Föreläsning *N–1* av *N**: Avslut, början, projekt

Tobias Wrigstad

*) med reservation för off-by-one errors

Retrospektiv

Funktionell

PKD

Haskell

Rekursion

Imperativ

IOOPM

C Java

Loopar Iteration

Aliasering

Värdeöverföring

Föränderligt tillstånd

Photo by Alexander Milo on Unsplash

Från ”programming in the small” till ”programming in the large”

”Skriv ett program som fungerar”

- Kodgranskning
- Debugging
- Versionshantering
- Pull-requests
- Testning
- Resurshantering
  - Minne
  - Svansrekursion
- Planering
- Dokumentation
- Refaktorering
- Designmönster
- Defensiv programmering
- Coupling och cohesion
- Profilering och optimering
- Kvalitativa kodattribut
- Icke-funktionella krav

En resa genom tiden

C → **Java**

Manuell minneshantering

Lågnivå

Procedurer

Ingenting är gratis

Automatisk minneshantering

Högnivå

Objekt

Språkstöd för ”best-practises”

Moduler, arv, klasser

Inkapsling

Parametrisk polymorfism

Reflektion

Gränssnnitt

Undantagshantering

Och nu?

Photo by Caleb Jones on Unsplash

Hur du kan gå vidare härifrån (exempel)

- Mjukvarutestning
- Underhållsprogrammering
- Avancerad mjukvarudesign
- Algoritmer och datastrukturer I, II, III
- Semantik för programspråk
- Programmeringsteori
- Parallellprogrammering…

- Agil mjukvaruutveckling
- Kravställning
- Komplexa IT-system

- Tillämpningar
  - Artificiell intelligens
  - Maskininlärning
  - Villkorsprogrammering
  - Bildanalys
  - Inbyggda system
  - …


Definition of Done
Board
ready for demo
Sprint calendar
Legacy tasks
The team
Stories
Burndown
Backlog
Tasks
Parking
Retrospective

Kursens bäring på yrkeslivet

- Kunskapsmässigt, jättestort

    Se föregående bilder!

- Utifrån hur arbetet utförs — **inte jättestort**

    Bygga program från scratch

    Inget team

    Nästan ingen förvaltning

    Parprogrammering?

    Väldigt mycket NIH och väldigt lite mjukvarubibliotek

Däremot

Uppdelning i sprintar

Planering och burndown

Arbete med tickets

Kodgranskning

Alla verktyg

…

Bonus: En redovisning ≈ whiteboard/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!

Programmering ≠ Programutveckling

- Programmering handlar om att orkestrera beräkningar

  Design, mätningar

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

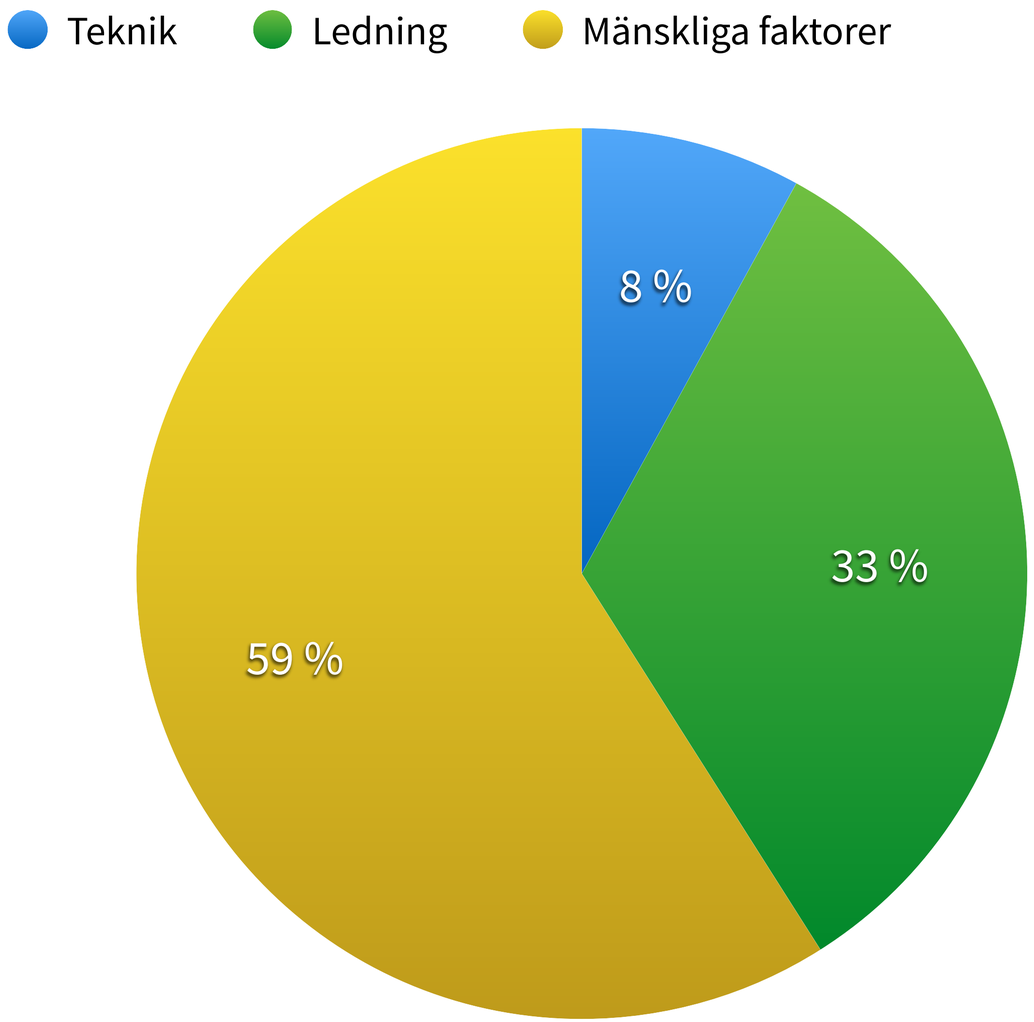
- Programutveckling handlar om att orkestrera människor

