

# Föreläsning 25

---

Tobias Wrigstad

*Den mänskliga faktorn i  
programutveckling*



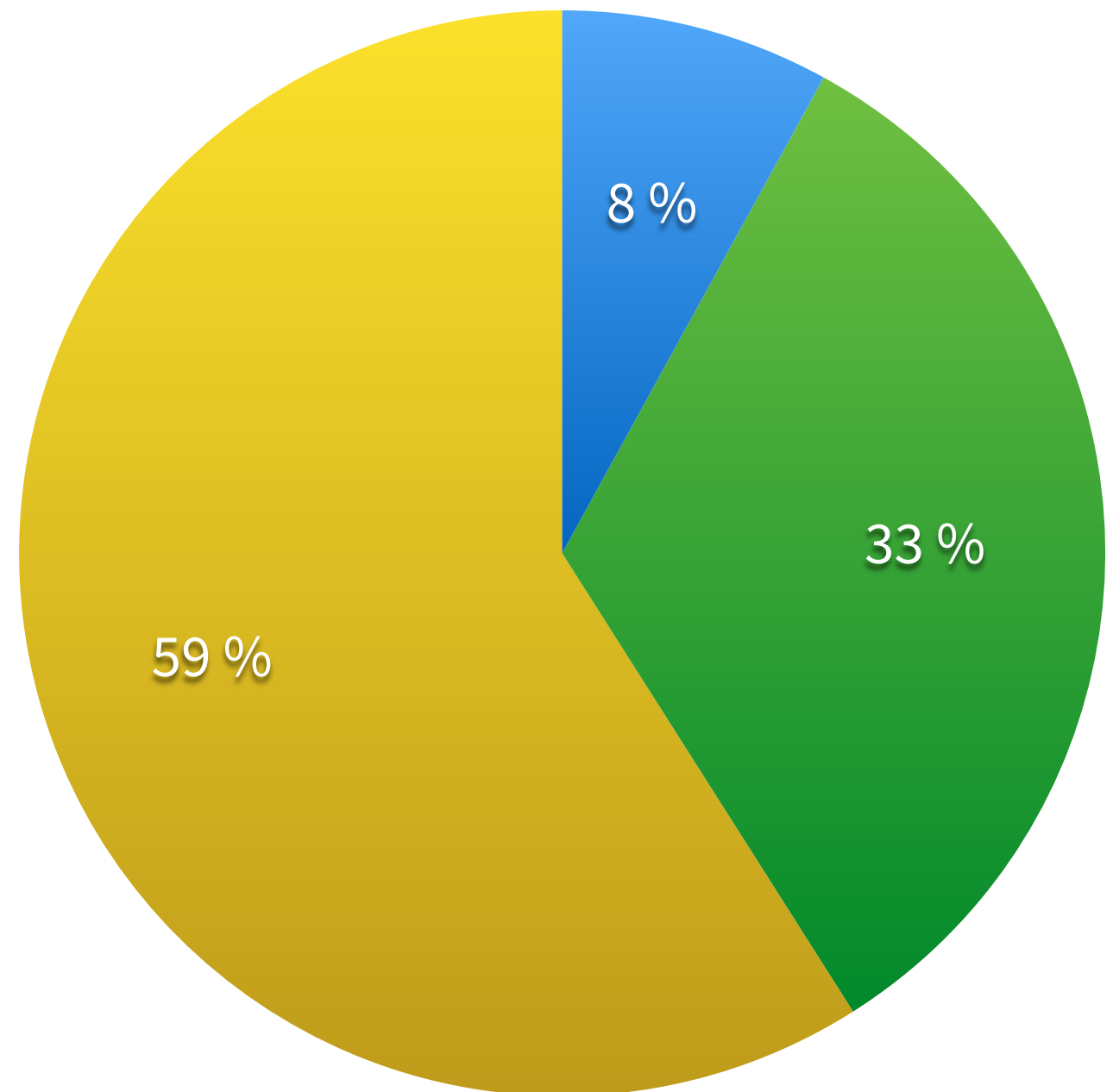
# Programutveckling handlar både om sociologi & teknologi



# En informell undersökning 2006–2009 på ca 350 seniora utvecklare

***Vad anser du skapar mest problem i utvecklingsprojekt?***

● Teknik ● Ledning ● Mänskliga faktorer



# Exempel på motivation till föregående bild

---

- People do not have the required competence or experience
- Formal/Informal communication does not work well, especially not between teams
- People are not assigned so that their competence is utilised in a good/the best way
- People resist changing the way they work
- People are not following/do not understand design rules/processes
- Lack of overview of the process
- Too much time spend on (not) understanding the requirements
- Lack of motivation
- Hard to understand existing code base
- Peoples' reluctance to test

# Programmering ≠ Programutveckling

---

- Programmering handlar om att orkestrera beräkningar

Design, mätningar

Loopar, klasser, if-satser, procedurer, metoder, etc.

