

# **Att arbeta i projekt**

## **En kort introduktion till software engineering**

Presenteras av Morgan Ericsson (@morganericsson)



# **Idag**

**Hur arbetar man **tillsammans**?**

**Minimera slöseri, maximera **tid, kunskap, färdighet, ...****

**Kort introduktion till **software engineering, utvecklingsprocesser** och **varför** dessa behövs.**

**Hopplock av saker, teori kommer i senare kurser.**

**Hur använder man bäst **git** och **gitlab**?**

# **Software Engineering**

# Enkelt?

**1. Vad** skall vi göra?

**2. Gör** det

**3. Gjorde** vi det?

(Bara git och python som krånglar till det...)

# Vad skall vi göra?

## Taxes

In a (very) simplified version of the Swedish income tax system we have three tax levels depending on your monthly salary:

- You pay a 30% tax on all income below 38.000 SEK/month
- You pay an additional 5% tax on all income in the **interval** 38.000 SEK/month to 50.000 SEK/month
- You pay an **additional** 5% tax on all income **above** 50.000 SEK/month

Write a program `tax.py` which reads a (positive) monthly income from the keyboard and then prints the corresponding income tax.

# Gör det

```
def tax(income):  
    tx = income * 0.3  
    if income > 38000:  
        tx += (income - 38000) * 0.05  
    if income > 50000:  
        tx += (income - 50000) * 0.05  
  
    return tx
```

# Gjorde vid det?

```
assert tax(32000) == 9600  
assert tax(46000) == 14200  
assert tax(79000) == 27200
```

**Verkar funka? Men, är vår tolkning rätt?**

# Så?

**Även enkla problem kan vara tvetydiga.**

**Vår implementation kan innehålla buggar ...**

**... som vi kanske inte upptäcker.**

**Hur tror ni det ser ut när problemen  
blir **större** och **mer komplexa**?**

# Mjukvarukrisen

**Mjukvara var ineffektiv och svarade inte mot kraven.  
Projekt drog över tid och budget, och var ohanterbara.**

**Software engineering utvecklades som ett svar på  
mjukvarukrisen: “Application of engineering to software”**



*“The major cause of the software crisis is that the machines have become several orders of magnitude more powerful! To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem.”*

# “Engineering Seeks Quality”

**Målet med software engineering är att skapa mjukvara  
med hög kvalitet.**

**Men, mjukvara har en inbyggd komplexitet och  
mjukvarusystem är ibland så komplexa att vi inte har  
kapacitet att förstå dem.**

**Så, software engineering handlar till stor del om att  
*skapa enkelhet* (illusion of simplicity)**

# De tre stegen



(Vad skall vi göra?)

(Gör det)

(Gjorde vi det?)

En **sekventiell** mjukvaruutvecklingsprocess.