  Få intressanta program skrivs och underhålls av en enda person

  Det kräver samarbete, också över tid

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

Projektarbete

Introduktion till problemet ”storskalig” mjukvaruutveckling

- Mer sociologi, mindre teknik
- Kanske 1–2 personer kan utföra arbetet, men det fungerar inte i praktiken
- Förståelse för – och samförstånd kring tolkning av – en abstrakt specifikation
- Gemensam process

Syftet är inte att lära sig mer om programmering

- Även om det naturligtvis kan vara en sidoeffekt

Att skapa en förståelse för problem och möjligheter för framtida kurser att bygga vidare på — det kräver dock din eftertanke men framförallt dokumentation

Informella kanaler

Vad ni skall tänka på under projektet

- Hur kan man säkerställa leverans på utsatt datum?
- Hur lastbalanserar man så att ingen blir utbränd och så att alla får vara med?
- Hur vet vi att vi levererar rätt produkt?
- Hur vet vi att det vi levererar fungerar korrekt? (Och vad betyder korrekt?)
- Hur kan man kommunicera effektivt i ett team?
- Hur hanterar vi t.ex. projektmedlemmar som är omotiverade, stressade kring saker utanför projektet, okommunikativa, etc. — på ett hållbart vis?
- Hur bygger man ett team som fungerar bra?

