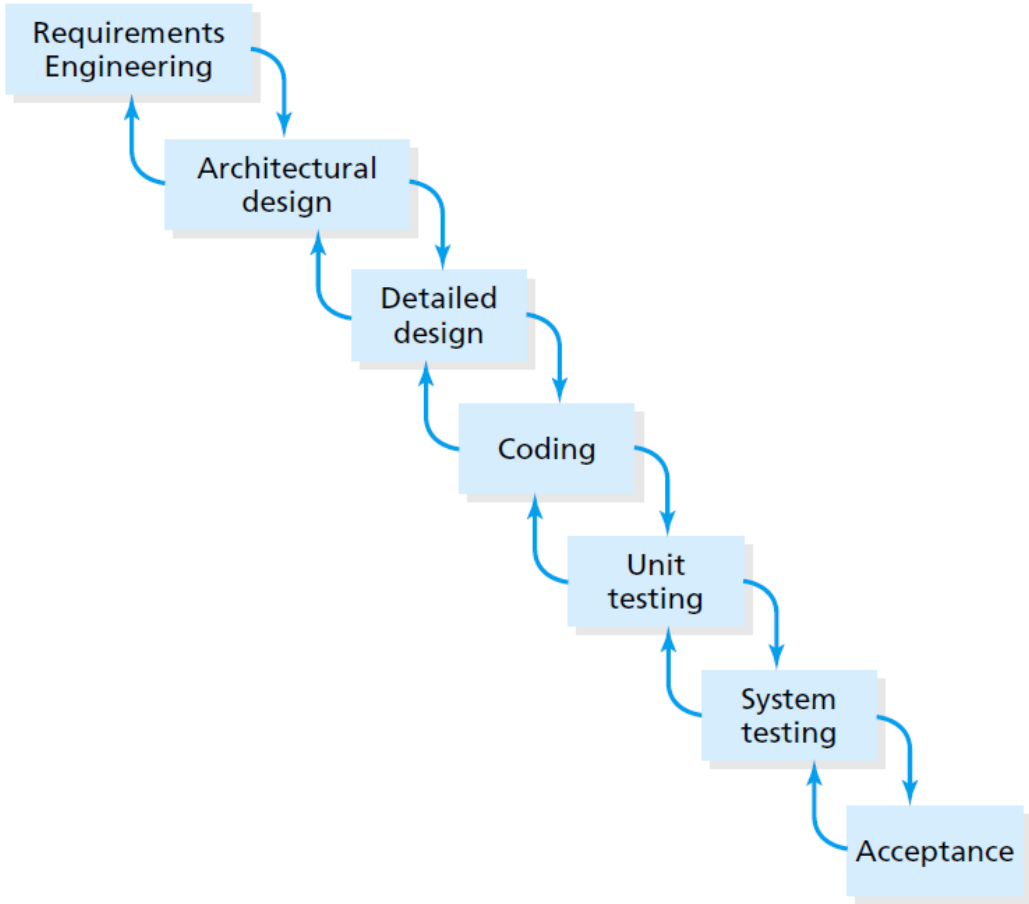
| | |
|---|---|
| 2. | Fiender ska röra på sig |
| 1. | Styra spelarens figur |
| 3. | Kollision |
| 5. | Svårare fiender |
| 4. | Välja tangenter |