# ... blir ofta fem

Krav

Implementation

Evolution



Design

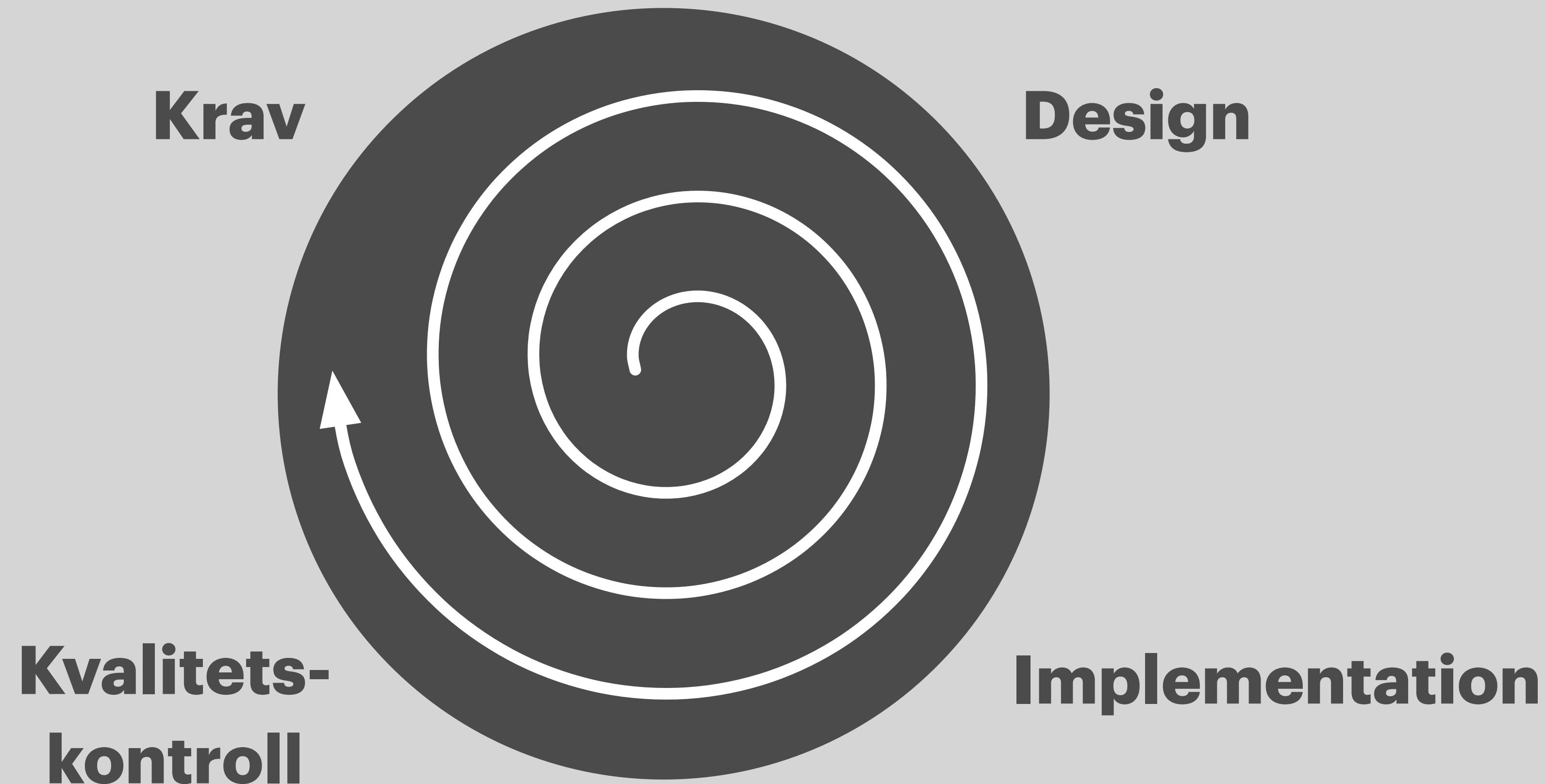
Kvalitetskontroll

En **sekventiell** mjukvaruutvecklingsprocess.

# De fem stegen

1. **Krav** handlar om att förstå problemdomänen och användarna.
2. **Design** handlar om att skapa en lösning, det vill säga välja teknik, algoritmer, interaktionsmodeller, ...
3. **Implementation** handlar om att realisera designen.
4. **Kvalitetskontroll** handlar om att kontrollera att implementationer uppfyller kraven, har kvalitet, ...
5. **Evolution** handlar om att åtgärda fel och brister, införa ny funktionalitet, ...

# Förändring sker hela tiden



Processen är snarare **iterativ** på grund av **förändring** och **evolution** innehåller ofta alla steg...

# Produktion eller skapande?



**Definierad eller empirisk process?**

# Kanban

# Men! Hur gör man?

**Det finns flera olika metoder, t.ex. vattenfallsmodellen,  
spiralmodellen, agile, lean, scrum, ...**

**Dessa kommer i mer detalj i senare kurser.**

**Vi antar en **empirisk** process, som blandar från några kända,  
t.ex. Lean/Kanban, XP, osv.**

# Något om krav...

**Krav kommer att behandlas i detalj i en senare föreläsning.**

**Just nu antar vi att:**

- **det finns en mängd krav som skall uppfyllas.**
- **denna mängd kan förändras under projektet.**
- **kraven ordnas i en product backlog**
- **kraven kan innehålla prioritet och något mått på komplexitet.**

# Exempel

*“De 10 mest köpta produkterna de senaste 30 dagarna ska visas på förstasidan.”*

# “Product Backlog”

“Product Backlog” innehåller de krav som skall implementeras.

Är **aldrig fullständig**, bara en uppskattning av de krav som skall implementeras. Den kan och kommer att **växa** och **förändras** under projektets gång.

Lever så länge projektet lever. När den är tom är ni klara. Så, fokus på att arbeta med krav från den.

# “Product Backlog” på gitlab

The screenshot shows the GitLab interface for creating a new issue. The left sidebar lists various project management sections: Project overview, Repository, Issues (3), Boards, Labels, Service Desk, Milestones, Iterations, Merge Requests (0), Requirements, CI / CD, Security & Compliance, Operations, and a Collapsible sidebar. The main area is titled "New Issue". It includes fields for "Title" (with a placeholder "Add description templates to help your contributors communicate effectively!"), "Type" (set to "Issue"), "Description" (with "Write" and "Preview" tabs, a rich text editor toolbar, and a note "Write a comment or drag your files here..."), and "Markdown and quick actions are supported" (with an "Attach a file" button). Below the description is a checkbox for "This issue is confidential and should only be visible to team members with at least Reporter access." Further down are fields for "Assignee" (set to "Unassigned" with an "Assign to me" link), "Weight" (with a placeholder "Enter a number"), "Milestone" (with a dropdown menu), "Due date" (with a placeholder "Select due date"), and "Labels" (with a dropdown menu).

# “Product Backlog” på gitlab

The screenshot shows the GitLab interface for the project "gitdemo2". The left sidebar is visible with options like Project overview, Repository, Issues (selected), Boards, Labels, Service Desk, Milestones, Iterations, Merge Requests, Requirements, CI / CD, Security & Compliance, Operations, and Collapse sidebar. The main area shows the "Issues" page with a header for Morgan Ericsson > gitdemo2 > Issues. It displays 3 Open issues, 0 Closed issues, and 3 total issues. The issues listed are:

- Lägga till produkt  
#3 · opened just now by Morgan Ericsson updated just now
- Förstasidan laddar långsamt.  
#2 · opened just now by Morgan Ericsson updated just now
- De 10 mest köpta produkterna de senaste 30 dagarna ska visas på förstasidan.  
#1 · opened 1 minute ago by Morgan Ericsson updated 1 minute ago

Below the issues, there are buttons for RSS feed, calendar, file upload, and download, along with links for Edit issues and New issue.