    Det blir inte ett team bara för att man är >1 person

    Samarbete uppstår inte alltid spontant

    Undergrupper, småpåvar, galenpannor…

EGO

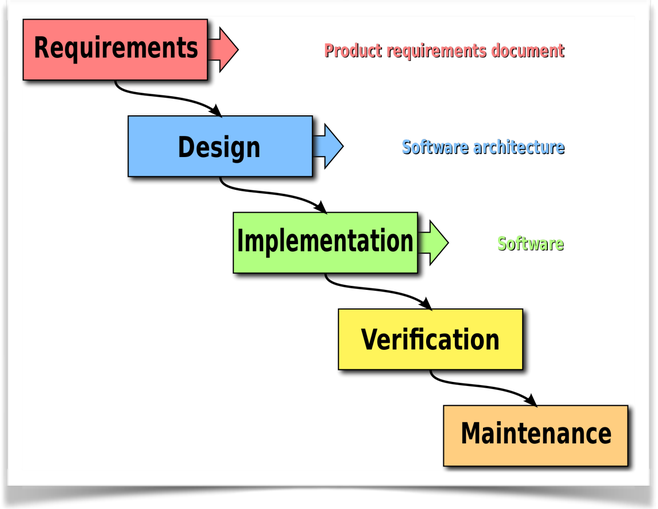
Vad du skall tänka på under projektet

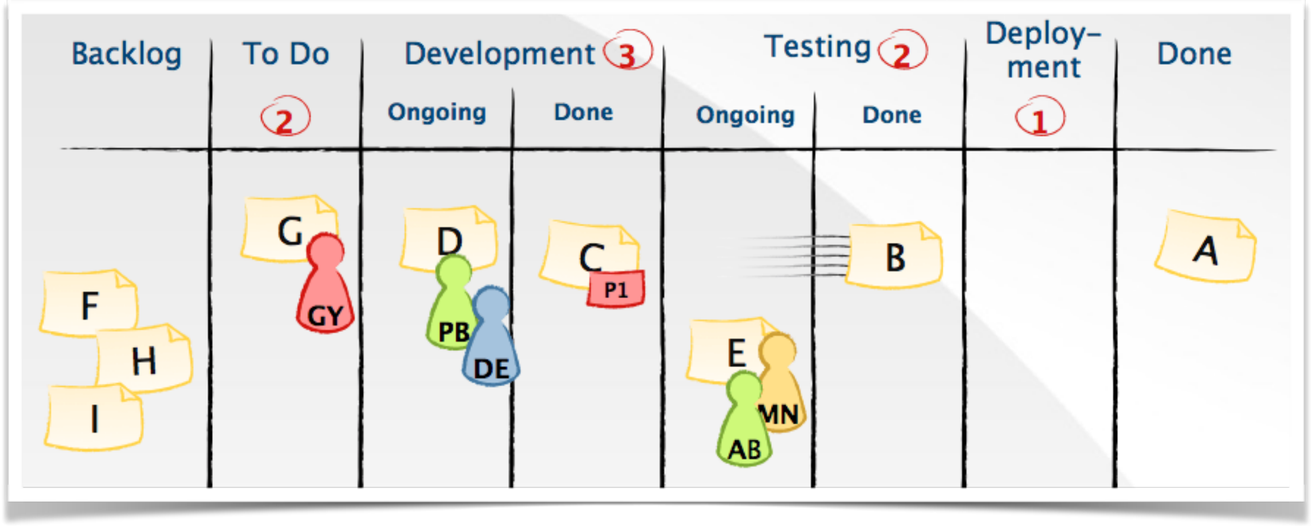
- Hur fungerar jag i ett team? Vilka är mina styrkor och svagheter?
- Hur bör jag förändra mitt arbetssätt för att fungera bättre i ett team?
- Vad motiverar mig?
- Vad får mig omotiverad?
- Hur ser jag till att hålla mig motiverad, glad, fokuserad, och produktiv?
- Hur bra är jag på att uppskatta hur lång tid någonting kommer att ta så att jag kan delta i planering?
- Vad har jag för process för att göra, följa upp och förbättra uppskattningar jag gör?
- Vilken roll vill jag ha i ett team?

Exempel på saker man inte skall göra

- Läsa specifikationen inifrån och ut
- ”Make it right the first time”
- Dela upp arbetet i N delar och så ses vi om 1 månad
- Kuppa eller slarva med förankring
- Slåss mot alla bossar samtidigt
- Tävla
- Jobba parallellt på samma sak
- Inte ha väldefinierade roller / ansvar

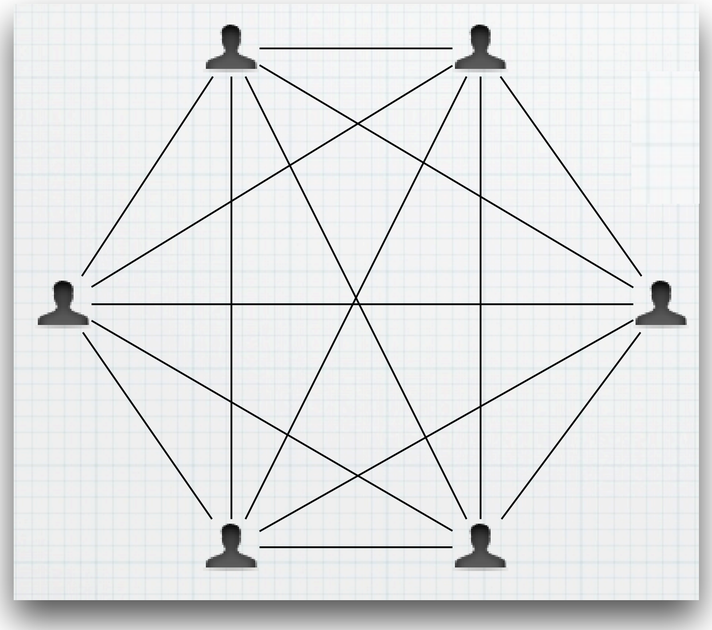
Hur kan man säkerställa leverans på utsatt datum?





Ett team kan inte vara produktivt från ruta 0

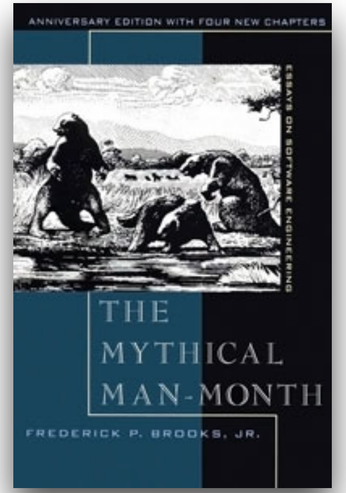
Forming → Storming → Norming → Performing

