LUNDS UNIVERSITET

Föreläsning 3: Test II, Design II, Robotmässa

Programvaruutveckling - Metodik 2019 | Markus Borg

Agenda F3

Programvarutestning - del 2

- Black-box testning
- Inför Lab 3

Programvarudesign - del 2

- Arkitekturdesign

Robotmässa

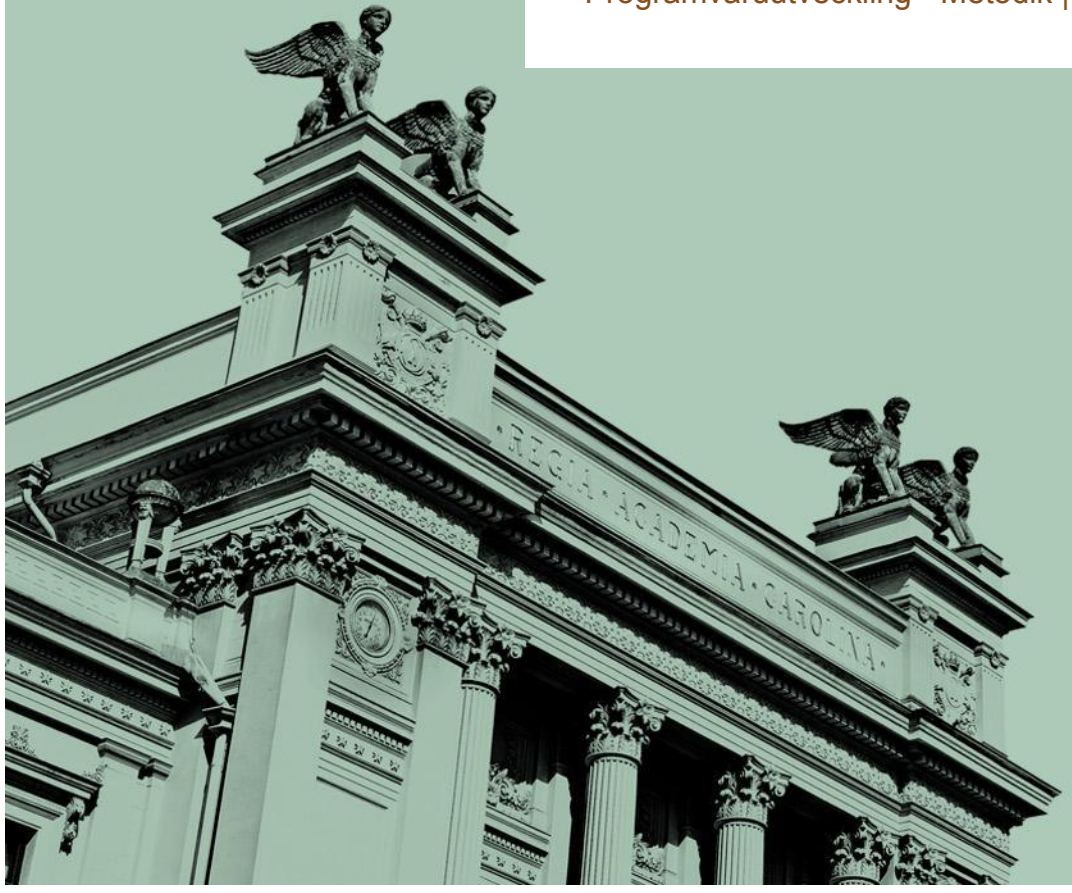
- Status i projekten
- Videovisning

LUNDS UNIVERSITET

LUNDS UNIVERSITET

Test II

Programvaruutveckling - Metodik | Markus Borg



Black-box vs. White-box

White-box

- Kräver tillgång till koden
- Testar utfall och inre funktion
  - täcker vi raderna?
  - täcker vi vägarna?

Black-box

- Programmet ses som en ”svart låda” och man utnyttjar inte någon kunskap om koden i samband med definition av testfall
- Kravspecifikationen används för att ta fram testfall
- Testar utfall/resultat

LUNDS UNIVERSITET

Black-box-testning

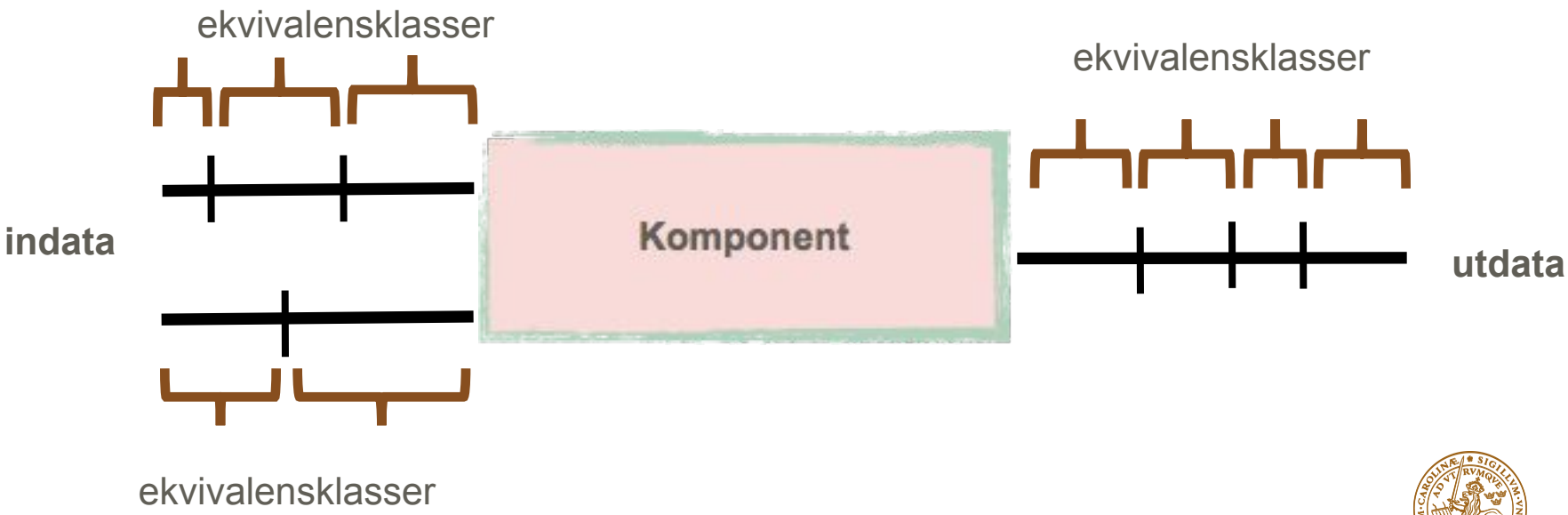
Några vanliga tekniker för testdesign

- Ekvivalenspartitionering
- Gränsvärdestestning
- Stresstestning
- Parvis testning

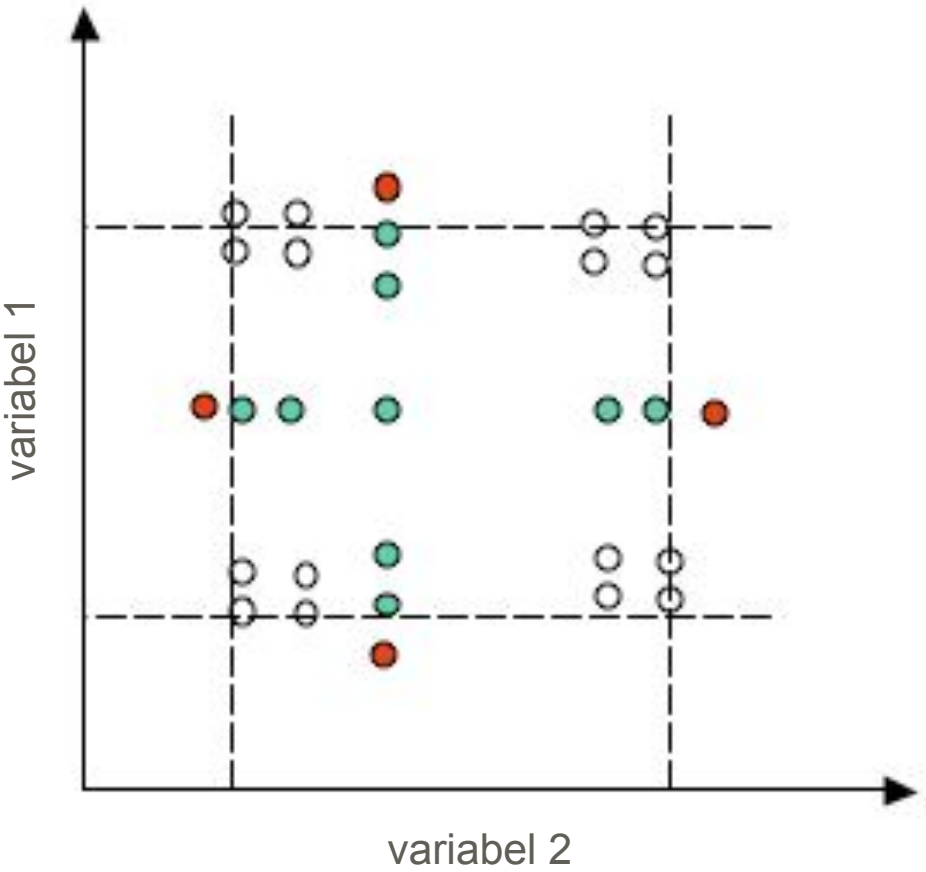
LUNDS UNIVERSITET

Ekvivalenspartitionering

Hitta värden för in och utdata som behandlas på inbördes enhetligt sätt



Gränsvärdestestning
- exempel med två variabler



För n variabler, ett intervall var

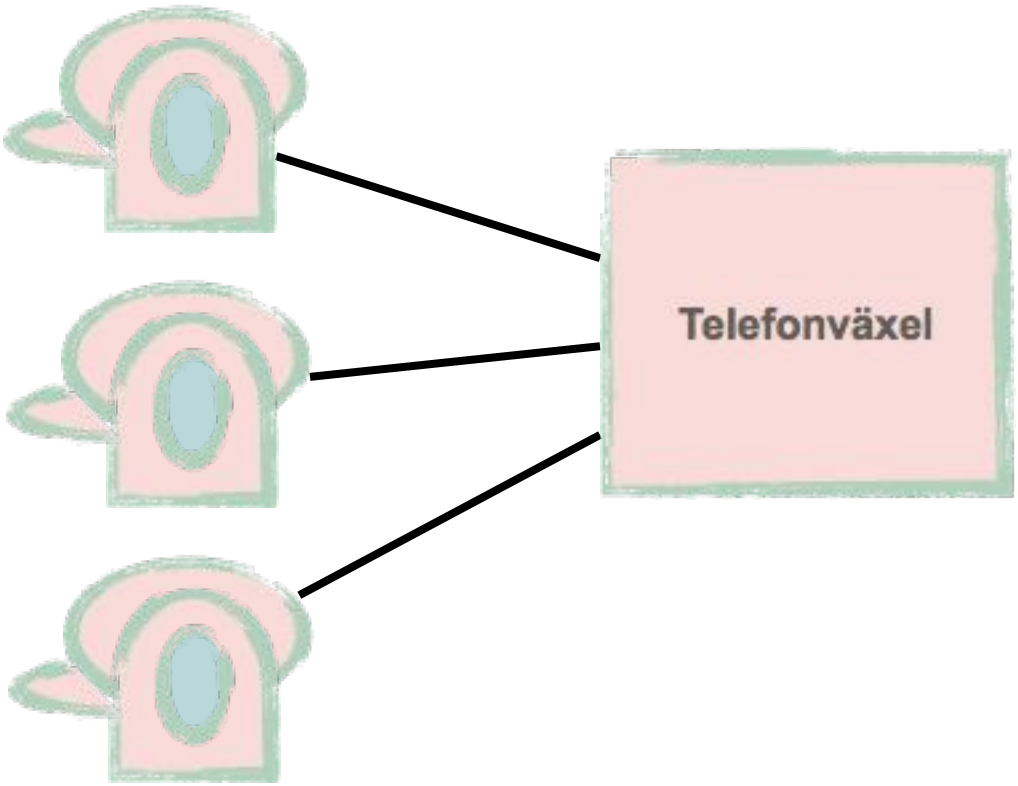
- Vanliga gränsvärden: innanför eller på gränser => 4n+1 testfall

- Robust-test: även utanför gränserna => 6n+1 testfall

- Kritisk systemtestning: även alla kombinationer av gränsfall: 5^n testfall

Stresstestning

Kontrollera vad som händer vid hög belastning, t ex:



telefoner > max
samtidiga samtal > max
…

ERROR: ArrayOutOfBounds ?