- Programutveckling handlar om att orkestrera människor

Få intressanta program skrivs och underhålls av en enda person

Det kräver samarbete, också över tid

# Människor är inte maskiner (!)

---

- Hur du presterar skiljer sig från din kollega — och dig själv över tid
- Det estetiska är viktigt (var skriver du ut {-tecknet?, tabbar eller mellanslag?)
- Personkemi är viktigt
- Din chef kan vara en inspiration (eller inte)
- Arbetsmiljön påverkar hur bra vi presterar
- Kulturella faktorer påverkar samarbetsmöjligheterna
- etc...

# Vad är programutveckling?

---

- 80% av en utvecklares tid går åt till att tänka
- <20% av tiden går åt till att vara kreativ
- Resten av tiden går åt till att skriva ned resultatet (bl.a. "*koda*")

**Källa:** Robert L. Glass, Facts and Fallacies of Software Engineering, Addison-Wesley, 2003

# Vad är programutveckling?

---

- Teknologin löser inte programmeringsproblemen

...bara de ointressanta

Dess mål är att låta programmeraren koncentrera sig på **”det som är viktigt”**

- Det betyder att det (i regel) inte spelar någon avgörande roll om...

Du använder Vim, Emacs eller Netbeans

Du använder tabbar eller mellanslag

Du använder C eller Java

...



# Vad är en bra programmerare?

---

- Team player
- God kommunikatör
- Noggrann och skeptisk
- Lat på ett bra sätt
- Tänker före hen kodar
- Ödmjuk
- Tror på testning och tar tid att testa
- Drar sig inte för att kasta kod
- Är inte sin kod
- Bred kunskap om programspråk
- Skriver bra kod
- Skriver bra kod snabbt

# Vad är en bra programmerare?

---

- Vad är skillnaden mellan en "superprogrammerare" och en vanlig programmerare?
- Vad betyder det för hur det fungerar i projekt?
- Exempel:
  - Källarprogrammerare vs. extroverta programmerare
  - OS/2:s SMP-mikrokärna i assembler
- Vem skall bestämma? De som skall skriva koden idag eller de som skall underhålla den i 15 år?

# Vad är bra kod? [recap]

---

- Utöver korrekt...
- Lätt att förstå/läsbar
- Lätt att testa
- Har test
  - ... projektet
- Lätt att underhålla
  - ... repot
- Low coupling och high cohesion
  - ... språket X
- Lämplig abstraktion
- Abstraktionerna läcker inte
- Robust
- Återanvändbar
- Portabel
- Effektiv
- Ser ut som all annan kod i...