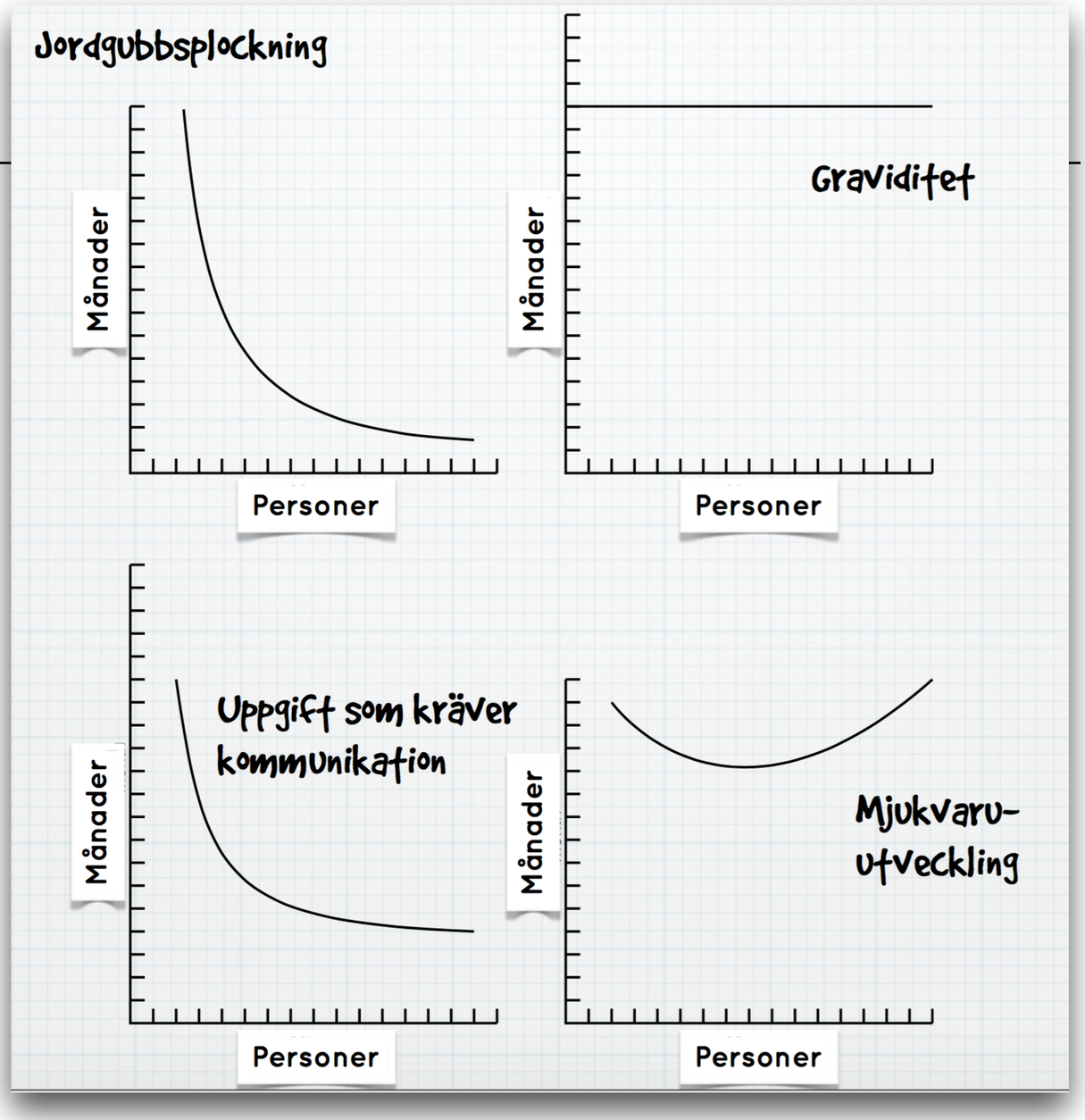


Med varje ny medlem växer kostnaden för koordination och kommunikation

Brooks lag

> Adding more manpower to a project that is already late is only going to delay it more





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	They Really Value	We Fucking Do
Individuals and interactions	*Tons of billable hours*	Programming, Motherfucker
Working software	*Tons of pointless tests*	Programming, Motherfucker
Customer collaboration	*Bleeding clients dry*	Programming, Motherfucker
Responding to change	*Instability and plausible deniability*	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.

Hur lastbalanserar man så att ingen blir utbränd och så att alla får vara med?