Parvis testning

En strategi för testdesign som reducerar antalet testfall

- Test bör undersöka kombinationer av parametrar
- Antalet möjliga testfall blir fort väldigt stort
  - » 5 olika operativsystem
  - » 10 olika versioner av Java
  - » 15 olika webläsare
  - » …
- Istället för att testa alla kombinationer, testa alla parameterpar
  - » Visat sig vara en effektiv strategi (dvs. hittar stor andel buggar)

5 x 10 x 15 = 750 testfall

LUNDS UNIVERSITET

Parvis testning för systemtest av diskmaskin

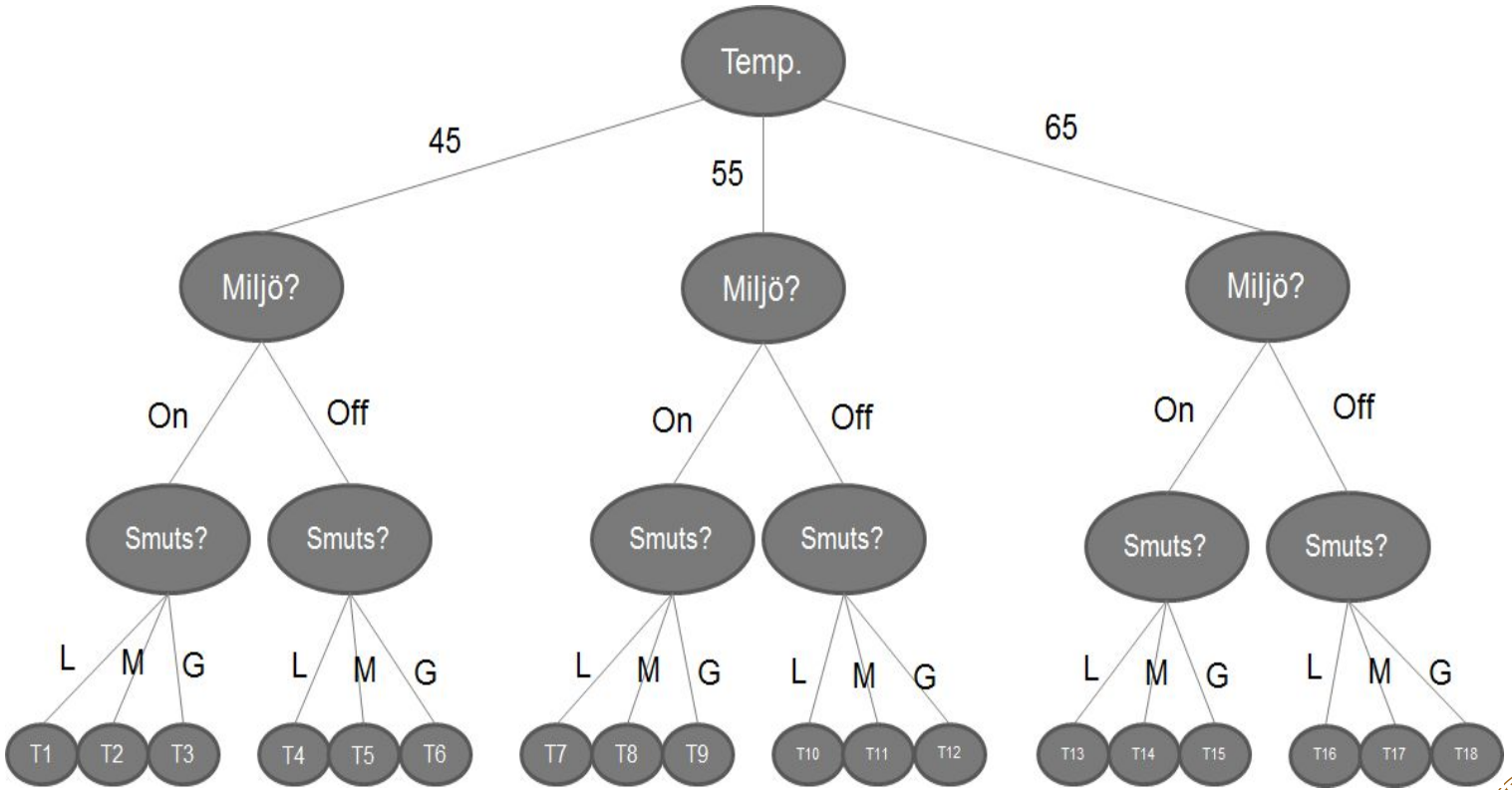
Testparametrar:

Temperatur	(45, 55, 65)
Miljöläge	(on, off)
Nedsmutsningsgrad	(lätt, måttlig, grov)

Utfall:

Diskresultat	(rent, smutsigt)

Samtliga möjliga testfall



⇒ **3 x 2 x 3 = 18 möjliga testfall**

LUNDS UNIVERSITET

Använd parvis testning!

Samtliga kombinationer av parameterpar testas

Färre testfall än vad som krävs för uttömmande testning

- Men en rimlig nivå för att hitta defekter

Generera testdata som uppnår parvis täckning med ett minimalt antal testfall är svårt

- ett kombinatoriskt optimeringsproblem
- verktyg används för att generera testdata

LUNDS UNIVERSITET

Systemtest diskmaskin: Möjliga par

Temperatur-Miljöläge (3 x 2 komb.)

- 45-on, 45-off
- 55-on, 55-off
- 65-on, 65-off

Temperatur-Nedsmutsningsgrad (3 x 3 komb.)

- …

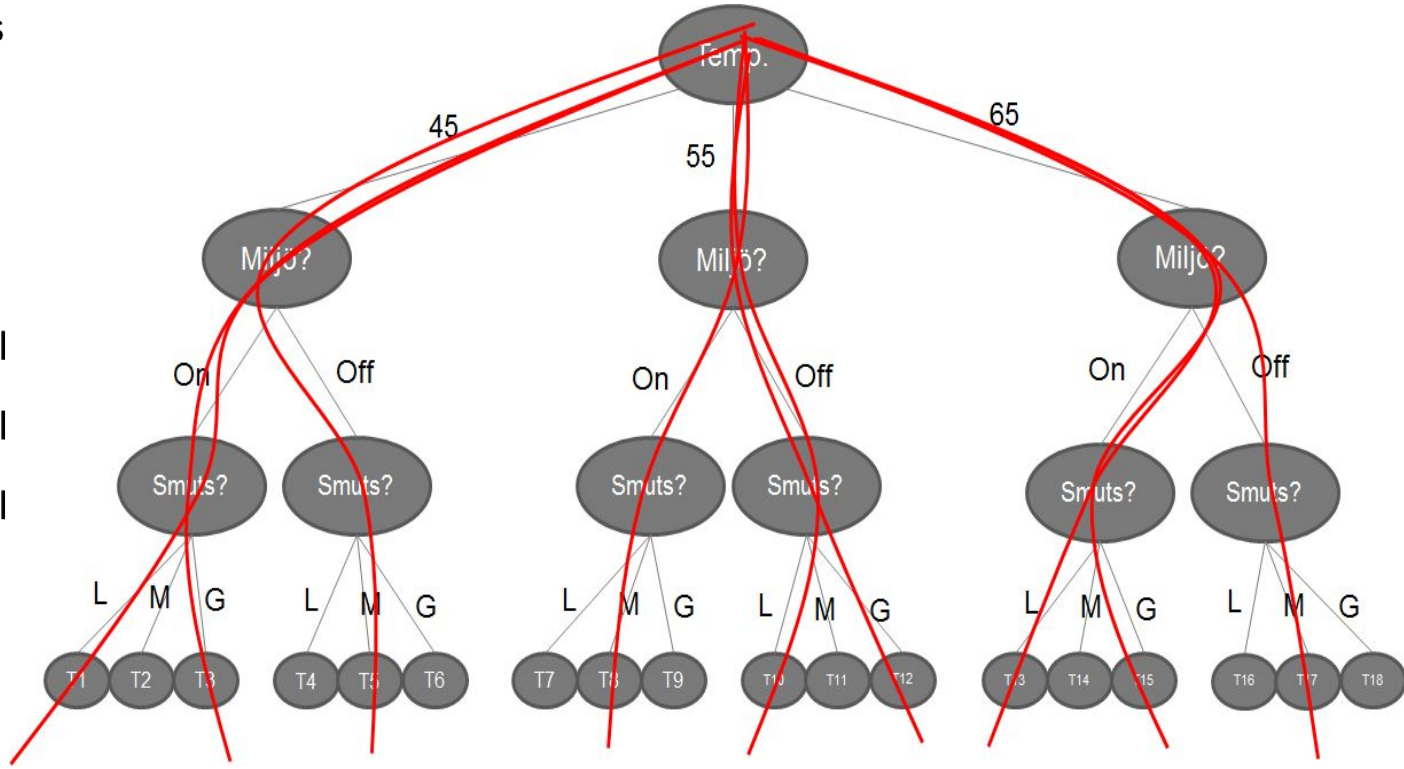
Miljöläge-Nedsmutsningsgrad (2 x 3 komb.)

- …

LUNDS UNIVERSITET

Lösning med 9 testfall => halverat testbehov

	Temp.	Miljö.	Smuts
T1	45	On	Lätt
T2	55	Off	Lätt
T3	65	On	Lätt
T4	45	Off	Medel
T5	55	On	Medel
T6	65	Off	Medel
T7	45	On	Grov
T8	55	Off	Grov
T9	65	On	Grov



→ **9 testfall räcker för att testa alla parameterpar**

Genomföra, dokumentera och rapportera test

I simulerad miljö

- Endast i utvecklingsmiljön (SW)

I mer verklig miljö

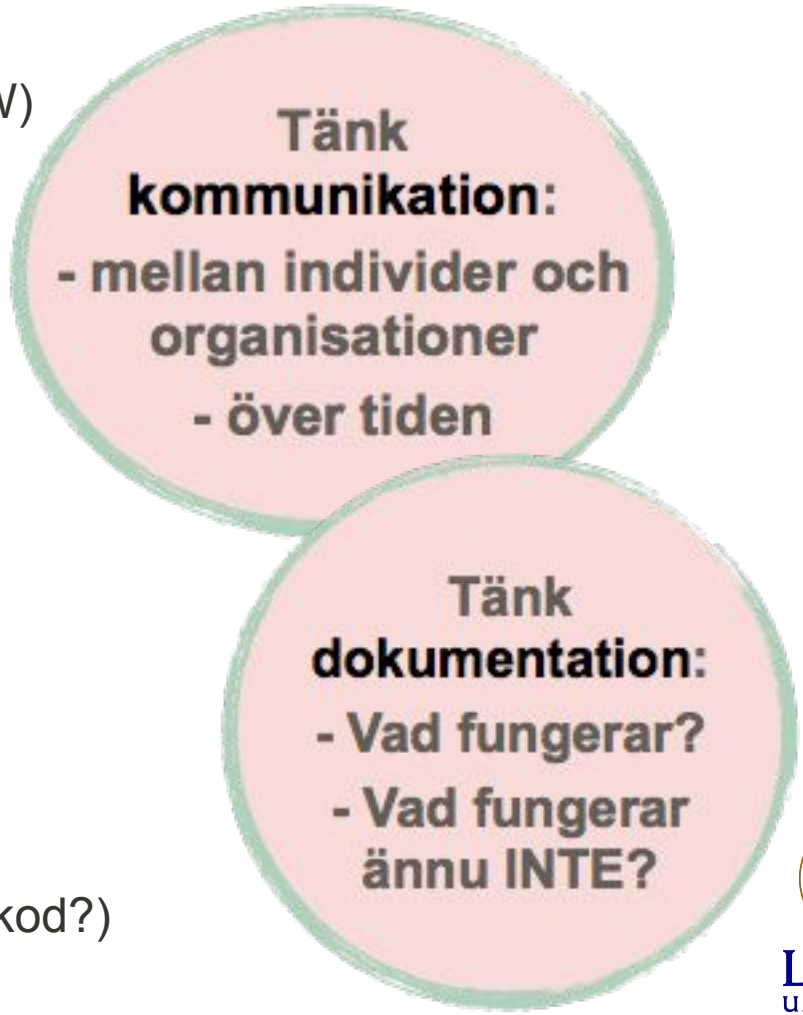
- I testuppsättning (SW/HW)

I verklig miljö

- I äkta (SW/HW)
  med äkta användare

Alla fel rapporteras/registreras

- Testfall
- Resultat
- Feltyp
- Allvarlighet
- Ev. ytterligare beskrivning (felkod?)



LUNDS UNIVERSITET

Slutord - Test

Testning ofta den enskilt dyraste aktiviteten i ett utvecklingsprojekt

När är testningen färdig?

Finns inga garantier att alla defekter är hittade!

- Man kan alltid testa mer…

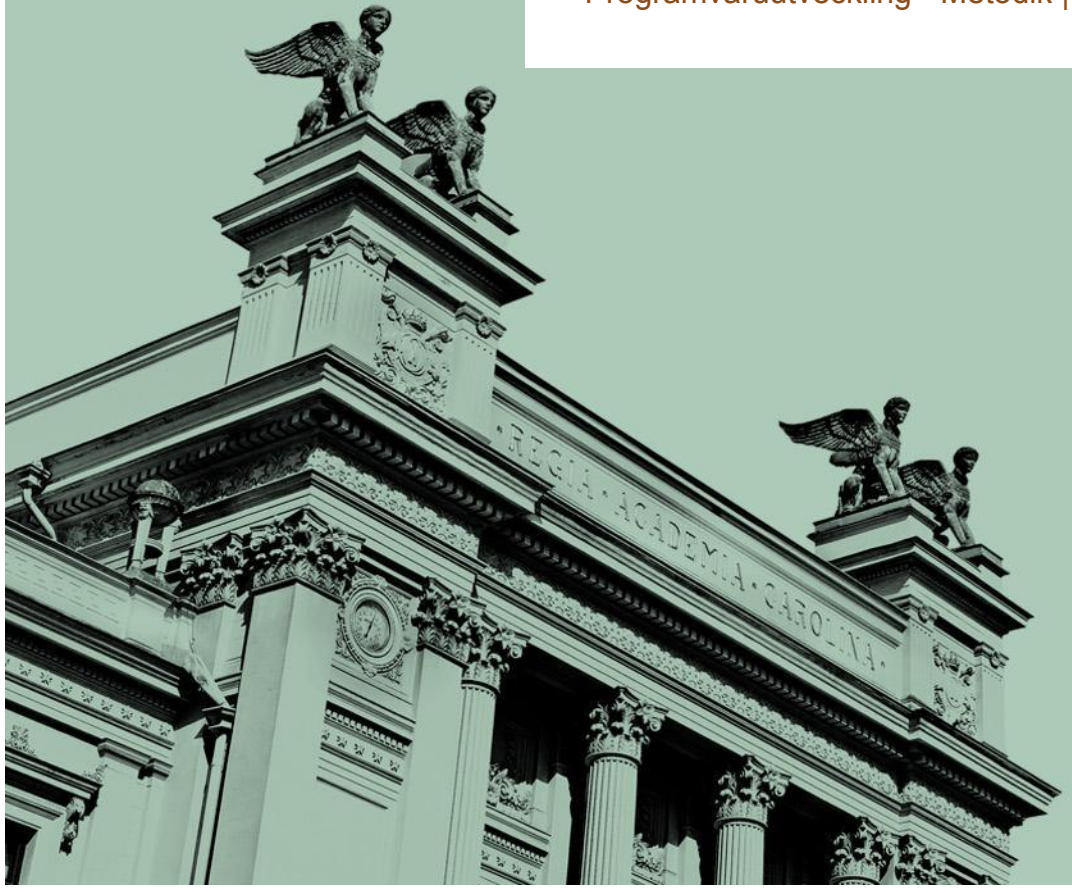
- Slösade resurser
- Försenad release
- Ökade kostnader

- Defekter finns kvar
- Missnöjda användare
- Dyr support
- Dyrt underhåll

LUNDS UNIVERSITET

Inför Laboration 3

Programvaruutveckling - Metodik | Markus Borg



Testautomation för enheter och system

Enhetstester

Skrivs för varje klass som ett skyddsnät - utan test ingen tillit!

