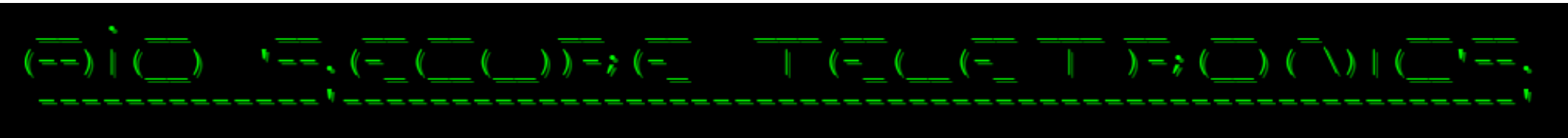
Scrum & agila arbetsmetoder

En Scrum-buffé

Om Joakim

- 41 år
- Skrev mitt första dataspel för 33 år sedan
- Dataingenjör 2006
- Särbo
- Katt-ägd
- Musiker
- Gokartnörd
- Lagar och renoverar 80-talstekonologi

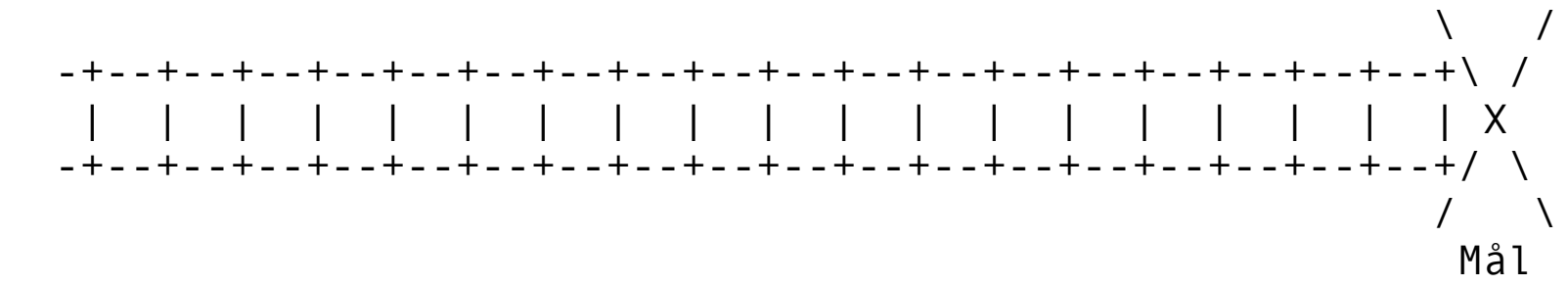


Vad är Agile och LEAN?

* Sätt att optimera processer för flöde av värde (kundnytta) och förändring.
  * I LEAN pratar man om "Stream Value Mapping" som styrmetod
    - Man tittar på led/ställtider mellan olika processer i produktionen
    - Försöker få ett pull-system där nästa delprocess tar sig ann uppgifter istället för att får uppgifter lagda på sig.
    - I slutändan handlar det om att få betalt för sina investeringar, men med fokus på tillfört kundvärde snarare än beläggning och effektivitet av resurser.
  * Kontinuerligt tillföra värde till kunden
    - Arbeta i korta inkrement som alltid är potentiellt levererbara.
  * Eliminera hinder för att tillföra värde
    - Vad kan jag göra för att förhindra att lager skapas mellan delprocesser?

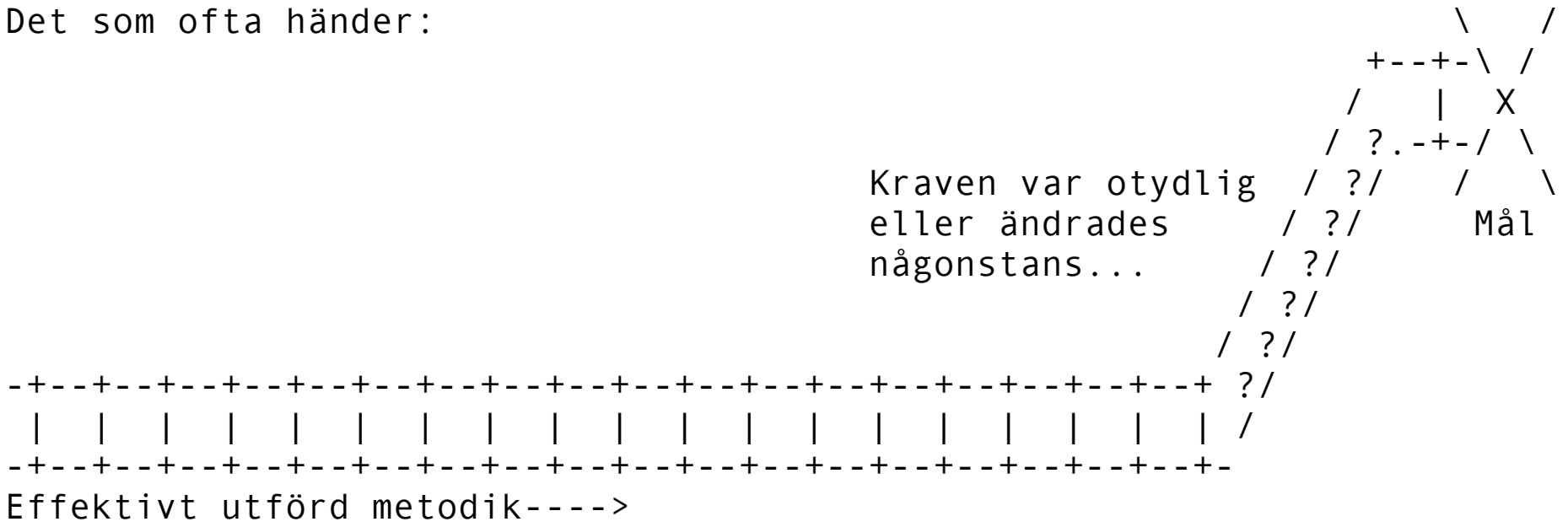
* Traditionella processer: Kan ses som en tågräls (ett effektivt sätt att förflyttas)

