



LUNDS  
UNIVERSITET

# Föreläsning 4: Projektplanering & Granskning

ETSA02 Programvaruutveckling – Metodik 2019 | Markus Borg



# Agenda F4

---

Projektplanering och projektplaner

Statisk testning - granskning

Om robotprojekten och laboration 3

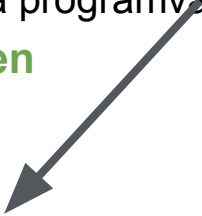


**LUNDS**  
UNIVERSITET


# 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**



LUNDS  
UNIVERSITET

# Programvaruprojekt – Grundläggande begrepp

ETSA02 Programvaruutveckling – Metodik | Markus Borg



# Projektplanering och mjukvara: Why care?

---

1. Programvara utvecklas (nästan) alltid i projektform
2. Programvaruutvecklingsprojekt har frekvent misslyckats sedan 60-talet
  - Kartläggning från Standish Group (2003)
    - 13 522 programvaruprojekt
    - 82% försenade
    - 43% sprängde budget
  - Huvudförklaring inte tekniska problem
    - Mänskliga faktorer dominerar!



LUNDS  
UNIVERSITET





# Projekt - Ingen allennarådande definition...

---

”a planned piece of work that has a specific purpose”



Svenska  
Akademiens  
ordlista

“planerat arbete av större omfattning”

## Återkommande egenskaper på projektarbete

- görs ej på rutin - osäkerhet råder
- planering är nödvändigt - även för det osäkra
- finns ett uttalat mål
- förutbestämd tidsram
- resurserna är begränsade



LUNDS  
UNIVERSITET

# Programvaruprojekt vs. traditionella ingenjörprojekt

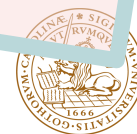
---

Programvara är ingen fysisk produkt – bara information!

- ”osynlig” produkt, framsteg mindre tydliga
- påverkas inte av välkända fysiska lagar
- kan förändras sent – både styrka och utmaning
- komplexitet per \$ hög



**Vi bygger inte ett hus till eller  
ännu en bro... Vi utför  
innovation på beställning!**



**LUNDS**  
UNIVERSITET



# Grundläggande begrepp

---

**Milstolpe** = en utvecklingsaktivitets slutpunkt, t.ex.

- Scope freeze
- Code complete
- Conclusion of test

**Leverabel** = konkret projektresultat som tas emot av någon intressent. Produceras ofta i samband med milstolpe. Exempel:

- Prototyp
- Kravspecifikation 1.0
- Testrapport



**LUNDS**  
UNIVERSITET

		Monday						Tuesday						Wednesday						Thursday						Friday						Sa	Su
w		8	10	12	13	15	Late	8	10	12	13	15	Late	8	10	12	13	15	Late	8	10	12	13	15	Late	8	10	12	13	15	Late		
13	Activity	25/3		<-	-	-	-	-	-	Lab0	-	-	-	-	-	-	->	Lab1			O1												
	Groups 1-4																	Alfa			2116												
	Groups 5-8		F1															Beta			3308												
	Groups 9-12																		Alfa			2116											
	Groups 13-16																		Beta			3308											
14	Activity	1/4																Lab2			O2												
	Groups 1-4																		Alfa			2116											
	Groups 5-8		F2																Beta			3308											
	Groups 9-12																			Alfa			2116										
	Groups 13-16																			Beta			3308										
15	Activity	8/4																Lab3			O3												
	Groups 1-4																		Alfa			2116											
	Groups 5-8		F3																Beta			3308											
	Groups 9-12																			PW			1147										
	Groups 13-16																			PW			1149										
16	Activity	15/4																															
	Groups 1-4																																
	Groups 5-8		F4																														
	Groups 9-12																																
	Groups 13-16																																
17	Activity	Annandag påsk						Exam period						Exam period						Exam period						Exam period							



LUNDS  
UNIVERSITET



LUNDS  
UNIVERSITET

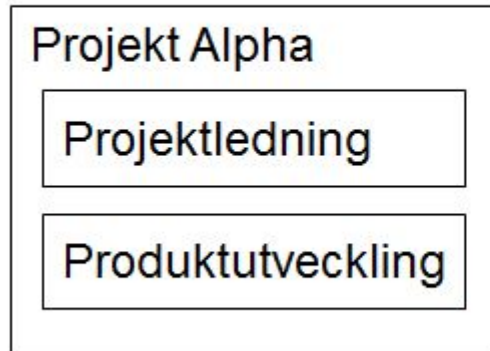
# Projektplanering och Projektplaner

ETSA02 Programvaruutveckling – Metodik | Markus Borg



# Vad kommer först: Projektplan eller kravspecifikation?

---



- Beroende av varandra
- Kraven är en del av produkten. Sista versionen måste sparas.
- Planen hör till organisationen. Erfarenheterna bör sparas.





**Bra planering garanterar  
inte lyckade projekt...**

**... men dålig planering  
leder ofta till misslyckande!**



**LUNDS**  
UNIVERSITET

# Planeringen färdig först vid projektslut

---

“In preparing for battle I have always found that plans are useless, but planning is indispensable.”

- *Dwight D. Eisenhower*



- **Planering är en iterativ process som pågår under hela projektet**
- **Uppföljning under utvecklingen kritiskt!**



**LUNDS**  
UNIVERSITET

# Underskatta inte kommunikation!

---

Effektiv kommunikation nödvändigt för lyckade projekt

- Fysiska möten bäst

Global software engineering svårt, men allt vanligare

- Videokonferens, telefonmöten, mail, intranät etc.

**Outsourcing** –  
Köpa utvecklingsarbete av  
annat bolag  
**Offshoring** –  
Utlandsentreprenad, etablera  
utvecklingscenter i annat land

“An Empirical Study of Speed and Communication in Globally Distributed Software Development”, Herbsleb and Mockus, 2003

[http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=1205177&filter%3DAND%28p\\_IS\\_Number%3A27132%29](http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=1205177&filter%3DAND%28p_IS_Number%3A27132%29)

“Conflict Management in Student Groups - A Teacher’s Perspective in Higher Education”, Borg et al., 2011

<http://journals.lub.lu.se/ojs/index.php/hus/article/view/4923>

# Förmedla förväntningar och framsteg

---

- Programvara är en osynlig produkt
  - Utvecklingsarbetet måste aktivt synliggöras
- Förväntningar och framsteg måste kommuniceras
- Bryt ned krav till konkreta arbetspaket
  - Följ upp hur arbetet fortskrider
  - Rapportera kontinuerligt till alla inblandade



LUNDS  
UNIVERSITET



# KPI - Key Performance Indicator

---

- Nyckeltal för att värdera en organisations verksamhet
  - Antal nya per vecka: krav, rader kod, testfall
  - Förseningar
  - Försäljningsresultat
  - Fel som hittas under test
  - Rapporterade buggar från kunder