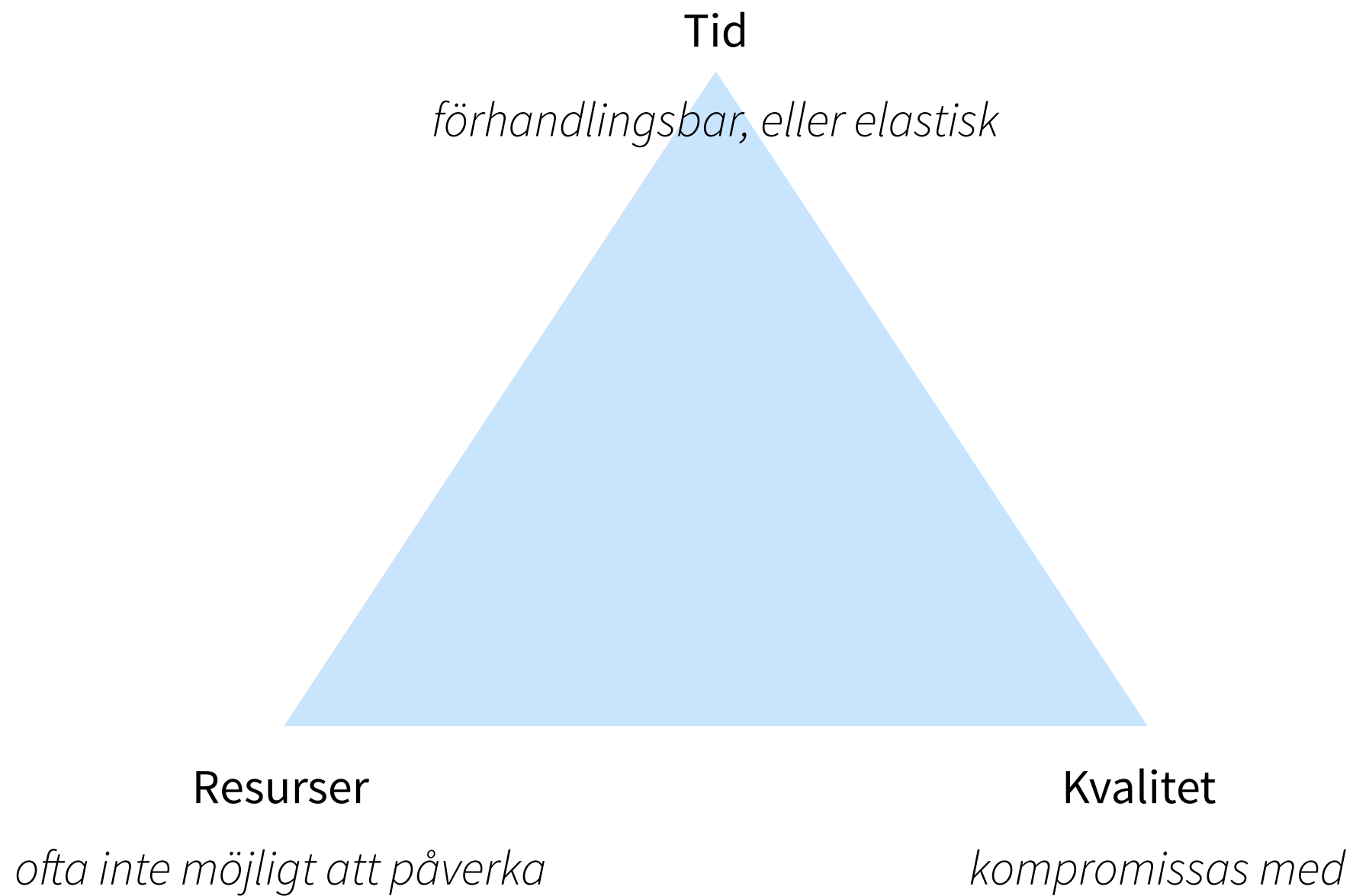
# Programvaruteknik [Eng: Software Engineering]

---

- Hur bygger vi mjukvara så att varje gång vi gör det levererar vi...
  - ✓ på utsatt tid
  - ✓ med givna resurser
  - ✓ med hög kvalitet