LINKÖPING UNIVERSITY

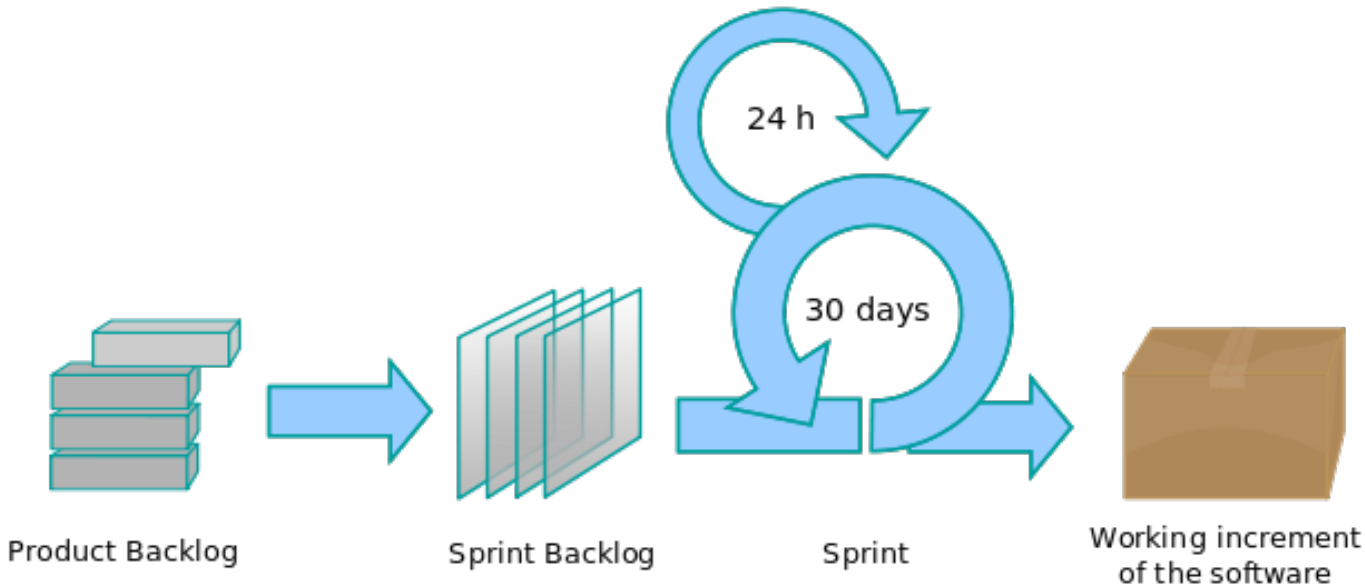
# Vattenfallsmodellen



# Vattenfallsmodellen med återhopp

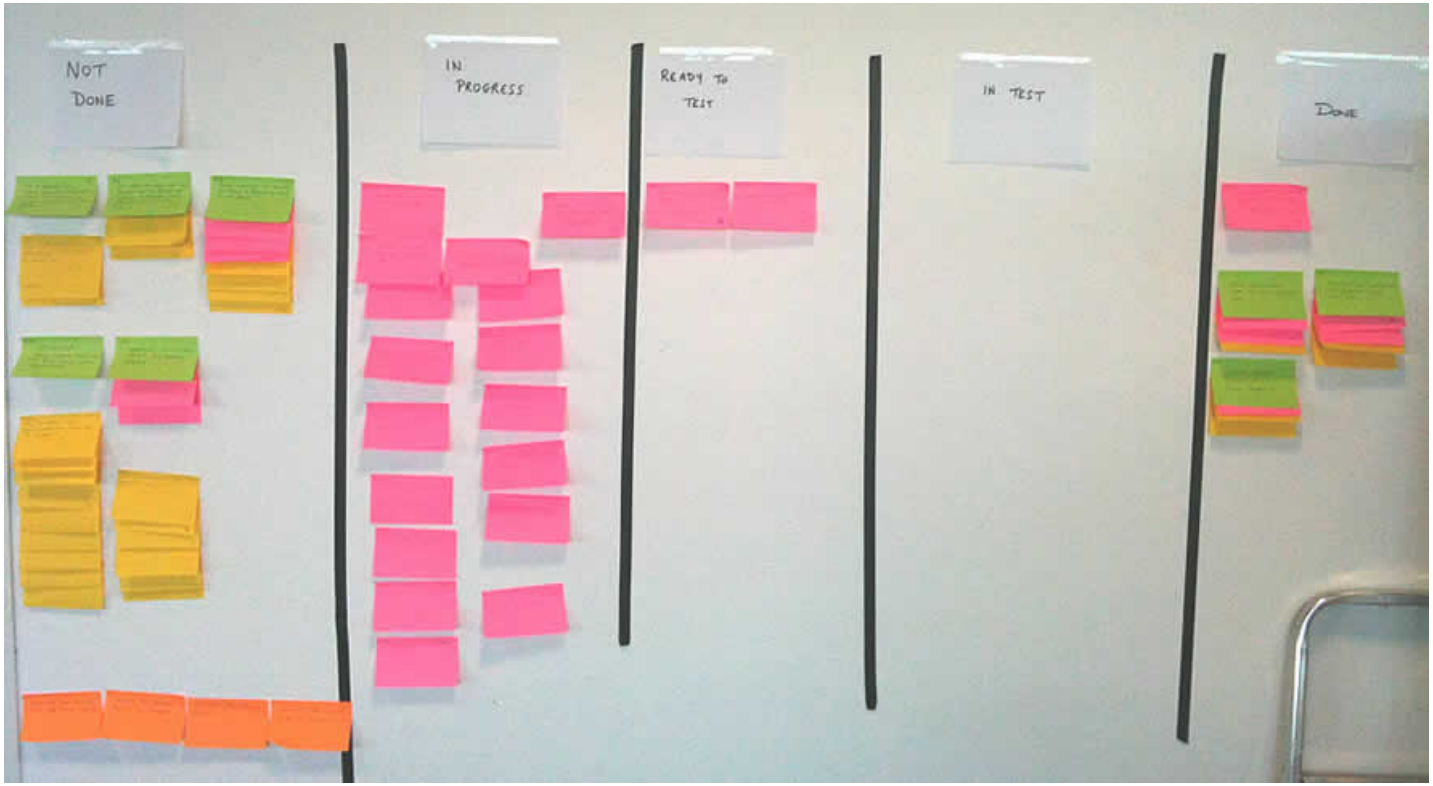


# Scrum

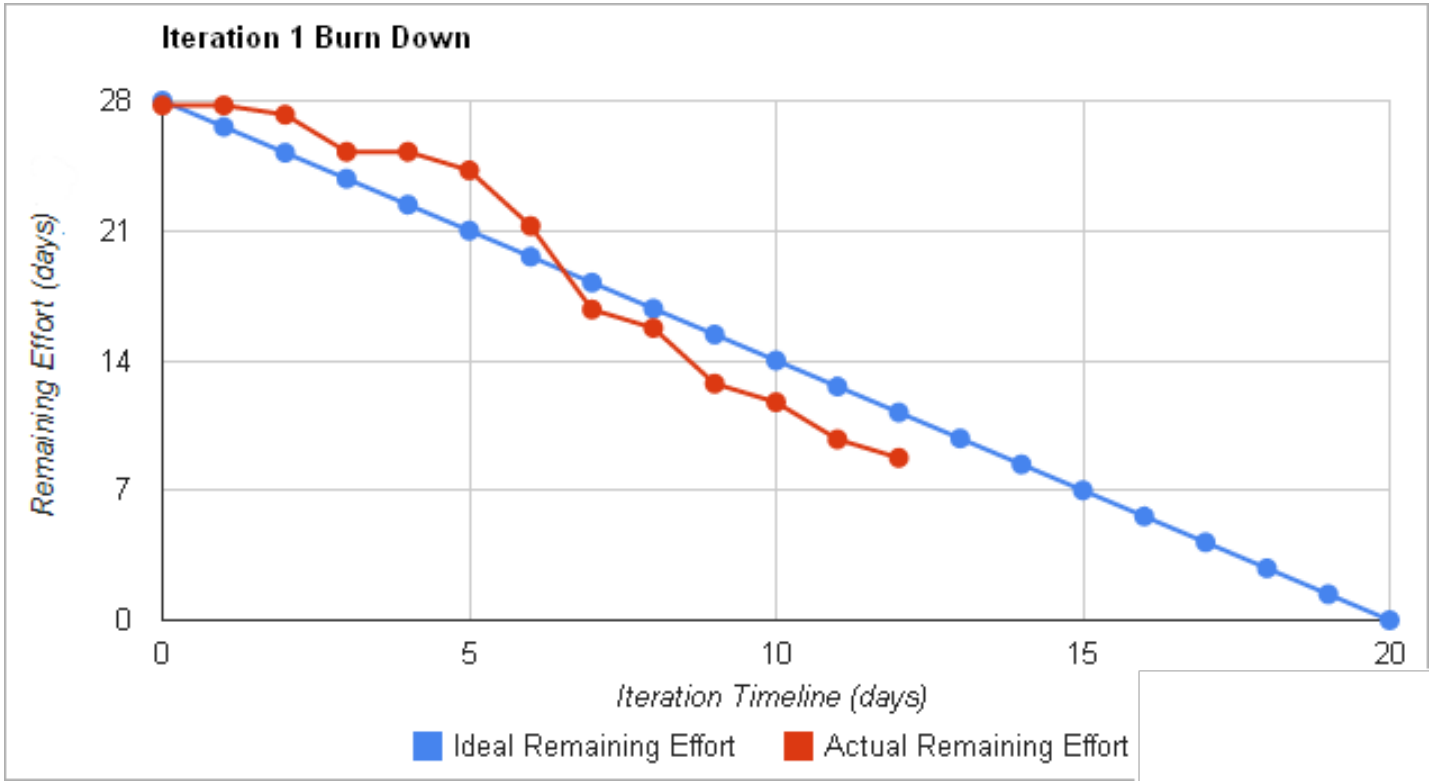



By Lakeworks - Own work, GFDL

# Scrum - TODO board



# Scrum - Burndown chart



# Prototyping

- Bygg en enkel prototyp
- Visa den för beställaren
- Använd den feedback ni fått för att förbättra 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 Examinerande delmoment
3 Mjukvaruprojekt
4 Kravspecifikation
5 Metoder
6 Systemdesign och OOP
7 Testning
8 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
- Strukturerad programmering
- Objektorientering
- Multiparadigm?

