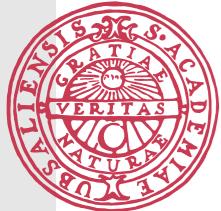


Föreläsning 25

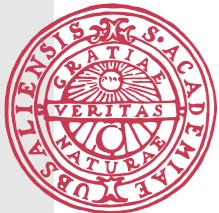
Tobias Wrigstad

*Programmering !=
programutveckling*



Computer science is the only
profession in which a single
mind is obliged to span the
distance from a bit to a few
hundred megabytes

—Steve McDonnel



Fakta om projektet

- Det är inte speciellt stort – 1000 rader kod + 1–2000 rader test är vanligt/förväntat
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**

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 uppgiftens tekniska komplexitet

- Studentprojekt tenderar att vara mer demokratiska än i verkligheten

Målet är att utveckla ett program, men syftet är att lära sig/träna färdigheter

Projektledarens verkliga makt är liten — svårt att sparka eller omplacera någon

Ett studentprojekt blir därför som ett rollspel — **man måste ge makt**

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.

Exempel på hur ett Trello-bräde kan se ut

UpScale Uppsala Programming Languages  

Doing

- Integration with new PonyRT
  2 
- Implement dependencies on tasks

- Basic data structures
  3  4/5 
- Module system
 8/22 

Add a card...

Next

- Traits
  25  2/4 
- Parametric polymorphism
  18  0/2 
- Resolving deadlocks due to self-fulfilled futures
  0/3  AN  
- Implement Par data structure.
  10   
- Finish the IMDB top 256 CS flicks
 AN   
- 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  20/22  AN 
- Streams
  20  1  
- Document the type system in comments in the code
  3 
- Fix comment rot in CodeGen

Exempel på hur ett Trello-bräde kan se ut

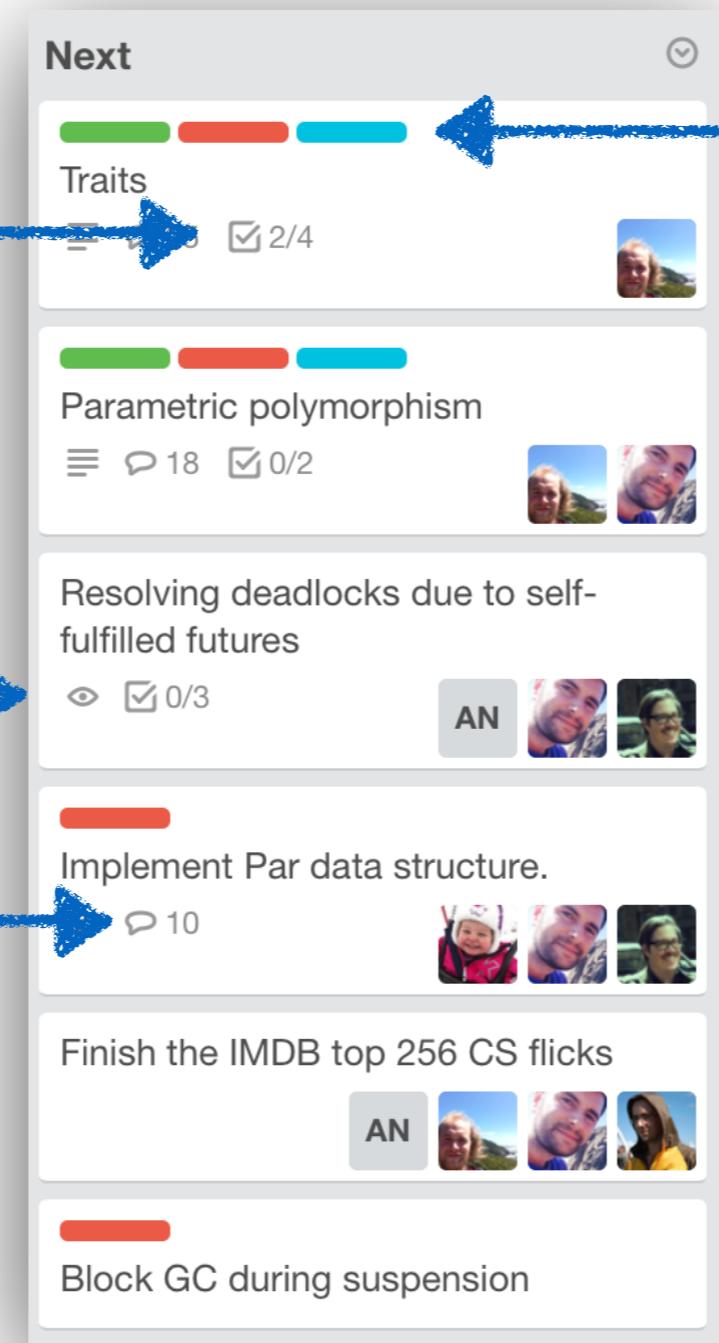
Framstegsfunktion

Bevakar jag det?

Diskussionstrådar

Kategorier av tickets

Vilka jobbar på det?