- Objekt och mockobjekt skapas, men hela systemet exekverar ej
- Testar att logiken är korrekt implementerad
- Används för regressionstestning
  - Exekveras kontinuerligt (varje commit) - **alltid automatiska**
  - Detekterar fel tidigt
  - Mäts med kodtäckningsmått

Systemtester

Testar att kraven är uppfyllda

- Hela systemet exekverar
- Testar beteende och kvalitet
- Oftast manuella, men **automation välkomnas**

LUNDS UNIVERSITET

Lab 3 - Automatiserad systemtestning

Lab 2

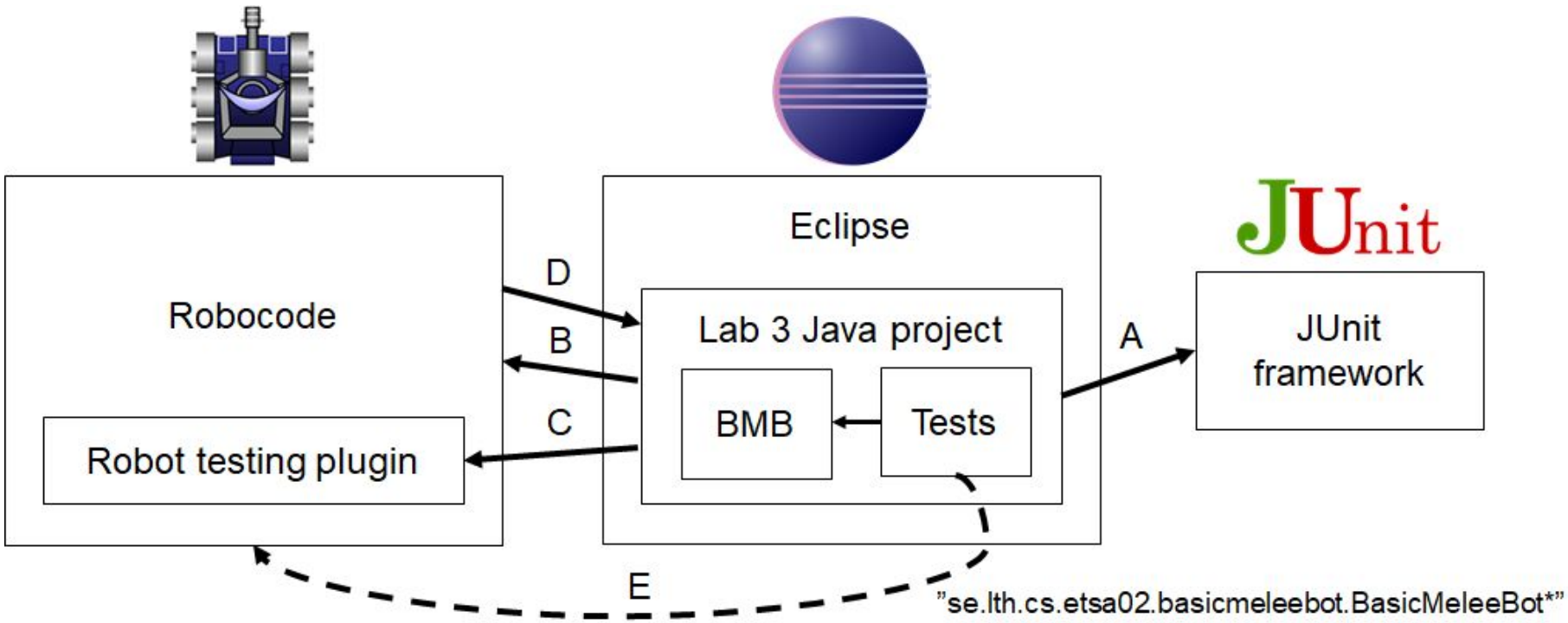
- “Mekanisk” refaktorisering (Copy-Paste)
- Låg nivå av kreativitet
- Färdiga enhetstester som skyddsnät

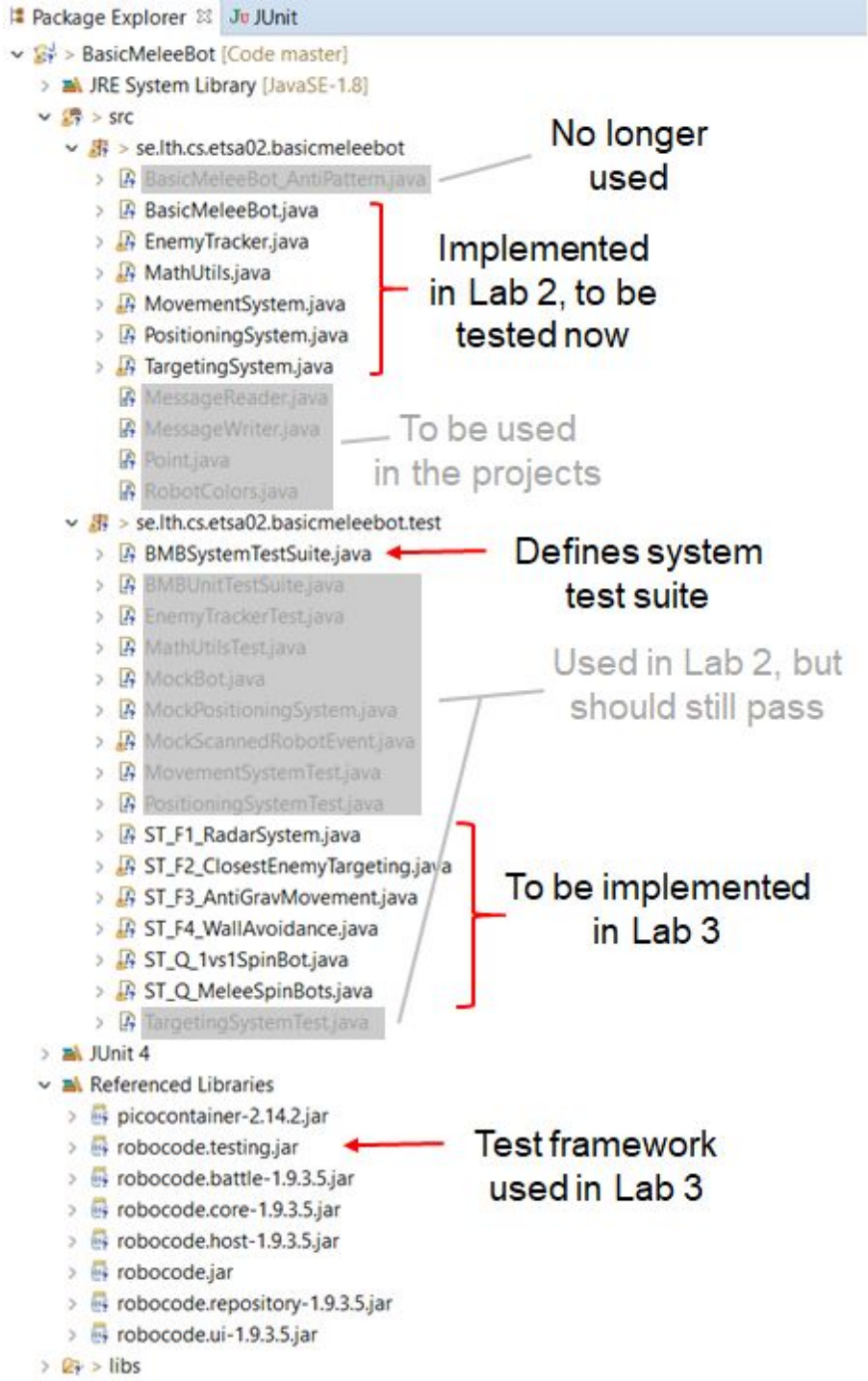
Lab 3

- Utgå från den färdiga implementation från Lab 2
- Fokus på testdesign av automatiska systemtestfall
- Väldigt kreativt

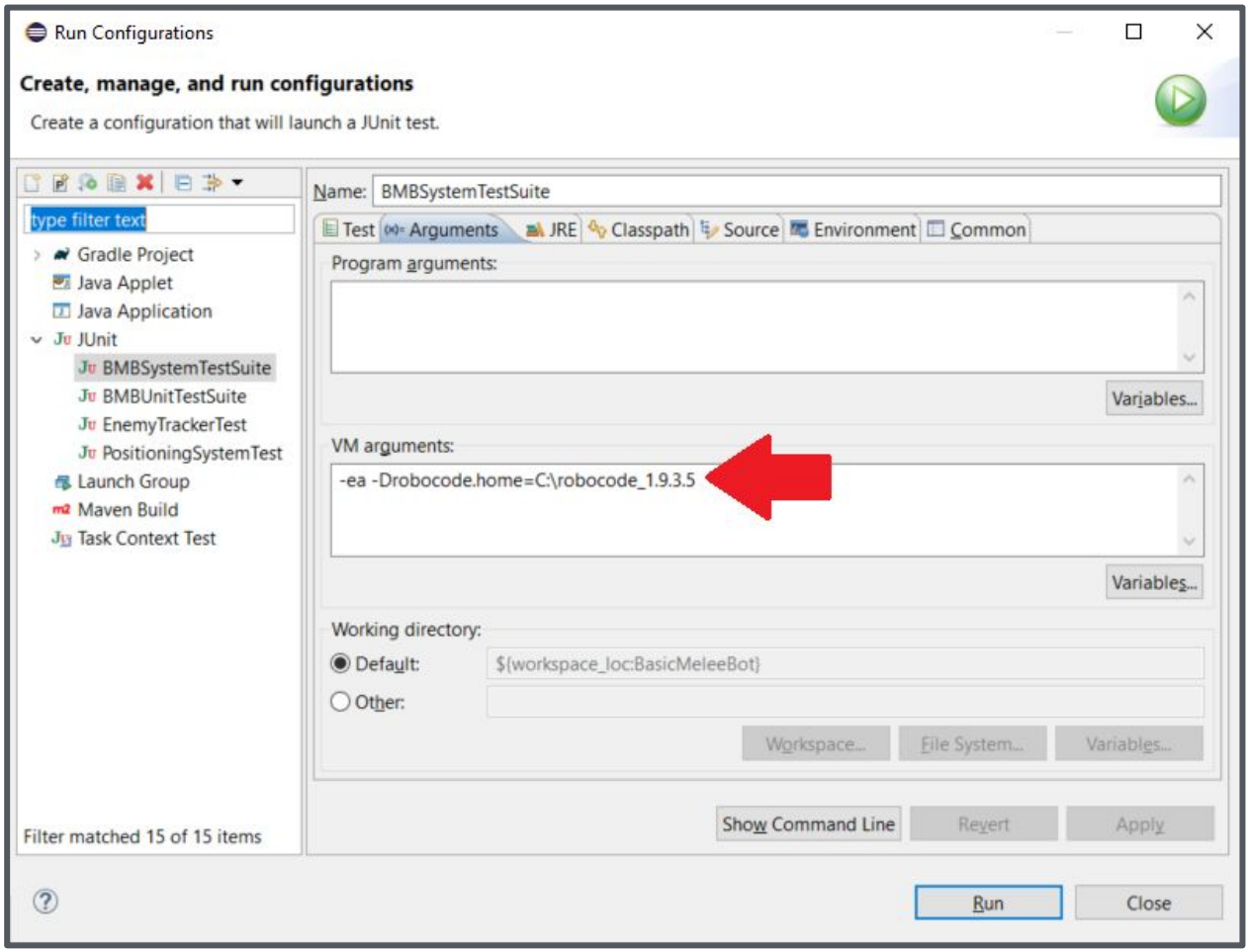
LUNDS UNIVERSITET

Lab 3 - Översikt




Package Explorer
JUnit
BasicMeleeBot [Code master]
JRE System Library [JavaSE-1.8]
src
se.lth.cs.etsa02.basicmeleebot
BasicMeleeBot_AntiPattern.java
BasicMeleeBot.java
EnemyTracker.java
MathUtils.java
MovementSystem.java
PositioningSystem.java
TargetingSystem.java
MessageReader.java
MessageWriter.java
Point.java
RobotColors.java
se.lth.cs.etsa02.basicmeleebot.test
BMBSystemTestSuite.java
BMBUnitTestSuite.java
EnemyTrackerTest.java
MathUtilsTest.java
MockBot.java
MockPositioningSystem.java
MockScannedRobotEvent.java
MovementSystemTest.java
PositioningSystemTest.java
ST_F1_RadarSystem.java
ST_F2_ClosestEnemyTargeting.java
ST_F3_AntiGravMovement.java
ST_F4_WallAvoidance.java
ST_Q_1vs1SpinBot.java
ST_Q_MeleeSpinBots.java
TargetingSystemTest.java
JUnit 4
Referenced Libraries
picocontainer-2.14.2.jar
robocode.testing.jar
robocode.battle-1.9.3.5.jar
robocode.core-1.9.3.5.jar
robocode.host-1.9.3.5.jar
robocode.jar
robocode.repository-1.9.3.5.jar
robocode.ui-1.9.3.5.jar
libs
No longer used
Implemented in Lab 2, to be tested now
To be used in the projects
Defines system test suite
Used in Lab 2, but should still pass
To be implemented in Lab 3
Test framework used in Lab 3