## Fördelar

- Enkelt att mäta och förstå
- Kan jämföras över tid och mellan projekt

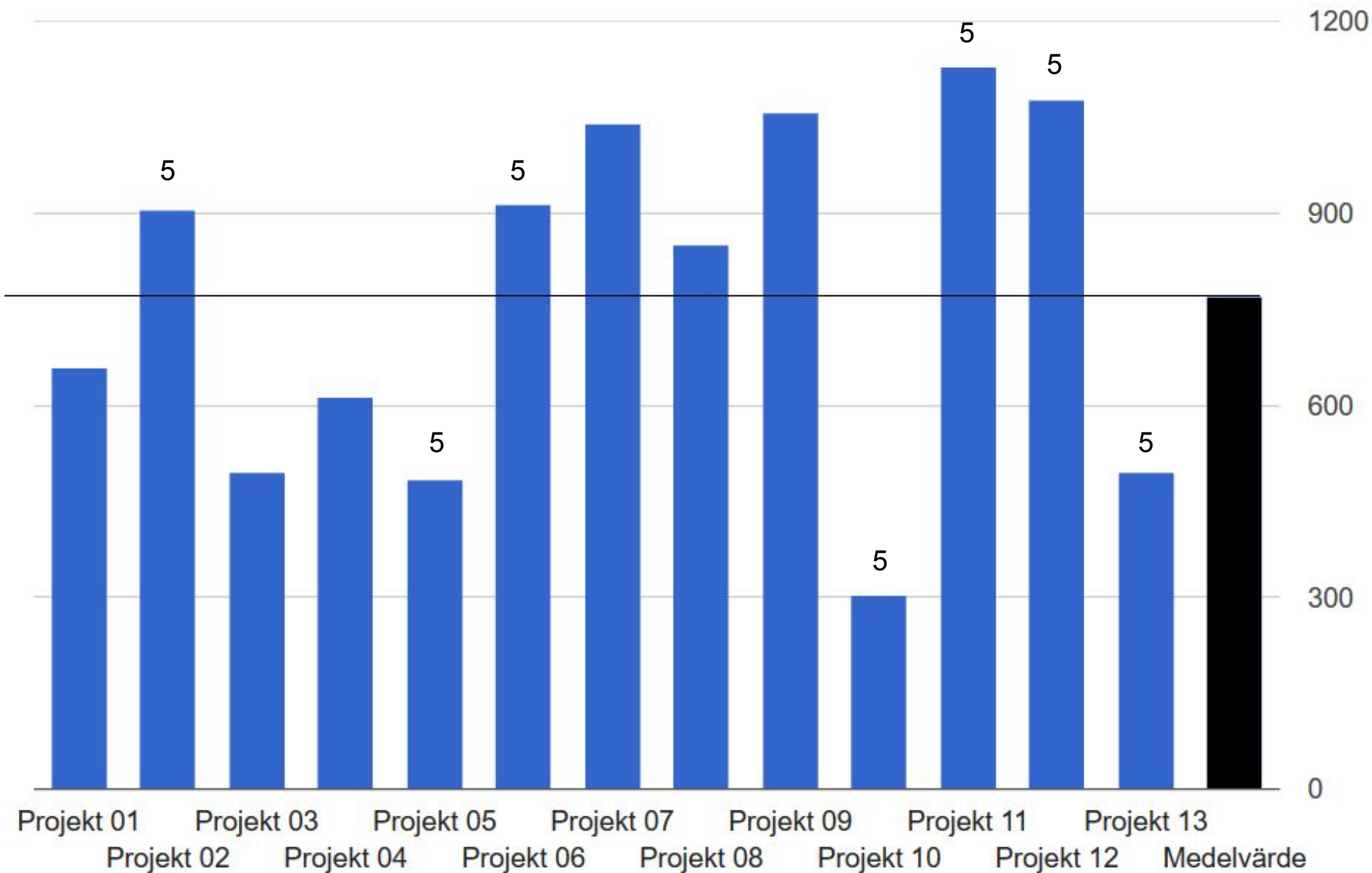
## Nackdelar

- Förenklad bild av verkligheten
- Risk för suboptimering



LUNDS  
UNIVERSITET

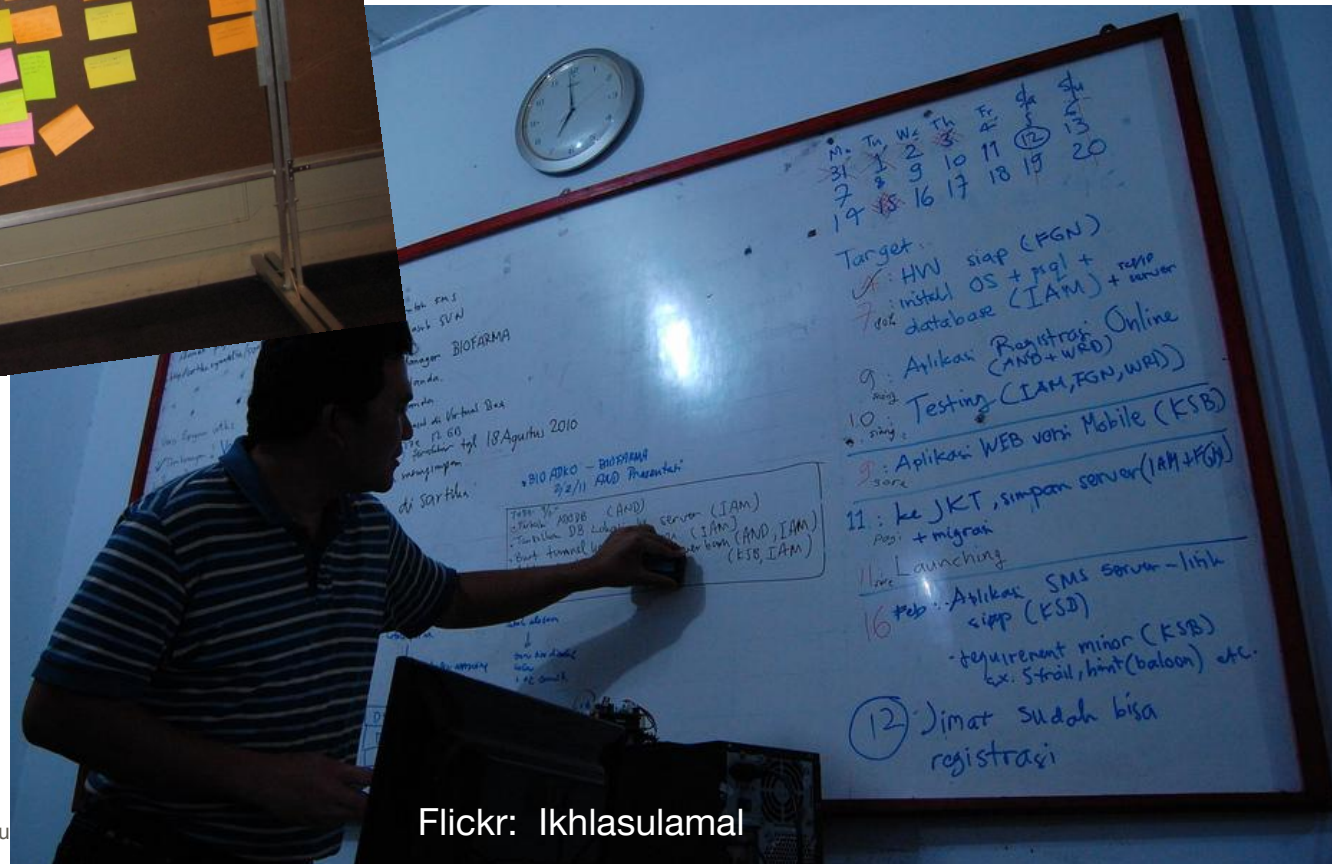
# KPI: Kravvolym - Antal ord i tidig version av SRS



# Kraftfulla verktyg: Whiteboards och post-its!



Flickr: geodog



Flickr: Ikhlusalamal

DID YOU GET THE  
NOTE I LEFT ON YOUR  
MONITOR?

YES

DID YOU GET MY  
VOICEMAIL?

YES

DID YOU GET  
MY E-MAIL?

YES

SHOULD I  
TELL YOU  
WHAT THE  
NOTE AND  
VOICEMAIL  
AND E-MAIL  
SAID?

THERE'S  
SOME-  
THING  
WRONG  
WITH  
YOU.

www.dilbert.com scottadams@aol.com

8407 © 2007 Scott Adams, Inc./Dist. by UFS, Inc.