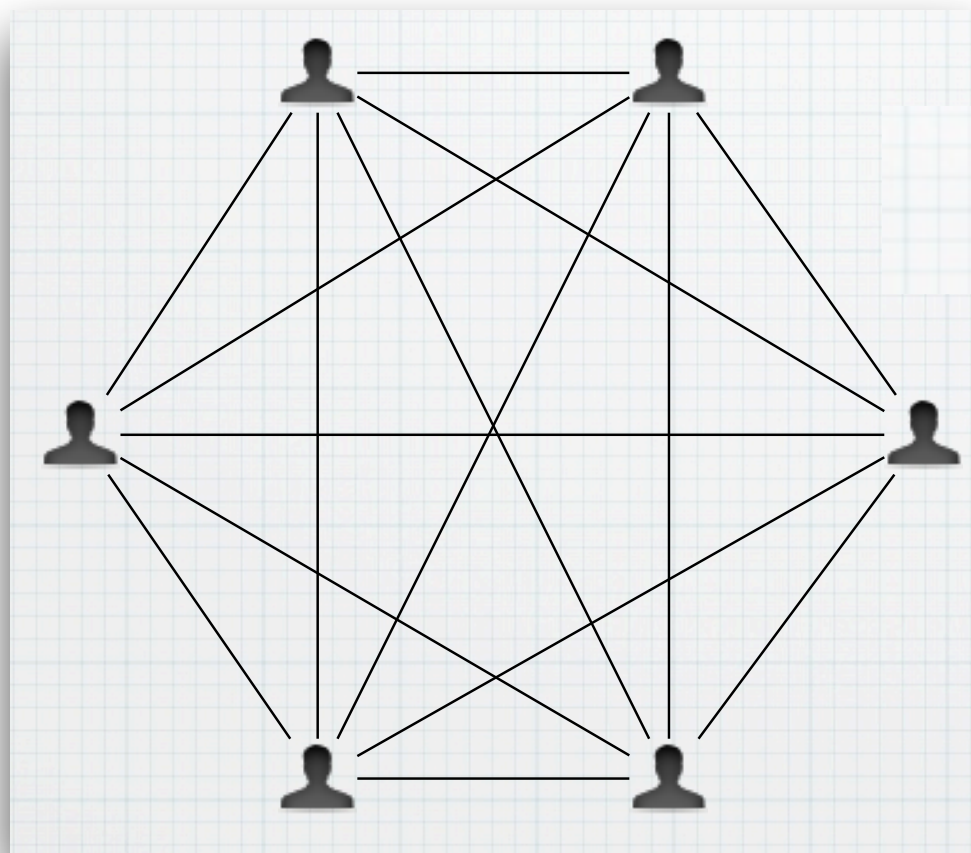
# Tid—Resurser—Kvalitet

---



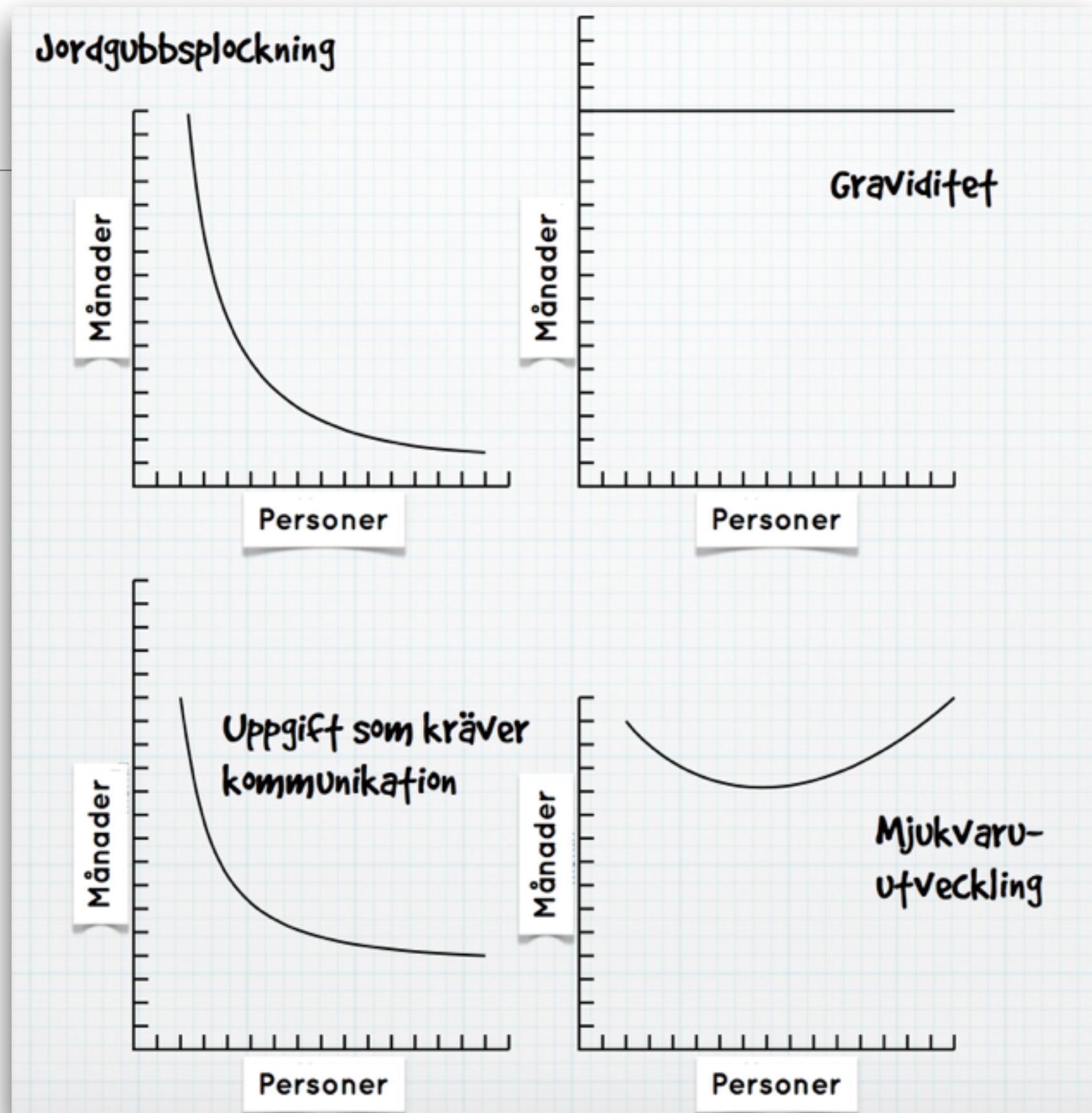
# Brooks lag

*”Om du stoppar in fler utvecklare på ett försenat projekt så kommer det att försenas ytterligare”*



- Kan detta team göra på 1 månad vad en person kan göra 6 månader?
- Minimal kalendertid för ett projekt
- Det tar också tid för en grupp att bildas!

# Brooks lag



# Personlighetssammansättningar

---

## **The chair**

bra på att hålla möten, vara lugn och tolerant (men inte nödvändigtvis smart)

## **The plant**

bra på att vara kreativ och komma på lösningar

## **The monitor-evaluator**

bra på att utvärdera olika lösningar och välja den som är ”bäst”

## **The shaper**

den som hela tiden oroar sig och hjälper teamer att inte glömma bort de viktiga/svåra bitarna

**Källa:** R. Meredith Belbin, Management Teams: Why They Succeed or Fail, (Heinemann, 1981)



# Personlighetssammansättningar

---

## **The team worker**

bra på att skapa bra arbetsklimat och hålla medarbetarna glada

## **The resource investigator**

bra på att spåra upp resurser och information

## **The completer-finisher**

mål-orienterad och pushar mot målet

## **The company worker**

lagspelare som är villig att "ta en för laget, d.v.s. göra mindre intressanta/roliga uppgifter"

**Källa:** R. Meredith Belbin, Management Teams: Why They Succeed or Fail, (Heinemann, 1981)

# Kodkriget [Tom DeMarco & Tim Lister]

---

- Studera programmerare i ”deras naturliga miljö”
- Programmerare på olika företag – mäta företagens produktivitet