# Kanban

**Kanban är ett system för att hantera lager som har sitt ursprung i “Lean Production”.**

**Huvudtanken i Lean är att eliminera slöseri och på så sätt maximera värde. Att hålla saker i lager är slöseri. Kanban förespråkar att behov styr produktion och att det skall finnas ett jämt flöde.**

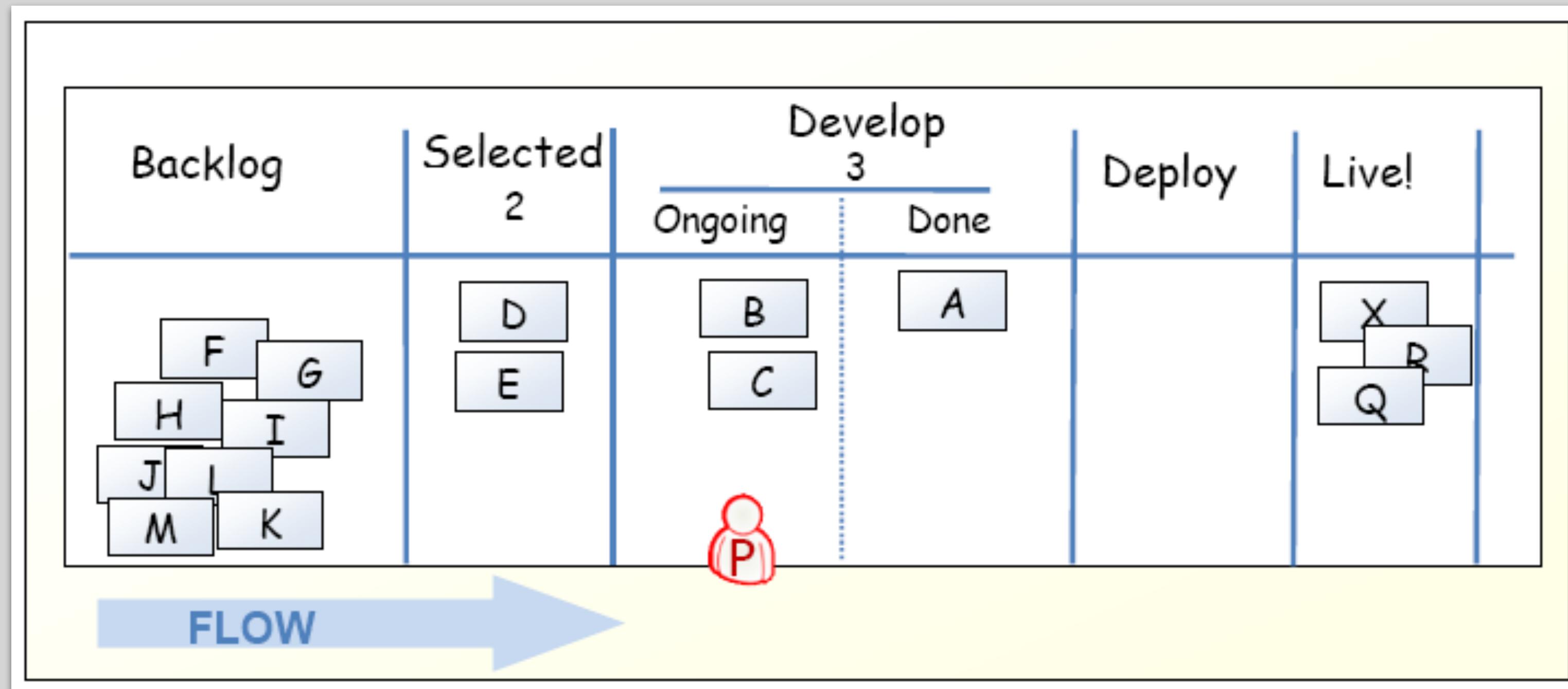
**Kanban är ett visuellt system som ursprungligen bygger på fysiska kort på en tavla, en Kanban.**