# Fyra viktiga moment i projektplanering

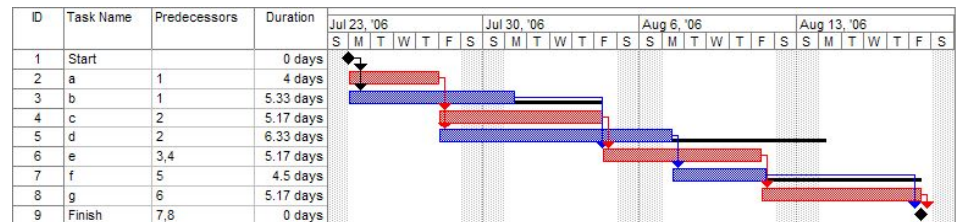
Intressentanalys



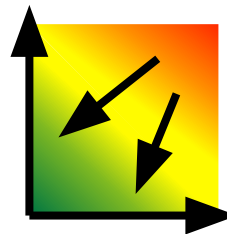
Kostnadsskattning



Schemaläggning



Riskhantering



LUNDS  
UNIVERSITET

# Intressenter och olika typer av mål

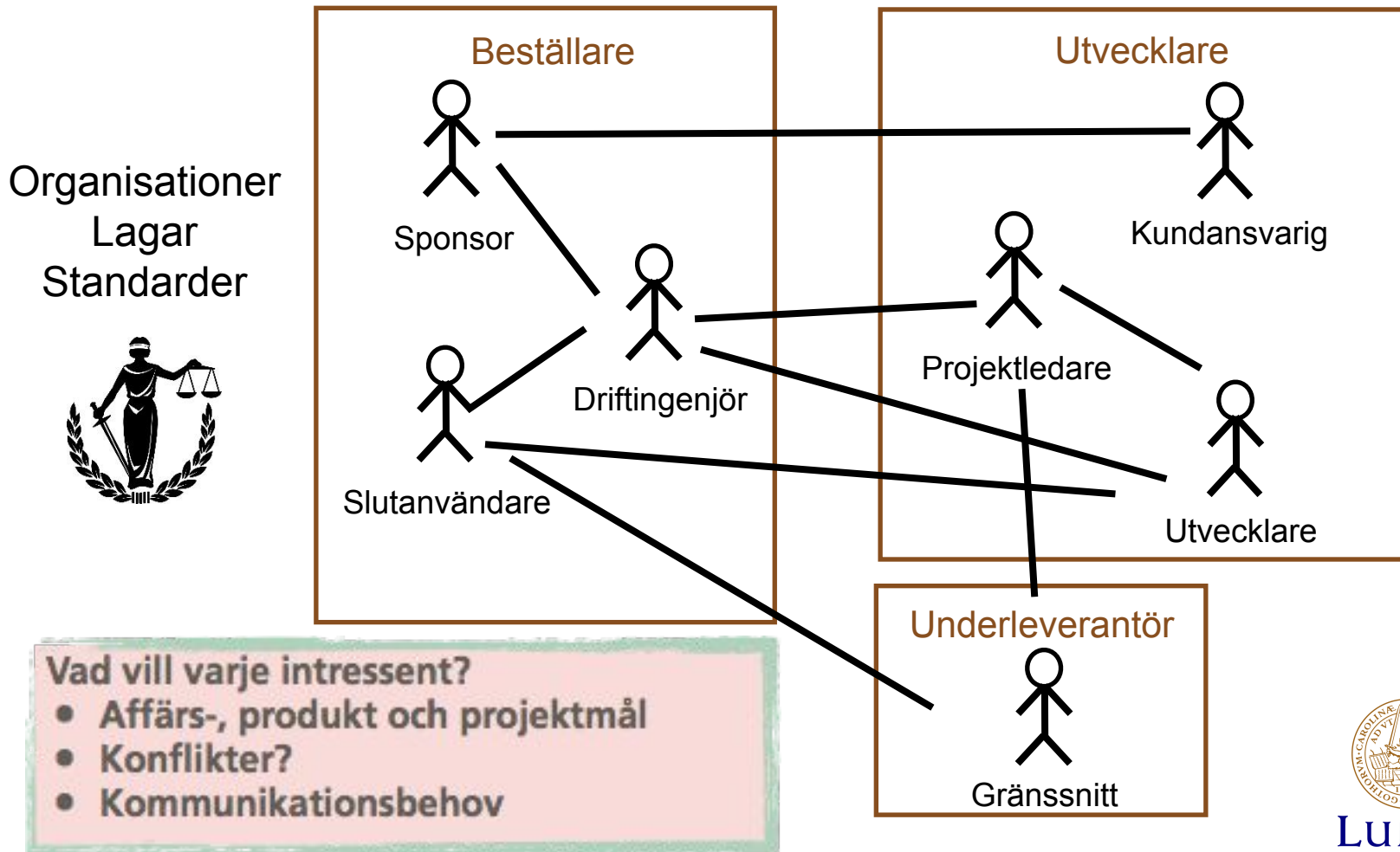
---

- Affärsmål - i linje med beställarens affärsidé
  - Hur genererar detta intäkter?
  - Varför ser aktieägarna värde i detta?
- Projekt mål - verksamhetens målsättningar
  - Lära sig ny domän, introducera nyanställda, utveckla nya arbetsmetoder, prova ny teknik
- Produktmål - ambitioner med den nya produkten
  - Vad utmärker produkten på marknaden?
  - Vilken nytta erbjuds användarna?



**Uppdelningen gör att motsättningar kan identifieras!**

# Intressentanalys (stakeholder analysis)



# Kostnadsskattning

Vid programvaruutveckling  
domineras kostnaderna av persontid

- Kostnadsskattning => tidsuppskattning



Enkelt mått på programvarans storlek

- rader källkod (lines of code)

Naivt mått på utvecklarens produktivitet

- rader källkod per personmånad

**Varierar enormt!**

Stora komplexa system:

~30 rader/personmånad

Enkel välkänd domän:

~900 rader/personmånad

Programmerarens  
förmåga kan påverka  
med en faktor 10



**LUNDS**  
UNIVERSITET



# Kostnadsskattning – Tre metoder

---

## Expertbedömning

- Flera erfarna personer gör kvalificerade gissningar
- Top-down: övergripande funktioner → subfunktioner → integration
- Bottom-up: komponenter → subsystem → system

## Estimat baserade på analogier

- Jämför med tidigare utvecklingsprojekt
- Liknande storlek? Komplexitet? Motsvarande utvecklarkompetens?

## Algebraiska metoder

- Räkna ut ett estimat, till exempel:  $\text{kostnad} = A \times \text{Size}^B \times M$   
A = komplexitet, B = extrakostnad för stora system, M = mognad
- Parametrarna bestäms baserat på databas med historiska projekt

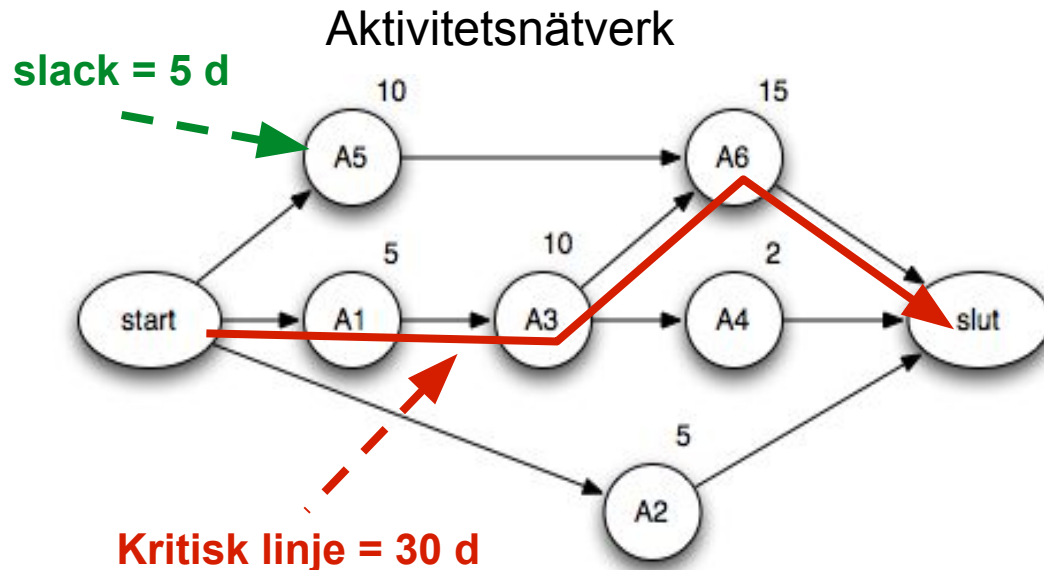


LUNDS  
UNIVERSITET

# Schemaläggning - Aktivitetsnätverk

- Bryt ned projekt i arbetspaket
- Estimera tidsåtgång och beroenden
- Identifiera kritisk ledtid, dvs. minimal genomförandetid

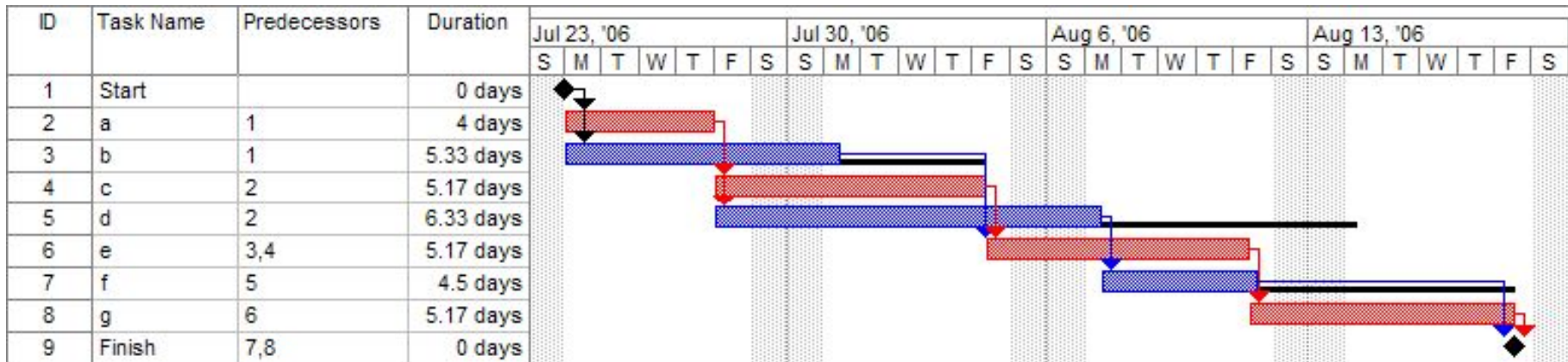
Aktivitet	Tid (d)	Beroenden
A1	5	
A2	5	
A3	10	A1
A4	2	A3
A5	10	
A6	15	A3, A5



- Störningar på **kritisk linje** försenar projektet
- Övriga aktiviteter har **slack**, dvs. utrymme för försening

# Schemaläggning – Gantt-diagram

- Horisontella stapeldiagram med tidsaxel
- En managementrevolution vid 1900-talets början!



- Hämtad från Wikipedia, skapad med Microsoft Project
- Kritisk väg presenteras i rött
- Slack representeras av svart linje



LUNDS  
UNIVERSITET

# ”Naturlagar” inom software engineering

---

Mest på skoj, men sätter fingret på upplevda fenomen.  
Urval, fritt översatta:

## **Parkinsons lag**

”En arbetsuppgift kommer att ta den tid som är avsatt för ändamålet.”

## **Hofstadters lag**

“En arbetsuppgift tar alltid längre tid än du förväntar dig, även om du tar Hofstadters lag med i beräkningen.”

## **Brookes lag**

”Att tilldela fler utvecklare till ett försenat projekt försenar det ytterligare.”