- Arbeta ensam, bedömmas i par
- Kriga på arbetstid, under normal kontorstid, etc.

# Utfallet

---

- Valet av programmeringsspråk är mindre viktigt än vem som sitter framför tangentbordet (med undantag för assembler)
- Hur många års erfarenhet man har är inte signifikant (förutom om man inte har arbetat mer än 6 månader i aktuellt språk)
- Den som blir fort klar gör få fel
- Inget statistiskt samband mellan lön och resultat...
- ...däremot mellan programmerarna i paren

Kanske spelar organisationen roll, eller så dras bra programmerare till varandra, eller båda delar

- Produktivitetsskillnad mellan organisationer – 1 : 10

# Utfallet

Arbetsmiljö	Bästa 25%	Sämsta 25%
Egen arbetsyta	78 ft <sup>2</sup>	46 ft <sup>2</sup>
Tillräckligt tyst	57% ja	29% ja
Tillräckligt privat	62% ja	19% ja
Får stänga av telefonen	52% ja	10% ja
Får vidareända samtal	76% ja	19% ja
Blir ofta störd	38% ja	76% ja

# Utfallet

Arbetsmiljö	Bästa 25%	Sämsta 25%
Egen arbetsyta	78 ft <sup>2</sup>	46 ft <sup>2</sup>
Tillräckligt tyst	57% ja	29% ja
Tillräckligt privat	60% ja	19% ja
Får stänga av telefonen	52% ja	10% ja
Får vidarese andra samtal	76% ja	19% ja
Blixta störd	38% ja	76% ja

# Utfallet

Arbetsmiljö	Bästa 25%	Sämsta 25%
Egen arbetsyta	78 ft <sup>2</sup>	46 ft <sup>2</sup>
Tillräckligt tyst	57% ja	29% ja
Tillräckligt privat	62% ja	19% ja
Får stänga av telefonen	52% ja	10% ja
Får vidareända samtal	76% ja	19% ja
Blir ofta störd	38% ja	76% ja

# EGO och arbetsplatsen

---

- Det finns många faktorer som påverkar vad jag vill syssla med i mitt arbete

Teknik...