Om 12 månader ska vi vara "där borta":



... Och det skall gå så effektivt som möjligt!

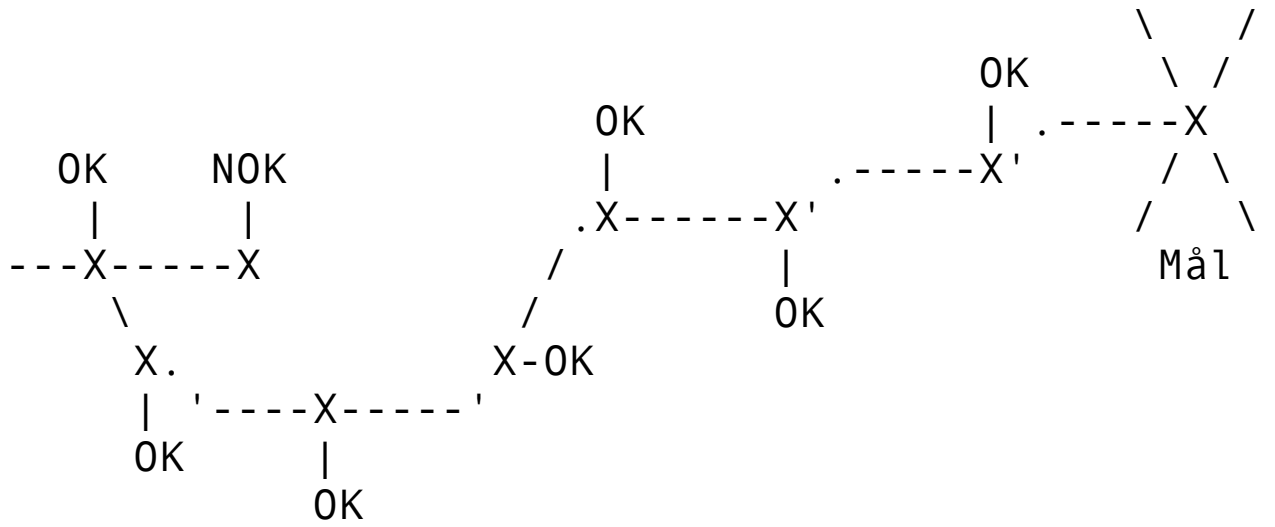
Det som ofta händer:



Göta kanal - Ett ingenjörsmästerverk!

Göta kanal – Går långsammare och är mer osäkert än ångloken!

Agilt tankesätt:



"Fail fast"

* Manifest för Agil systemutveckling

Vi finner bättre sätt att utveckla programvara genom att utveckla själva och hjälpa andra att utveckla. Genom detta arbete har vi kommit att värdesätta:

Individer och interaktioner framför processer och verktyg
Fungerande programvara framför omfattande dokumentation
Kundsamarbete framför kontraktsförhandling
Anpassning till förändring framför att följa en plan

Det vill säga, medan det finns värde i punkterna till höger, värdesätter vi punkterna till vänster mer.

Kent Beck
Alistair Cockburn
James Grenning
Ron Jeffries
Robert C. Martin
Jeff Sutherland
Mike Beedle
Ward Cunningham
Jim Highsmith
Jon Kern
Steve Mellor
Dave Thomas
Arie van Bennekum
Martin Fowler
Andrew Hunt
Brian Marick
Ken Schwaber

* Tolv principer (Liter mer mjukvaruperspektiv)

"Vi följer dessa principer", sa de kloka personerna:

1. Vår högsta prioritet är att tillfredsställa kunden genom tidig och kontinuerlig leverans av värdefull programvara.

2. Välkomna förändrade krav, även sent under utvecklingen. Agila metoder utnyttjar förändring till kundens konkurrensfördel.