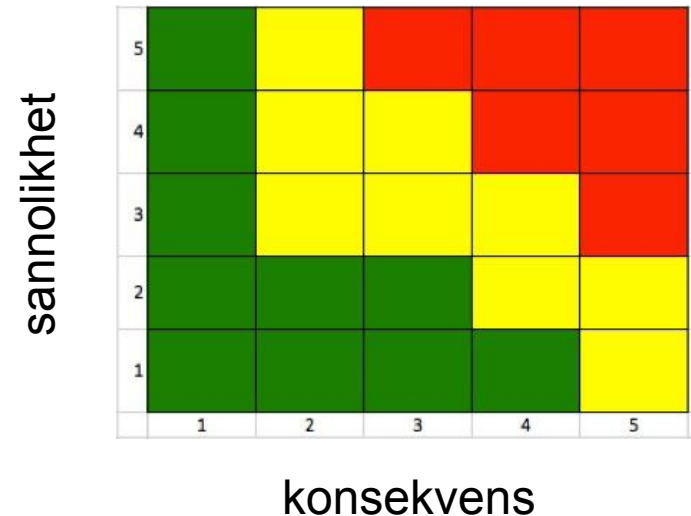
# Riskhantering

## Definition av risk

sannolikhet för  
oönskad konsekvens

×

konsekvensens storlek



Olika risktyper har olika påverkan

- **Projektrisker:** projektplan, tillgängliga resurser
- **Produktrisker:** programvaran som utvecklas
- **Affärsrisker:** påverkar utvecklingsorganisationen



LUNDS  
UNIVERSITET



# Riskhanteringsprocessen

---

**Aktivitet**

1) Identifiera  
risker

2) Bedöm  
risker

3) Behandla  
risker

4) Följ upp risker  
under projektet

**Leverabel**

Lista med  
risker

Priorite-  
rade  
risker

Planer för  
risker

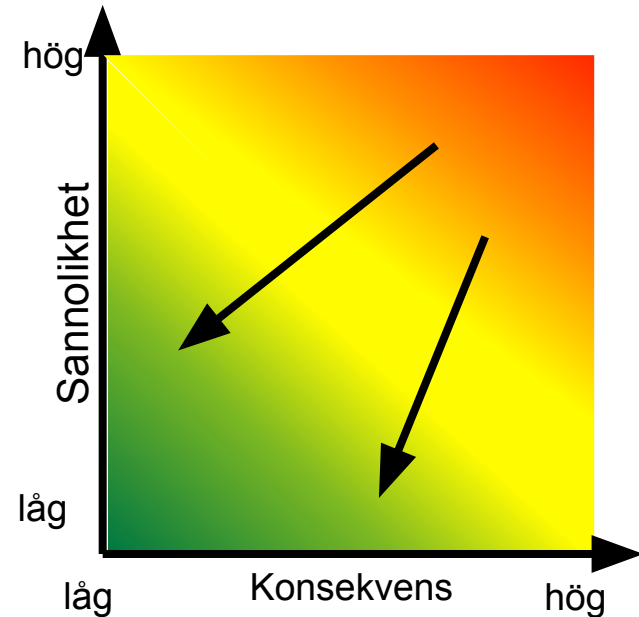


**LUNDS**  
UNIVERSITET

# Riskhantering i praktiken

## Strategier

- Reducera konsekvens
- Minska sannolikhet
- Alternativ (plan B)



Riskkälla	S	K	Risk (S x K)	Strategi
Hårdvara försenad	2	5	10	Undersöka alternativ
				Konstruera simulator
Sjukskrivningar	1	2	2	Begränsa övertid
Krav förändras	4	3	12	Veckomöten med kund

# LTH-gemensam avslutning: Riskhantering (2 år)



# Innehåll i en projektplan

---

## Inledning

projektmodell, övergripande produktbeskrivning, målsättningar, begränsningar

## Projektorganisation

utvecklingsorganisation, testorganisation, andra intressenter

## Hårdvara och programvara

Resurser som krävs för projektets genomförande

## Arbetsnedbrytning

aktiviteter, leverabler, milstolpar

## Tidplan

när varje aktivitet påbörjas och avslutas, när varje milstolpe ska uppnås

## Uppföljning och rapportering

hur framsteg mäts och hur det kommuniceras

## Risikanalys



**LUNDS**  
UNIVERSITET

# Tidpunkt, kostnad eller kvalitet?

---

**Tidpunkt**  
När ska vi leverera?



**Kostnad**  
Vad får det  
få kosta?

**Kvalitet**  
Hur bra ska  
det bli?

Tre önskvärda  
egenskaper:

- Leverans i tid
- Utveckling inom budget
- Programvara med god kvalitet

Bara en eller två kan  
prioriteras!



**LUNDS**  
UNIVERSITET



# Sammanfattning projektplanering

---

- Programvaruprojekt speciella eftersom de innebär komplex innovation av osynlig produkt
- Planering pågår till projektet är avslutat
- Fyra centrala aktiviteter i projektplanering: intressentanalys, kostnadsskattning, schemaläggning och riskhantering
- Projektplanen beskriver bl.a. projektorganisation, arbetsnedbrytning, tidplan och riskanalys





LUNDS  
UNIVERSITET

# Statisk testning - Granskning

ETSA02 Programvaruutveckling – Metodik | Markus Borg



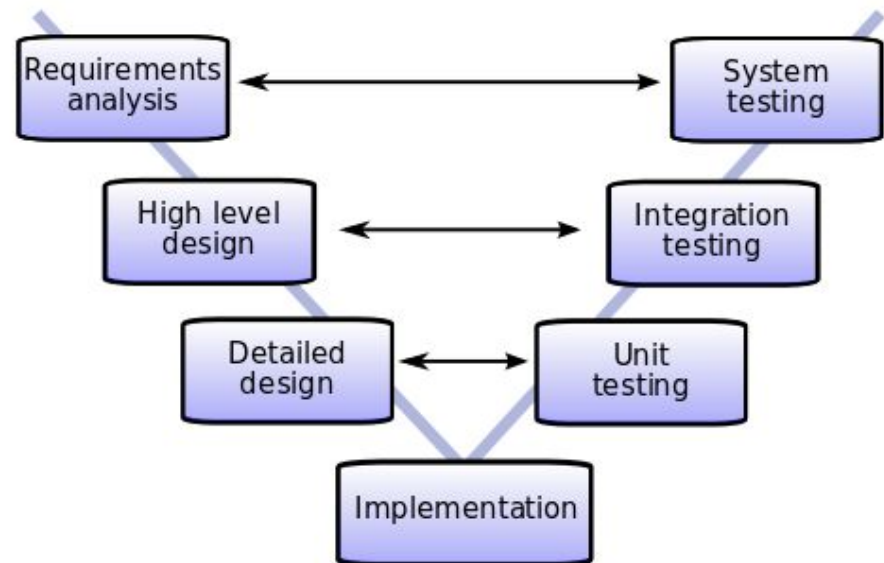
# Granskningar – grundläggande idé

---

Hitta fel tidigt utan att exekvera kod – dvs. statisk testning

Alla artefakter kan granskas (kravspecifikation, testspecifikation, design, källkod, testfall etc.)

- Läs artefakt på ett strukturerat sätt
- Rätt personer ska läsa
- Personerna ska läsa på rätt sätt
- Alla viktiga delar av dokumenten ska läsas



# Typer av granskningar

---

- Kodgranskning
  - Systematisk undersökning av källkod.
- Inspektion
  - Formell insats för att identifiera brister eller säkerställa kvalitet enligt väldefinierad process.
- Audit
  - Genomlysning av oberoende organisation för att säkerställa att produkt eller process uppfyller standarder, regelverk, lagstiftning, etc.



LUNDS  
UNIVERSITET

# Kodgranskning - Vad och varför?

---

- Okulärbesiktning av någon annans källkod som komplement till dynamisk testning
- Huvudsyften
  - Identifiera fel
  - Upprätthålla konventioner
  - Identifiera sårbarheter
  - Sprid kunskap om koden

```
///</summary>
///<param name="orderedChildIds">A collection of child ids.</param>
///<param name="movedChildId">The id of the moved child.</param>
public void ChangeChildSortOrder(int[] orderedChildIds, int movedChildId)
{
    + if (orderedChildIds == null)
    + {
    +     throw new ArgumentNullException("orderedChildrenIds");
    + }

    + bool found = false;
    + ItemToItem moved = null;
    + ItemToItem previous = null;
    + ItemToItem next = null;
    + foreach (int orderedChildId in orderedChildIds)
    + {
    +     ItemToItem current = ChildItems.FirstOrDefault(c => c.ChildId == orderedChildId);
    +     if (current != null)
    +     {
    +         {
    +             if (current.ChildItem.ItemId == movedChildId)
    +             {
    +                 moved = current;
    +                 found = true;
    +             }
    +         }
    +         else
    +         {
    +             if (current.ChildItem.ItemId == movedChildId)
    +             {
    +                 moved = current;
    +                 found = true;
    +             }
    +         }
    +     }
    + }
```

# Kodgranskning - När och hur?

---

- Utvecklingsprocessen kan specificera kodgranskning vid olika tillfällen på olika vis. Till exempel:
- Innan commit and push (eller pull request)
  - Annan utvecklare måste granska innan kod hamnar i kod-repository
- Efter commit and push
  - Annan utvecklare måste granska ny kod som integrerats i kod-repository
- Inför release
  - Flera utvecklare genomför formell inspektion



LUNDS  
UNIVERSITET



# Kodgranskning - Tips!

---

- Kodgranskning är bra och viktigt. Källkoden blir bättre och kunskap sprids i organisationen.
- Mängder av “best practices” på Internet. Exempelvis:
  - Granska inte för många rader kod åt gången (<400)
  - Stressa inte. Det ska inte gå fort.
  - Granska inte för längre åt gången (<1 h)
  - Notera brister i en granskningslogg
  - Planera för hur identifierade brister ska åtgärdas