Beskrivning
av vad som
skall göras

 Traits in list [Next](#) X

Members Labels Add

 + 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();
    }
```

→ →

Members Labels Checklist Due Date Attachment

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



Stämöten (Stand-up meeting)

- He regelbundna möten för att synkronisera
- Jämför med uppföljningsmöten från kurserna 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

Processer

Ingen process

Äsch mjukvara – hur svårt kan det vara!?

Vilken process som helst är bättre än ingen process

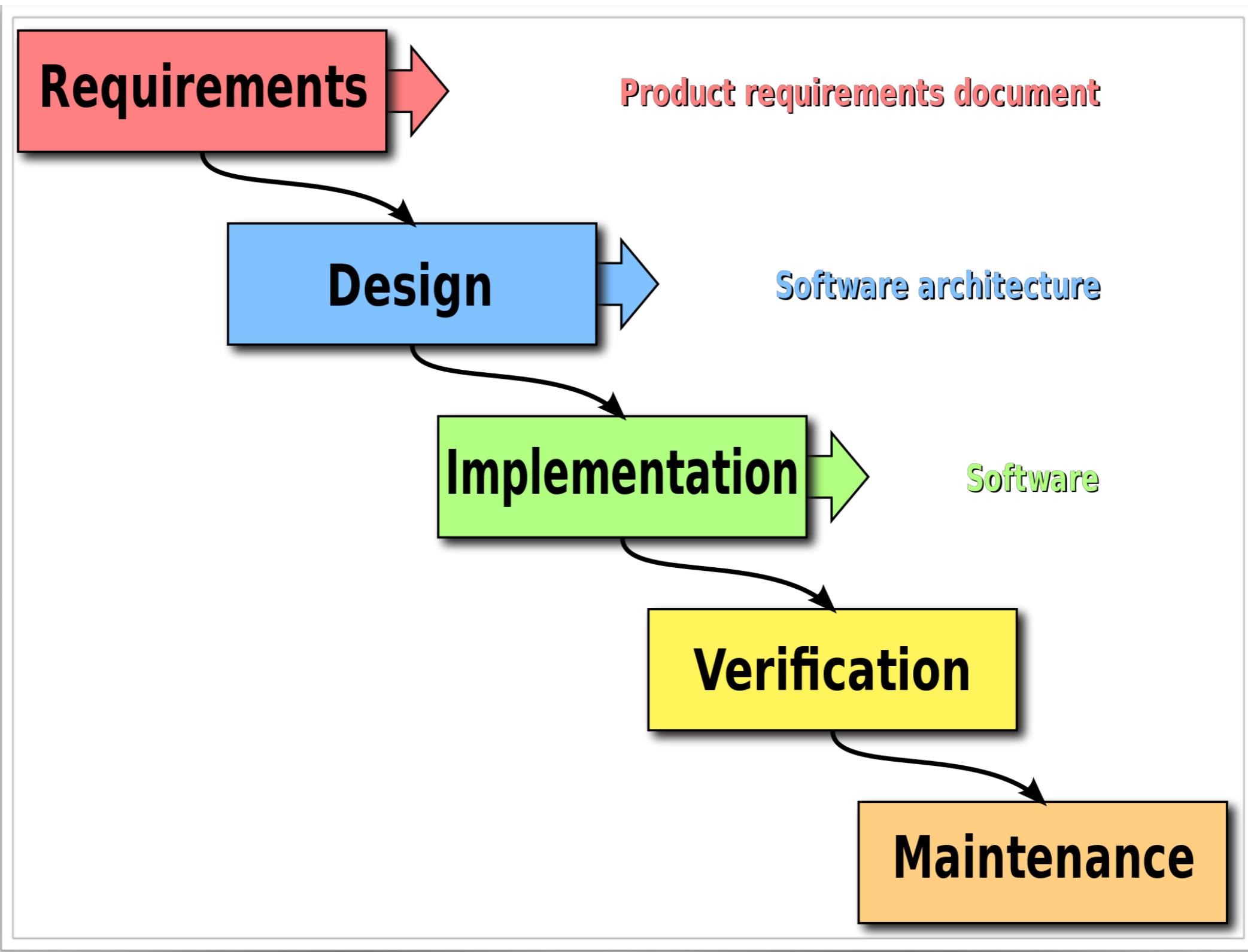
Vattenfallsmodellen

Iterativa modeller

Dokumentationsdrivna modeller

Lättrörliga (Agila) modeller

...



Systems Development Life Cycle



1. Preliminary Analysis and Planning

- System objectives
- Scope of the problem
- Information gathering
- Project plan



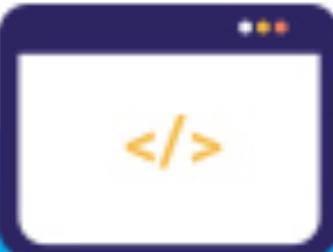
2. Requirement Analysis

- Functional Requirements Document created



3. System Design