3. Leverera fungerande programvara ofta, med ett par veckors till ett par månaders mellanrum, ju oftare desto bättre.

4. Verksamhetskunniga och utvecklare måste arbeta tillsammans dagligen under hela projektet.

5. Bygg projekt kring motiverade individer. Ge dem den miljö och det stöd de behöver, och lita på att de får jobbet gjort.

6. Kommunikation ansikte mot ansikte är det bästa och effektivaste sättet att förmedla information, både till och inom utvecklingsteamet.

* Tolv principer (Liter mer mjukvaruperspektiv)

7. Fungerande programvara är främsta måttet på framsteg.

8. Agila metoder verkar för uthållighet. Sponsorer, utvecklare och användare skall kunna hålla jämn utvecklingstakt under obegränsad tid.

9. Kontinuerlig uppmärksamhet på förstklassig teknik och bra design stärker anpassningsförmågan.

10. Enkelhet - konsten att maximera mängden arbete som *inte* görs - är grundläggande.

11. Bäst arkitektur, krav och design växer fram med självorganiserande team.

12. Med jämna mellanrum reflekterar teamet över hur det kan bli mer effektivt och justerar sitt beteende därefter.

* Fungerar detta bara för mjukvara? - Vi kommer till det senare...

Men under tiden... Låt oss titta på ett talande exempel på skillnaderna mellan traditionell process och agil process och vilka konsekvenser de båda kan ha för den som verkligen behöver något löst.

Traditionell processeffektivitet i sjukvård: resurseffektivitet (fokus ligger på att använda resurserna effektivt)

* Effektiva doktorer och maskiner som kan göra en sak väldigt bra och fort

* Patienten behöver vänta på att doktorerna blir tillgängliga

* Kliniker som är specialiserade på specifika undersökningar

* Patienten behöver vänta på remisser

Karina har upptäckt en knöl i bröstet och vänder sig till vårdcentralen. Distriktsläkaren undersöker och sänder remiss till bröstkliniken. Hon kallas till mammografi och senare till bröstkirurgen, som sänder en ny remiss till cytologen. Sex veckor efter första besöket får hon besked. Karinas flöde från första läkarbesöket till diagnos tog 42 dagar.

En vård där LEAN fungerar: flödeseffektivitet (fokus sätts på den enhet som förädlas i verksamhetens flöden).

* En klinik med ett team av duktiga doktorer som jobbar tillsammans

* De kan olika områden men fokuserar på bröstcancer

* All utrustning på plats på kliniken som behövs för att undersöka och behandla bröstcancer

* Patienten får diagnos och behandling på samma plats.

Eva känner också en knöl i bröstet och besöker en klinik där man samlat all bröstcancerverksamhet. Efter en undersökning av läkaren görs en finnålspunktion och provet skickas till laboratorium. Evas flöde tog 2 timmar.

För den som vill lära sig mer om detta exmpel:

"Detta är LEAN" av Niklas Modig och Per Åhlström, jobbat åt Ericsson m.fl.

http://www.dettaarlean.se

Vad är det för skillnad på "effektiv"?

* Engelskans effective och efficient

Effective (adj.): Adekvat att uppnå ett syfte, producera vad som förväntades eller uppnår det förväntade resultatet.

Efficient (adj.): Utför eller fungerar på bästa möjliga sätt med minsta slöseri av tid och energi.

Lite förenklat skulle man kunna säga att Effective innebär att göra rätt sak, efficient innebär att göra på rätt sätt.

Eller så kan vi kalla Effective för ledtidseffektiv och efficient för ställtidseffektiv, enligt tidigare exempel.

* Vi (inte alla, men många) tekniskt inriktade satsar ofta på att göra praktiska saker på ”rätt sätt”...

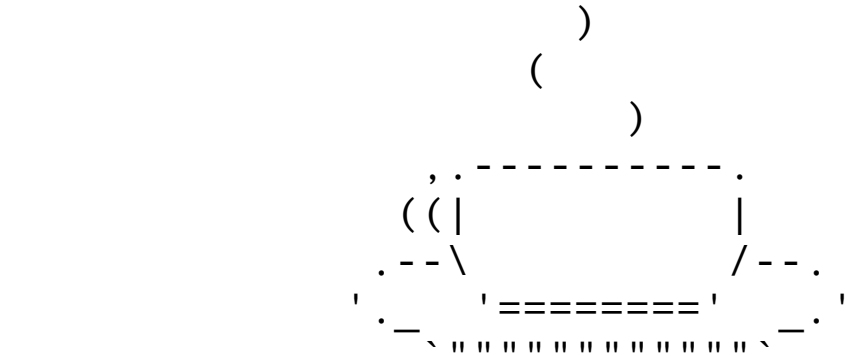
  ... Så motsatsen kan vara lite icke-intuitiv och kanske något vi behöver träna på lite för att bli både "effektiva" och "effektiva". :)