LUNDS  
UNIVERSITET

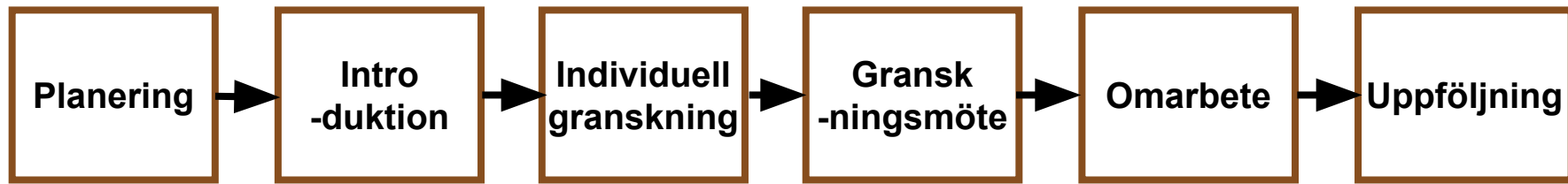
# Kodgranskning - Verktøgsstöd

---

- Vissa typer av brister kan man verktyg för att identifiera
  - Låt människorna fokusera på resten
- Statisk kodanalys (Lab 4)
  - Verktøg som analyserar källkoden utan att exekvera den (antingen källkod eller kompilerad kod)
  - Kan identifiera när kod bryter mot konventioner
  - Kan identifiera kodkonstruktioner som ofta leder till buggar
  - Kan identifiera sårbarheter



# Inspektioner (för både dokument och kod)



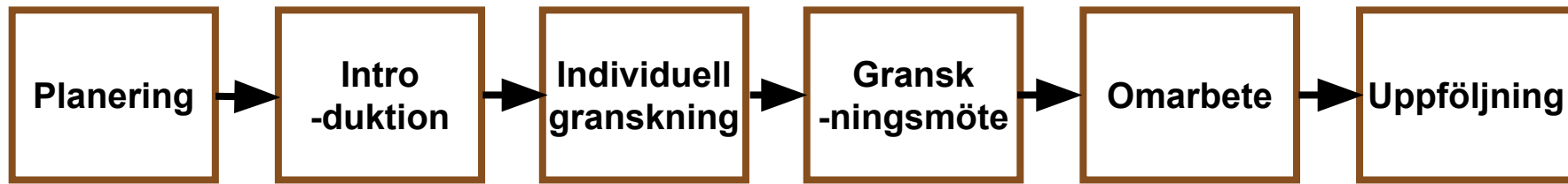
- Formell process för strikt granskning
- Utvecklad på 70-talet av IBM
- Ofta processkrav i kritiska domäner
- Resultat dokumenteras nogga
- Övning 4
  - Granskningsprotokollet del i Final Release

The screenshot shows a web-based form titled "Granskningsprotokoll för Dokument 0.XX". It includes a menu bar (File, Edit, View, Insert, Format, Data, Tools, Add-ons, Help) and a toolbar with various icons. The form is divided into several sections:

- Grupp**: A table with columns A through F.
- Dokument**: A section with fields for "Vilket dokument och dokumentversion som har granskats - Länka om på webben", "Datum", "När granskningsmötet ägde rum", "Granskare", "Vilka personer som har medverkat i granskningen / granskningsmötet", and "Dokumenterat av".
- Uppföljning**: A section with a field for "Person som dokumenterat granskningsprotokollet i projektwebben".
- Prioriteter**: A table with columns A through F.
- Checklista**: A section with a field for "Namn och länk till eller referens till checklista som använts vid granskningen".
- Övergripande synpunkter, åtgärdsförslag och eller beslut**: A text area for overall comments, suggestions, and decisions.
- Problem #**: A table with columns for "Position i dokument", "Problemtyp", "checklista\*", "Problem", "Prioritet", and "Uppföljning\*".

# Inspektioner - Steg 1-3

---

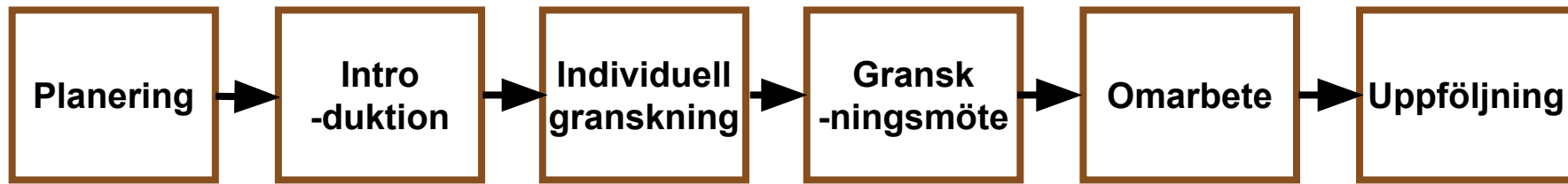


- Planering
  - Förbered material, bjud in personer till möte
- Introduktion
  - Instruera granskarna, tilldela granskningsroller
- Individuell granskning
  - Granskarna identifierar brister
  - Bristerna dokumenteras noga



**LUNDS**  
UNIVERSITET

# Inspektioner - Steg 4-6



- Granskningsmöte

- Alla samlas på möte. Artefakten går igenom från början till slut.
- Defekter funna vid individuell granskning dokumenteras.

- Omarbete

- Ansvarig utvecklare åtgärdar bristerna

- Uppföljning



**LUNDS**  
UNIVERSITET

# Roller vid granskningmötet

---

- Moderator
  - Leder mötet.
  - Styr genomläsning av artefakt från början till slut.
- Sekreterare
  - Författar granskningsprotokollet
- Författare (skapare av artefakten)
  - Svarar på eventuella frågor som dyker upp
- Granskare
  - Personer som genomfört individuell granskning



LUNDS  
UNIVERSITET



# Lästekniker vid individuell granskning

## Ad-hoc

- Upp till granskaren

## Checklist-baserad

- Stöd av en checklista
- Vanligen framtagen av kvalitetsingenjörer

## Scenario-baserad

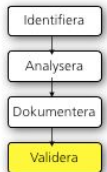
- Följ ett användningsscenario under granskningen

## Perspektiv-baserad

- Granska som en specifik roll: användare, testare, operatör, utvecklare, etc.

### Checklista för granskning för kursens projekt

1. Saknas några krav?
2. Är samtliga krav nödvändiga?
3. Finns det några motstridiga krav?
4. Kan samtliga krav verifieras?
5. Är samtliga krav tydligt formulerade eller kan några krav misstolkas?
6. Finns samtliga nödvändiga definitioner?
7. Är det möjligt för dokumentets målgrupp att förstå dokumentet?
8. Följer kravspecifikationen sin dokumentmall?
9. Är något krav formulerat för detaljerat?
10. Har något krav formulerats på för hög abstraktionsnivå?
11. Är all text och illustrationer nödvändiga?
12. Har samtliga krav unika identifierare?

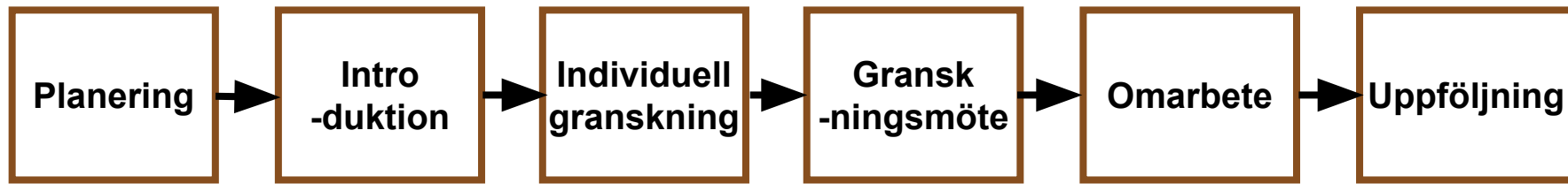


Guidat alternativ



LUNDS  
UNIVERSITET

# Vad kostar det?



- Planering och introduktion: ? h
- Individuell granskning:
  - Kravspecifikation: 5 sid/h
  - Design: 4 sid/h
  - Källkod: 150 rader/h (utan kommentarer)
  - Testdokumentation: 4 sid/h

**Bjud in till granskningsmöte först när det är meningsfullt!**

(Ebenau et.al., Software Inspection Process, 1994)

