



# Project management

*Control Project Course*

*Joakim Lilliesköld*

*Royal Institute of Technology*

*Stockholm, Sweden*

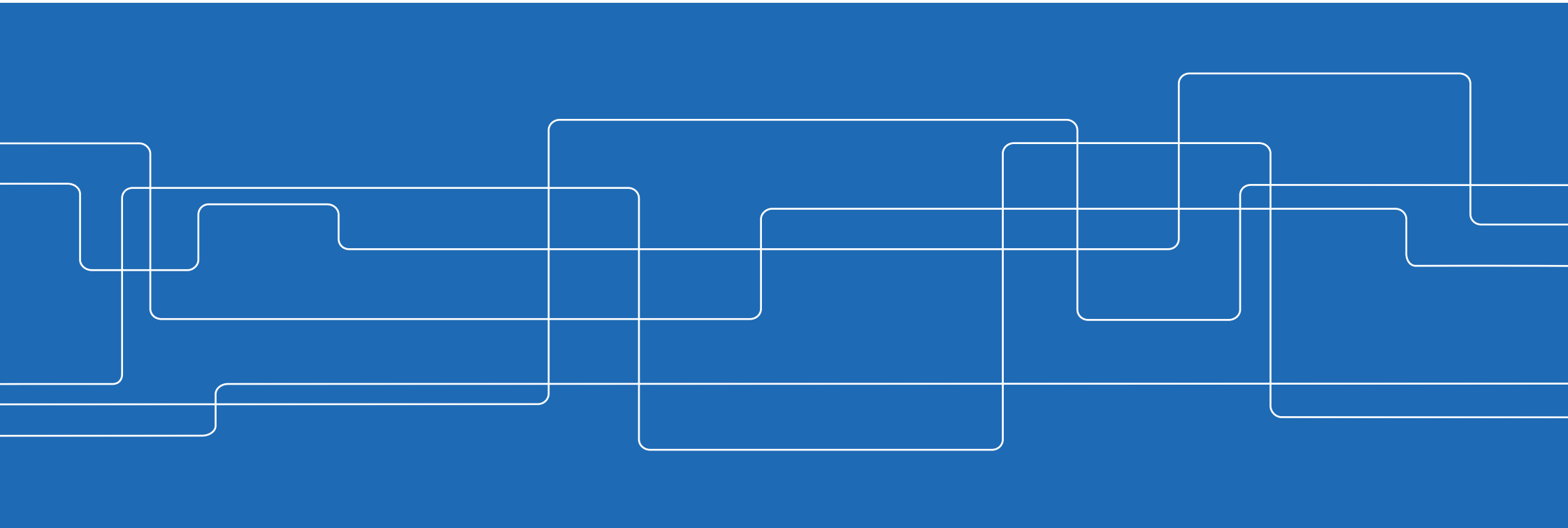


# Agenda

What is project management  
Terminology  
Project planning

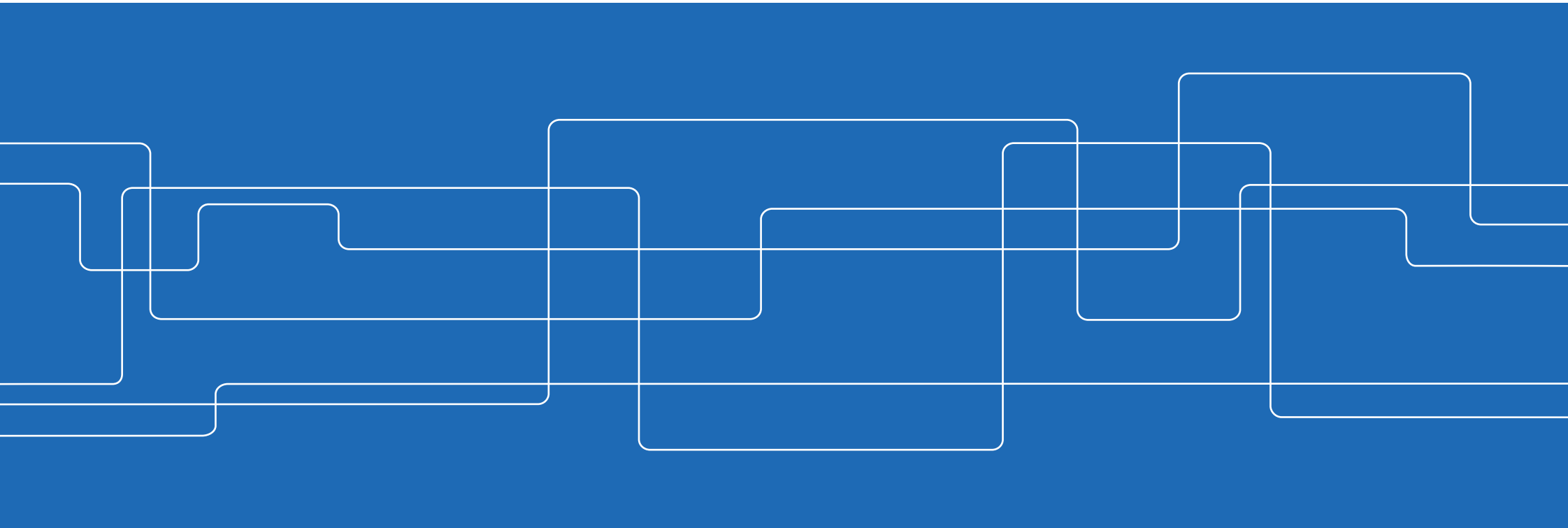


# What is a project?





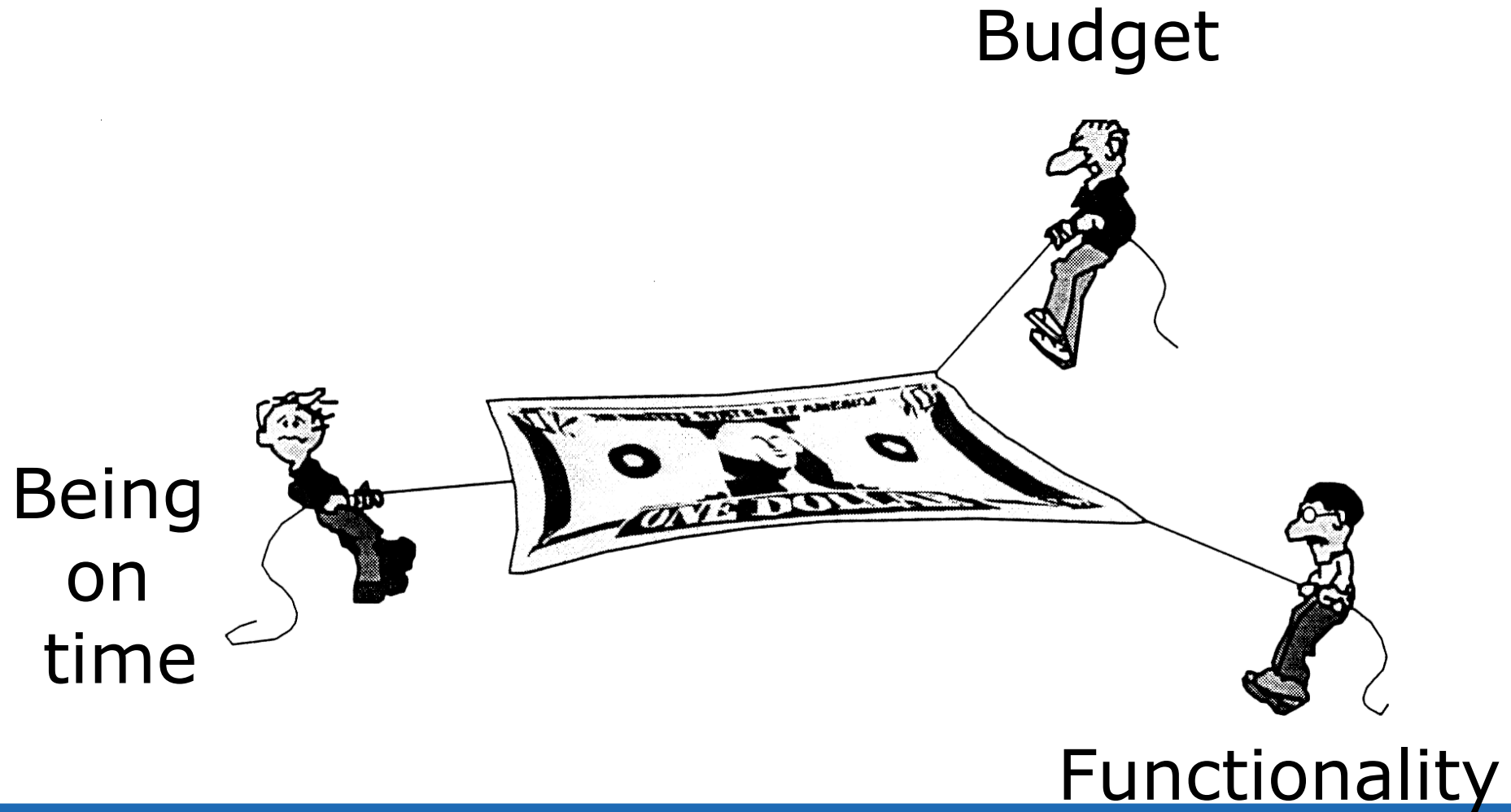
# What is project management?





**Team**  
**Responsibility**      **Resource control**  
**Control**      **Organization**      **Risk**  
**Limitations**      **Planning**      **Expectation**      **Possibilities**  