* Agile och LEAN (Scrum och Kanban söker efter både efficient och effective)

* Kanban och Scrum är ramverk för lättviktiga processer som (om korrekt införda) bör kunna utföras snabbt och korrekt (efficient) för att få mer tid att göra rätt sak åt kunden (effective).

* Kanban och Scrum är tänkta att göra det enkelt att byta riktning när kraven från omvärlden förändras, så att man inte designar in ett potentiellt felaktigt slutmål i produkterna från början

* Scrum är ett bra ramverk ihop med eXtreme Programming eller eXtreme Manifacturing för att med korta intervall kunna visa och potentiellt Leverera ett produktinkrement. D.v.s. fokus på fungerande produkter, hela tiden.



- Finns det något ni känner igen?
- Har ni stött på arbetsplatser där man applicerat Agile?
  - Har det fungerat?
- Hur arbetar ni idag?

Historian om FPGA-teamet 2008 (Kanban)

* Sju personer, 25-57 år
* Aldrig klara i tid (i alla fall inte kvalitetssäkrat)
* Massor med övertid framåt slutet på projekt
* En teammedlems hjärtinfarkt
* Vår första backlog
* Vi lärde oss "hållbar hastighet"

Tre rattmuffar av fem

Historien om ComSwitch-teamet (Scrum)

* Backlog och storynedbrytning
* Projektstart
* Estimering
* OK på tidsåtgång med guldkanter borttagna
* Diagram
* Vi gjorde det, utan en enda timmas övertid
* Korta sprintar
* Åka runt i organisationen och dela med oss.

Nio skummtomtar av tio

Historien om processor-design-teamet (Kanban)

* Produktägare i teamet
* Svårt med små inkrement
* Lika personligheter
* Ingen estimering (taktik-röstning) överanalys stor diff
* Missnöje av projekt och linje
* Blev aldrig klara, trots långa sprintar
* Global Scrum Gaterhing - Berlin

Två Nintendo-konsoller av fem

Historien om verktygs-teamet (Scrum)

* Global Scrum Gaterhing – Prag
* Initial misstro
* Tog själva på oss rollen att prata med kunder
* Produktägare i teamet
* Olika personligheter
* Total transparens, lyfta konflikter
* Små inkrement och korta sprintar
* Estimat som skiljde 1-2 poäng mellan alla individer
* 5min sprintplanering, 30min backloghantering, 1tim retrospektiv
* Fungerande demo

Åtta hallongrottor av tio

Scrum/Kanban som hjälpmedel

* Något att bli efficient på, så man kan bli mer effective på att framställa sin produkt.

* Se till att eliminera konflikt-roller

  - Lönesättande chef ska inte vara Scrum Master
  - Produktägare skall inte vara Scrum Master
  - Produktägare skall (helst inte) vara implementatör

* Sätter ficklampan på organisatoriska problem (smärtsamt) (impediments)

* Självorganiserande team - jobba med tillit och mod

* Co-location, co-location, co-location (Mtp: Agile Manifesto)

* Sprint-backlog/Produkt-backlog/Visualisering/Transparens

  - Tydlig produktvision
  - Backlog grooming
  - Sprintplanering (empiriskt åtagande)
  - Korta dagliga möten
  - Sprint burn-down-chart för att på en sekund kunna se hur det går

  Visualiseringen är mycket viktigt för tilliten för ett självorganiserande team, och för att slippa förklara på timslånga möten för hierarkin.

Vilket passar i just min sorts utveckling?

* Förmodligen funkar både Scrum och Kanban, beroende på behovet av sprintar

* Scrum är nog mer lämpligt för nyutveckling och hanterar förändring aningen bättre än Kanban

* Kanban är nog mer lämpad för repetitiv utveckling eller som ett första Steg i en agil transformation för att börja jobba med momenten i en Lättviktig process.