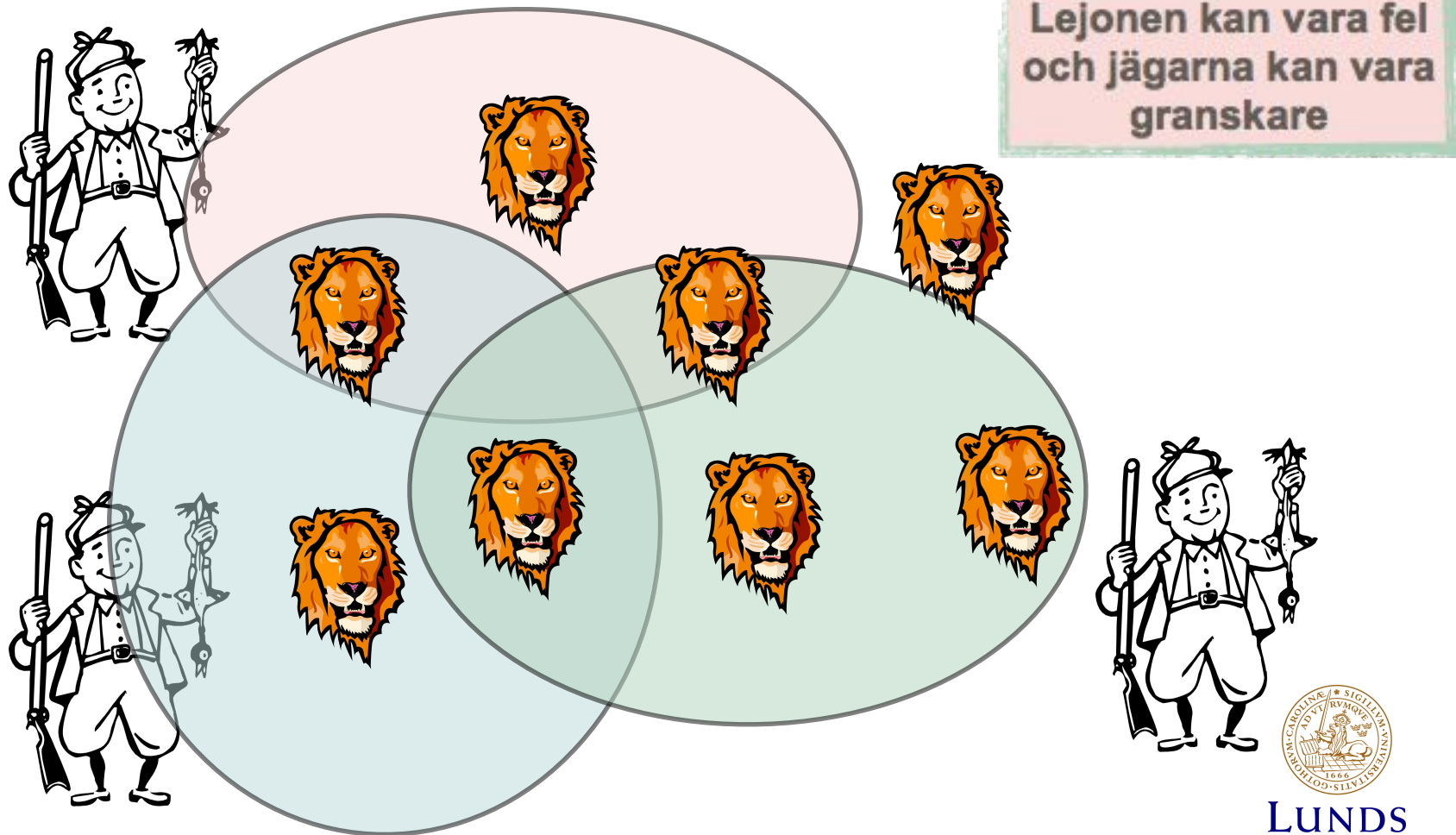
- Granskningsmöte: 4-10 personer  $\times$  2 h
- Omarbete och uppföljning: ? h



**LUNDS**  
UNIVERSITET

# Har vi hittat alla brister

## - eller hur många lejon finns det i skogen?



# Capture-recapture

En metod från ekologin för att estimerar  
djurpopulation

Antag två likvärdiga granskare samt brister som är  
lika enkla att finna, samt:

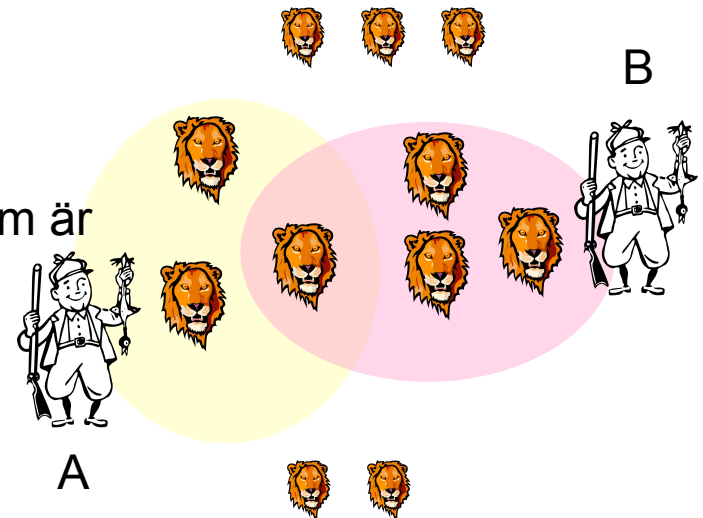
$N$  = totalt antal brister

$N_A$  = antal brister som granskare A hittar

$N_B$  = antal brister som granskare B hittar

$N_{AB}$  = antal brister som båda hittar

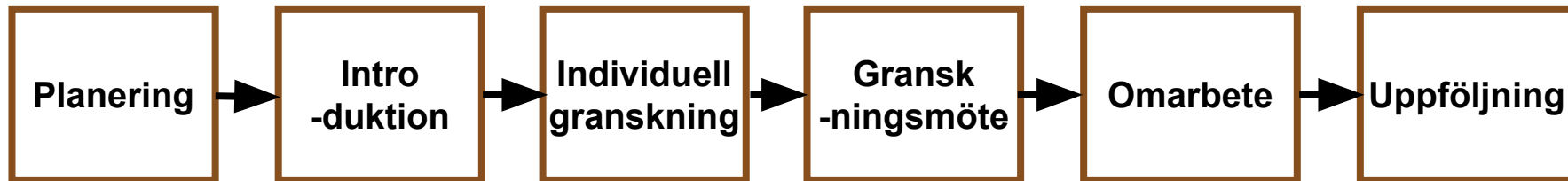
Andel brister som båda hittar bland granskare As  
resultat ( $N_{AB} / N_A$ ) motsvarar andelen brister  
granskare B hittade bland samtliga ( $N_B / N$ )



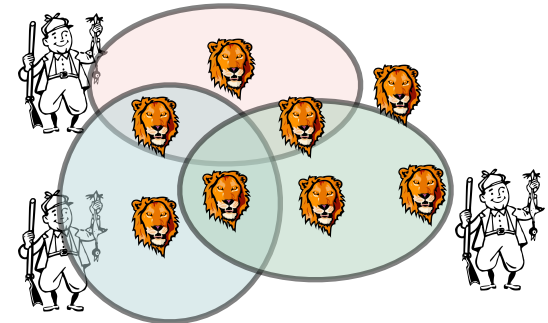
$$\frac{N_{AB}}{N_A} = \frac{N_B}{N}$$

$$N = \frac{N_A \times N_B}{N_{AB}} = \frac{3 \times 4}{1} = 12$$

# Statisk testning - sammanfattning



- Systematisk metod för att identifiera brister i artefakter utan exekvering
  - Kodgranskning
  - Inspektion
  - Audit
- Använd verktyg för statisk kodanalys
- Granskningsprocesser är en naturlig del av kvalitetssäkrande arbete
  - Dyrt men bra!



```
/// </summary>
/// <param name="orderedChilds">A collection of child ids.</param>
/// <param name="movedChildId">The id of the moved child.</param>
public void ChangeChildSortOrder(int[] orderedChilds, int movedChildId)
{
    if (orderedChilds == null)
    {
        throw new ArgumentNullException("orderedChilds");
    }

    bool found = false;
    ItemToItem moved = null;
    ItemToItem previous = null;
    ItemToItem next = null;
    foreach (int orderedChildId in orderedChilds)
    {
        ItemToItem current = ChildItems.First(c => c.ChildId == orderedChildId);
        if (current != null)
        {
            if (current.ChildItem.ItemId == movedChildId)
            {
                moved = current;
                found = true;
            }
            else
            {
                // ...
            }
        }
    }
}
```