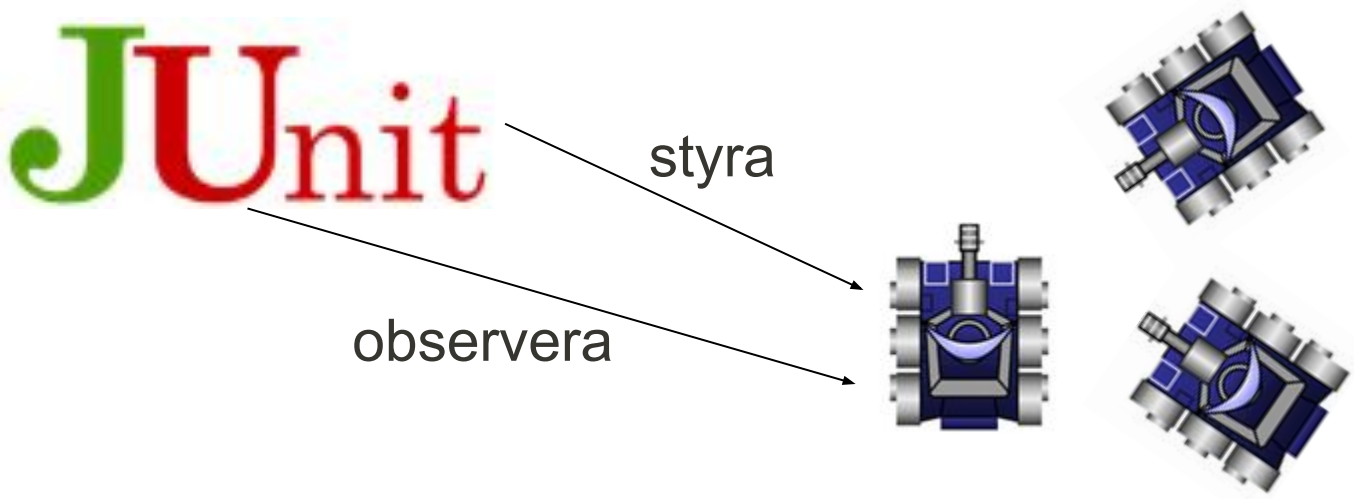
Lab 3 - Ny JUnit Run Configuration



Lab 3 - RobotTestBed

Ramverk Robocode-testning

- Möjliggör test av exekverande robotar
- Exekvera matcher utan att presentera GUI
- Överlagrar Robocode-metoder i BattleAdaptor
  - Styrbarhet: vissa tillstånd kan ändras
  - Observerbarhet: tillstånd kan testas med JUnit assertions



Lab 3 - Designa testfall i överlagrade metoder

Styrbarhet genom överlagrade get-metoder

Konfigurera vilka robotar som ska kämpa (kommaseparerad lista)

```
public String getRobotNames() {
      return "etsa02_lab4.BasicMeleeBot*,sample.SittingDuck";
}
```

Konfigurera robotarnas startposition (notera formateringen!)

```
public String getInitialPositions() {
      return "(25,25,0), (350,300,0), (700,500,0)";
}
public boolean isDeterministic() {
      return true;
}
```

Konfigurera antal rundor som ska utkämpas

```
public int getNumRounds() {
      return 500;
}
```

LUNDS UNIVERSITET

Lab 3 - Designa testfall i överlagrade metoder

Observerbarhet i metoder som tar emot events, dvs. här lägger vi till JUnit assertions

[1] Robocode Battle
void onBattleStarted(BattleStarted event)
    [1-N] Robocode Round
    void onRoundStarted(RoundStartedEvent event)
        [1-N] Robocode Turn
        void onTurnStarted(TurnStartedEvent event)
        void onTurnEnded(TurnEndedEvent event)
    void onRoundEnded(RoundEndedEvent event)
void onBattleCompleted(BattleCompletedEvent event)

Övning 3 - Kravworkshop

I Beta-releasen ska SRS v0.9 innehålla nedbrytning av features till detaljerade krav - som skall kunna verifieras

- Se exempel för Basic Melee Bot i Lab 3

Schemalagd tid utan sal för att ni säkert ska ha hunnit komma igång inför övningen

På övningen

- 45 minuter finslip + 45 min peer review mellan grupperna

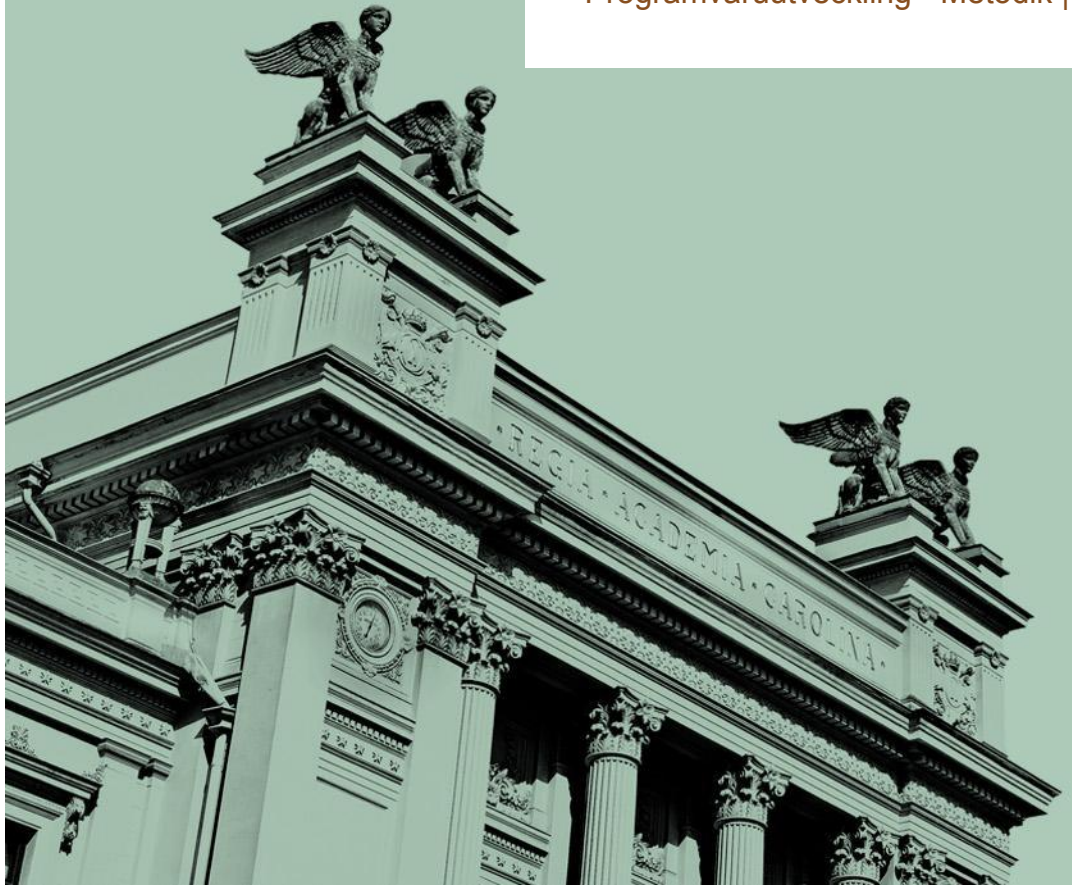
Grupp 9-12 + 15-16 kör gemensam övning i E:1147+E:1149

LUNDS UNIVERSITET

LUNDS UNIVERSITET

Programvarudesign 2

Programvaruutveckling - Metodik | Markus Borg



Programvarudesign - agenda

- F2: Objektorienterad design
- F3: Arkitekturdesign
- (Ej: Interaktionsdesign)

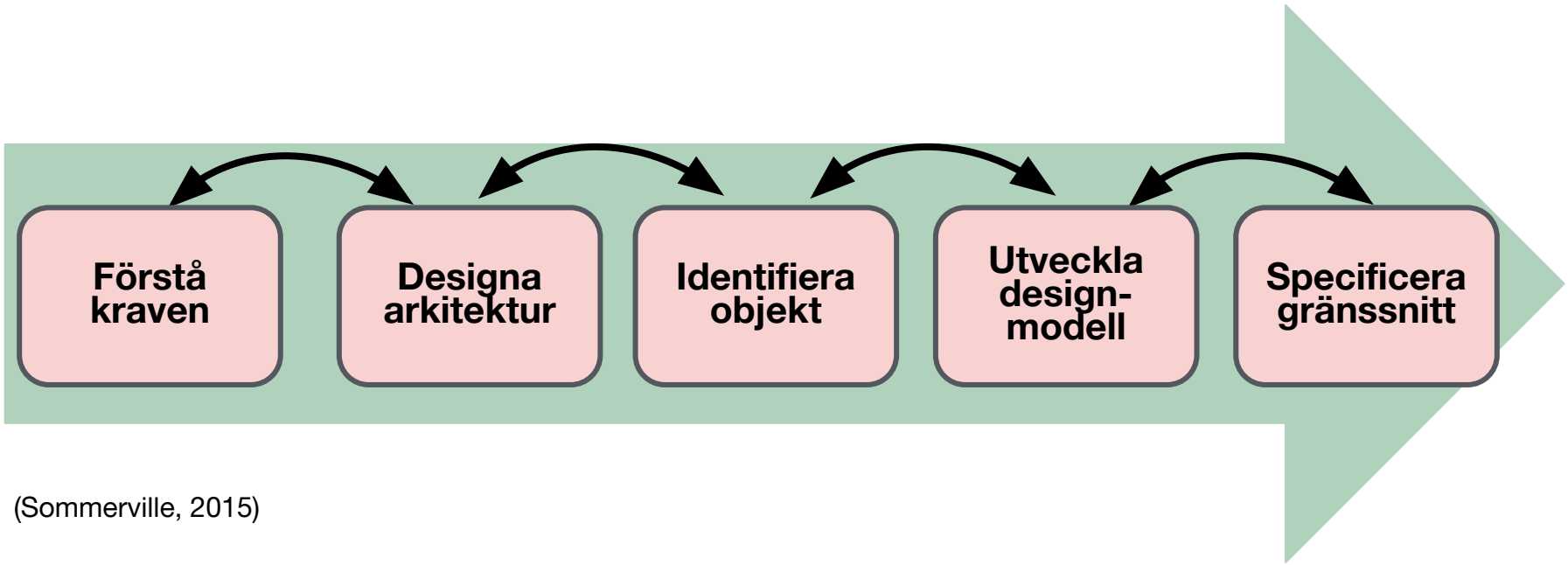
MAMA15 - Interaktionsdesign, grundkurs

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

Modell för design av objektorienterad programvara

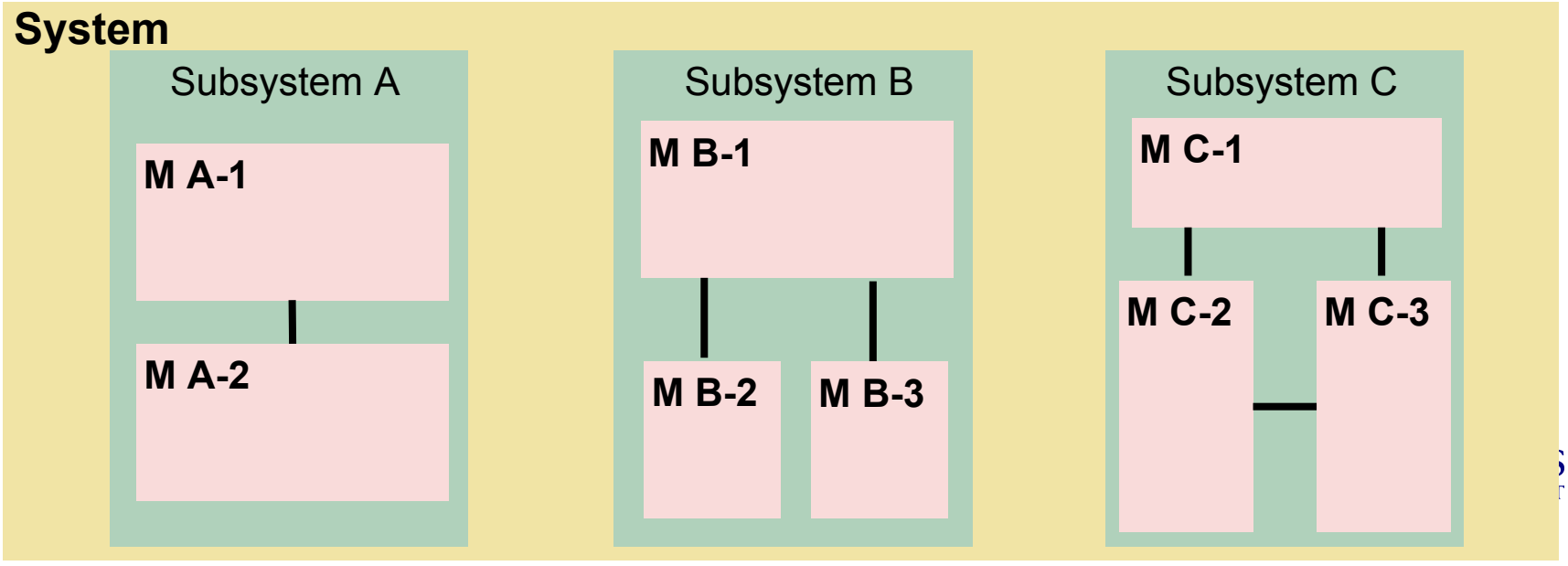


(Sommerville, 2015)