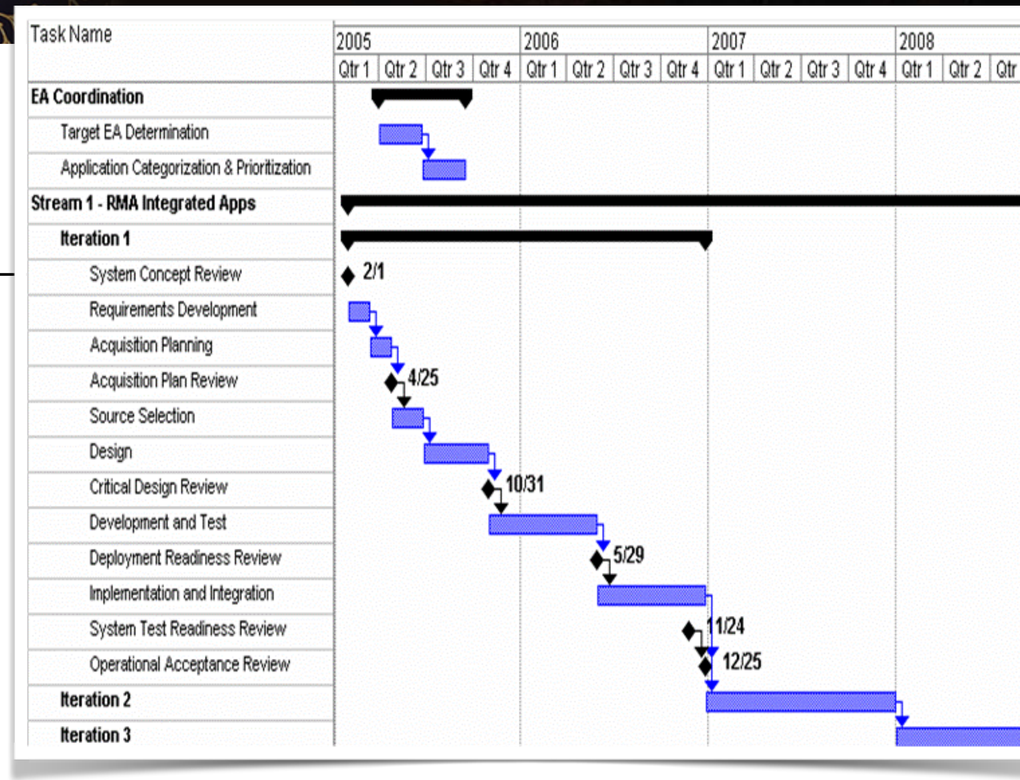
1. Bicycle Work Breakdown Structure (WBS)
   - 1.1. Bicycle Product (Deliverable) Scope
     - 1.1.1. Frame Set
       - 1.1.1.1. Frame
       - 1.1.1.2. Handlebar
       - 1.1.1.3. Fork
       - 1.1.1.4. Seat
     - 1.1.2. Crank Set
       - 1.1.2.1. Pedals
       - 1.1.2.2. Cranks
       - 1.1.2.3. Bearing Assembly
     - 1.1.3. Wheels
       - 1.1.3.1. Front Wheel Assembly
         - 1.1.3.1.1. Front Rim
         - 1.1.3.1.2. Spokes
         - 1.1.3.1.3. Hub
       - 1.1.3.2. Rear Wheel Assembly
         - 1.1.3.2.1. Rear Rim
         - 1.1.3.2.2. Spokes
         - 1.1.3.2.3. Hub
     - 1.1.4. Braking System
       - 1.1.4.1. Front Brakes
       - 1.1.4.2. Rear Brakes
     - 1.1.5. Shifting System
       - 1.1.5.1. Front Gear Shift System
       - 1.1.5.2. Rear Gear Shift System
   - 1.2. Supporting Project Scope
     - 1.2.1. Project Management
     - 1.2.2. Safety
     - 1.2.3. Systems Engineering
       - 1.2.3.1. Integration
       - 1.2.3.2. Testing
       - 1.2.3.3. Documentation

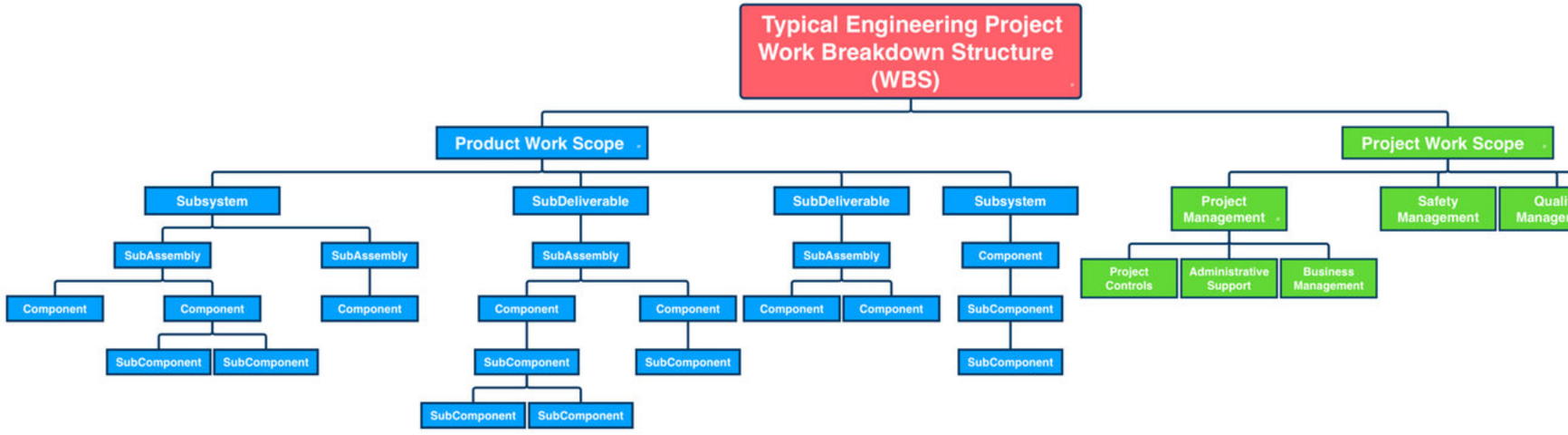
- Work-breakdown structure

  Vilka är de ingående delarna?