# Kanban och mjukvara

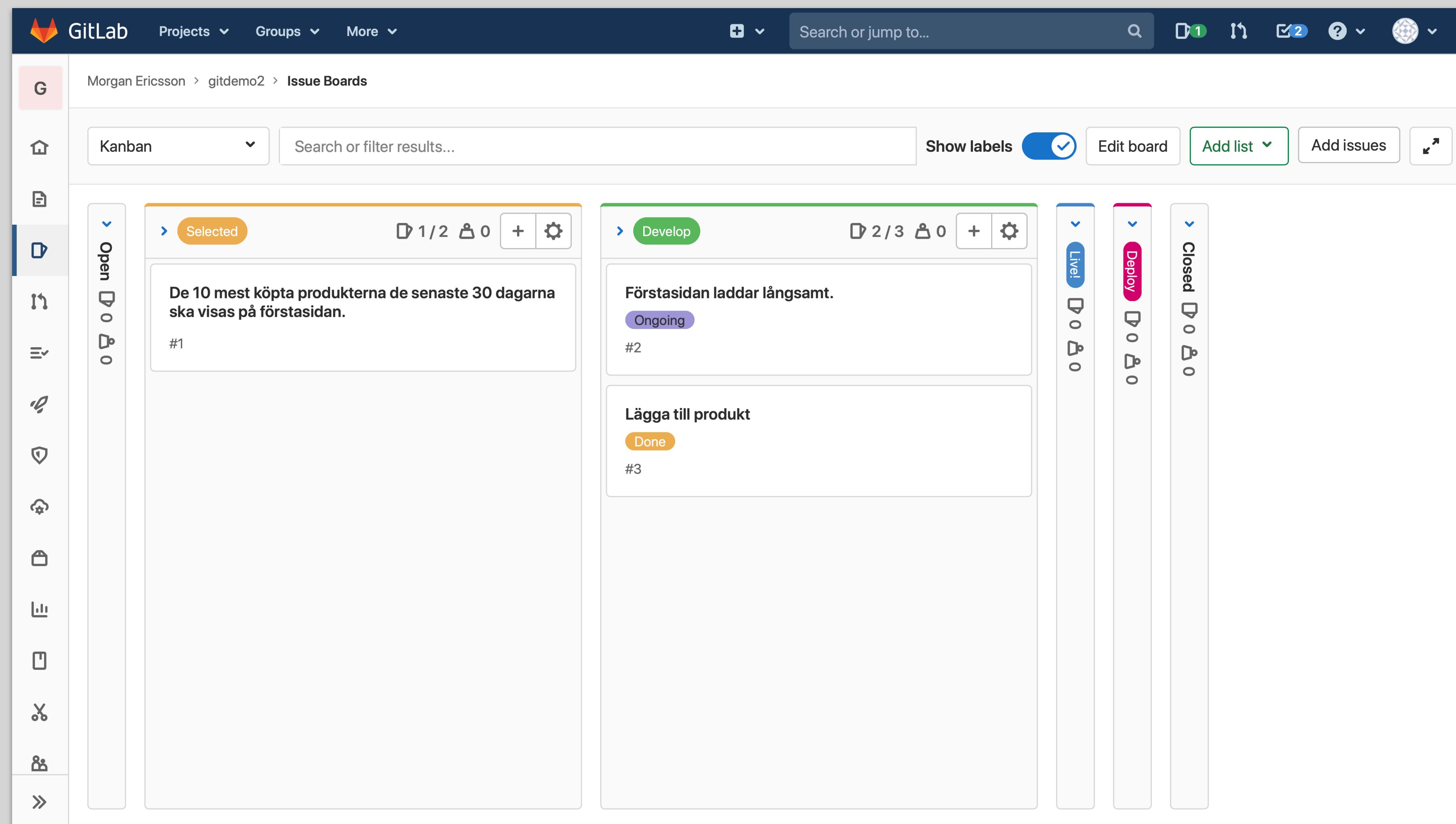
Med Kanban kan/skall vi:

1. **visualisera arbetet**
2. **begränsa hur mycket arbete som pågår vid en given tidpunkt.**
3. **kontinuerligt förbättra.**

# Exempel



# Gitlab



The screenshot shows a GitLab Issue Boards Kanban view for the project 'gitdemo2'. The board has four columns: 'Selected' (orange), 'Develop' (green), 'Live!' (blue), and 'Closed' (grey). The 'Selected' column contains one card with the text: 'De 10 mest köpta produkterna de senaste 30 dagarna ska visas på förstasidan.' and '#1'. The 'Develop' column contains two cards: one labeled 'Förstasidan laddar långsamt.' with status 'Ongoing' and '#2'; and another labeled 'Lägga till produkt' with status 'Done' and '#3'. The 'Live!' and 'Closed' columns are currently empty. On the left, a sidebar lists various project management icons. At the top, there's a navigation bar with links for 'Projects', 'Groups', 'More', a search bar, and user notifications.

Morgan Ericsson > gitdemo2 > Issue Boards

Kanban ▼ Search or filter results... Show labels ✓ Edit board Add list ▼ Add issues ◀

Selected ▶ 1 / 2 0 + ⚙️

De 10 mest köpta produkterna de senaste 30 dagarna ska visas på förstasidan.  
#1