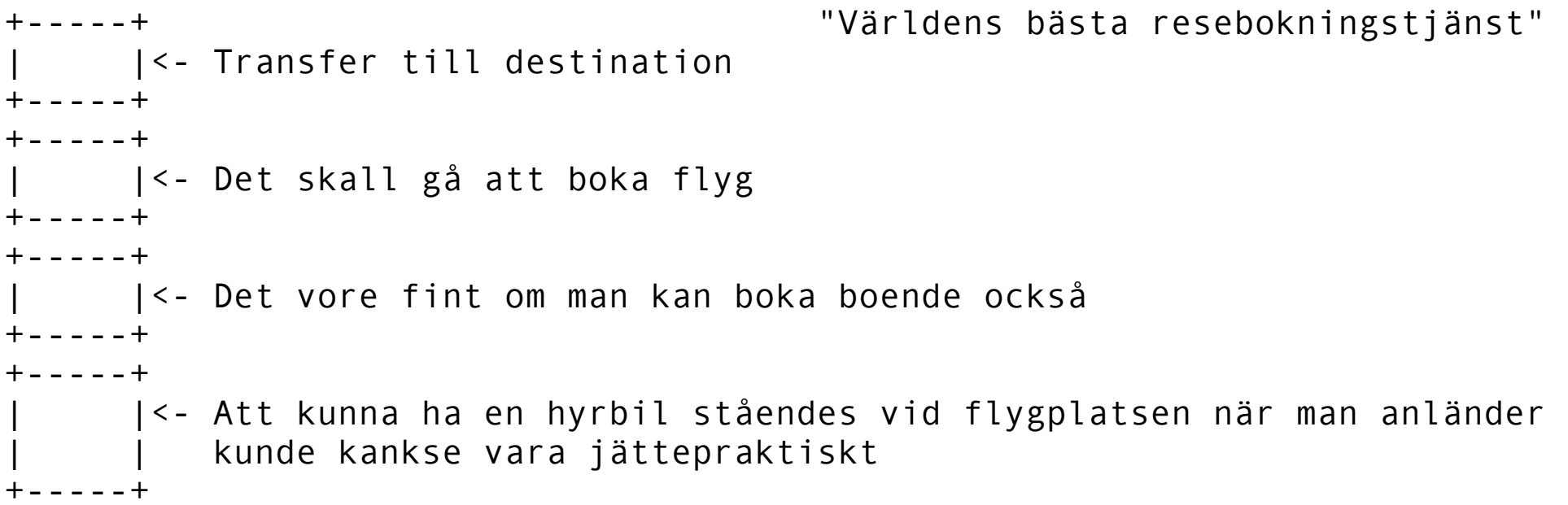
Roller

* Implementatör ”Team”
  * Ansvarar för att organisera sig själva
  * Många olika kompetensområden (”Tvärfunktionellt”)
  * 3-9 personer

* Scrum Master
  * Fokuserar på att Agile-principerna följs
  * Coach och facilitator
  * Lyhörd för vad som hindrar implementatörerna och försöker avlägsna hinder

* Produktägare
  * Representerar kunder och håller i den långsiktiga visionen
  * Måste vara produktexpert