**Deadlines**      **Goals**      **Education**  
**Software**      **Project Management**      **Reports**  
**Specifications**  
**Culture**      **Budget follow-up**      **Ownership**  
**Communication**      **Risk management**  
**Leadership**      **Time management**  
**Milestones**      **Priorities**      **Budget**

# "The Tripple Constraint"





# Japanprojektet

Vad handlar det om  
Låna pengar



# What is important?

Functionality

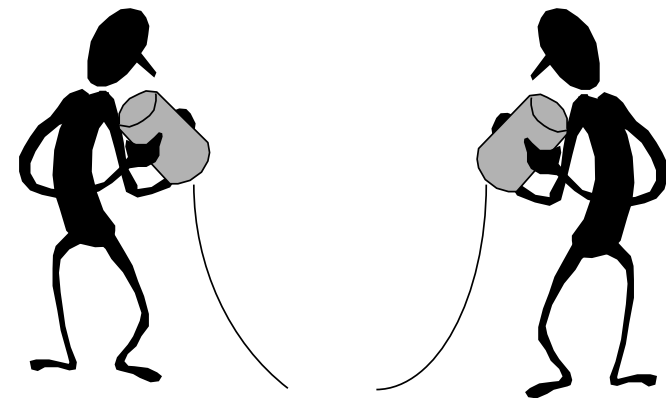
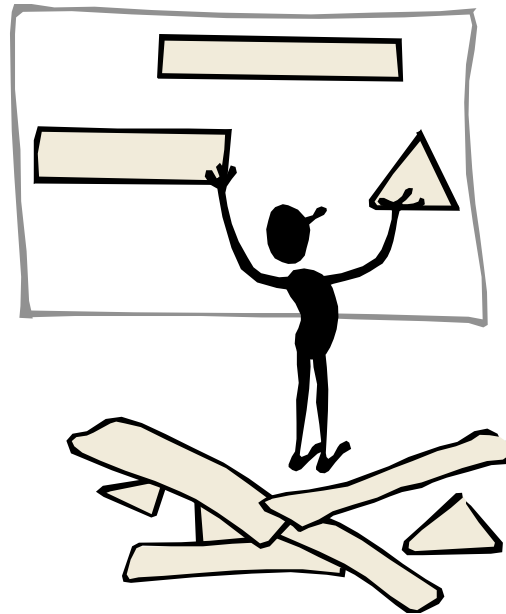
Being on time/meeting deadline