Develop ▶ 2 / 3 0 + ⚙️

Förstasidan laddar långsamt.  
Ongoing  
#2

Lägga till produkt  
Done  
#3

Live! ▶ 0 0 + ⚙️

Closed ▶ 0 0 + ⚙️

G

Open ▶ 0 0 + ⚙️

Projects ▼ Groups ▼ More ▼

Search or jump to... 🔍

1 2 ? ◀ ▶

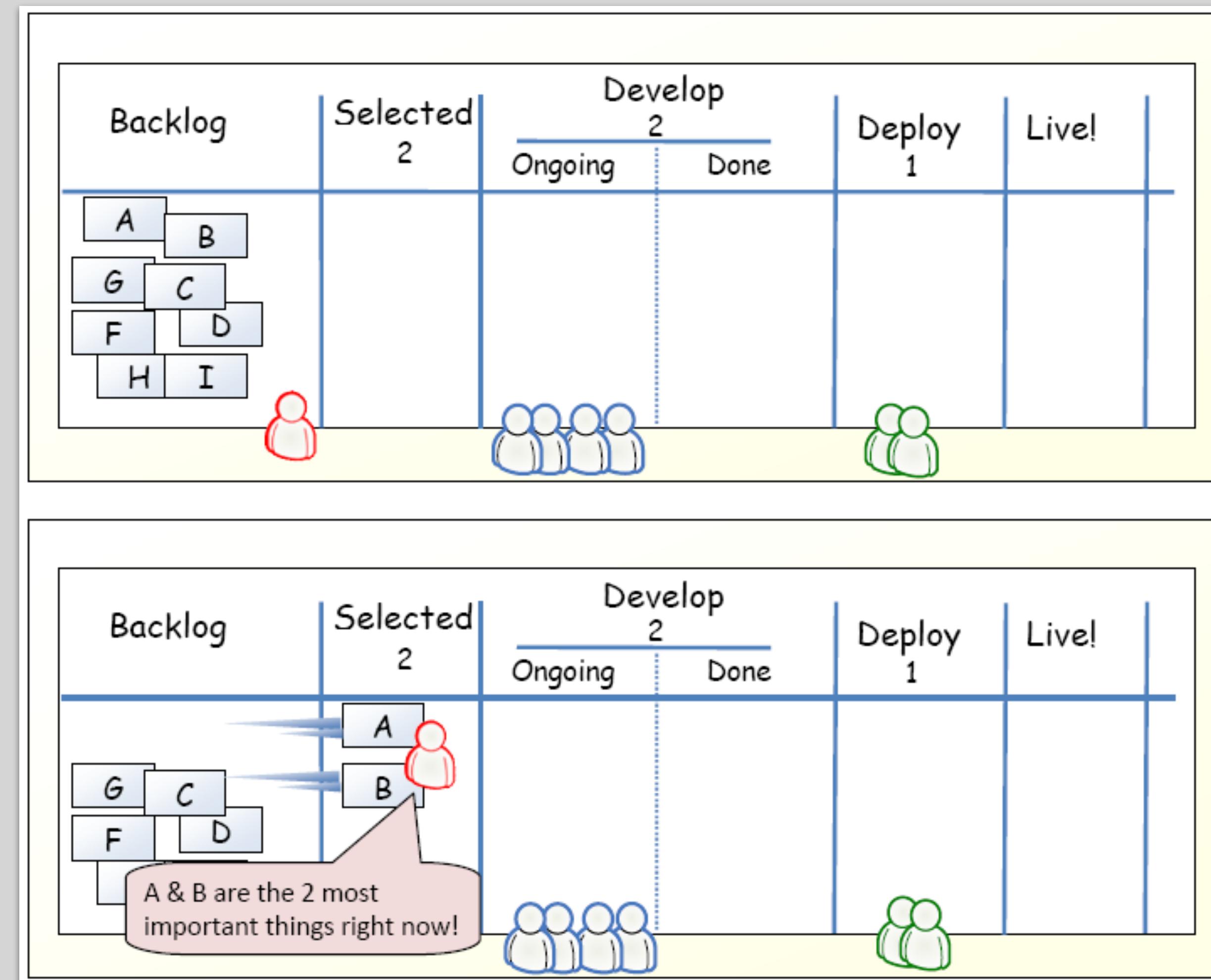
# Varför begränsa och hur?

**Minskar multitasking**, vilket också minskar hur ofta man byter vad man jobbar med. Ger snabbare resultat

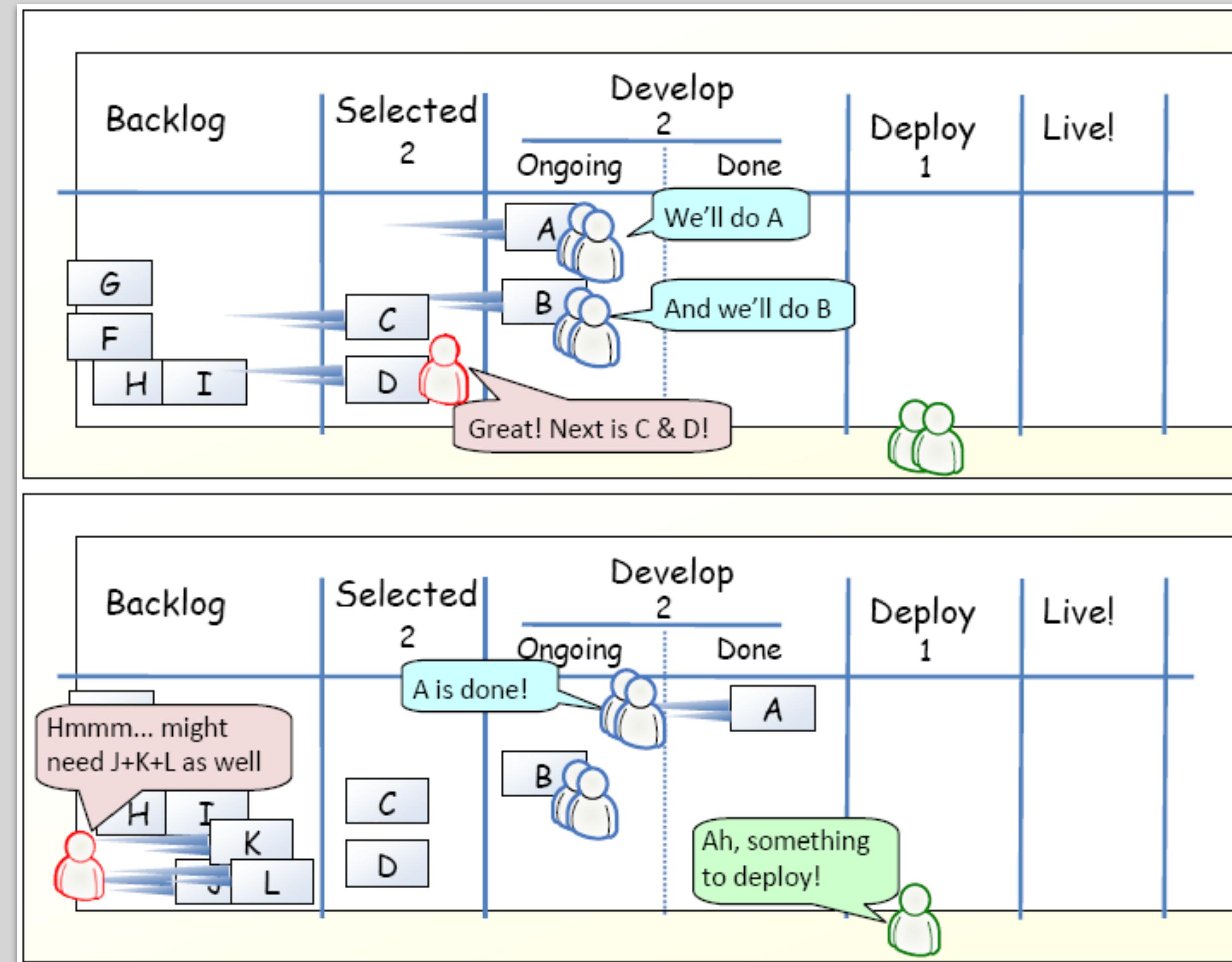
**Ökar samarbetet** inom gruppen då man måste hjälpas åt när nya uppgifter inte får påbörjas.