- Design Document
- Very detailed
- Explains features and operation
- Includes screen layouts, business process rules, etc.



4. Development

- Actual code written
- Testing procedures are defined



7. Operation and Maintenance

- Training
- Documentation
- System assessment
- Changes
- Evaluation



5. Integration and Testing

- Rigorous testing
- Errors, bugs and interoperability issues resolved



6. Implementation

- Deployment into the production environment

5. Integration and Testing

- Rigorous testing
- Errors, bugs and interoperability issues resolved



Development process





Phases

Core Process Workflows

Business Modeling

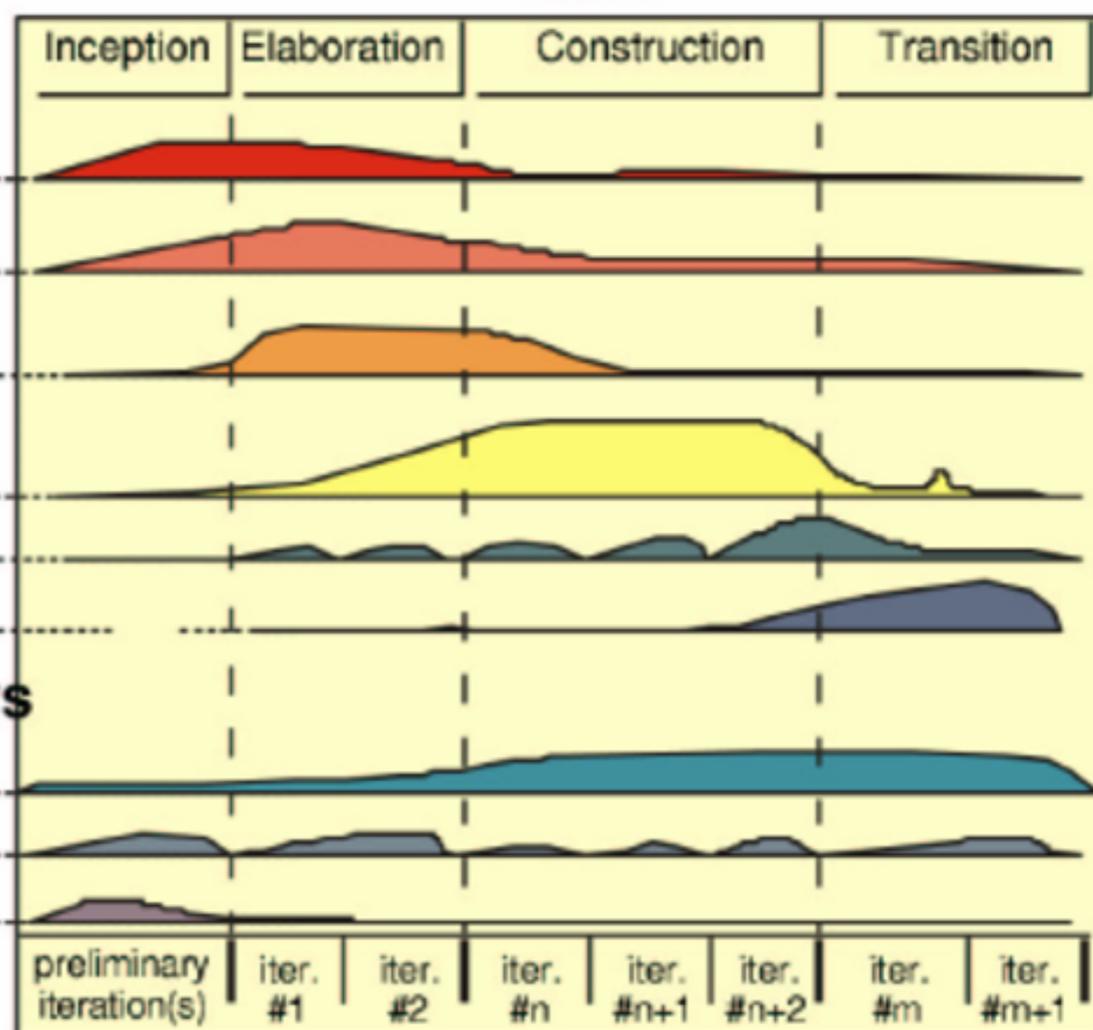
Requirements

Analysis & Design

Implementation

Test

Deployment

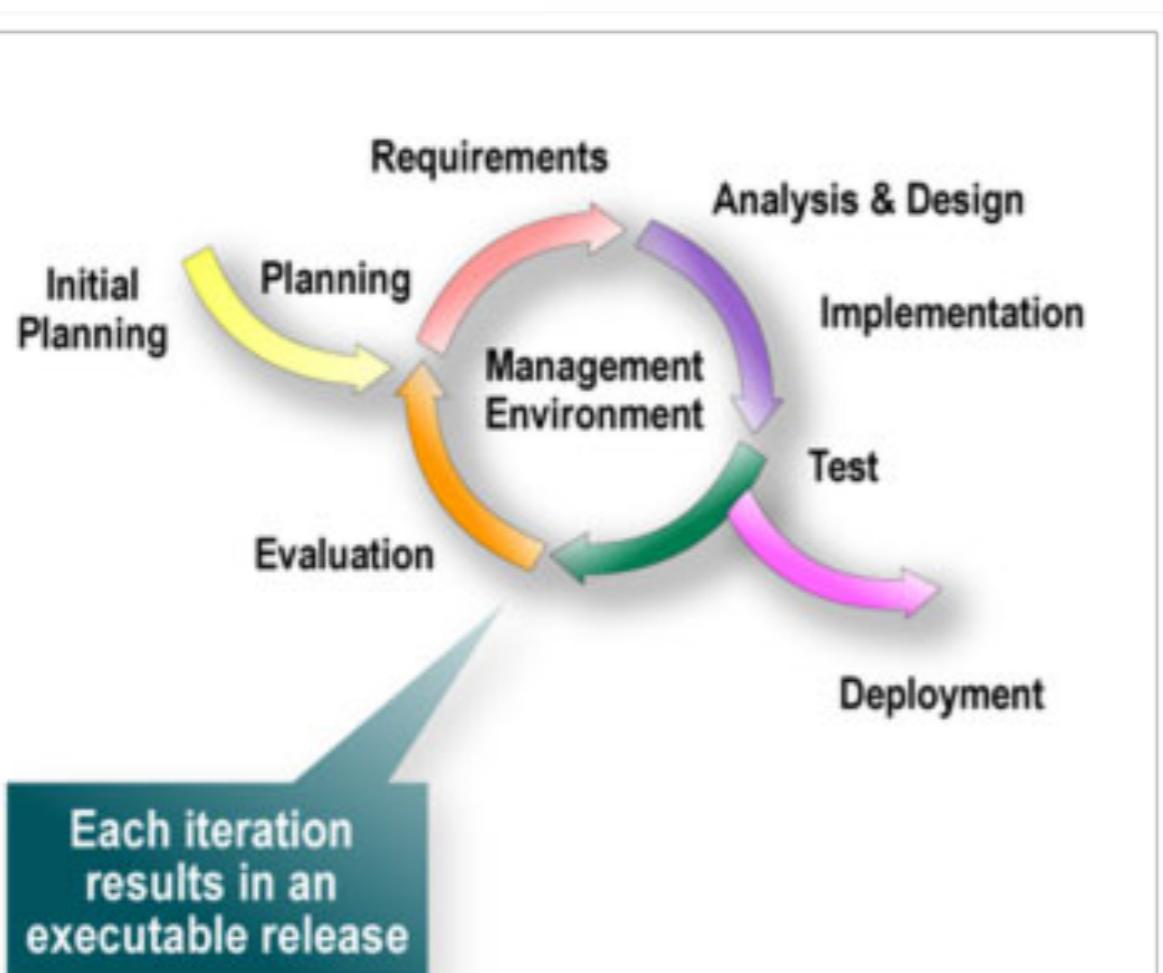


Core Supporting Workflows

Configuration & Change Mgmt.

Project Management.....

Environment.....



Capability Maturity Model



Manifesto for Agile Software Development



We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value:

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.



Principles behind the Agile Manifesto

We follow these principles:

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals.

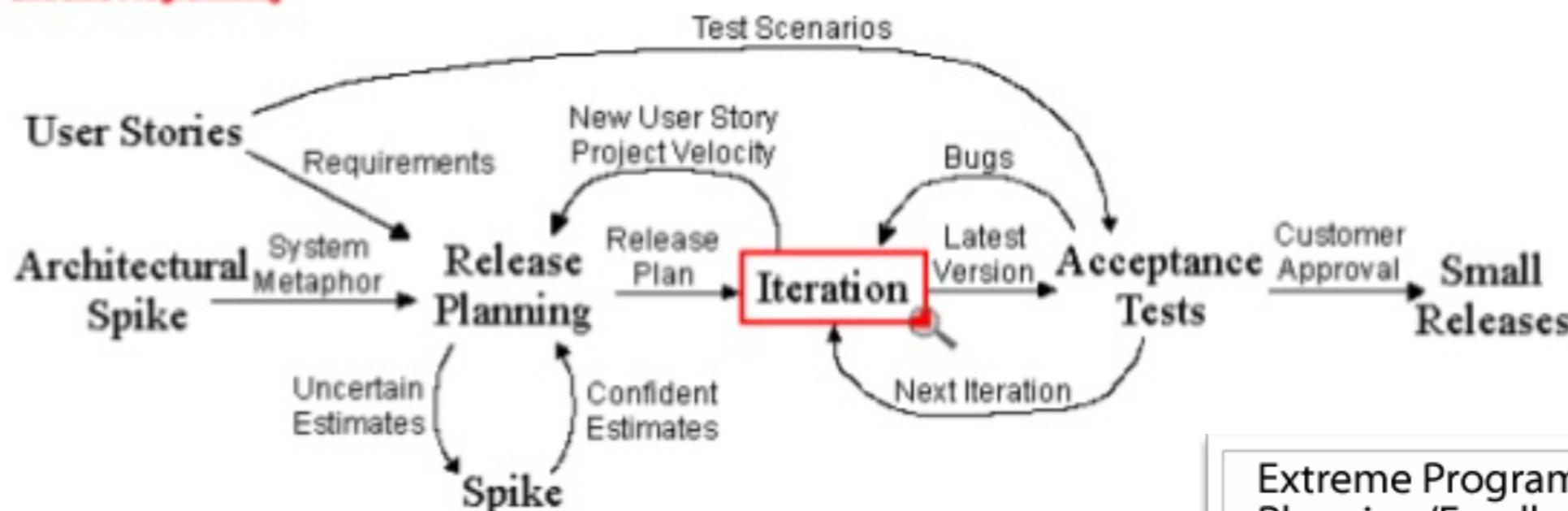
Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.





Extreme Programming Project



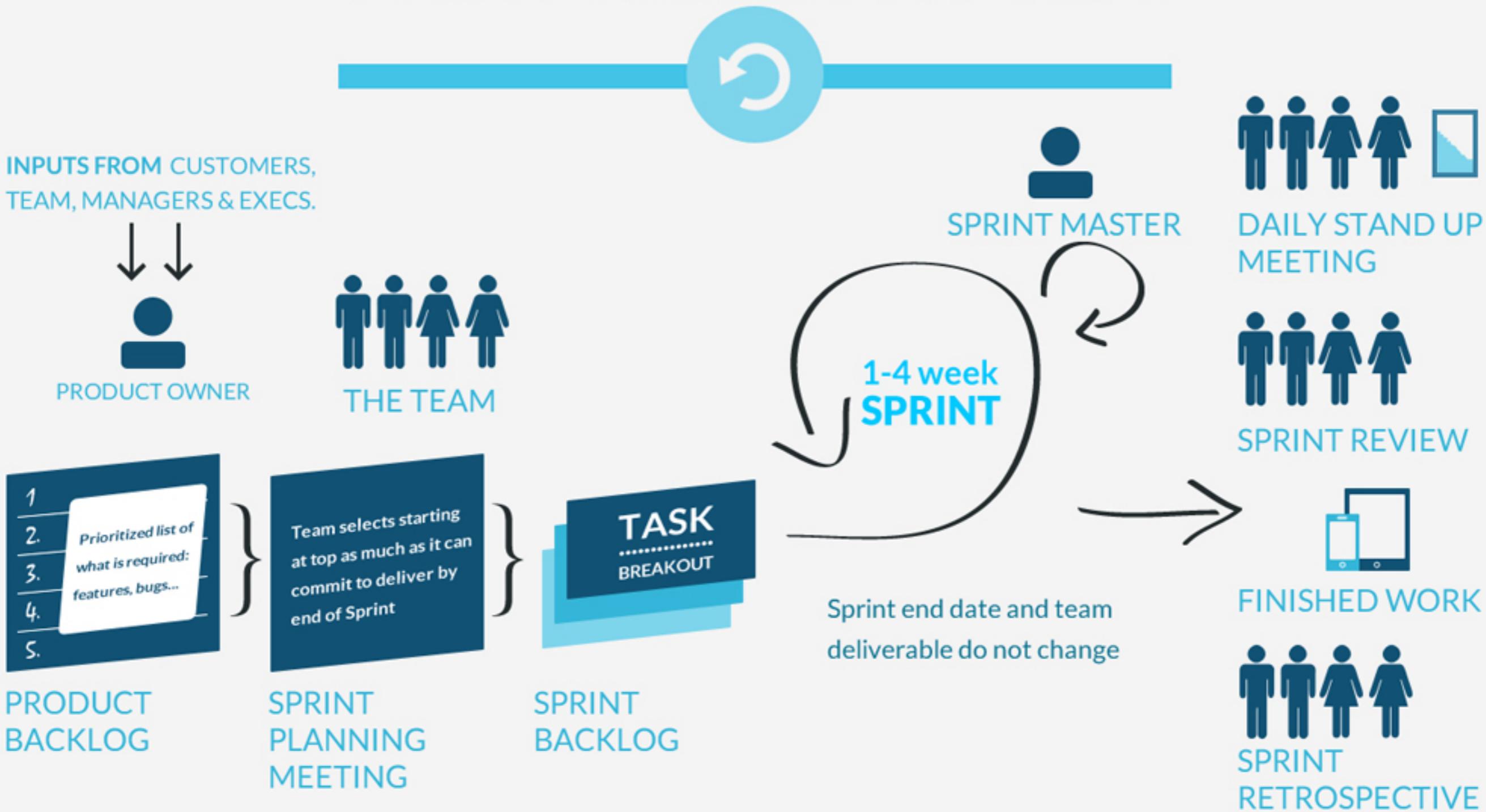
Extreme Programming Planning/Feedback Loops



© J. Donovan Wells

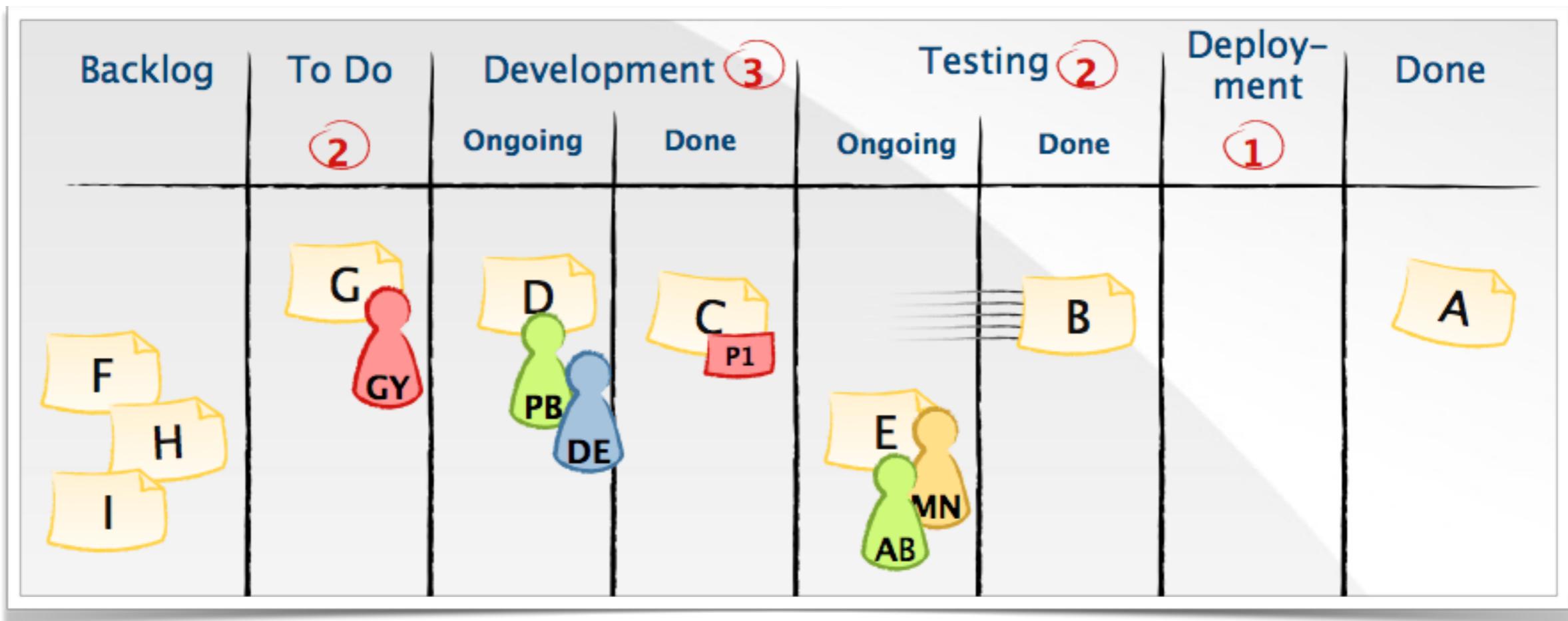


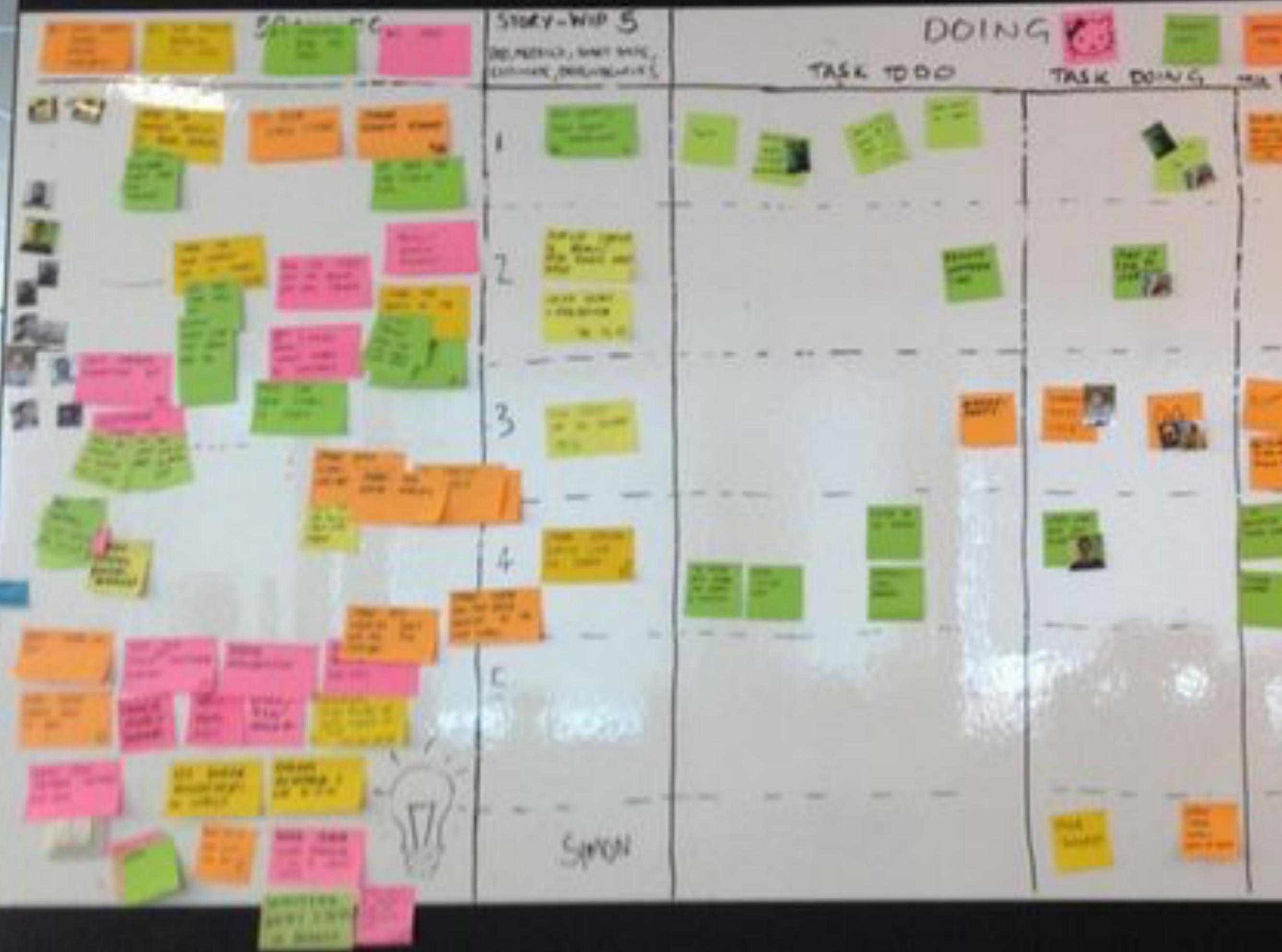
the SCRUM SOFTWARE DEVELOPMENT PROCESS



Kanban

- Visualisera, gör processen mer konkret (och gärna taktil!)





DOING

卷之三

A|B TESTS



Programming, Motherfucker

Do you speak it?

We are a community of motherfucking programmers who have been **humiliated** by software development methodologies for years.

We are tired of *XP*, *Scrum*, *Kanban*, *Waterfall*, *Software Craftsmanship* (aka *XP-Lite*) and anything else getting in the way of...**Programming, Motherfucker.**