Budget





# Planning and communication





# What is project management?

The application of:

Knowledge

Skills

Tools and techniques

In order to:

Meet project requirements and expectations



# **The key (to becoming successful)**

***Meet the expectations!***



# Triangeln

Vilka förväntningar har var och en



# Disclaimer!!

Unique, depending on the context

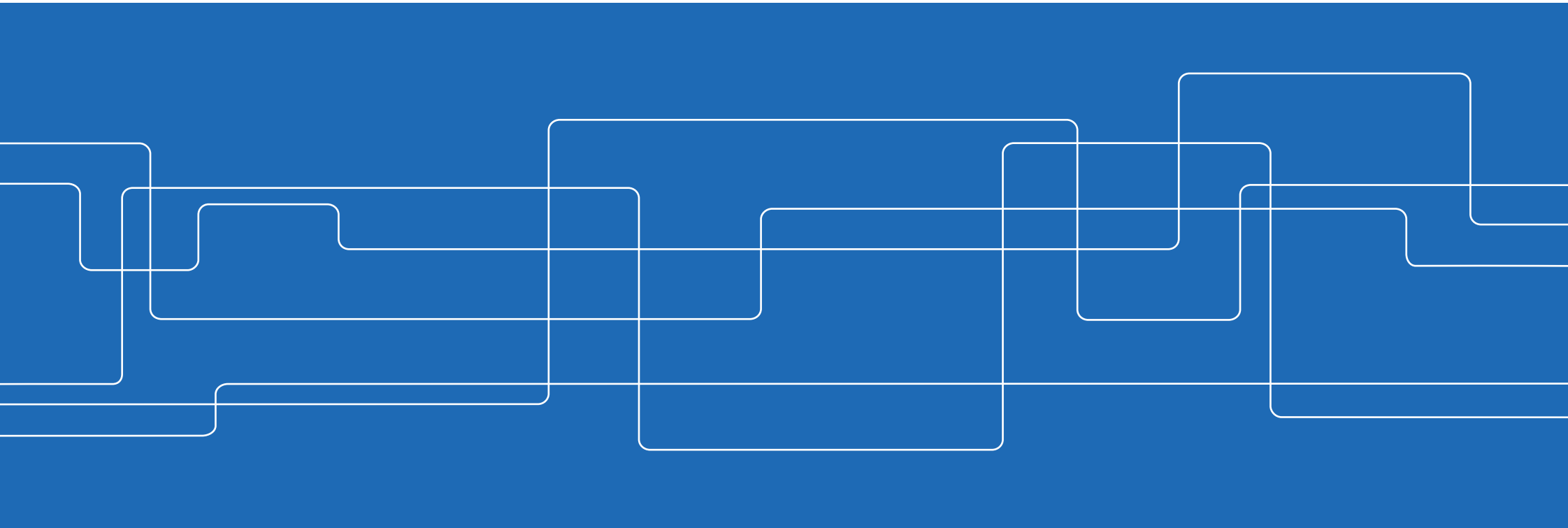
Theory is not enough: skill of  
craftsmanship is also needed

There is no universal truth...

“Do not try to reinvent the wheel”:  
“stolen with pride!!”



# Basic terminology





# Project phases

A project is always divided into phases

Each phase includes activities that have to be executed

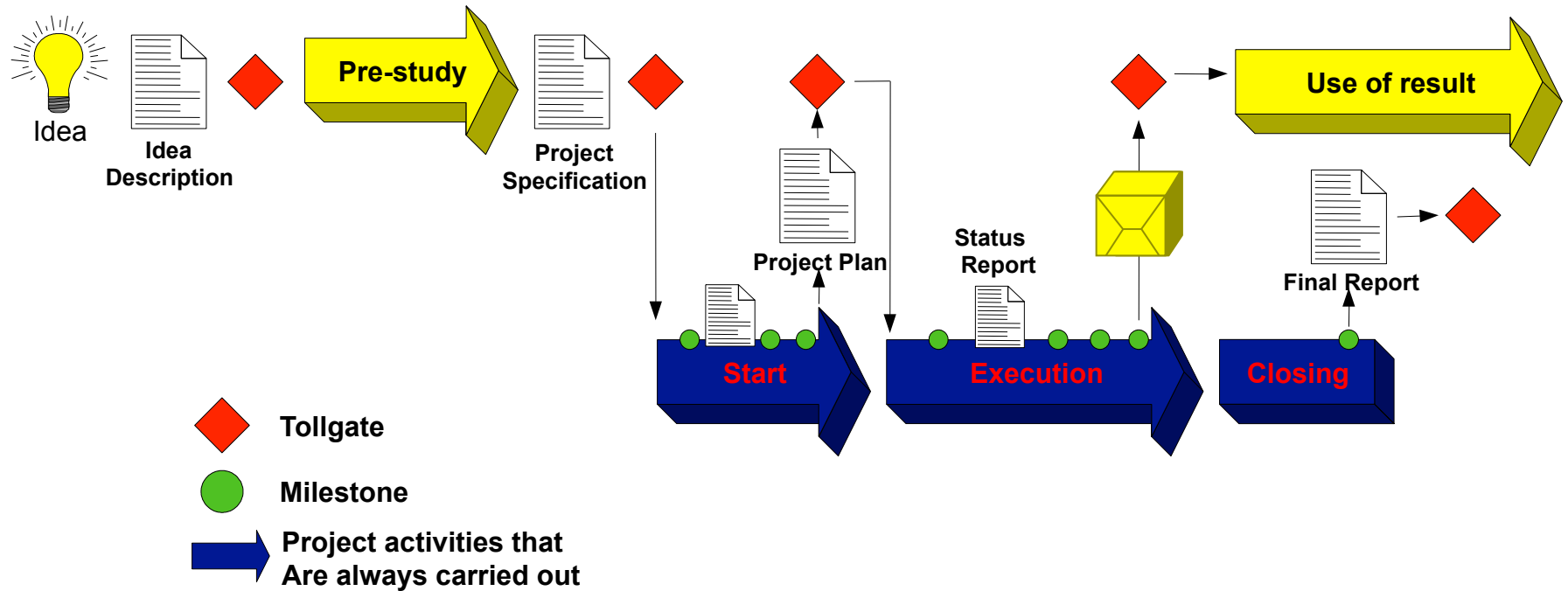
Different check points are defined for each phase:

- Milestone
-  Tollgate
- 

Experience shows that such an approach provides a well thought through structure guiding the work to focus on the most important activities, making the final results obtain a high quality



# Project phases



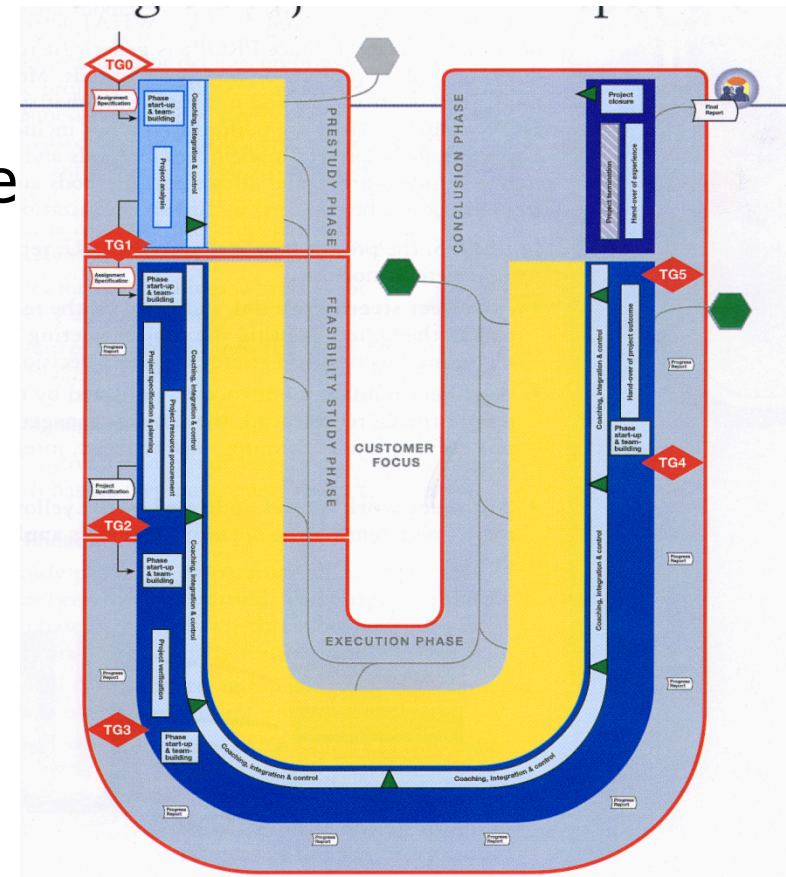
# Example PROPS

## Phases in PROPS

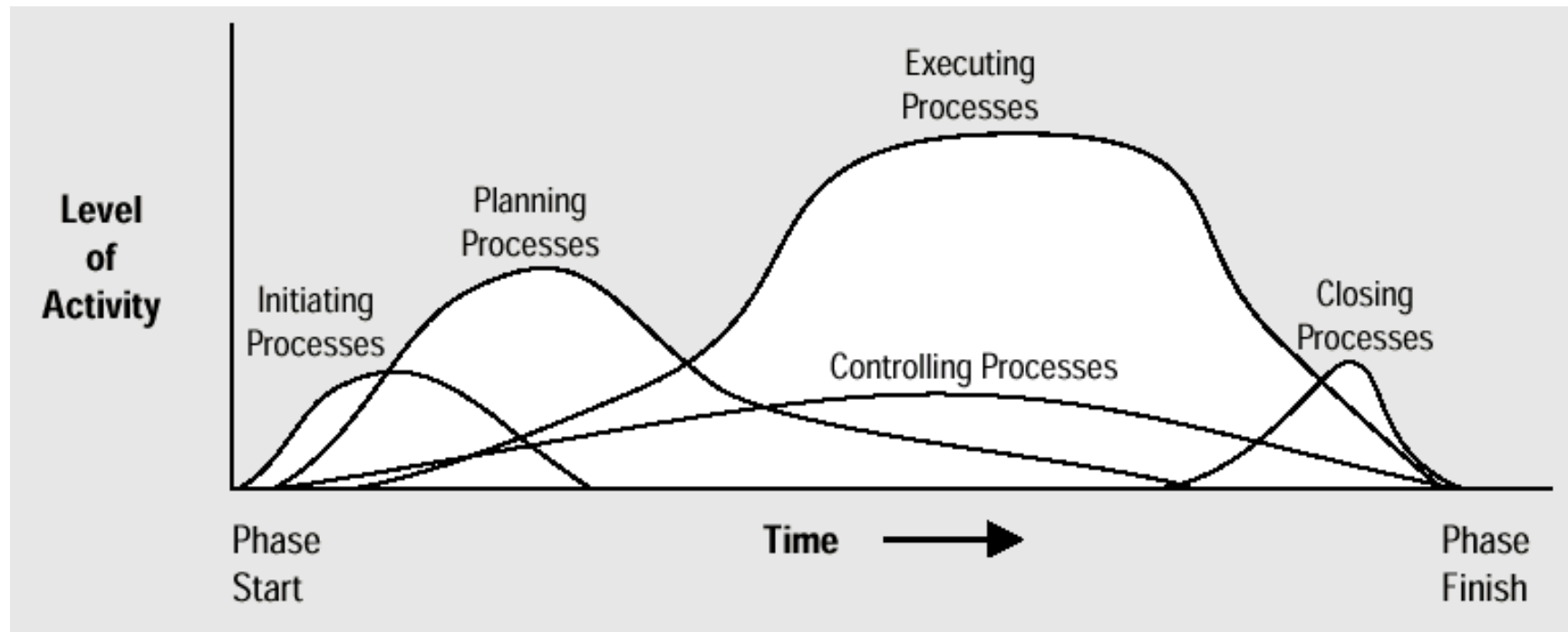
- Pre-study phase
- Feasibility study phase
- Execution phase
- Conclusion phase