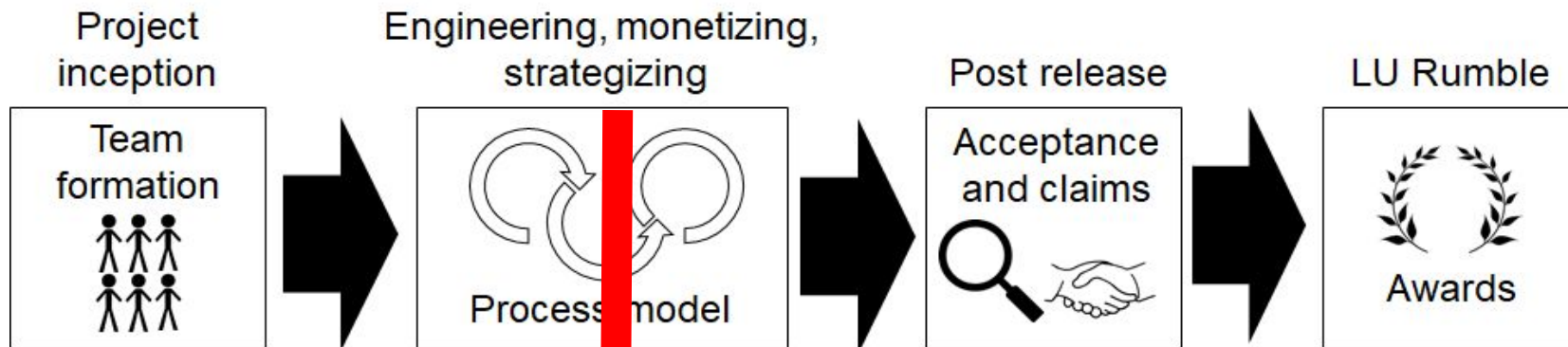
LUNDS  
UNIVERSITET

# Robotprojekten

Programvaruutveckling - Metodik | Markus Borg

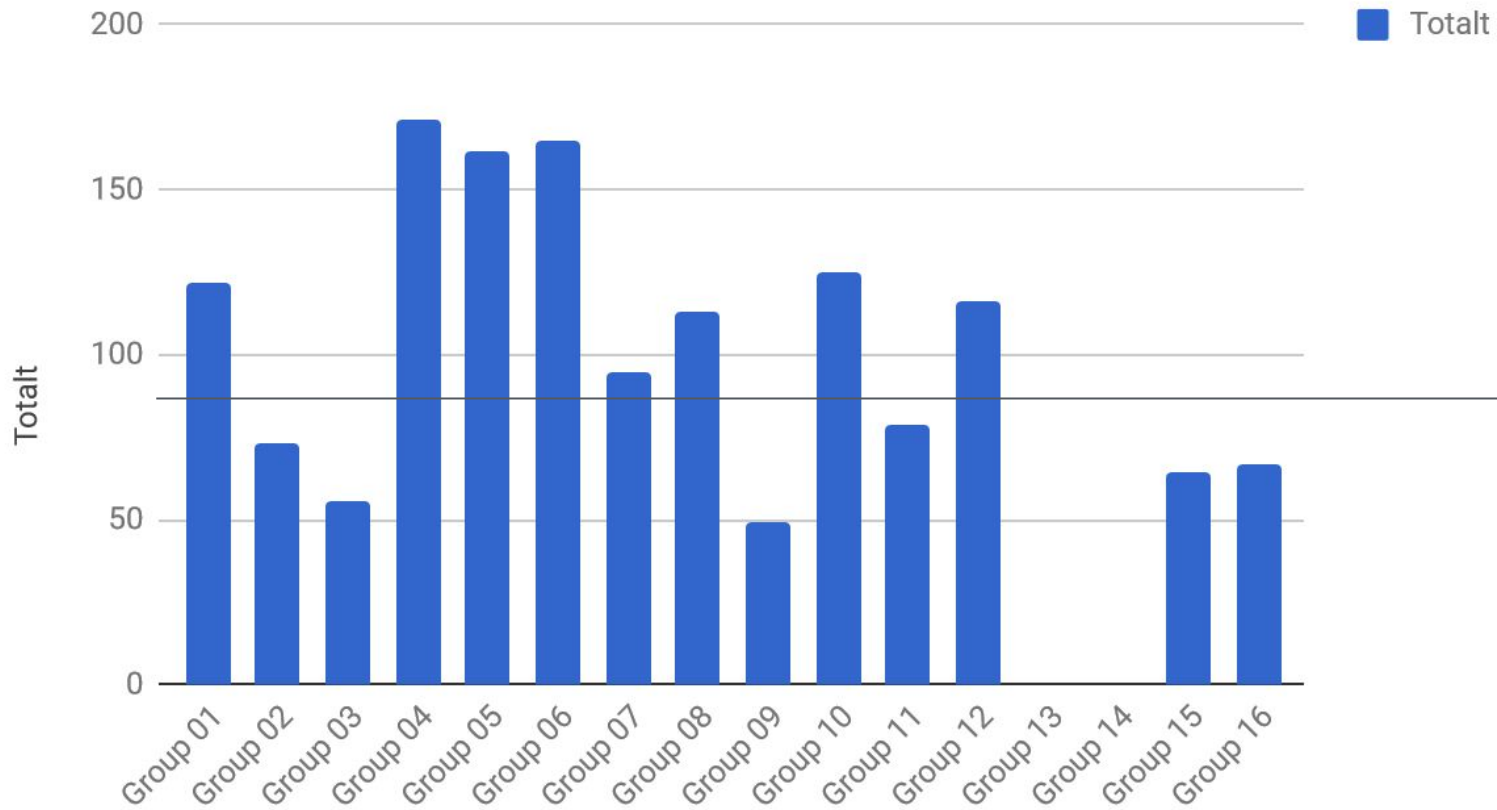




[illegible]