Utmaningar...

Kollegor...

Chefer...

...men så länge Maslows första 3 är uppfyllda – inte lön

# Status

---

- Olika programmeringsuppgifter anses ofta ha olika status

Underhåll

Testning

Att programmera webbsidor

- Bör man monopolisera kunskap?

Haha! Bara jag kan teknologi X!

- Vad ger status i den här klassen, t.ex.?

Hur får man den?

Hur tar man den?

Hur behåller man den?



# Informella kanaler

---

- Outsourcing – vem gör ditt jobb imorgon? Eller snarare var görs det?
  - Vad händer egentligen vid kaffemaskinen?
  - Var finns motsvarande kanaler här på universitetet?
- 
- Exempel:

Nya hissar, flytta kaffemaskinen, sekreteraren med utsikt, kunskapsutbyte i kaffekön...

# Mått om mått

---

<i>Estimat gjort av</i>	<i>Produktivitet</i>	<i># Projekt</i>
<i>Programmer</i>	<i>8.0</i>	<i>19</i>
<i>Chef / Lead</i>	<i>6.6</i>	<i>23</i>
<i>Tillsammans</i>	<i>7.8</i>	<i>16</i>
<i>Extern analytiker</i>	<i>9.5</i>	<i>21</i>
<i>Inget estimat</i>	<i>12.0</i>	<i>24</i>

- Baserat på en studie av 103 utvecklingsprojekt av Lawrence & Jeffrey (1985)
- Undersöka sanningen bakom "the folklore belief that 'programmers are more productive when they work towards their own estimates'"

# En redovisning har drag av en anställningsintervju

---

- Många företag, inklusive t.ex. Apple, Google, Spotify, Netflix, Twitter, Facebook, men också mindre företag har en intervjuprocess som **innefattar problemlösning**
- Ett intressant CV kan ta dig till en intervju, men för att lyckas måste du inte bara vara tekniskt skicklig, utan också  
kunna förmedla din kunskap och  
redogöra för din process
- Man söker efter personer som  
kan interagera med sin omgivning,  
dela med sig av sin kunskap, och  
är mottaglig för kritik
- Att träna sina färdigheter att på ett avslappnat sätt ”förmedla kunskap” är viktigt!

# Projektarbetet



# Att projektleda ett studentprojekt

- Det är viktigt att varje projektgrupp har en projektledare

Det handlar inte om att bestämma, utan om att det finns **en** central kontaktyta

- Forskning kring studentprojekt visar två tendenser hos projektledare

Projektledaren är äldst i gruppen (för någon definition av ålder)

Projektledaren har en teknisk vision/är inte rädd för komplexitet

- Studentprojekt tenderar att vara mer demokratiska

Målet är att utveckla ett program, men syftet är att lära sig

Projektledarens verkliga makt är liten — svår att utöva

Ett studentprojekt blir därför som ett rollspel



# Ett demokratiskt projekt?

---

- Alla är med och fattar alla beslut

Inte nödvändigtvis effektivt

Ofta leder detta till att man inte för bok över vad man bestämmer, ingen spårbarhet

- Saknar ofta en process för viktiga beslut

Beslutet fattades av ”de som var där, då”

Man kan fatta felaktiga beslut — snabbt!

- Beslut gäller bara så länge som majoriteten fortfarande håller med om det

- Meritokrati: man har inflytande i samma utsträckning som man bidrar (ung.)

# Hur bör det då gå till?

---

- Välj en projektledare på riktigt

Kanske den som är mest tekniskt skicklig bör/vill fokusera på det?

Kanske den som redan driver kåren redan har fullt upp?

Kanske den som är (upplevs som) äldst ändå inte är det bästa valet?

- Ge projektledaren makt

Som projektmedlem: öva dig i att låta en like bestämma

Som projektledare: ta ansvar för att driva på planering, uppföljning, etc.

Projektledaren tar mer tid till uppföljning och planering, ngt. mindre tid till kodning

- Det är bra med roller och ansvar i projektet — minskar overhead

Tech lead, doc lead, design lead, etc.