**Börja med att sätta små värden**, t.ex. hur många ni är i gruppen och justera baserat på flöde. Mät hur länge ett issue får vänta, hur mycket som produceras över tid och så vidare...

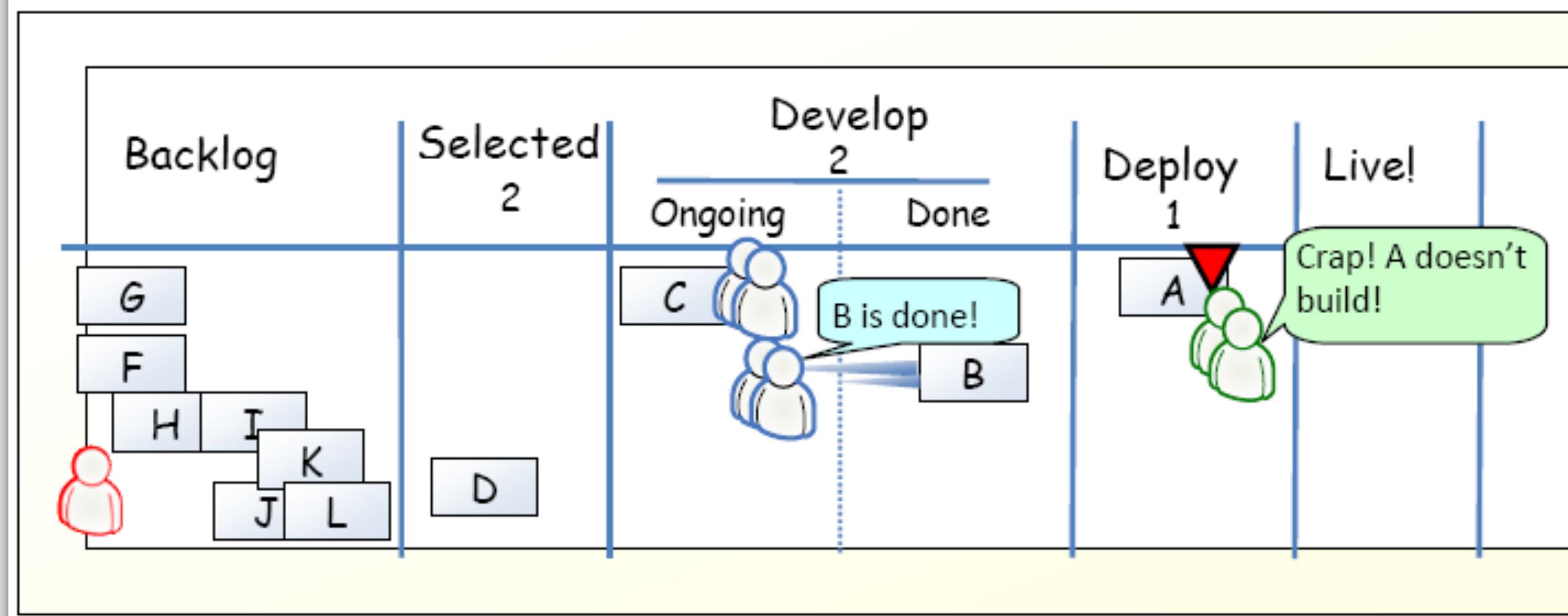
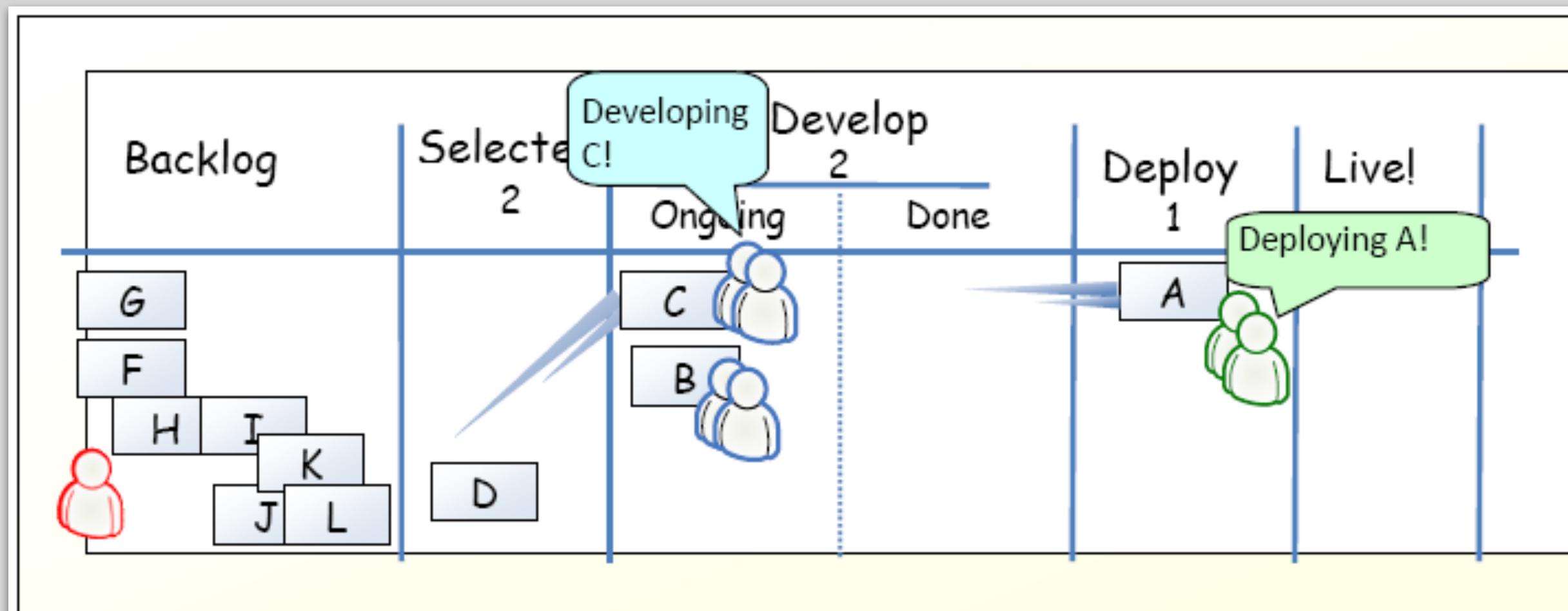
# Exempel