We are tired of being told we're socialy awkward idiots who need to be manipulated to work in a Forced Pair Programming chain gang without any time to be creative because none of the 10 managers on the project can do... **Programming, Motherfucker.**

We must destroy these methodologies that get in the way of...**Programming, Motherfucker.**

* * * *

Our Values

They Claim To Value

Individuals and interactions

Working software

Customer collaboration

Responding to change

They Really Value

Tons of billable hours

Tons of pointless tests

Bleeding clients dry

Instability and plausible deniability

We Fucking Do

Programming,
Motherfucker

Programming,
Motherfucker

Programming,
Motherfucker

Programming,
Motherfucker

We think the shit on the left, is really just the con in the middle, and that we really need to just do the thing on the right...Programming, Motherfucker.

Processer

Ingen process

Äsch mjukvara – hur svårt kan det vara!?

Vilken process som helst är bättre än ingen process

Vattenfallsmodellen

Iterativa modeller

Dokumentationsdrivna modeller

Lättrörliga (Agila) modeller



Sprint 1: Design

- Design är ett ”wicked problem”
- Design är en rörig process, även om resultatet skall vara rent och snyggt
- Design handlar om trade-offs och prioriteter
- Design handlar om att lyfta och tillmötesgå (eller inte) krav
- Design drivs av heuristik
- Design är icke-deterministiskt

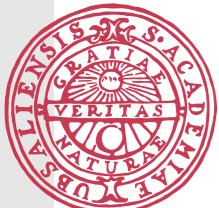
Produktivitet(Prog_A)

==

Produktivitet (Prog_B)

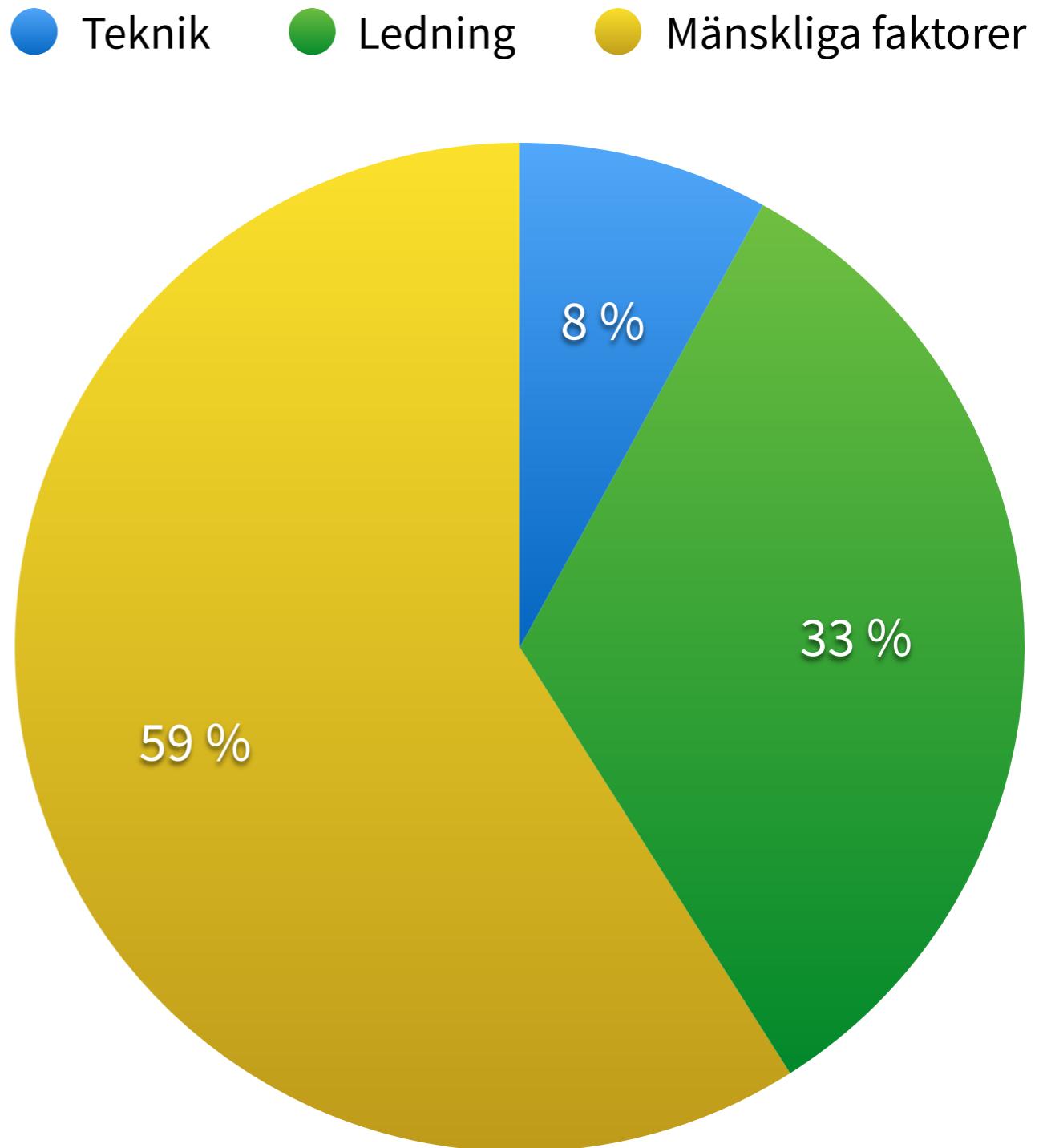


Programutveckling
handlar både om
sociologi & teknologi



En informell undersökning 2006–2009 på ca 350 seniora utvecklare

*Vad anser du skapar
mest problem i
utvecklingsprojekt?*



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
- Bred kunskap om andra verktyg
- Örat mot rälsen, men ingen vindflöjel
- 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
- Lätt att underhålla
- Low coupling och high cohesion
- Lämplig abstraktion