Arkitekturdesign

Nedbrytning av systemets övergripande struktur

- System - helheten
  - ↪Subsystem – enhet som ej beror på andra subsystem
    - ↪Moduler – enhet som verkar ihop med andra moduler
      - ↪(Komponenter – en eller flera klasser)



system = samling komponenter som samverkar för att uppnå ett mål

system-av-system = samling system som samverkar för att uppnå mål utöver summan av de ingående systemen (framträdande egenskaper)

Från militära tillämpningar till civilt bruk (t.ex. trafikmiljö eller industri 4.0)

Syfte med arkitekturdesignen

- Länk mellan kraven och detaljerad design
  - Grov ritning för implementation
- Kommunicerar designbeslut i organisationen
- Grund för systemanalys
  - Säkerhet
  - Prestanda
- Underlättar återanvändning
  - Använda delar i andra system
  - Utveckla produktlinjer

LUND UNIVERSITY

Val av arkitekturdesign

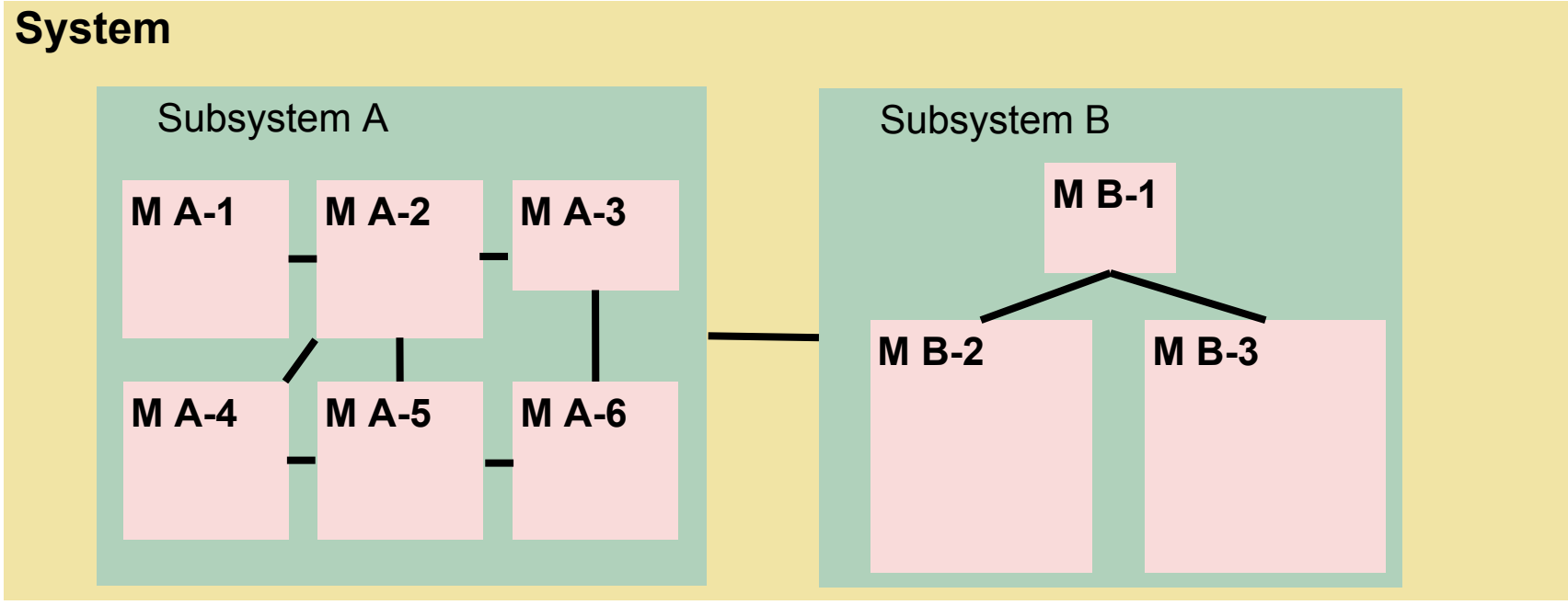
- Förståelse för kontext och intressenter nödvändig för bra beslut
- Kvalitetskraven avgör ofta beslutet
  - Arkitekturellt signifikanta krav
- Vad vill vi uppnå? Motsättningar vanligt!

Övriga faktorer som kan avgöra

- Organisationens tekniska kompetens och erfarenhet
- Återanvändning av tidigare arkitektur
- Standarder som behöver uppfyllas

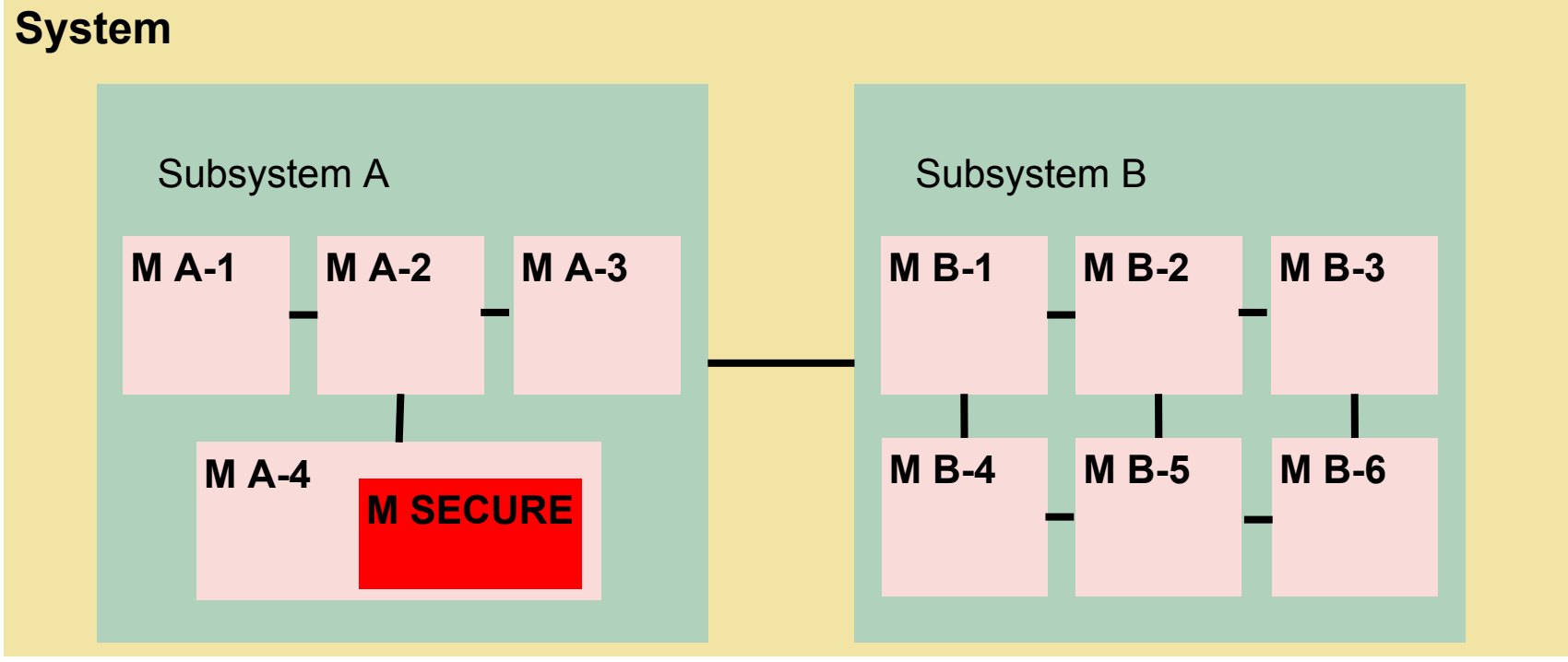
Prestanda?

- Kommunikation är en prestandatjuv!
- Samla tunga beräkningar i moduler som kommunicerar minimalt utåt
- Acceptera att beräkningsmoduler blir stora



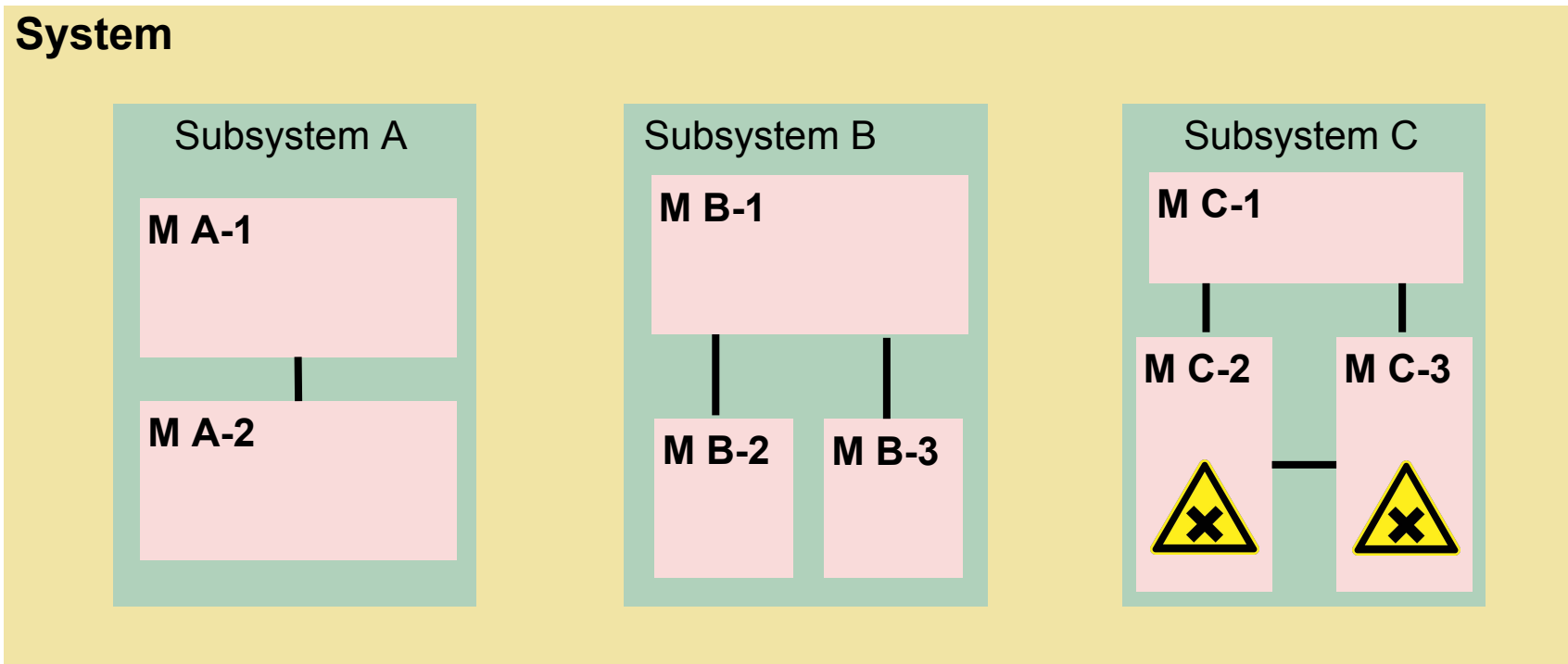
Säkerhet? (security)

- Åtkomstbegränsning viktigt
- Introducera säkerhet i olika lager
- Hantera den känsligaste informationen innerst



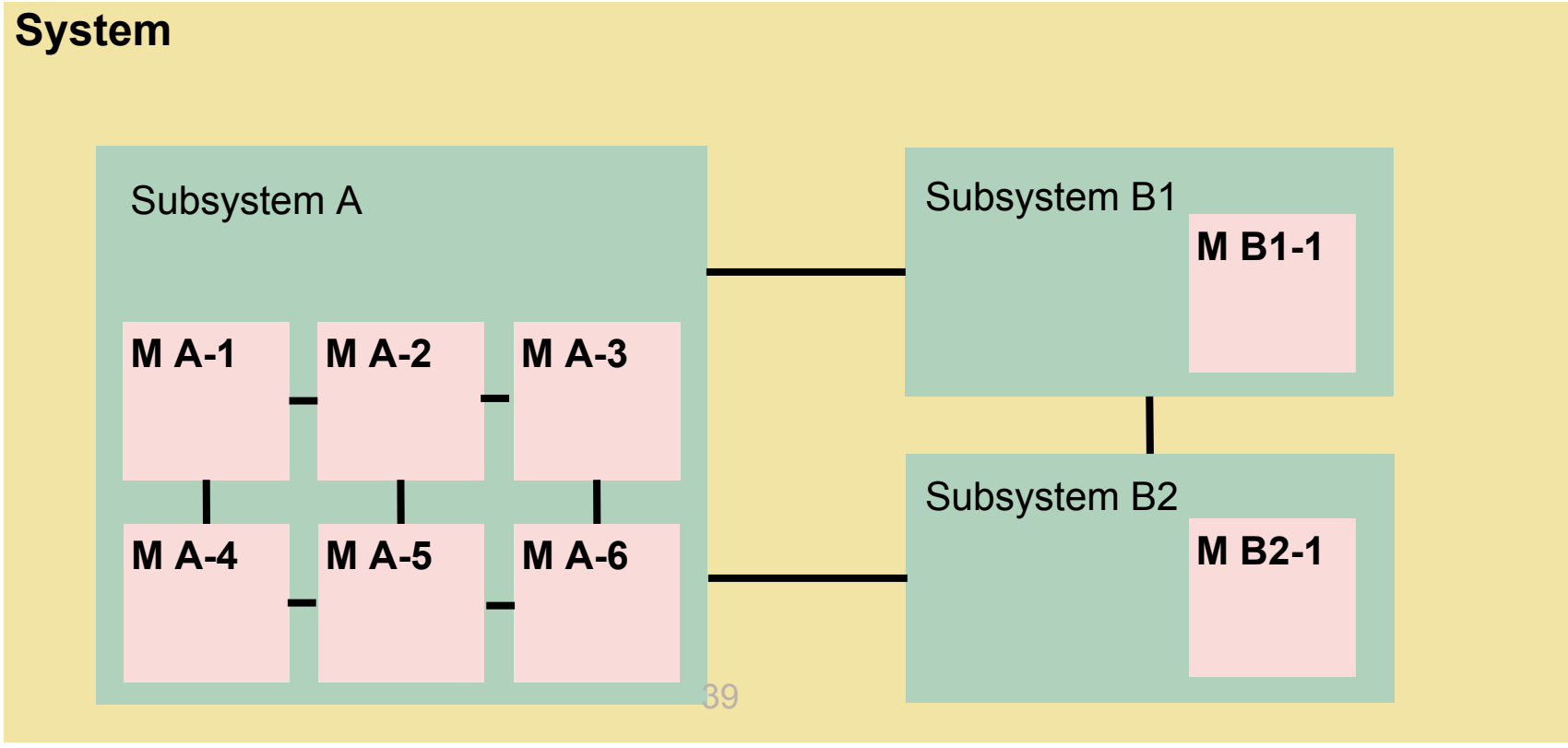
Säkerhet? (safety)