5 Tollgates

>8 Milestones



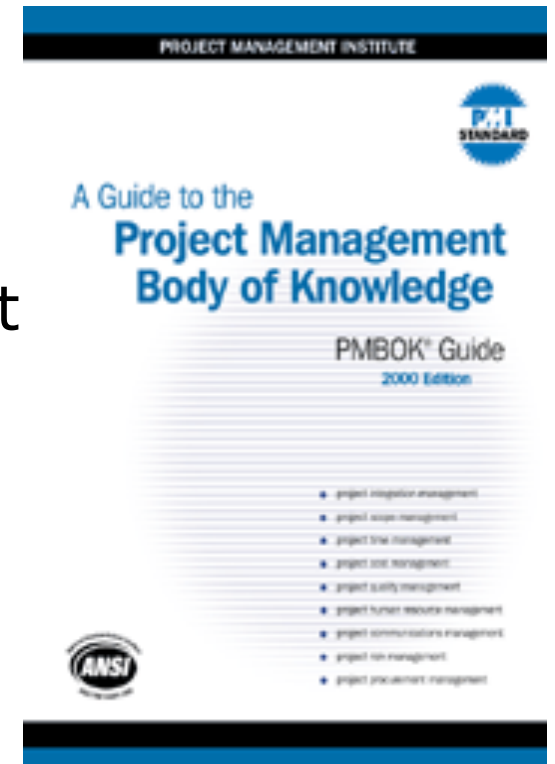
# Project phases



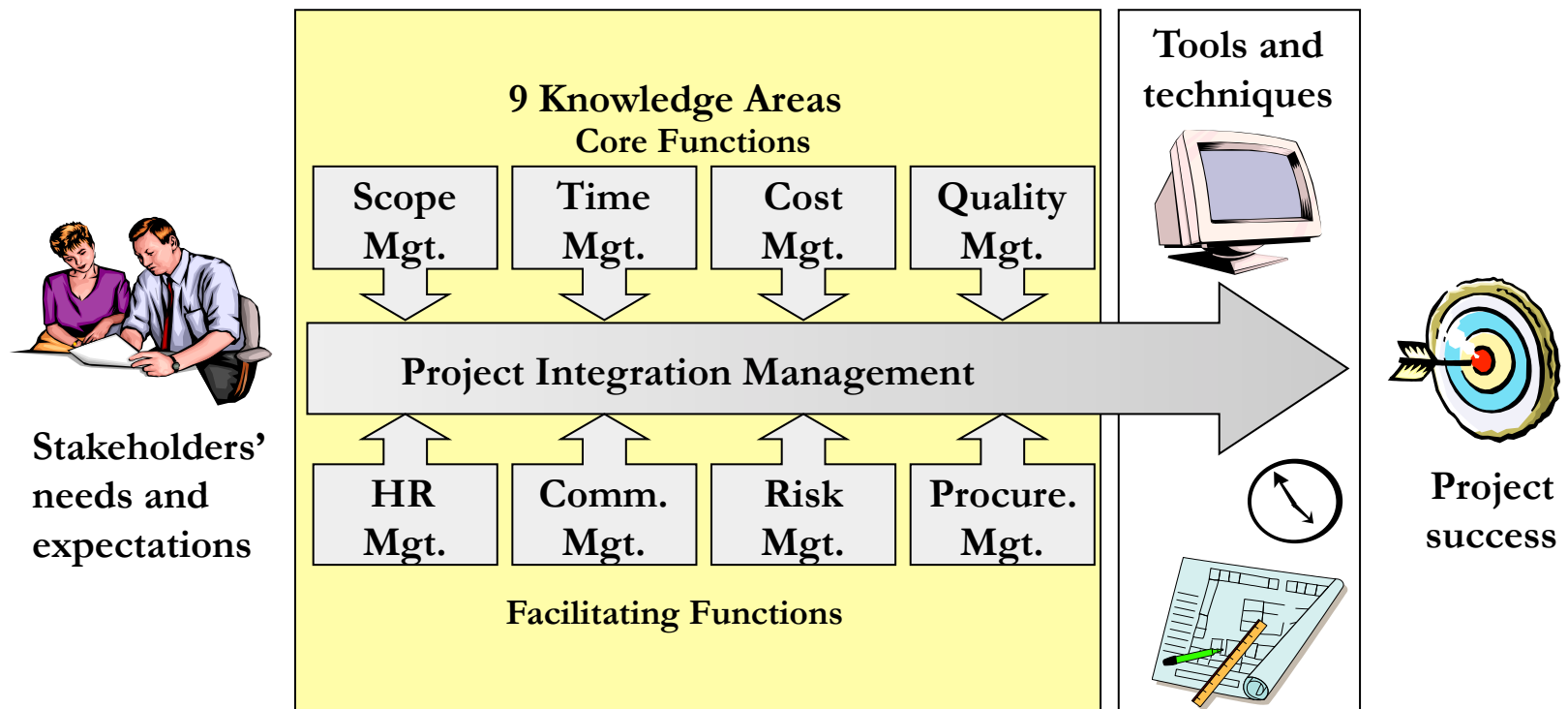


# The Project Management Body of Knowledge

Divided in two parts:  
The knowledge areas of project management  
Framework for project management