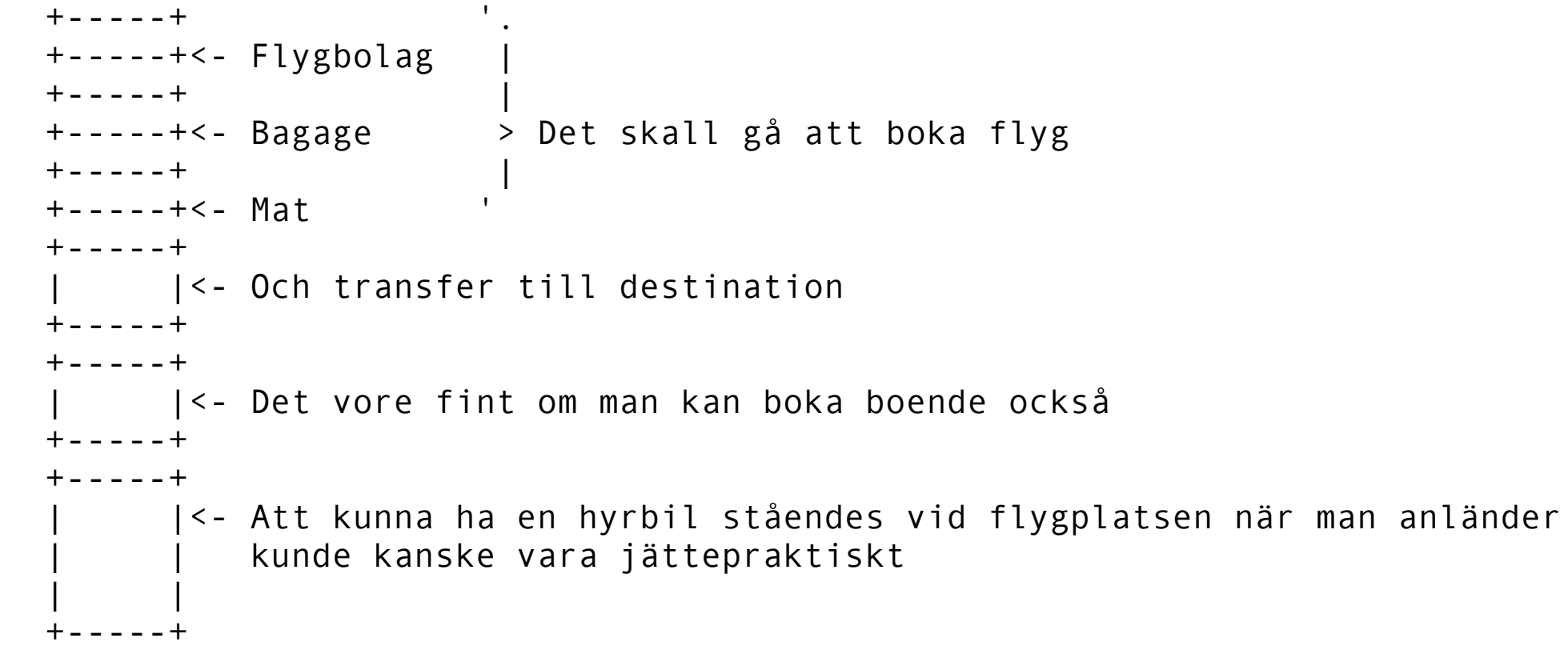
Produktvision och backlog



Backlogen tas fram av kund, produktägare och implementerande team

* Prioritering/Visualisering - Ägna inte för mycket tid på planering

  * Små och mer detaljerade stories i toppen av prioritering



Nedbrytning i user-stories görs av Scrum-teamet (implementatörer + produktägare), prioritering av stories görs av produktägaren ihop med kunden

* Planering/Visualisering

  * Detta är det vi skall jobba med härnäst

    Sprint-backlog: Slogan: Det skall gå att boka flyg

    +-----+
    +-----+<- Flygbolag [] [] [] [] []
    +-----+
    +-----+<- Bagage [] [] []
    +-----+
    +-----+<- Mat [] []

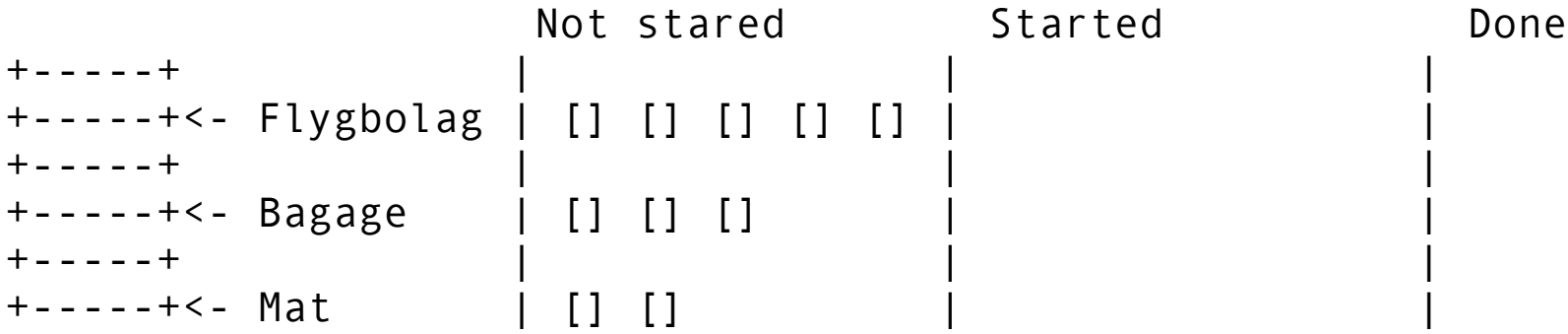
Till största delen sköter implementerande delen av teamet sprintplanering och vidare nedbrytning i små tekniska-detaljer. Med stöd av och insyn för produktägaren.

Redan nu kan teamet kalla till demo för kunden till sprint-slutet.