- Att verifiera säkerhetskrav är svårt och dyrt
- Samla alla säkerhetskritiska operationer i separat subsystem



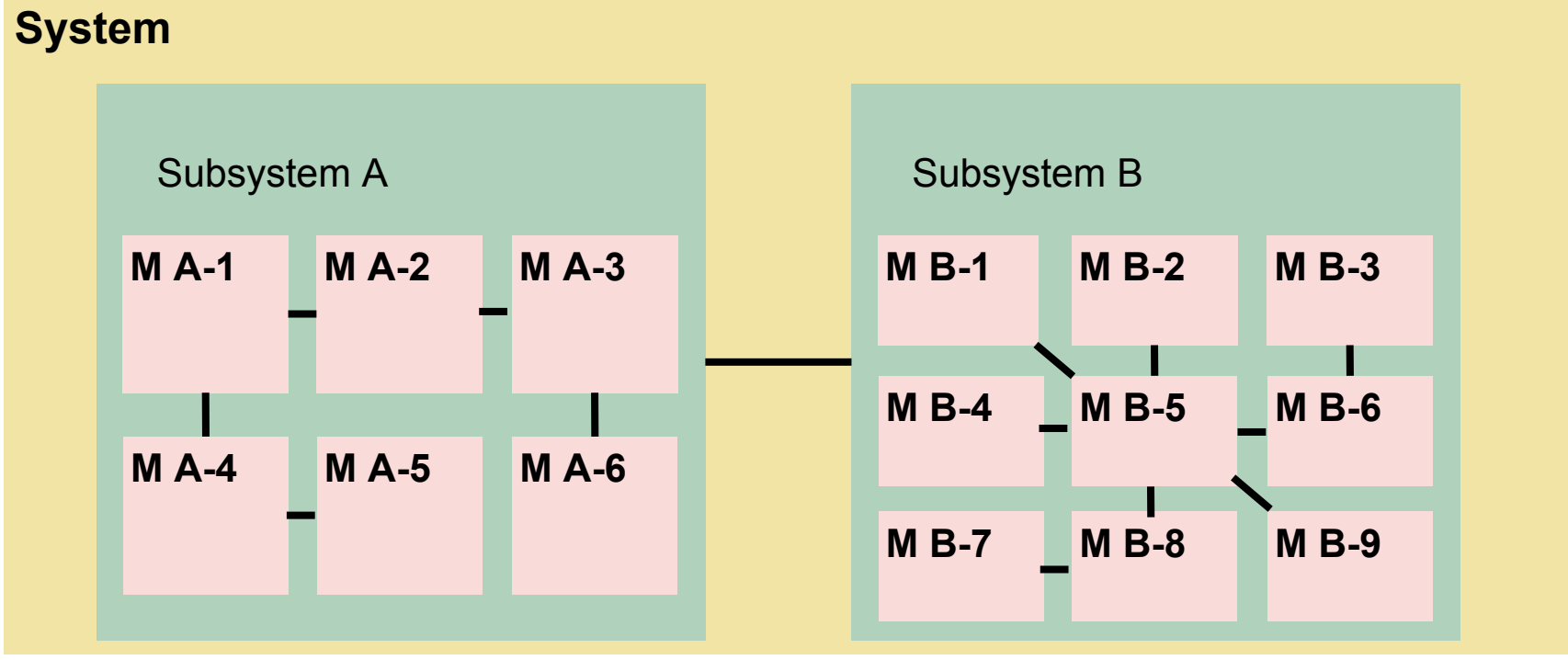
Tillgänglighet?

- Minimera risken att systemet är otillgängligt
- Introducera redundans

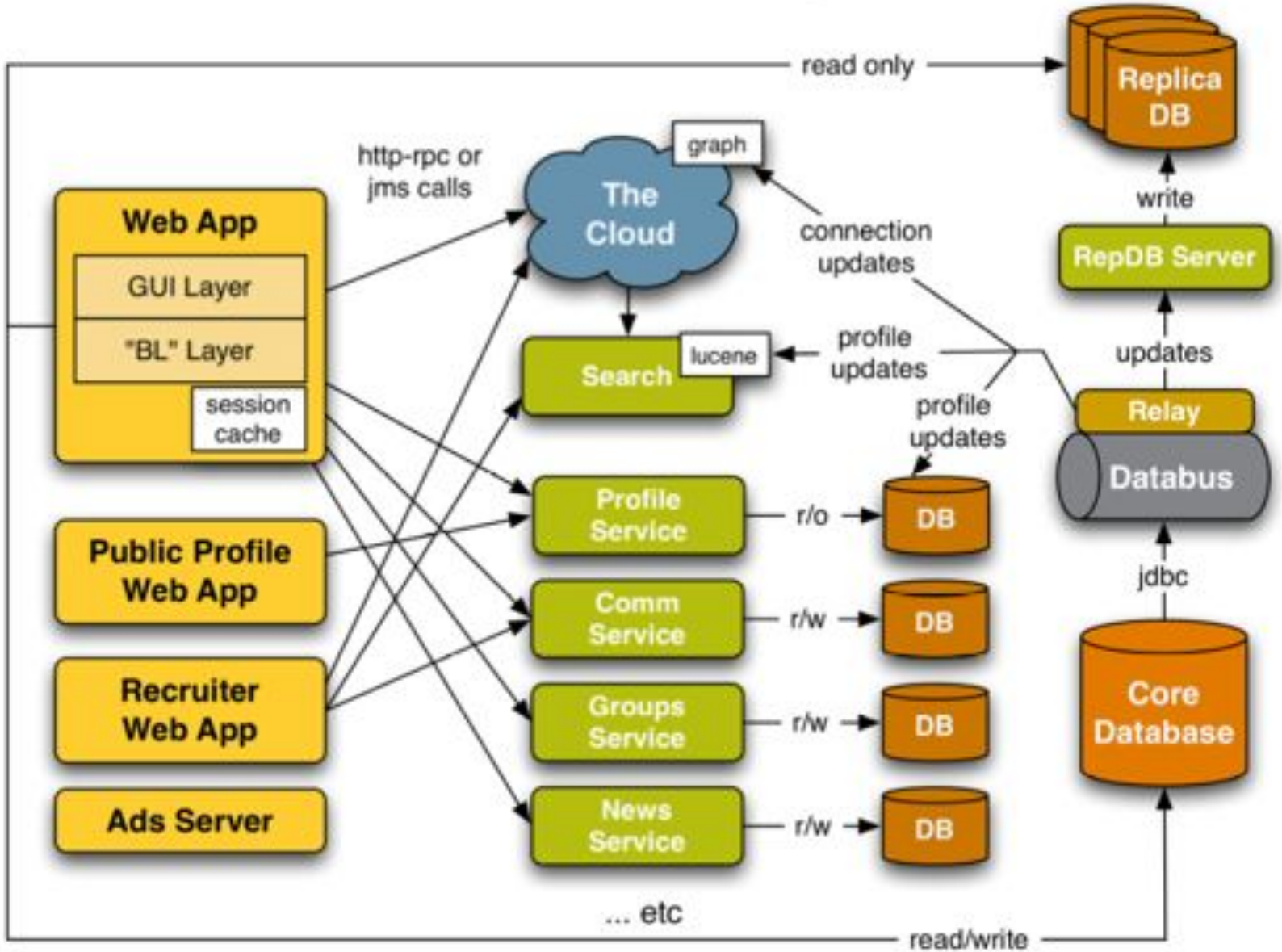


Enkelt underhåll/evolution?

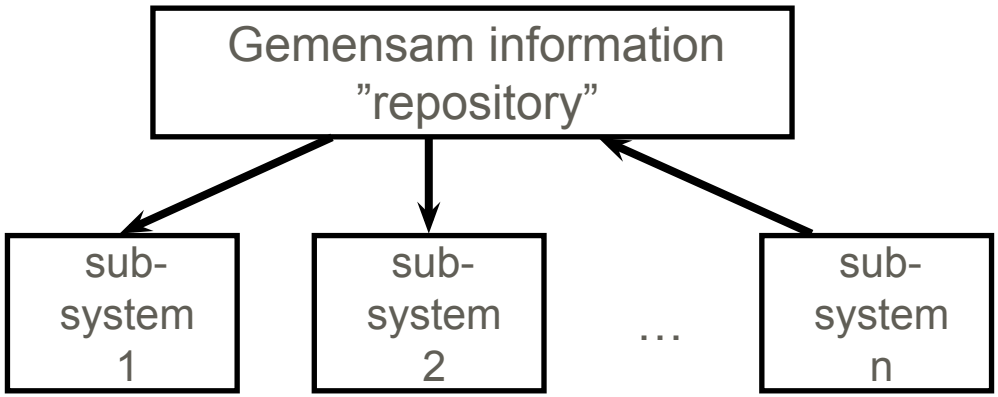
- Fokusera på små oberoende moduler
- Minimal kommunikation gör det lättare att ersätta moduler i framtiden
- Tjänsteorienterad arkitektur (microservices)



LinkedIn: Java-arkitektur



Typexempel – Repository (delad data)



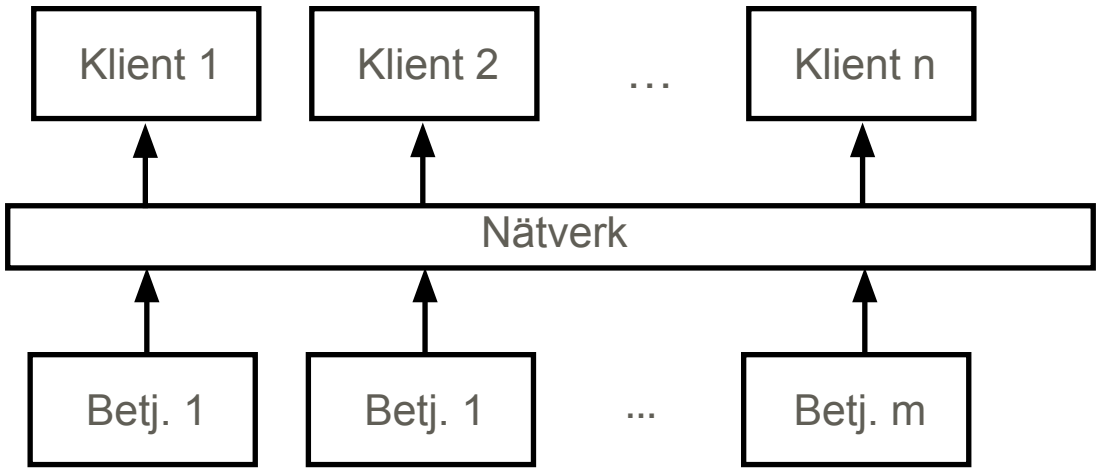
Fördelar:

- Effektivt med mycket data
- Data-producent måste inte veta så mycket om konsument
- Operationer på all data underlättas, t.ex. backup

Svagheter:

- Alla subsystem måste använda samma dataformat
- Vidareutveckling kan vara svårt eftersom mycket bygger på en viss datamodell

Typexempel - Client-server



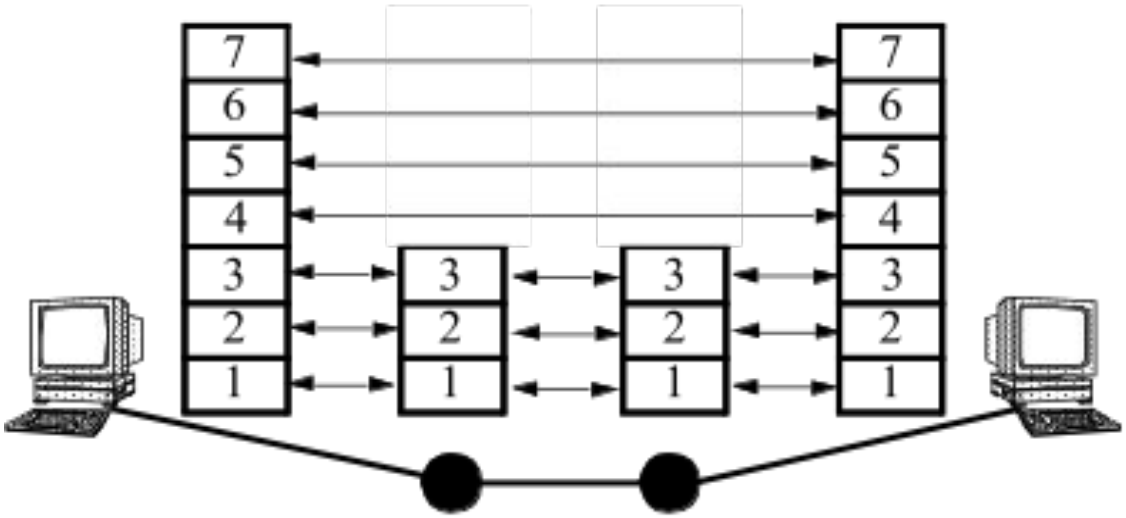
Fördelar:

- Distribuerad arkitektur
- Lätt att lägga till nya klienter och betjänare

Svagheter:

- Uppdatering av klient eller betjänare kan kräva uppdatering av samtliga
- Ingen gemensam datamodell

Typexempel – Abstraktionslager



Varje lager utgör en ”abstrakt maskin” som används av nästa lager

Fördelar:

- Stöd för inkrementell utveckling
- Underlättar portabilitet

Svagheter:

- Kan uppstå beroenden mellan flera lager
- Kan bli sämre prestanda

Programvarudesign - sammanfattning

Design är både en aktivitet och ett resultat

Arkitekturdesign är en övergripande nedbrytning av systemstruktur: System → Subsystem → Moduler

- Val av arkitektur beror på kvalitetskraven
- Exempel: repository, client-server, abstraktionslager

Objektorienterad design beskriver hur komponenter implementeras av klasser

- Beskrivs vanligtvis med UML
- Sträva efter låg koppling och hög sammanhållning

(Interaktionsdesign utanför kursen)

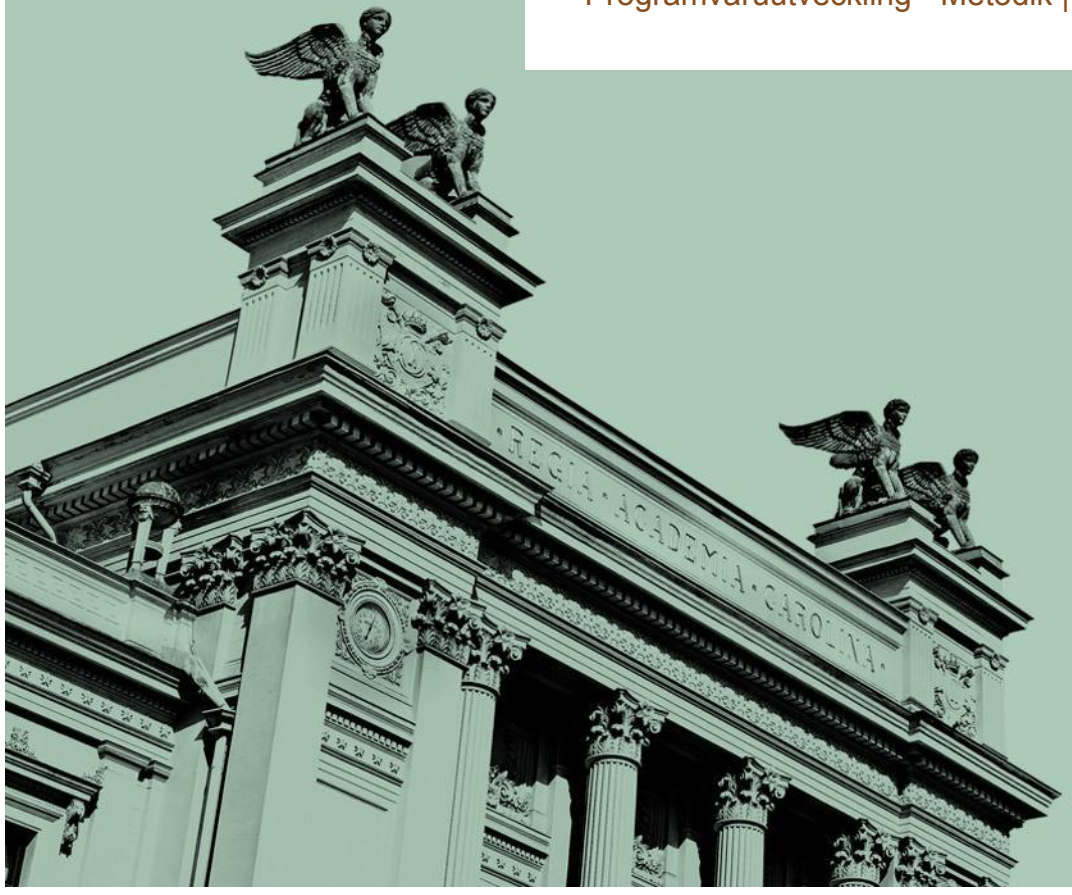
- Grafiska användargränssnitt

LUNDS UNIVERSITET

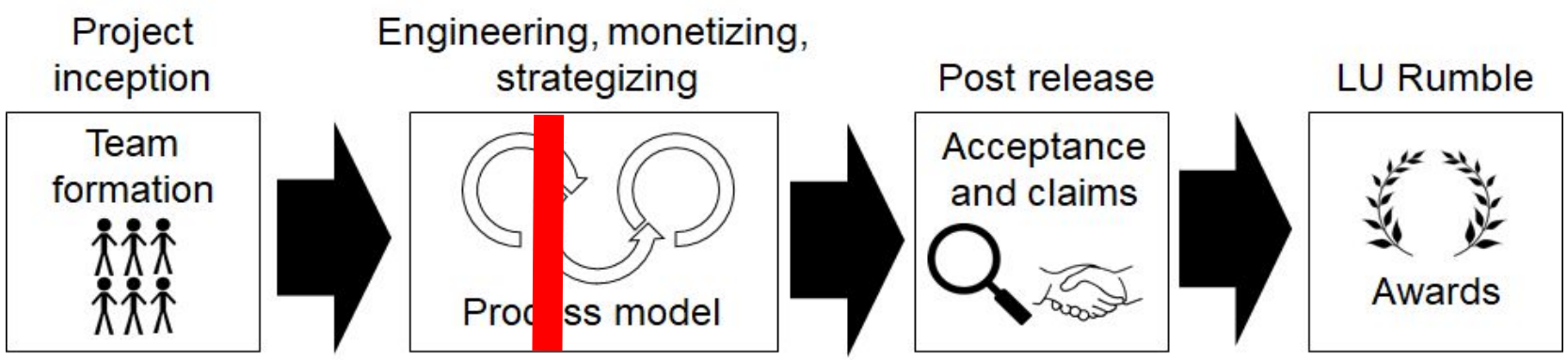
LUNDS UNIVERSITET

Robotmässa

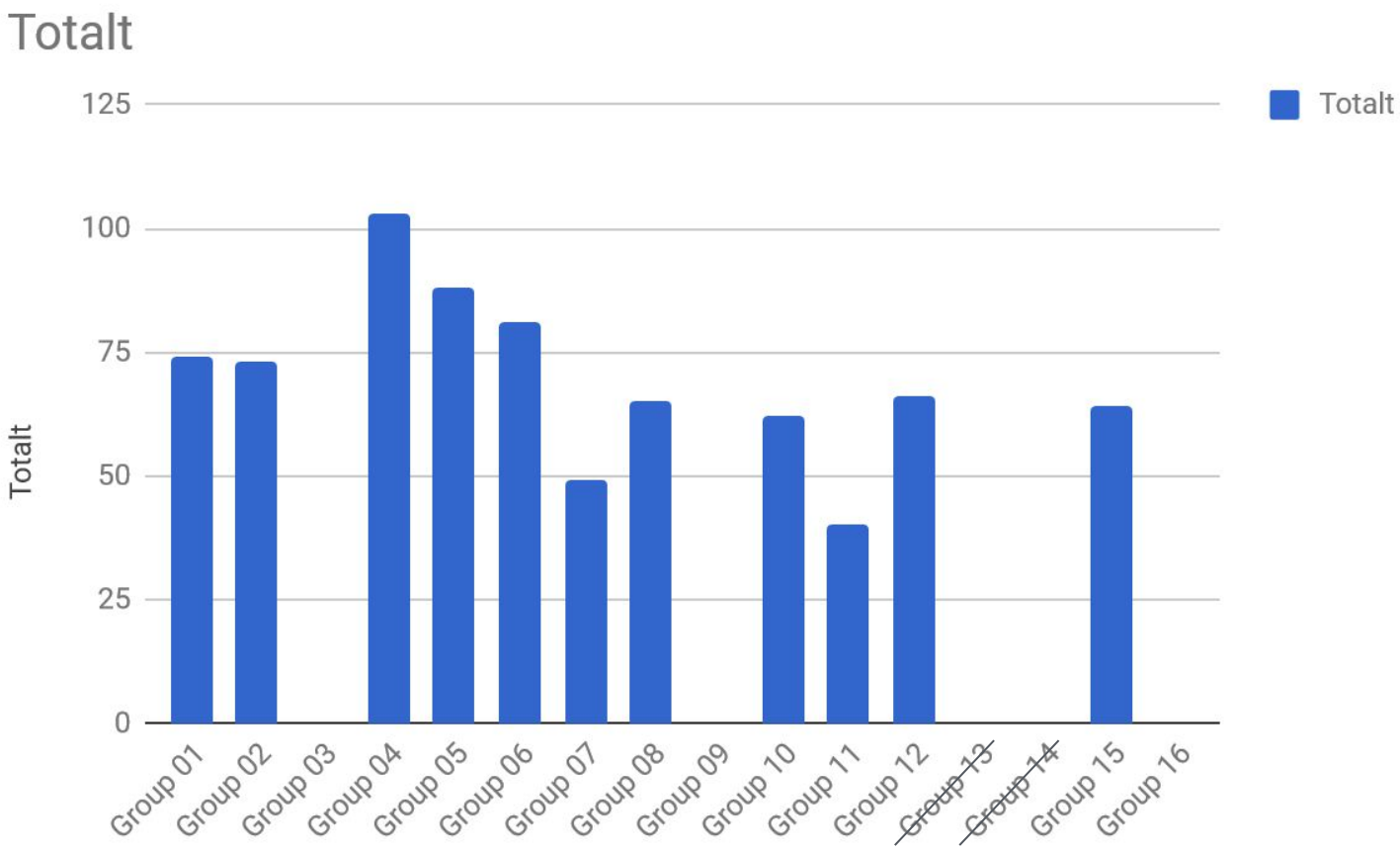
Programvaruutveckling - Metodik | Markus Borg



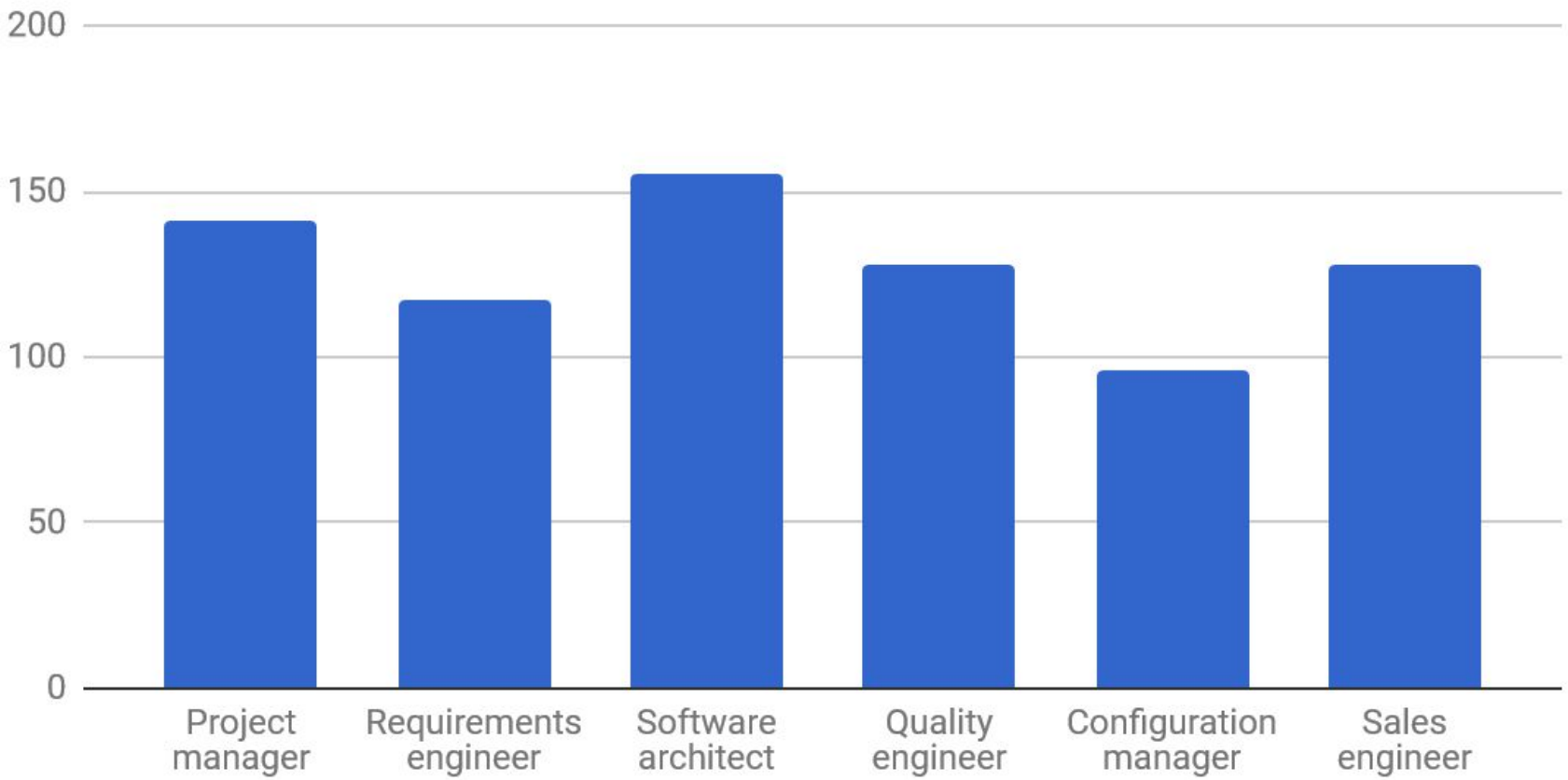
w		Monday						Tuesday						Wednesday						Thursday						Friday						Sa	Su
		8	10	12	13	15	Late	8	10	12	13	15	Late	8	10	12	13	15	Late	8	10	12	13	15	Late	8	10	12	13	15	Late		
13	Activity	25/3		<-	-	-	-	-	-	Lab0	-	-	-	-	-	->	Lab1			Ö1					L1								
	Groups 1-4																Alfa			2116												
	Groups 5-8		F1														Beta			3308												
	Groups 9-12																	Alfa			2116											
	Groups 13-16																	Beta			3308											
14	Activity	1/4															Lab2			Ö2												
	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			Ö3												
	Groups 1-4																Alfa			2116	PW											
	Groups 5-		F3 Fair	L3 = Alpha									L4				Beta			3308	PW											
	Groups 9-1																PW	Alfa			1147 + 1149											
	Groups 13-16																PW	Beta														
16	Activity																		L5 = Beta													
	Groups 1-4																															
	Groups 5-8		F4																	Skärtorsdag						Långfredag						
	Groups 9-12																															
	Groups 13-16																															
17	Activity	Annandag påsk						Exam period						Exam period						Exam period						Exam period							
	Groups 1-16																															
18	Activity	Exam period						Siste april						1 maj						Exam period						Exam period							
	Groups 1-16																															
19	Activity	6/5															Lab4			Ö4											L6 = Final		
	Groups 1-4																Alfa			2116												
	Groups 5-8		F5														Beta			3308												
	Groups 9-12																	Alfa			2116											
	Groups 13-16																	Beta			3308											
20	Activity	13/5																														
	Groups 1-4																															
	Groups 5-8		F6										L7						L8													
	Groups 9-12																															
	Groups 13-16																															
21	Activity	20/5	F7																													
	Groups 1-16																															
22	Activity	27/5																		Kristi himmelsfärdsdag												
	Groups 1-16																															
23	Activity	3/6						Take-home exam																								
	Groups 1-16																															