  Hur hänger de ihop?

  Kan visualiseras och organiseras på många sätt!





Hur lastbalanserar man så att ingen blir utbränd och så att alla får vara med?

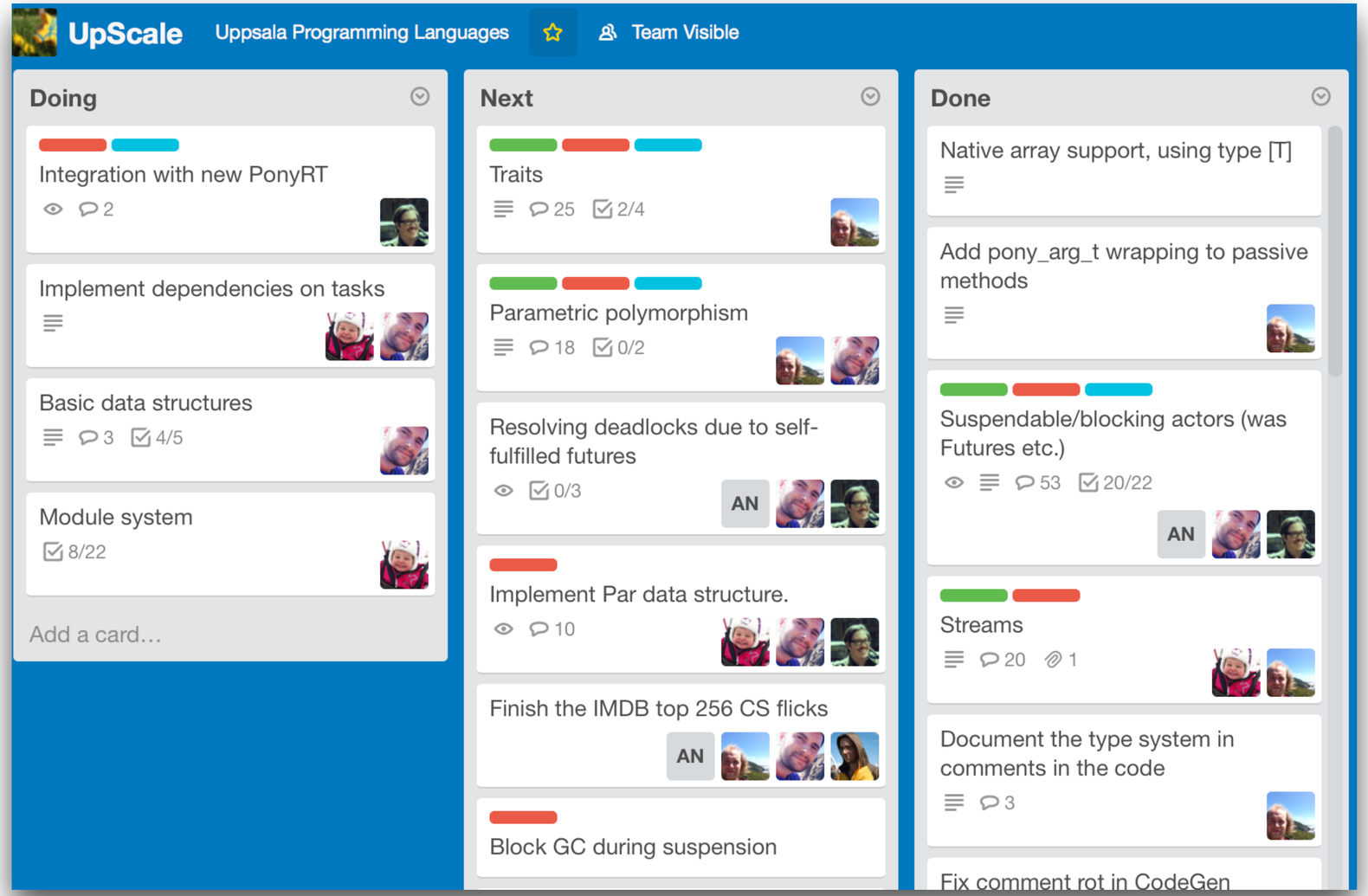
- Använd t.ex. Trello för att organisera information om ”tickets”
- Om det alltid finns något i ”next” så finns det alltid något att göra för den som är klar

  Vem prioriterar next?
- Minskar behov att planera vem gör vad i förväg — kan anpassa sig
- Lätt att se vem som gör vad

  …att alla är ”i spel”

  …vem som är överbelastad

  …vem man skall prata med om X
- Kan senarelägga nedbrytning av tickets



Hur vet vi att det vi levererar fungerar korrekt?