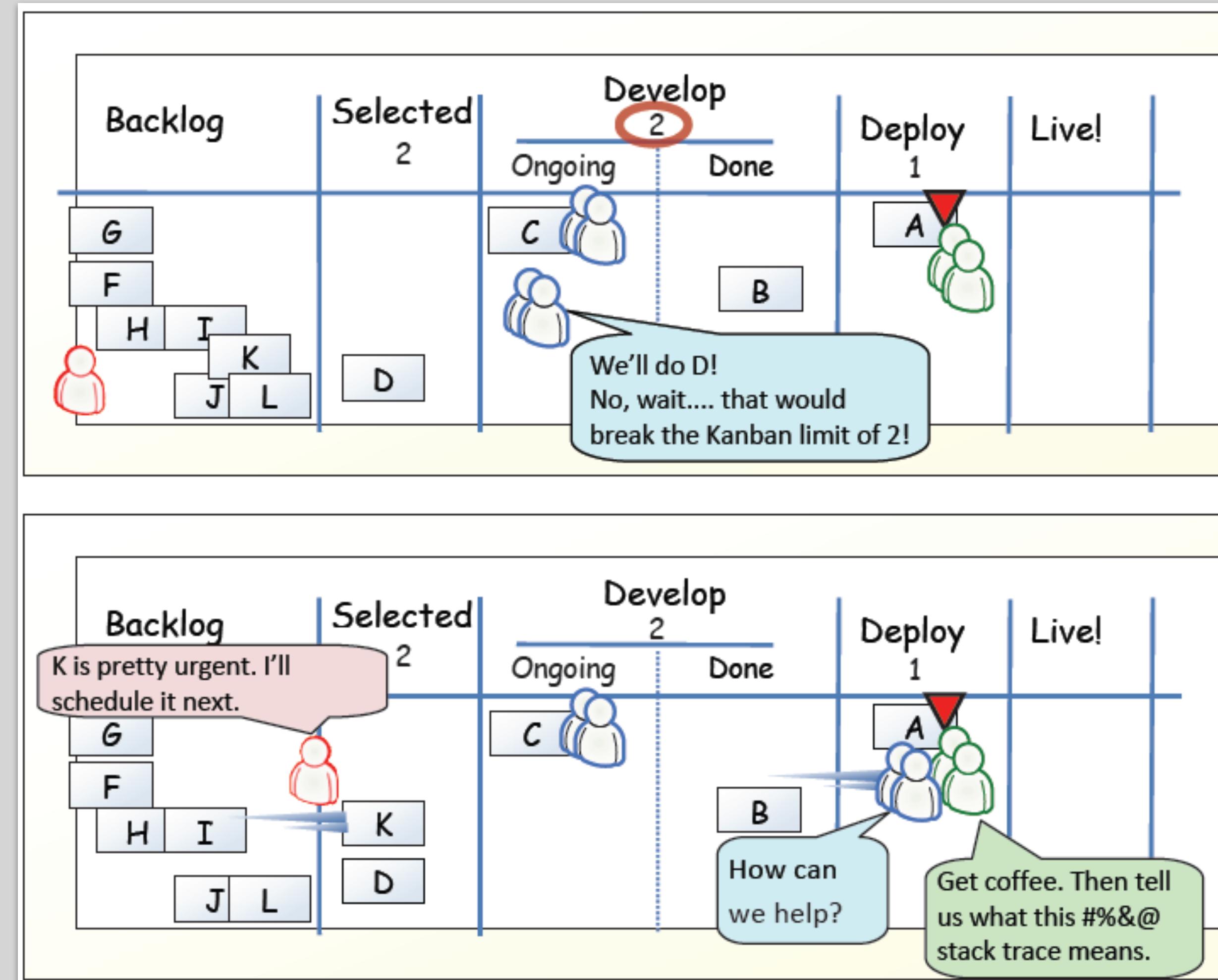
# Exempel



# Exempel



# Exempel



# Scrum?

**Scrum begränsar **tid**, dvs hur mycket hinner man göra på en tidsperiod istället för hur mycket för vara aktivt vid en given tidpunkt.**