- Abstraktionerna läcker inte
- Robust
- Återanvändbar
- Portabel
- Effektiv
- Ser ut som all annan kod i...
... projektet
- ... repot
- ... språket X

Programvaruteknik [Eng: Software Engineering]

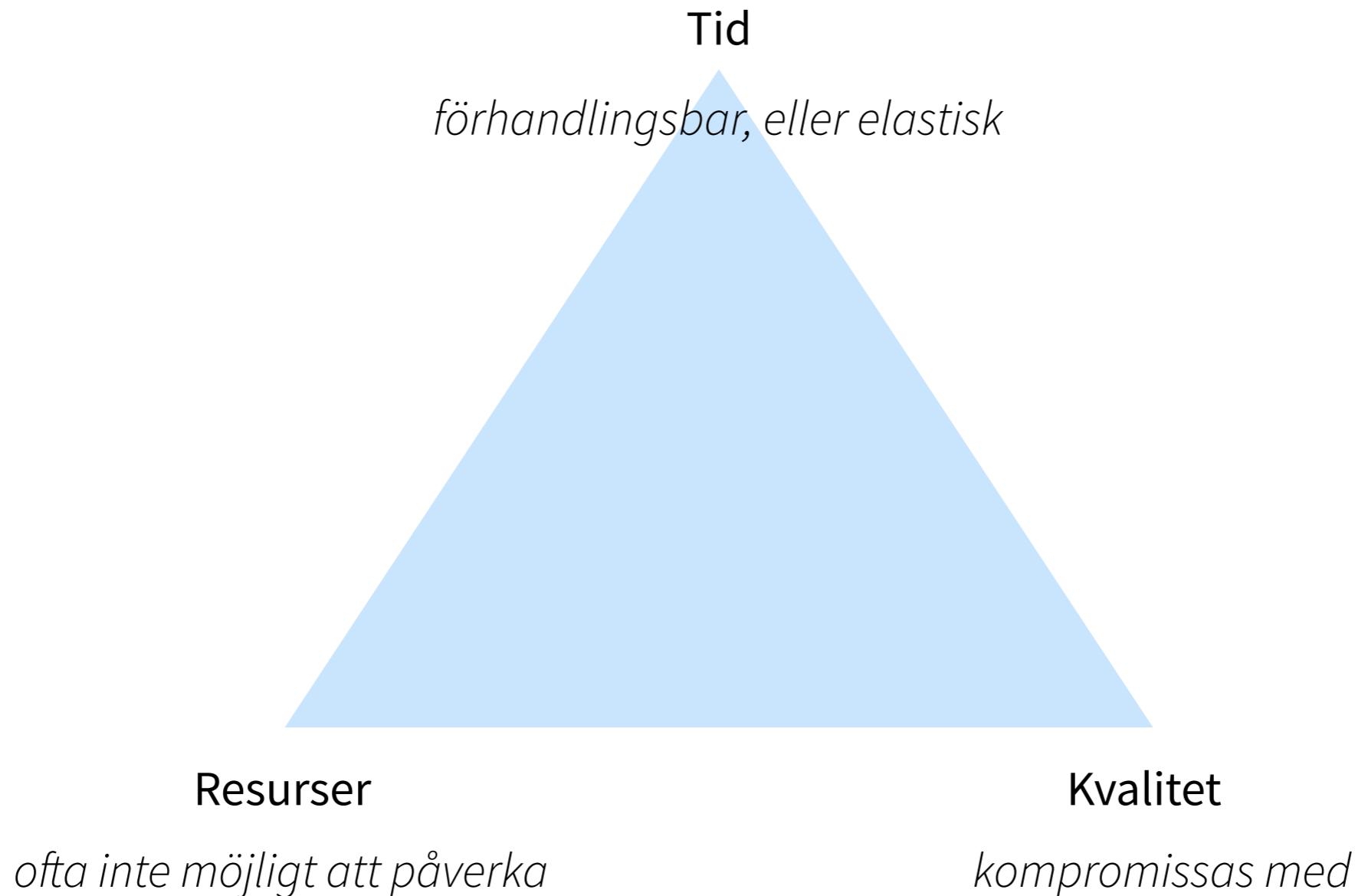
- Hur bygger vi mjukvara så att varje gång vi gör det levererar vi...

 på utsatt tid

 med givna resurser

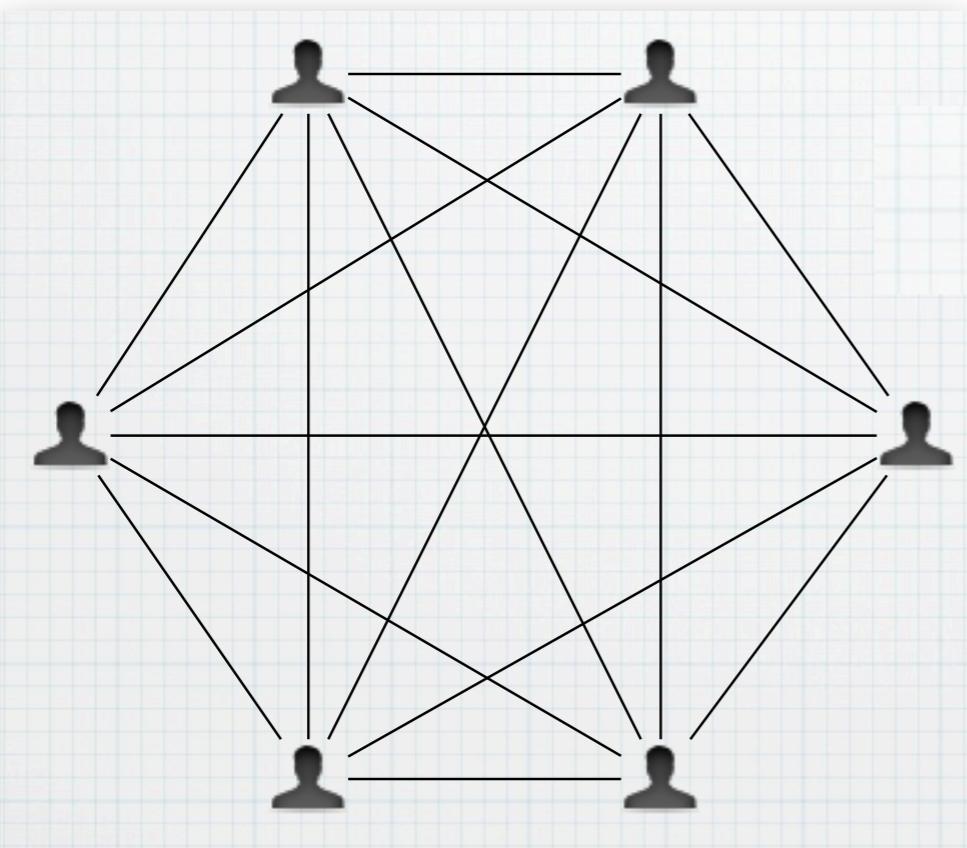
 med hög kvalitet

Tid—Resurser—Kvalitet



Brooks lag

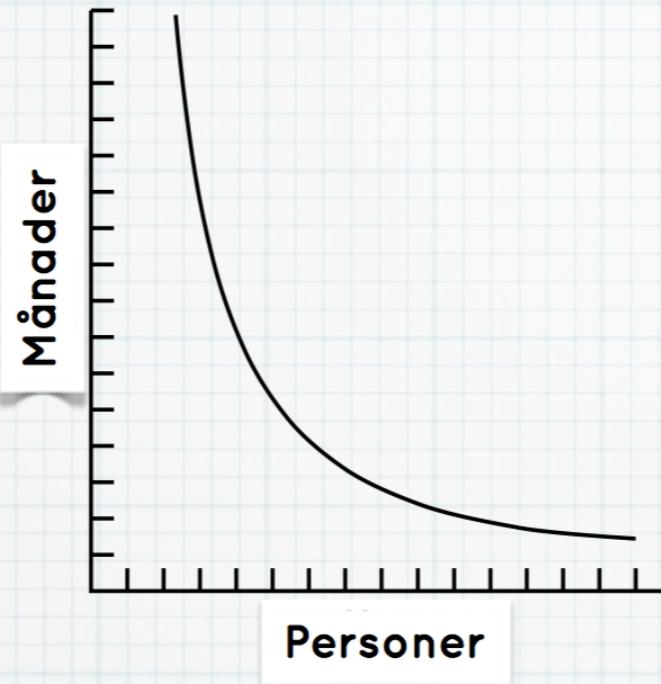
"Om du stoppar in fler utvecklare på ett försenat projekt så kommer det att försenas ytterligare"



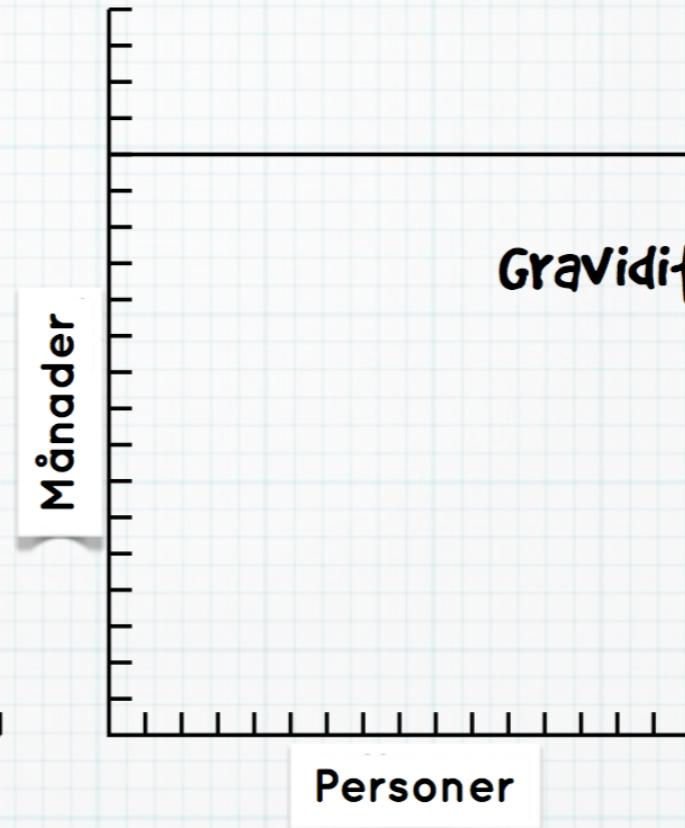
- Kan detta team göra på 1 månad vad en person kan göra 6 månader?
- Minimal kalendertid för ett projekt
- Det tar också tid för en grupp att bildas!

Brooks lag

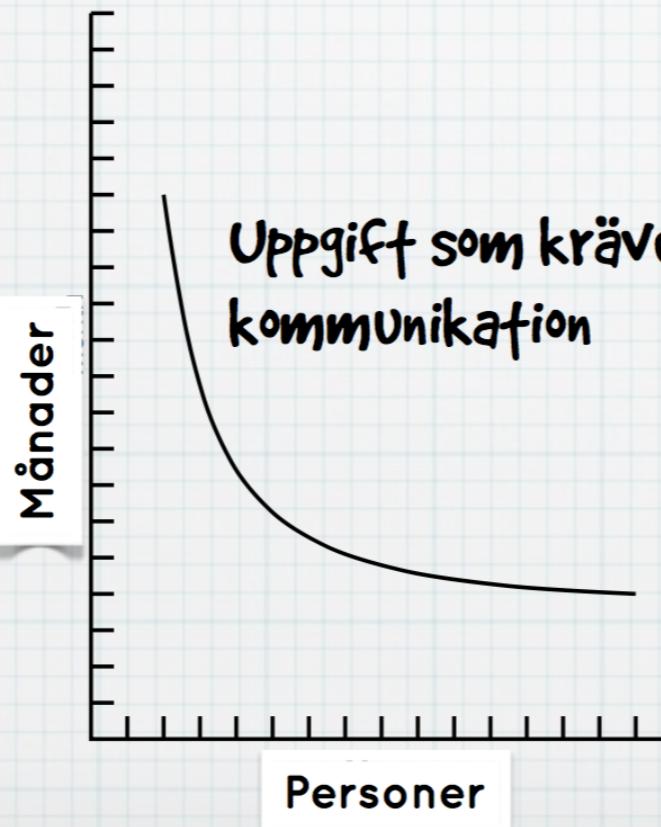
Jordgubbsplöckning



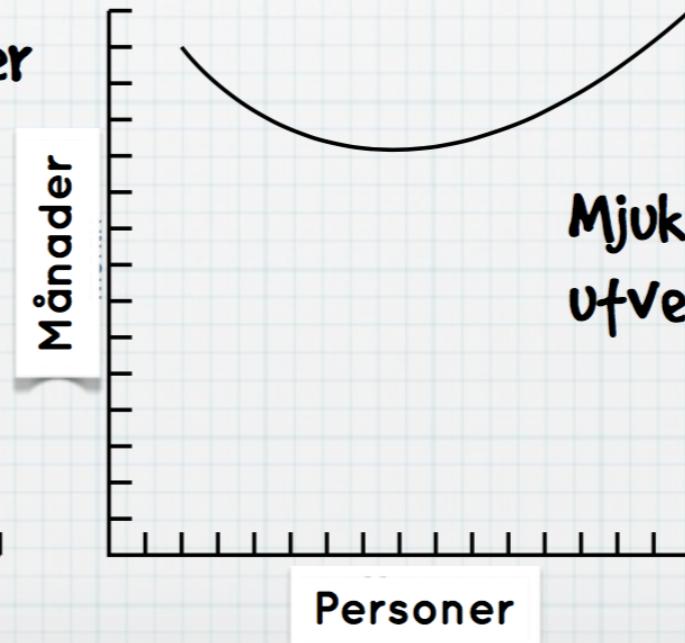
Graviditet



Uppgift som kräver
kommunikation



Mjukvaru-
utveckling