- Testning!
  - Enhetstester
  - Integrationstester
  - Systemtester
  - Stresstester
- Lägg ned tid initialt på att göra det enkelt att
  - Köra tester
  - Lägga till tester
  - Flagga till rätt person när ett test inte passerar

Låt 1–2 person(er) lösa detta en gång och för alla

Hur kan man kommunicera effektivt i ett team?

- Bestäm vilka kanaler som används och håll er till dem

  Se till att det är entydigt vad som skall kommuniceras via vilken kanal

  Gör det möjligt för de som inte kan närvara vid ett möte att ta del av diskussion / beslut

- Synkron och asynkron kommunikation

  Slack / Mattermost / …

  Epost

  Zoom / Discord / …

- Alla måste inte vara med på alla möten

  Men håll inte någon utanför!

- Låt inte skit ligga
- Don’t be the guy in the room
- Var ärlig med förutsättningar, kompetens, känslor etc.

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

Vad håller mig motiverad?

- The four major factors involved in satisfying programmers
  - Material reward and opportunities
  - Challenge and interest in the work itself
  - Employee benefits, working conditions and status of the organisation in its context
  - Competence of supervisors and leaders
- Variation in the programming task can be a bigger issue than salary
- Not uncommon for a programmer to quit right after a raise
- Social needs
  - Time to meet co-workers
- Esteem needs
  - Be valued by the organisation
  - Monetary rewards
- Self-realisation
  - Responsibility for their work
  - Demanding but not impossible tasks
  - Provide training to develop skills

Photo by Jordan Whitfield on Unsplash

Task-Oriented People

Motivation derived from the task at hand — the work is interesting

Self-Oriented People

Motivation derived from personal goals — the work pays well (which allows me to buy more guitars), or I am recognised as a great software designer, etc.

Interaction-Oriented People

Motivation derived from interaction with other people — going into work is fun

Unmotivated people are unproductive and demotivation can be contagious!

Gemensamma referensramar: Vad är en bra programmerare?

Gemensamma referensramar: 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

Gemensamma referensramar: 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?

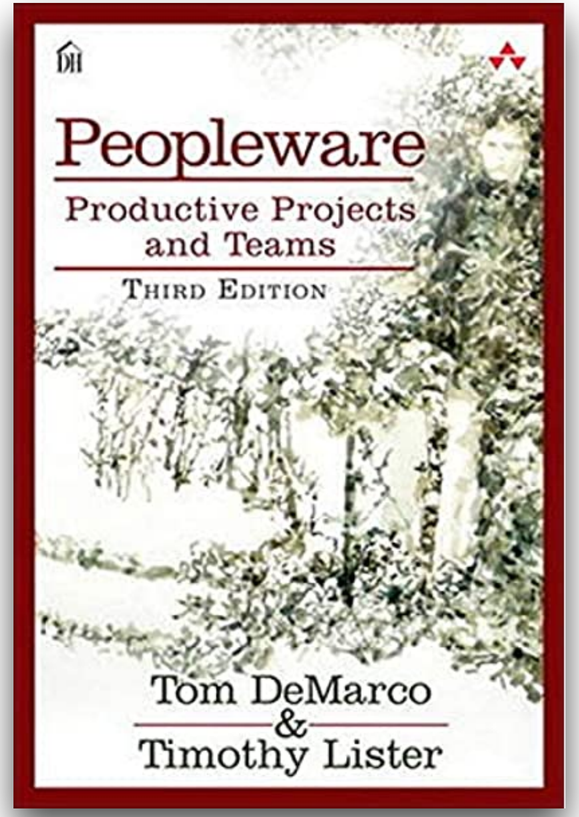
Gemensamma referensramar: Vad är bra kod? [recap]

Gemensamma referensramar: 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

Kodkriget [Tom DeMarco & Tim Lister]

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



Utfallet

- Valet av programmeringsspråk är mindre viktigt än vem som sitter framför tangentbordet (med untantag 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

Projektet

Domän: automatisk minneshantering i C

http://wrigstad.com/ioopm/projects/

Varför?

- Domän som alla kan
- Stålbad i pekarhantering

Projekt 1 (lättare)

- Implementation av API för referensräkning för godtyckliga C-program
- Användare manipulerar själva referensräknaren via API:t

Projekt 2 (svårare)

- Implementation av GC för godtyckliga C-program
- Biblioteket sköter minneshanteringen automatiskt

Fakta om projekten

- 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]