# Hur bör ett projekt gå till

---

- Varje individ ansvarar för att se till att hen aktivt deltar i de delar som examineras i projektet (X-målen)

Ni kommer att behöva få ett intyg av varandra i slutet av projektet

- Alla måste delta aktivt i programmeringen
- Alla måste **inte** göra allt i samma utsträckning
- Skapa task-forces för specifika problem

T.ex. vad är det för huvudproblem med att få ett C-program att fungera på både Linux och Solaris

- Time-boxa svåra saker eller experiment

Om vi inte kan få X att fungera på 1 dag överger vi den planen



# Kort om projektet [mer i specifikationen]

---

- Implementera en skräpsamlare i C och integrera med en Fas 1/Sprint 2-inlupp

I sort sett skall ni ta bort alla free och ändå inte ha minnesläckage

- Projektet skall bestå av **minst tre sprintar** av en längd som ni själva väljer
- All utveckling skall göras mot GitHub — inlämning sker via GitHub
- All parallell utveckling skall leda till skapandet av en Pull Request som skall accepteras av någon annan än den som programmerade den

Pull Requesten skall alltså granskas och eventuellt knuffas tillbaka

- Redovisning sker på slutseminarier

Bedömning på inlämnad kod, dokumentation, etc. — samt presentation

Slutseminarier i december och i januari (se schema) — välj själva

- Coach för varje grupp, separata möten med alla projektledare

# Sprint 1: Planering & Gruppformande

---

- Skapa en fungerande grupp (det kräver att man faktiskt ses)

- Förstå specifikationen **i lagom grad**

Identifiera osäkra punkter och attackera dem

Bryt ned specifikationen i mindre delar

Utgå från att ni **missförstår** specifikationen, detaljplanera utefter det

- Bestäm arbetssätt — top-down/bottom-up...

- Skapa mindre grupper för utveckling av specifika delar

Använd någon form av verktyg, t.ex. Trello, för att hantera "tickets"

- Använd projektspecifika burn-down charts för att se hur det går med planeringen

# Sprint 2–n

---

- Gör som ni vill!

# Exempel på hur ett Trello-bräde kan se ut

**UpScale** Uppsala Programming Languages ☆ Team Visible

### Doing

- Integration with new PonyRT (2 comments)
- Implement dependencies on tasks
- Basic data structures (3 comments, 4/5 completed)
- Module system (8/22 completed)

Add a card...

### Next

- Traits (25 comments, 2/4 completed)
- Parametric polymorphism (18 comments, 0/2 completed)
- Resolving deadlocks due to self-fulfilled futures (0/3 completed)
- Implement Par data structure. (10 comments)
- Finish the IMDB top 256 CS flicks
- Block GC during suspension

### Done

- Native array support, using type [T]
- Add pony\_arg\_t wrapping to passive methods
- Suspendable/blocking actors (was Futures etc.) (53 comments, 20/22 completed)
- Streams (20 comments, 1 comment)
- Document the type system in comments in the code (3 comments)
- Fix comment rot in CodeGen