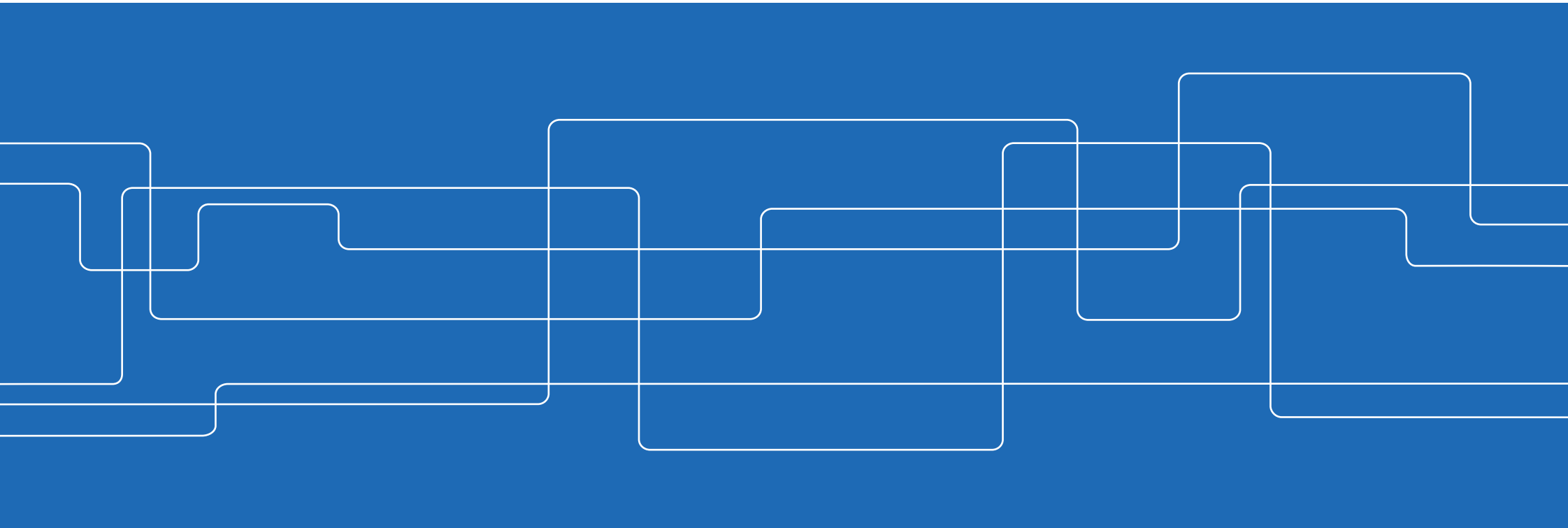


# The Project Management Body of Knowledge: Knowledge areas





# Different approaches





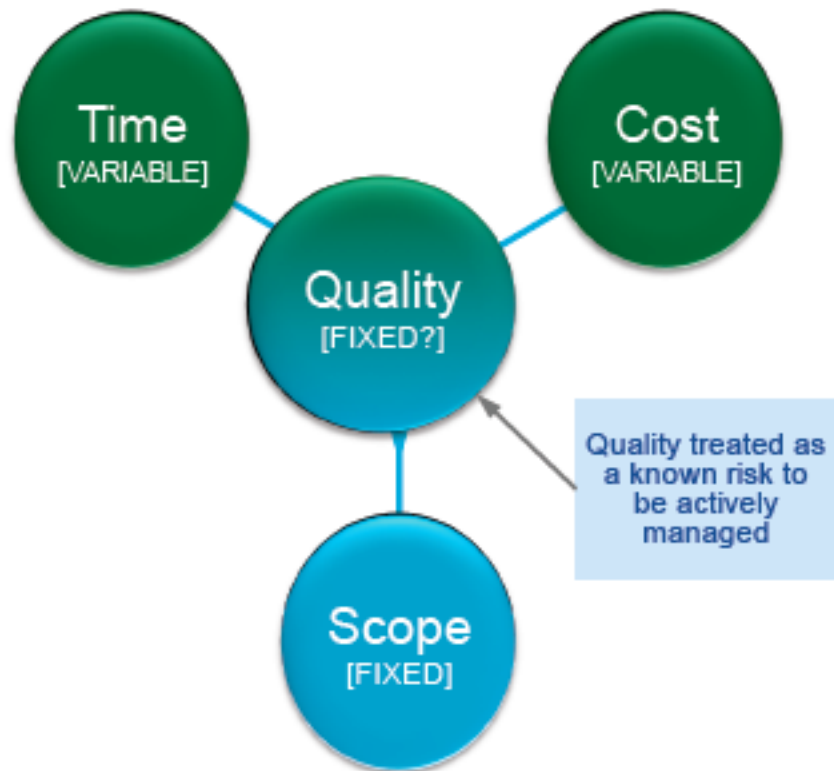
# Approaches

Agile

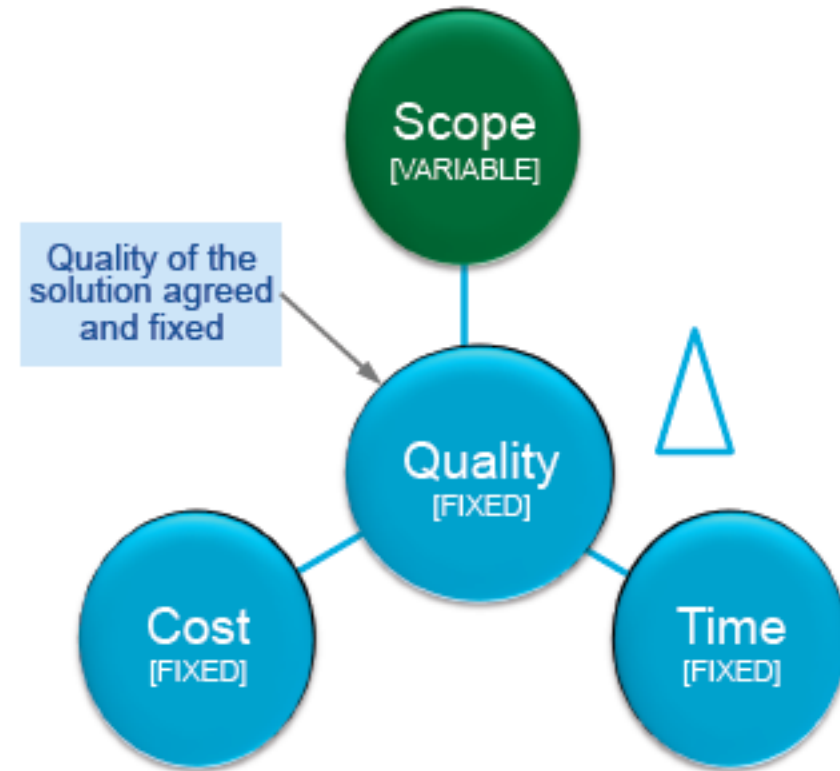
Based on planning

# Difference Agile vs traditional

## “Traditional” Approach



## Agile Approach







# Agile Manifesto - grunden

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.

# Plan for change

**Plan**  
(doomed to fail, but we don't know it yet)



**Big Bang scenario**  
"We will deliver ABCD in 4 weeks"



**Agile scenario**

- "We always deliver something every sprint (2 weeks)"
- "We think we can finish ABCD in 4 weeks, but we aren't sure"
- "We always deliver the most important items first"



Oops, our velocity is lower than we thought.  