**Kanban är bättre för denna typ av projekt då Scrum kräver att man uppskattar hur lång tid varje sak tar (för att kunna planera en tidsperiod) och att man kan förutse hur mycket tid man har under perioden. Detta kan vara **svårt** för korta studentprojekt.**

# eXtreme Programming

# eXtreme Programming (XP)

**Enkla **principer** som styr hur man arbetar inom ett projekt.**

**Om något är **bra**, gör det **hela tiden**. Om något är **dåligt**, gör det **inte**.**

# XP i projektet

Alla i projektet arbetar tillsammans, som **en grupp**. Sitt **tillsammans** (virtuellt, t.ex. via Discord) när möjligt.

Alla har ett **gemensamt ansvar** för **källkoden** och dess **kvalitet**. Vem som helst kan ändra den.

Bestäm en **standard** för hur **källkoden** skall se ut, t.ex. **filindelning, variabelnamn, och så vidare**, så att alla kan läsa och förstå den.

# XP i projektet

Använd en så **enkel design** som möjligt, gör den inte mera komplex än vad som **krävs** för den nuvarande versionen.

Gör **designbeslut** när du **måste**, dvs vänta så länge som möjligt. Försök alltid hitta sätt att **förbättra** designen när det är möjligt.

Sträva efter en **minimal** och **enkel** implementation.

**Metaforer** är effektiva när man kommunicrar kring designbeslut, t.ex. desktop-metaforen.

# XP i projektet

**Arbeta i ett hållbart tempo. Undvik “crunch-time” och en allt för hög belastning under vissa perioder. Kanban kan hjälpa här, dvs planera för rimlighet och följ upp.**

**Ha alltid en fungerande version av projektet. Uppdatera denna så fort ny eller uppdaterad funktionalitet finns tillgänglig.**

**Testa mycket och ofta.**

# Övrigt

**Kommunicera.** Ha regelbundna möten där ni **följer upp** hur arbetet fungerar. **Korrigera** saker som inte fungerar.

**Dessa möten skall ske utöver kommunikation kring övriga  
saker.**

**Kom överens om hur ni skall arbeta, vad som krävs av  
medlemmarna i gruppen och hur ni hanterar konflikter.  
Skriv gärna ned dessa i ett **socialt kontrakt**.**

# Verkar jobbigt!

**Mjukvaruutveckling kan vara jobbigt! Det vi diskuterat är (förenklade) “best practices” kring projektarbete som visat sig fungera.**

**Testa och anpassa!** Vi kräver att ni dokumenterar krav och använder gitlab för källkod. Varför inte testa issues och ett Kanban-board? Det är ju i princip gratis.

Lär er **anpassa** arbete efter övrig **belastning**. Det kommer att komma fler projekt med tidspress, så ju förr ni lär er hantera detta, desto bättre...

# Git

# **Men vänta! Git?**

**Git är en viktig del av det vi diskuterat. Med git kan ni äga koden tillsammans, det är möjligt att granska varandras kod och det är lätt att ändra design.**

**Vad är “best practice” för användande av git i projektet?**

# Git är distribuerat

**Git är ett distribuerat versionshanteringsystem där alla har en egen kopia. Det finns en “huvudsaklig” kopia, i ert fall på [gitlab.lnu.se](https://gitlab.lnu.se). Kopiorna på era datorer är de ni arbetar med för stunden.**

**Skapa en gemensam `.gitignore` så ni inte får in skräp!**

**Uppdatera mot den huvudsakliga kopian ofta, så att ni inte arbetar mot en gammal version. Börja med en pull/ fetch och avsluta med en push.**

**Man behöver inte göra push efter varje commit.**

# Dela upp arbetet

**Dela upp källkoden i filer. Det gör det lättare för flera personer att arbeta samtidigt.**

**Arbata i **branches** så mycket som möjligt, så att flera personer kan uppdatera samma fil utan att skapa konflikter.**

**Gör en **merge** när arbetet i en branch är klart eller när en naturlig brytpunkt. Gör merge från master **ofta**, för att minska konflikter.**

# Master skall alltid fungera

Den version som finns i **master** skall “**fungera**”. Det är upp till er att bestämma vad det innehär, men det skall t.ex. finnas syntaxfel eller motsvarande där.

Gör gärna en **merge-request** via gitlab om du gör en omfattande merge till master. Detta ger någon annan möjlighet att **granska** din källkod innan den hamnar i den fungerande versionen.

# Gitflow eller annan modell?

**Det spelar inte så stor roll vilken typ av branches ni skapar, t.ex. feature eller personliga, så länge ni är konsekventa. Hitta en modell som fungerar för er!**

**Bra namngivning och bra commit-meddelanden är mycket viktiga.**

**Men, feature-branches är generellt sätt lättare att hålla reda på. De har t.ex. en begränsad livslängd.**

# **Att arbeta i projekt**

## **En kort introduktion till software engineering**

Presenteras av Morgan Ericsson (@morganericsson)