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 ²	46 ft ²
Tillräckligt tyst	57% ja	29% ja
Tillräckligt privat	62% ja	19% ja
får stänga av telefonen	52% ja	10% ja
får vidaresända samtal	76% ja	19% ja
Blir ofta sförd	38% ja	76% ja

Utfallet

Arbetsmiljö	Bästa 25%	Sämsta 25%
Egen arbetsyta	78 ft ²	46 ft ²
Tillräckligt tyst	57% ja	29% ja
Tillräckligt privat	62% ja	19% ja
får stänga av telefonen	52% ja	10% ja
får vidaresenda samtal	76% ja	19% ja
Blir extra störd	38% ja	76% ja

Utfallet

Arbetsmiljö	Bästa 25%	Sämsta 25%
Egen arbetsyta	78 ft ²	46 ft ²
Tillräckligt tyst	57% ja	29% ja
Tillräckligt privat	62% ja	19% ja
får stänga av telefonen	52% ja	10% ja
får vidaresända samtal	76% ja	19% ja
Blir ofta sförd	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?

Bwahaha! 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>	8.0	19
<i>Chef/ Lead</i>	6.6	23
<i>Tillsammans</i>	7.8	16
<i>Extern analytiker</i>	9.5	21
<i>Inget estimat</i>	12.0	24

- 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'"

Slutsats?