It looks like we'll only finish AB by week 4.  
What should we do now?

Henrik Kniberg



# Project planning

1. Formulate the project goal
2. WBS – Work Breakdown Structure
3. Identify tasks
4. Identify dependencies
5. Estimate time
6. Identify the critical path
7. Distribute resources
8. Transfer to Gantt-schedules or other diagrams



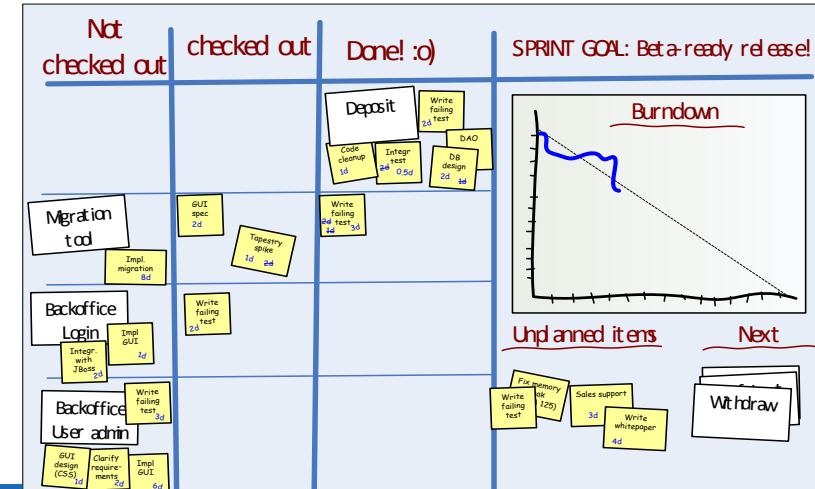
# Simplified planning (no or few parallel activities)

- 1. Formulate the project goal**
- 2. Divide the** project into phases and activities
- 3. Break down** the activities into work tasks
- 4. Time estimate** each work task
- 5. Schedule** and divide the time estimated work tasks on each project participant (resource planning)

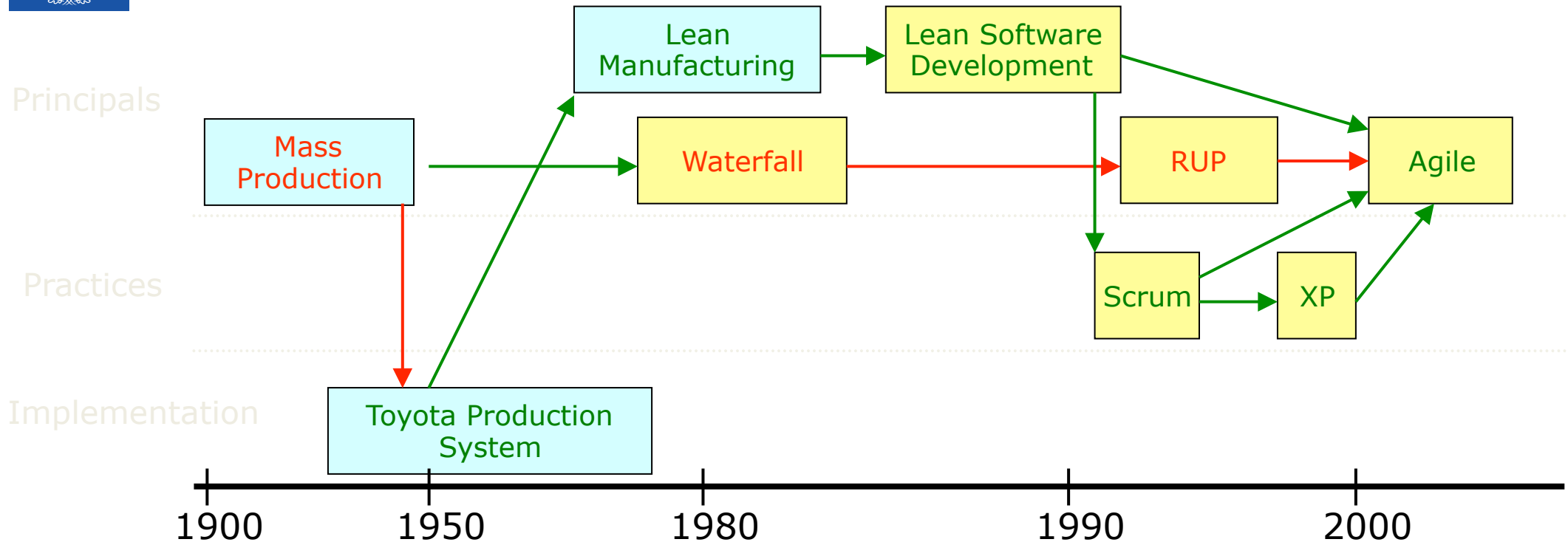
*Step 4 and 6 can you skip if you do not have any parallel activities*