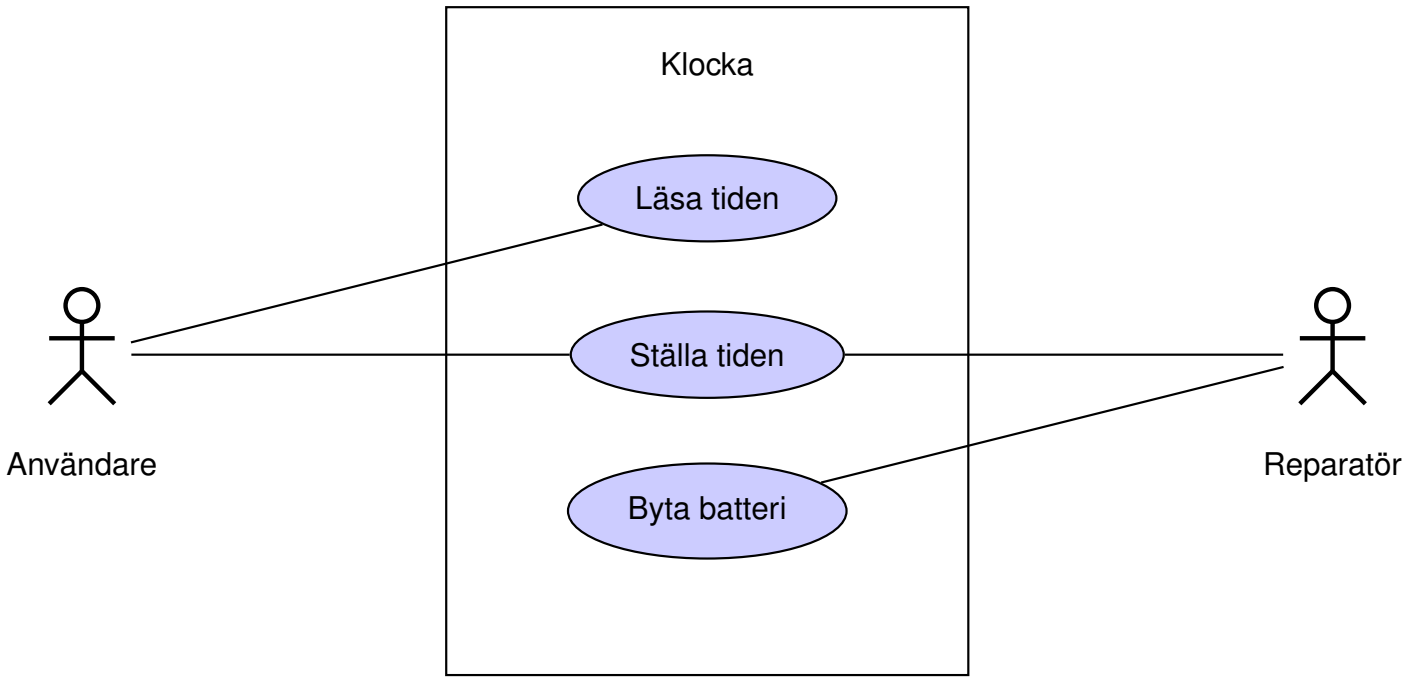
# Vad är ett objekt?

- 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

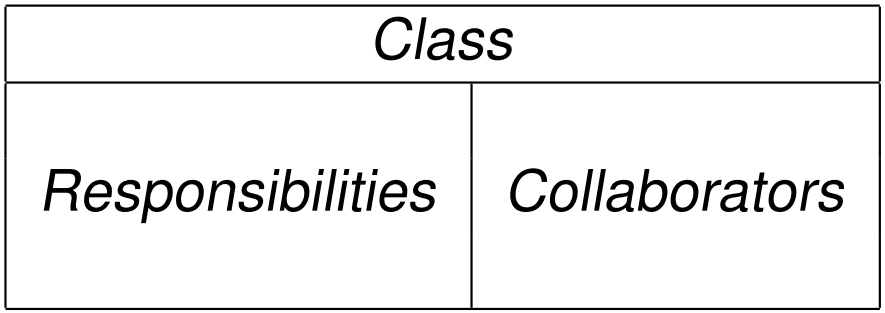
# Objektorienterad analys och design (OOA/OOD)

- 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



# CRC-kort

| *Class* | |
|---|---|
| *Responsibilities* | *Collaborators* |

Exempel:

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

# CRC-kort

| *Class* | |
|---|---|
| *Responsibilities* | *Collaborators* |

Exempel:

| **Player Character** | |
|---|---|
| Change position | Game world |
| Check collision | Game objects |
| Display | Controller |

LINKÖPING UNIVERSITY

# Testning - översikt

Tänk på att göra så att ni kan testa projektet. Det är förmodligen inte en rimlig ambition att skapa enhetstester för alla delar av projektet. Men det kan vara en rimlig sak att göra för vissa klasser. Ofta är det så att en bra objektorienterad design är lättare att testa en en dålig design. Detta beror mycket på coupling.

Framförallt bör ni se till att ni snabbt får igång en miljö där ni kan köra ert spel och testa att funktionaliteten ni skapar faktiskt fungerar som förväntat.

LINKÖPING UNIVERSITY

# Kom ihåg

- Webreg - se till att ni är anmälda
- GitLab - skapa repository, bjud in assistent och kursledare
- Första deadline är kravspecifikationen (se schema på kurssidan)

www.liu.se

LINKÖPING UNIVERSITY