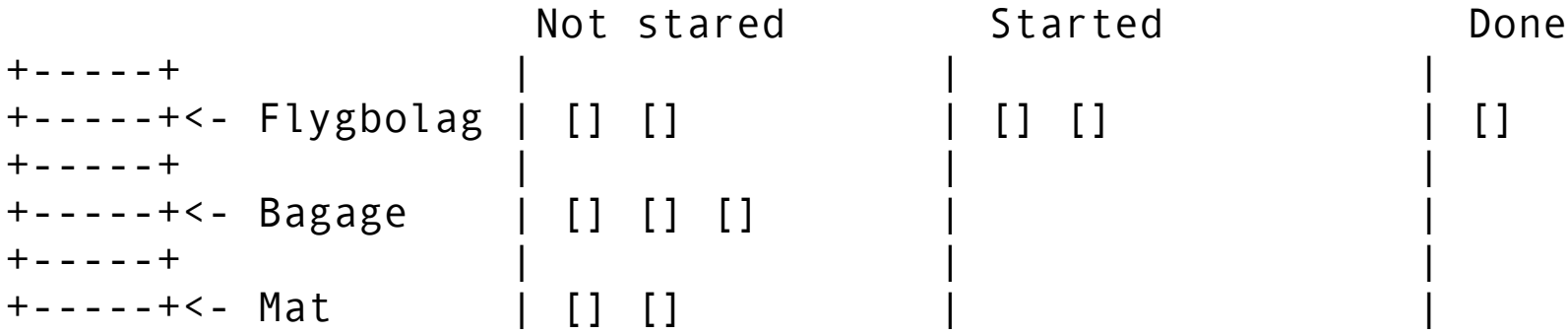
* Pågående arbete/Visualisering

  Tanken är att det skall gå att få en idé om hur det går för utvecklings-
  teamet med ett snabbt ögonkast på någon fysisk plats. I Scrum har man
  en så kallad Scrum-Board med lappar som flyttas allteftersom man förädlar
  produkten mot nästa potentiellt leveransbara inkrement.