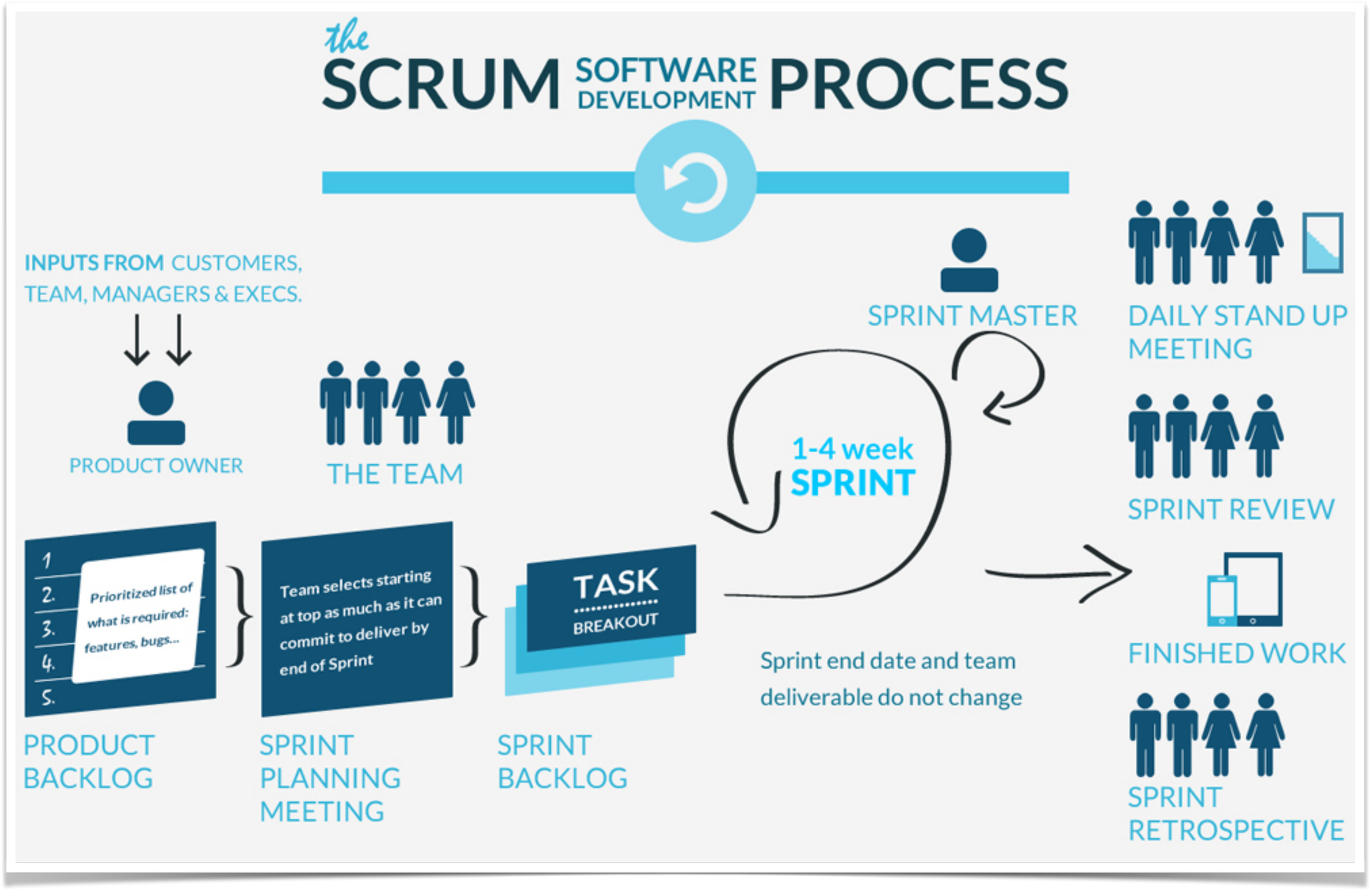
  Koordinera ett samarbete över tid

  Plattformsoberoende 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**



Lättrörligt

Enkelt att lastbalansera

Ingen lång uppstartssträcka

Passar explorativ ansats

Vanligt ”där ute”

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.

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

Vad skall lämnas in? [se http://wrigstad.com/ioopm/projects/ för detaljer]

- Högnivådesign
- Avvikelser
- Kod och tester och byggskript
- Rapport om kodkvalitet
- Testrapport
- Individuella reflektioner från varje projektmedlem
- Gemensam reflektion från teamet

Vad du skall tänka på under projektet

Vad ni skall tänka på under projektet

Vad händer om man inte blir godkänd?

- En grupp som inte lämnar in något i tid eller som lämnar in något som är alldeles för långt från målet blir underkänd och får komma tillbaka under 2021
- En grupp som lämnar in något som har enstaka eller smärre brister får en rest
- Bra att kommunicera och få acceptans för avsteg från spec löpande (se kurswebben)
- Projektarbetet examineras genom Y-målen och dessa redovisas i de olika dokumenten som lämnas in + seminarium

  Typiskt avser en rest att fixa till något enstaka av Y-målen
- Slutseminarium sista veckan på terminen

  Varje grupp deltar i 90 minuter

  Gör gemensam presentation inklusive demo, deltar på 2 andra gruppers presentation och deltar i diskussion