Tidrapporter - Total tid per grupp



Tidrapporter - Total tid per roll



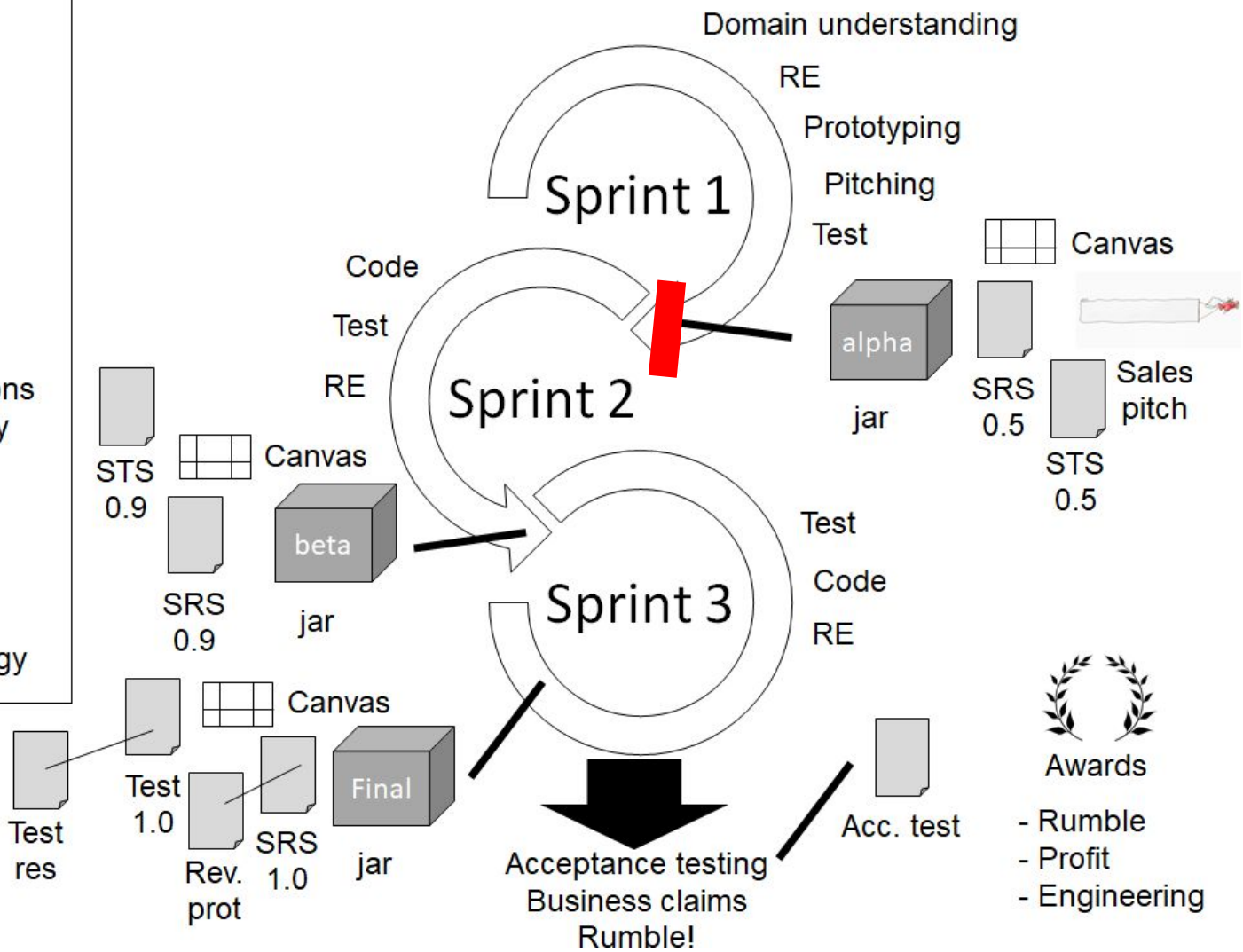
Flickr: vladislav@munich

High-level goals

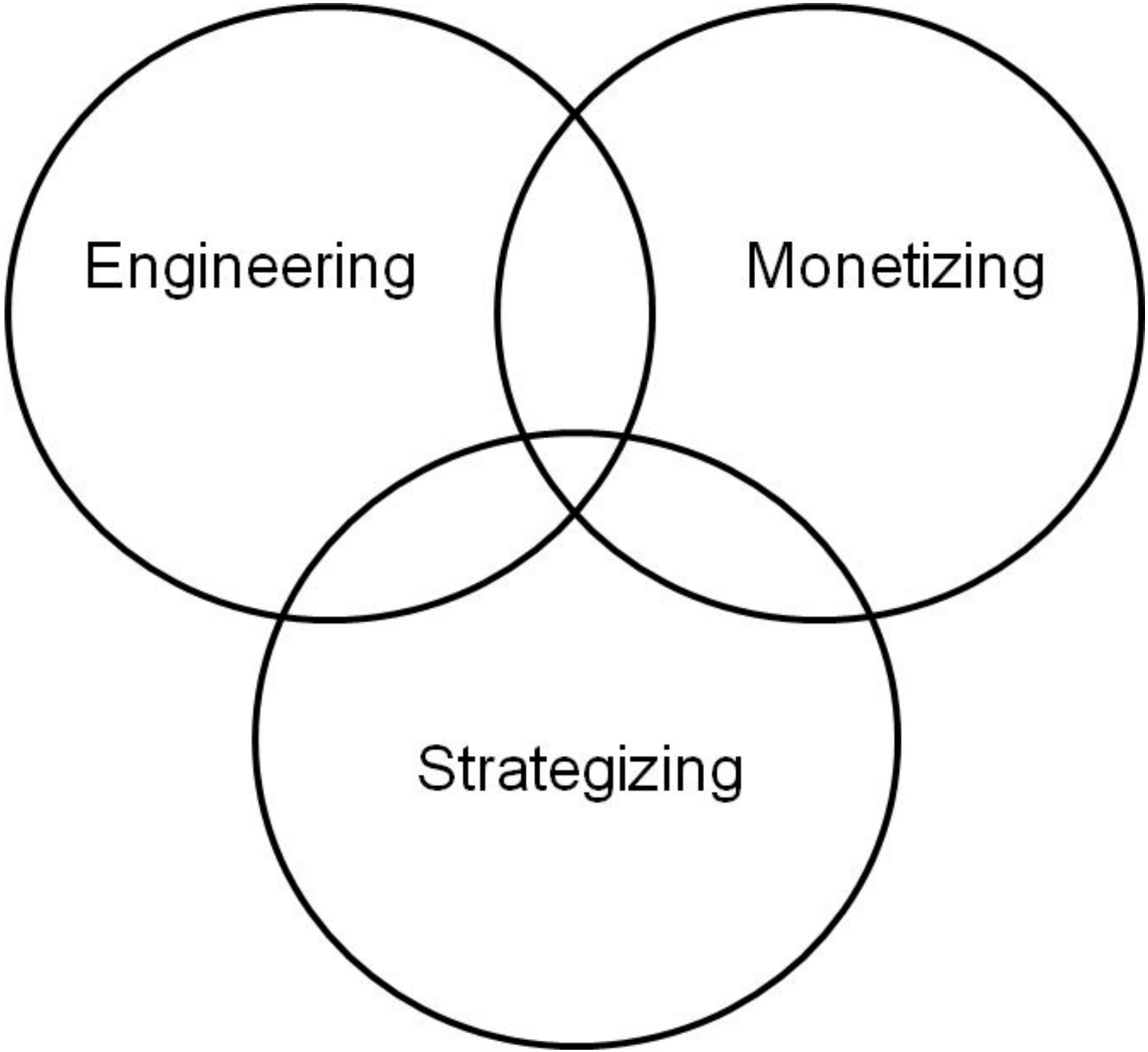
- Team formation
- Feature scoping
- Sales pitch

- Evolve product
- Maintain business relations
- Develop Rumble strategy

- Complete product
- High-volume sales
- Optimize Rumble strategy

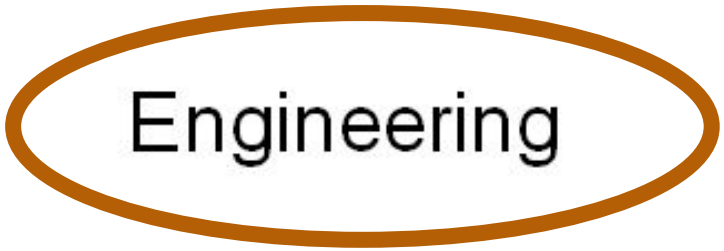


LUNDS UNIVERSITET


Engineering
Monetizing
Strategizing

Detta har hänt:

- Källkodshantering med git (F2)
- Objektorienterad design (Lab 2)
- Enhetstestning (Lab 2)



- Affärsutveckling och features (Ö1)
- Marknadsföring och video pitch (Ö2)

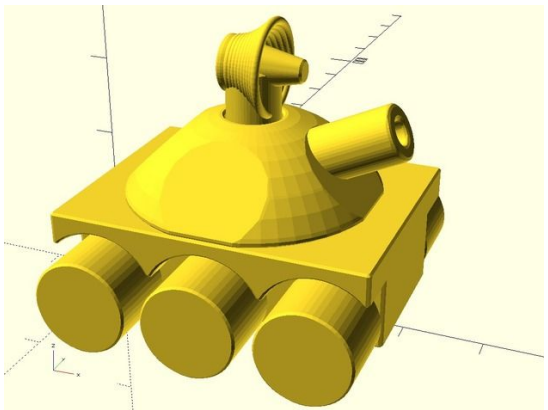
- Grundförståelse för ny domän: Robocode (Lab 1)

- Automatiserade systemtester (Lab 3)
- Kravspecifikation (Ö3)

- Affärsrelation beställare-leverantör

Engineering

Monetizing



Strategizing

Detta är på gång:

- Budgivning på robotar (L4)

Robotmässa - Filmvisning

Normal bots

1. Optimus Prime
2. CommandoBot
3. Terrabyte
4. Wall-I
5. LUDynamicsBot
6. Dagge
7. Prawn
8. Freja
9. Judas
10. xxNightStalkerxx
11. Sgt. Psycho

Droids

1. Double-O-Seven
2. iDroid

Leaders

1. NinjaBot

LUND UNIVERSITY

L4: Imorgon kl 23.59

Inlämning av inköpsvektor

- Alla grupper ska lägga bud på samtliga (övriga) gruppers robotar
- Alla grupper har €100
- Minsta bud är €10 (Motsvarande ETSA02 Basic Droid)
- Inköpsvektorn ~~ska skrivas under av projektledaren och lämnas i Grå Skåpet~~ - Projektledaren ska maila vektorn till markus.borg@cs.lth.se
  Exempel:

Inköpsvektor Grupp 02	
Grupp 01 - <ROBOTNAMN>	€15
Grupp 02 - <ROBOTNAMN>	N/A
Grupp 03 - <ROBOTNAMN>	€10
...	
Grupp 16 - <ROBOTNAMN>	€22