# Tidrapporter - Total tid per grupp

Totalt



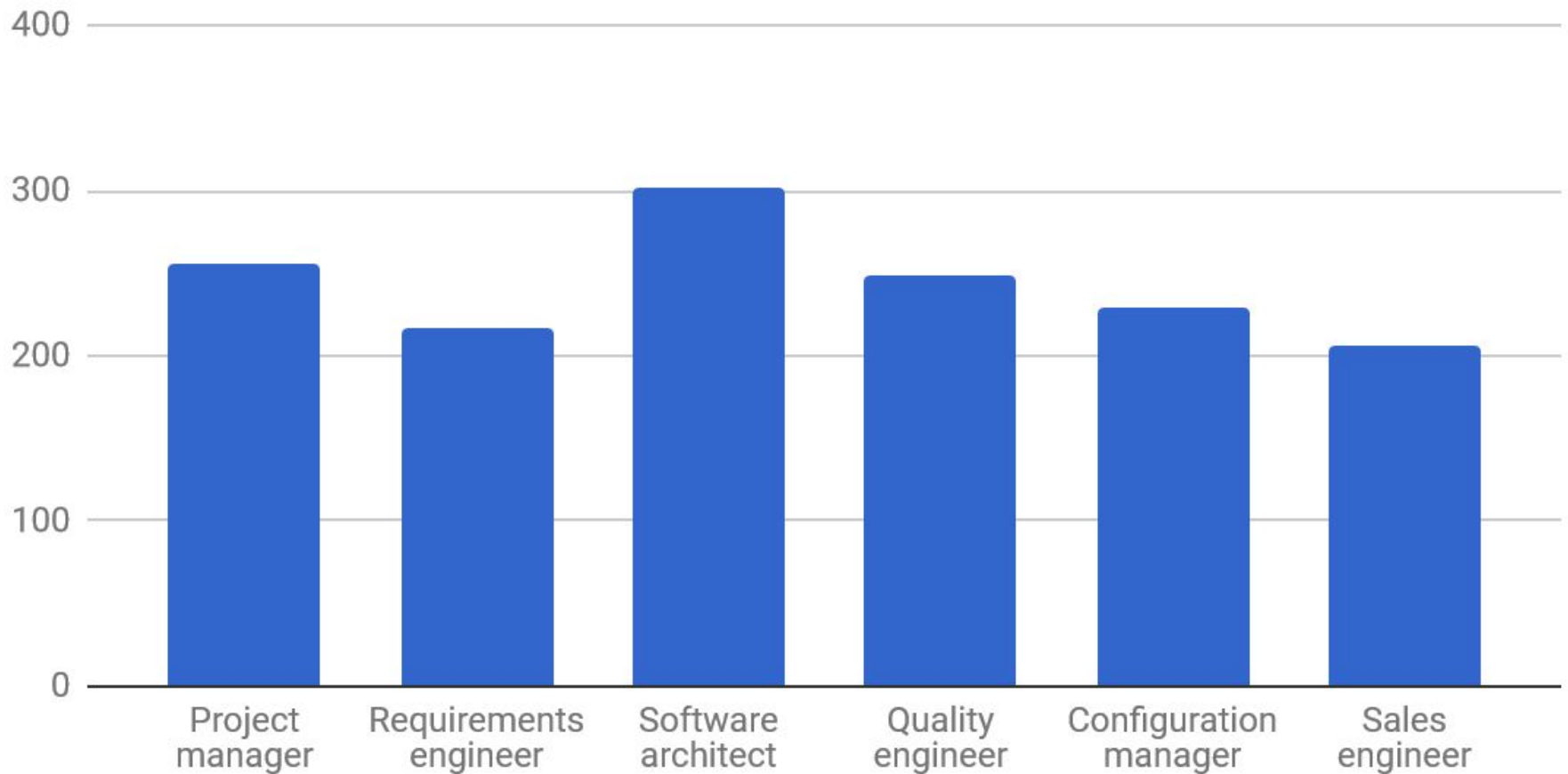
91 h



LUND  
UNIVERSITY

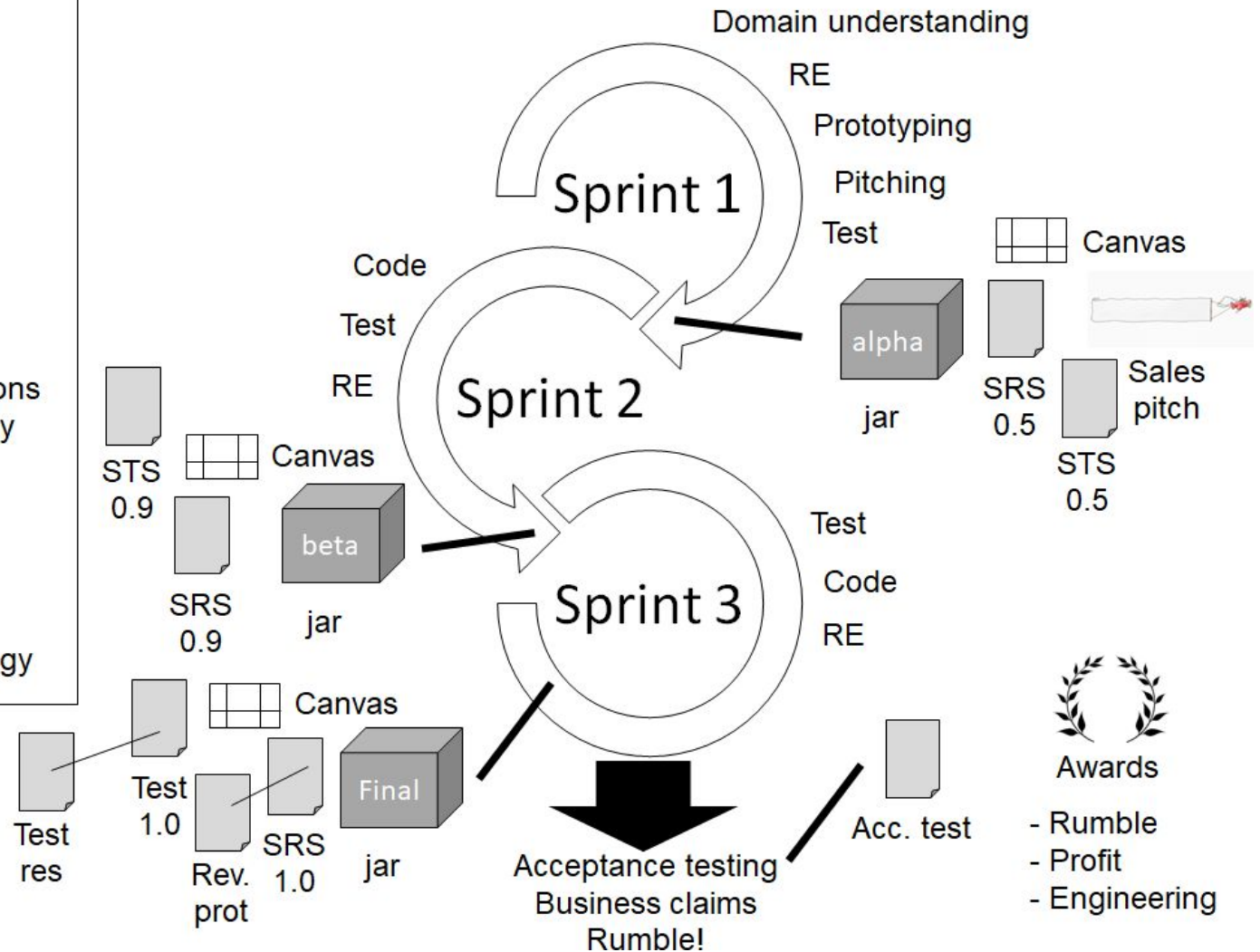
# Tidrapporter - Total tid per roll

---



# High-level goals

- Team formation
  - Feature scoping
  - Sales pitch
- 
- Evolve product
  - Maintain business relations
  - Develop Rumble strategy
- 
- Complete product
  - High-volume sales
  - Optimize Rumble strategy



**LUNDS**  
UNIVERSITET

# Lab 3 - Uttalande från Reguljära Enheten

---

## 1. Workarounds för buggen

- Nytt VM-argument: -DNOSECURITY=true
- Kopiera in robotarna i Robocode/robots
- Nedgradera till Robocode v1.9.3.2

## 2. Inga förväntningar på testkod för systemtest i Beta-releasen

- Det sparar vi till Final-release så att alla kan slutföra Lab 3

## 3. Valfritt labbtillfälle under omtentaperioden

- Info följer

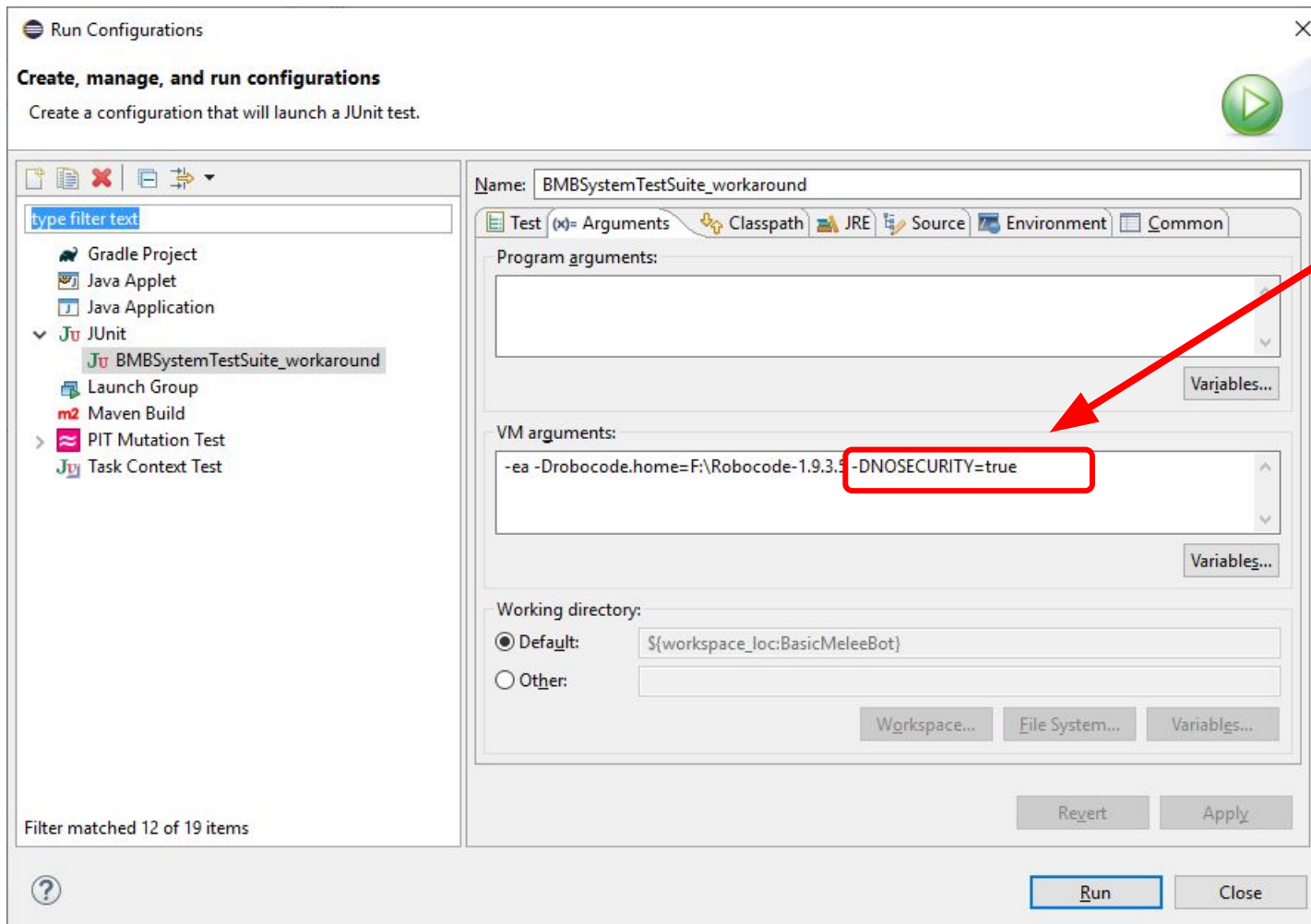
## 4. Vi rapporterar buggen till Robocode-projektet

- Vi har tillgång till väldigt rik information kring buggen!



LUNDS  
UNIVERSITET

# Lab 3 - Uttalande från Reguljära Enheten



LUNDS  
UNIVERSITET

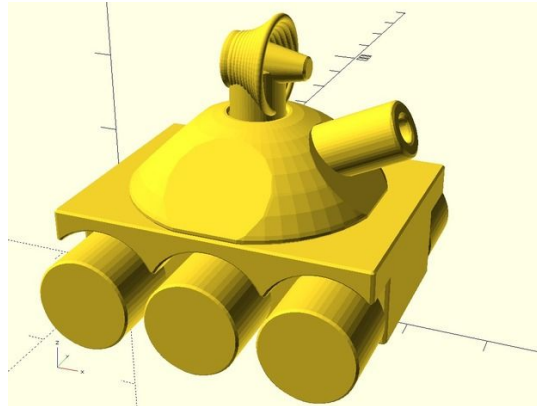


- Verkt yg f r kvalitetss kring (Lab 4)
- Granskningsm te SRS ( 4)

- Aff rsrelation best llare-leverant r

Engineering

Monetizing



Strategizing

- Fundera, fundera, fundera...

Detta  r p  g ng:

# Förtydliganden / Anvisningar

---

Får kod från labbarna användas i projekten?

- Ja, absolut.

Vår leverantörs sales engineer svarar inte på mail!

- Dåligt av försäljningsingenjören! Kunden är kung.
- Kommunikation med kunden är nyckeln till lyckad produktutveckling - missa inte den chansen.

Får man ändra på kravspecifikationen?

- Till viss del är detta oundvikligt!
- Detta ska ske i samråd med er beställare (och handledare). Kan ni argumentera väl för er sak?
- Dokumentera ändringshistorik noga. Uppdatera inte identifierare på enskilda krav, även om det uppstår “luckor”
- Att lägga till krav är mindre känsligt än att ta bort.



LUNDS  
UNIVERSITET

# L5: Onsdag kl. 23.59

---

## Beta Release - Minimum Viable Product

- SRS v0.9 (detaljerade kraven specificerade)
- STS v0.9
- Lean canvas (uppdaterad vid behov)
- Robot som jar-fil
  - Med källkod (+enhetstester, klassdiagram, Javadoc)
  - Utan källkod

ARTEFAKTER I RÖTT ÄVEN TILL KUND



LUNDS  
UNIVERSITET