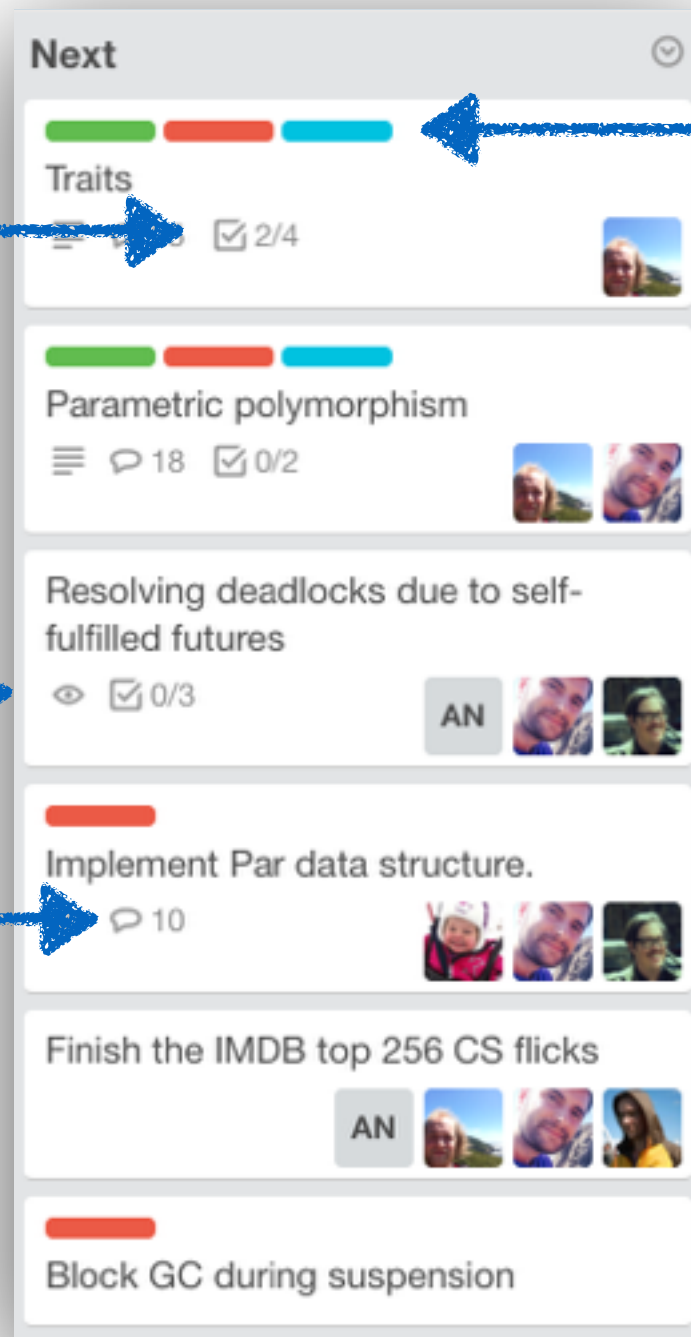


# Exempel på hur ett Trello-bräde kan se ut

Framstegsfunktion

Bevakar jag det?

Diskussionstrådar



Kategorier av tickets

Vilka jobbar på det?




Beskrivning  
av vad som  
skall göras



Traits in list [Next](#)

Members

 +

Labels

Front-end

Needed by review

Back-end

+

Description [Edit](#)

Replace the classes we have now with traits.

```
trait Set
  require f : int
  def set(x : int) : void
    this.f = x

trait Read
  require f : int
  def read() : int
    this.f

passive class Cell = Set + Read
  f : int
  def init(x : int) : void
    this.set(x)

class Main
  def main() : void
    let x = new Cell(42) in {
      print x.read();
      x.set(100);
      print x.read();
    }
```

Add

Members

Labels

Checklist

Due Date

Attachment

Actions

Move

Copy

Subscribe

Archive

[Share and more...](#)

☒ **Todo**

[Hide completed items](#) [Delete...](#)

50%

☒ *Trait-polymorphic-fields*

☒ *Closures-capturing-trait-polymorphic-variables*

☐ Returning passive objects from active methods

☐ Passing passive references as polymorphic arguments (when the parameter type



Delrapportering



# Fakta om projektet

---

- Det är inte speciellt stort — 1000 rader kod + 1000 rader test är vanligt

Flera av er skrev **ensamma** större lagerhanterare!

- Svårigheten ligger inte nödvändigtvis i koden utan i det kringliggande

Förstå specifikationen [eventuellt göra avsteg från den, motivera dem och få OK]

Koordinera ett samarbete över tid

Plattformsberoende C-program (t.ex. #makron, läsa manualer för OS, etc.)

- Ger er mer tid att fokusera på det som projektet (också) handlar om

Process, planering, samarbete, versionshantering, kommunikation, etc.

Modularisering, gränssnitt, dokumentation, **och inte minst testning**

# Ståmöten (Stand-up meeting)

---

- He regelbundna möten för att synkronisera
- Jämför med uppföljningsmöten från kursen hittills, men
  - Alla jobbar nu i samma projekt
  - Projektets framsteg är beroende av allas framsteg
  - Vissa tickets kommer att blockera på andra tickets
- Leds av projektledaren
  - ”Walk the board” — gå igenom alla tickets i doing, kolla att alla vet sina nexts
  - Behöver någon hjälp? Skall vi omfördela resurser?
  - Hur ligger vi till?
- Sedan: uppdatera projektets burn-down chart, räkna ut team velocity



# När formas projektgrupperna?

---

- Länk kommer idag i Piazza för att anmäla sig till att göra projektet
- Vi skapar **helt nya grupper** nu
- De första grupperna publiceras på onsdag nästa vecka
- Blandat IT & DV