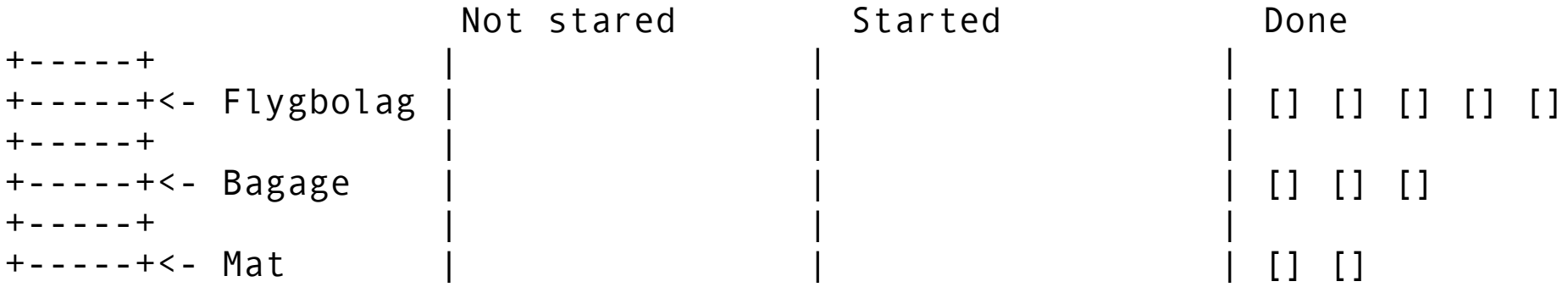
Daily Scrum



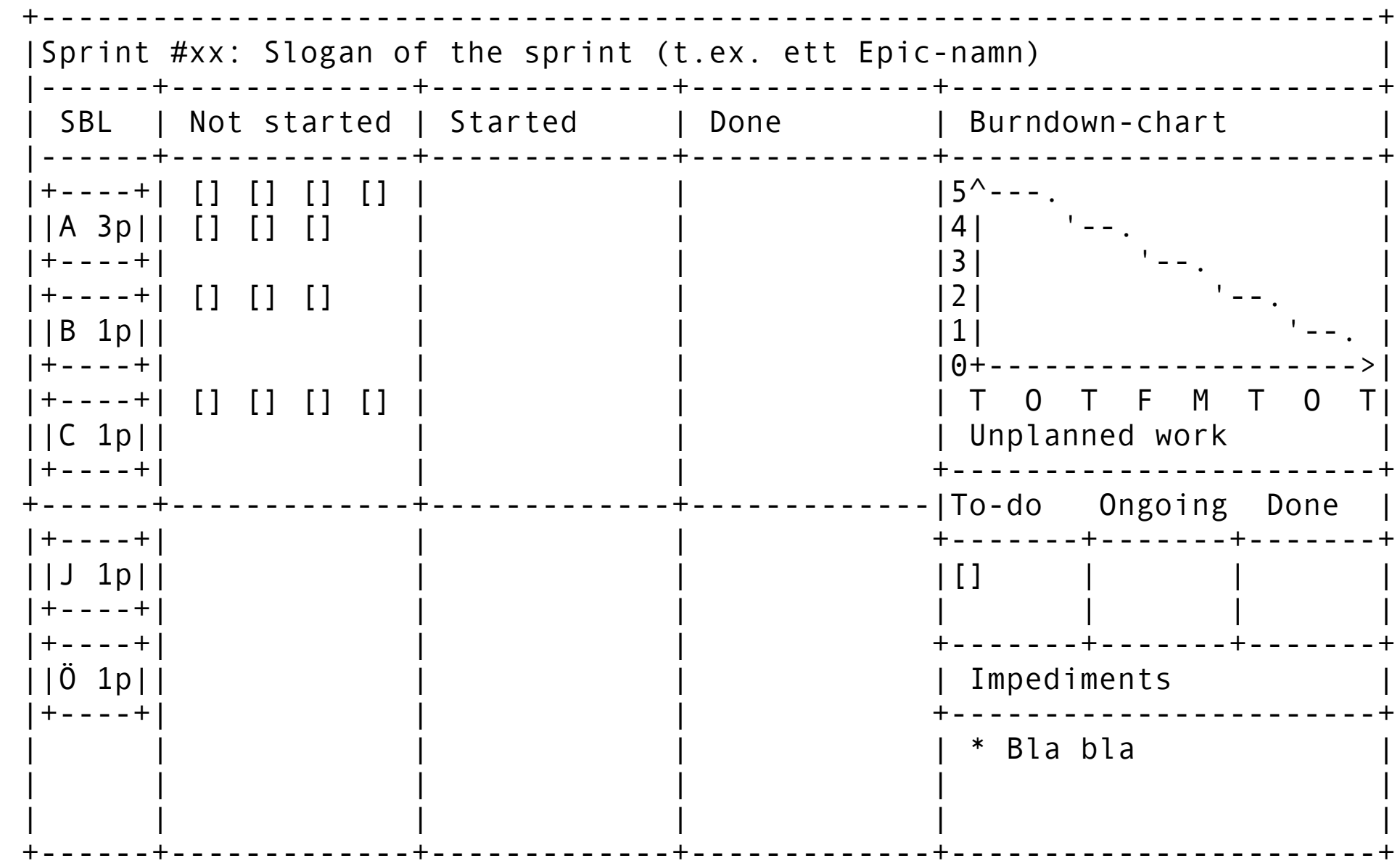
Daily Scrum



Daily Scrum



Håll ordning på oplanerat arbete och hinder (impediments) också.



--- BOK: Scrum And XP from the trenches

* Demonstration/Visualisering

  * Kolla att allt fungerar i det som vi åtog oss denna iteration!

  * Sprint Review

  * Kan hållas av utvecklingsteamet, scrum master, produktägare eller till och med kund.

* Kanban
  * Minska antalet byten av arbetsuppgifter: Work-In-Progress Limit
  * Påvisa trånga sektioner i produktionen
  * Hitta och hantera hinder
  * Man kan ha traditionella roller, men öka visuella återkopplingen
  * ... eller så adopterar man scrum-rollerna.

* Scrum
  * Sprintar med process-rutiner som återkommer (2-4 veckor)
  * Roller
    * Produktägare (PO = Product Owner)
    * Scrum Master
    * Utvecklande team (5-9 personer)
  * Ceremonier
    * Sprint planning
    * Daily scrum
    * Sprint-review
    ---
    * Retrospective
    * Backlog grooming/refinement
  * Förutsägbarhet
  * Hitta och hantera hinder

eXtreme Programming

- Återkoppling
  - Parprogrammering
  - Test-driven design
  - "Planning game"
  - Hela teamet
- Kontinuerlig process
  - Kontinuerlig integration
  - Små releaser
  - Kodningstandarder
  - Enkel design
  - Refaktorering eller designförbättring
  - Delat förstående
  - Gemensamt äga kodbasen
  - System-metaforer
- Välbefinnande
  - Hållbar hastighet
- Programmering
  - Kunden är alltid tillgänglig
  - Endast ett par integrerar kod
  - Ingen övertid
  - Utveckla enhetstestet först
  - Lämna optimering till sist
- Testning
  - All kod måste ha enhetstester
  - All kod måste passera alla enhetstester innan leverans
  - När en bugg hittas skapar man testfall före man börjar jobba med buggen. (En bug är inte ett logikfel, det är ett testfall som saknas)
  - Acceptanstester körs ofta och resultat visualiseras

eXtreme Manifacturing

* 10 principer
  * Optimera för förändring
  * Objektorientering (Modulär arkitektur)
  * Test-driven design (Röd -> Grön -> Refaktorera)
  * Contract-first design
  * Iterera designen
  * Agila hårdvarudesign-mönster
  * Kontinuerlig integration
  * Kontinuerlig leverans
  * Skalande mönster (Varje team har ansvar för integration och test av en del-modul andra får göra wrappers/gränssnitt för "stubbar" så länge)
  * Partner-mönster (Wrappers/gränssnitt för utbytbara delar - d.v.s. ingen sub-optimering)

* Motivation är kanske en av de mest produktivitetsökande faktorerna

  --- BOK: Joy Inc.

* WikiSpeed

* RSAnimations - Drive, the suprising truth about what motivates us

* Lite om era erfarenheter som utvecklare hos olika kunder