*Step 7 can be done with step 8*

# Scrum



# History



- 1986: The New, New Product development Game
- 1993: First Scrum team created by Jeff Sutherland
- 1995: Scrum formalized by Jeff Sutherland & Ken Schwaber
- 1999: First XP book
- 2001: Agile Manifesto
- 2001: First Scrum book by Ken Schwaber & Mike Beedle
- 2003: Scrum alliance formed, certification program started



# Scrum

En metodik för systemutveckling skapad av Jeff Sutherland och Ken Schwaber.

Ordet kommer från rugby.

- I rugbyliknande utveckling samarbetar ett tvärfunktionellt team för att göra klart produkten på samma sätt som ett rugbylag spelar tillsammans för att föra bollen upp för planen.
- Kontrasten är en arbetsform gentemot mer stafettliknande processer. I dessa färdigställs arbetet i funktionella faser med tydliga överlämningar mellan grupper när arbetet går från en fas till en annan.

Scrum har tillämpats sedan tidigt 1990-tal



# Scrum

- Transparency
- Short feedback loops
- Clear prioritization
- Continuous improvements
- Self-organizing team
- Face-to-face communication
- Simple tools
- Frequent and regular deliveries of the whole system
- Plans are needed, but often faulty





# Planning according to Scrum

A methodology for software system development created by Jeff Sutherland and Ken Schwaber.

The word is taken from rugby.

- In a rugby alike development effort cross functional teams cooperate to finish the product in the same way a team plays to get the ball up the rugby field.
- It is a contrary methodology to methodologies similar to relay running. In these approaches, work is finished in one phase by one member and then handed over, like a baton, to another group that continues.

Scrum has been used since early 1990



# Planning according to Scrum

**Product backlog.** All requirements on the product. Owned and managed by product owner. No upper limit, but requirements are prioritized.

**Sprint backlog.** The part of the product backlog that the scrum-team accept to implement the upcoming sprint.

**Sprint.** A work period. A sprint is an undisturbed work period, usually 2-4 weeks. Every sprint starts with a planning session and ends with a review of the accepted requirements. Every day has short status meetings. The last session in a sprint is a process improvement activity.

# Planning according to Scrum

**Daily scrum.** Short status meetings (15 min). All scrum team members answer one by one:

- What have you done since yesterday?
- What are you planning to do today?
- Any imperiments/stumbling blocks?
  - If there are, these are documented and managed outside the meeting

**Sprint review.** A meeting that is decided on the first day of the spring and can not be moved, on this meeting the work performed during the sprint is reviewed.

**Sprint retrospective.** All members reflect on the past sprint. A few suggestions get picked out and improved till next sprint.

# Planning according to Scrum

**Sprint planning.** A full day during which the product owner goes through all requirements for the scrum-team. The scrum-team will then break down the requirements. The activities that the group accept for the sprint is then called sprint backlog.



# Buzzwords

**Stories.** A describing story trying to capture the requirements of what needs to be done

As a <role>

I want to <what>  
so that <why>

As a buyer

I want to save my shopping cart  
so that I can continue shopping later

Backlog item #55

## Deposit

Importance

30

Estimate

Notes

Need a UML sequence diagram. No need to worry about encryption for now.

How to demo

Log in, open deposit page, deposit €10, go to my balance page and check that it has increased by €10.

## Informal User Stories

Customer can Browse Products and Place Orders

Customer can do Keyword Search for Products

Customer alerted when No Products Selected

Customer alert on Invalid Payment Details.

Customer should be able to Track Orders

- 
- 
- 

## Formal User Stories

**Epic: As a Customer I want to be able to Browse Products and Place Orders in the system**  
**So that I can purchase goods 24/7 without having to visit the store**

- Q1. How should products be browsed?  
A1.
- Q2. What are the select criteria for products?  
A2.
- Q3. How should payment details be provided?  
A3.
- Q4. How should delivery details be provided?  
A4.
- Q5. How should purchase be Confirmed?  
A5.
- US2: Customer can do Keyword Search for products
- US3: Customer alert when no products selected
- US4: invalid payment details

**As a Customer I would like to Track Orders in the system and know at any given point what is the status of the order**  
**So that I can chase up any bottlenecks and delays**

- Q1. How should Order Details be provided?  
A1.
- Q2. How to Submit a "Tracking Request"?  
A2.
- ...



**Supplementary Requirements**

**Reliability**  
*As a Customer I need the system will be available 24 hours per day every day of the year*  
*So that maximum service level can be achieved with optimized sales. Some limited system down time is acceptable....*

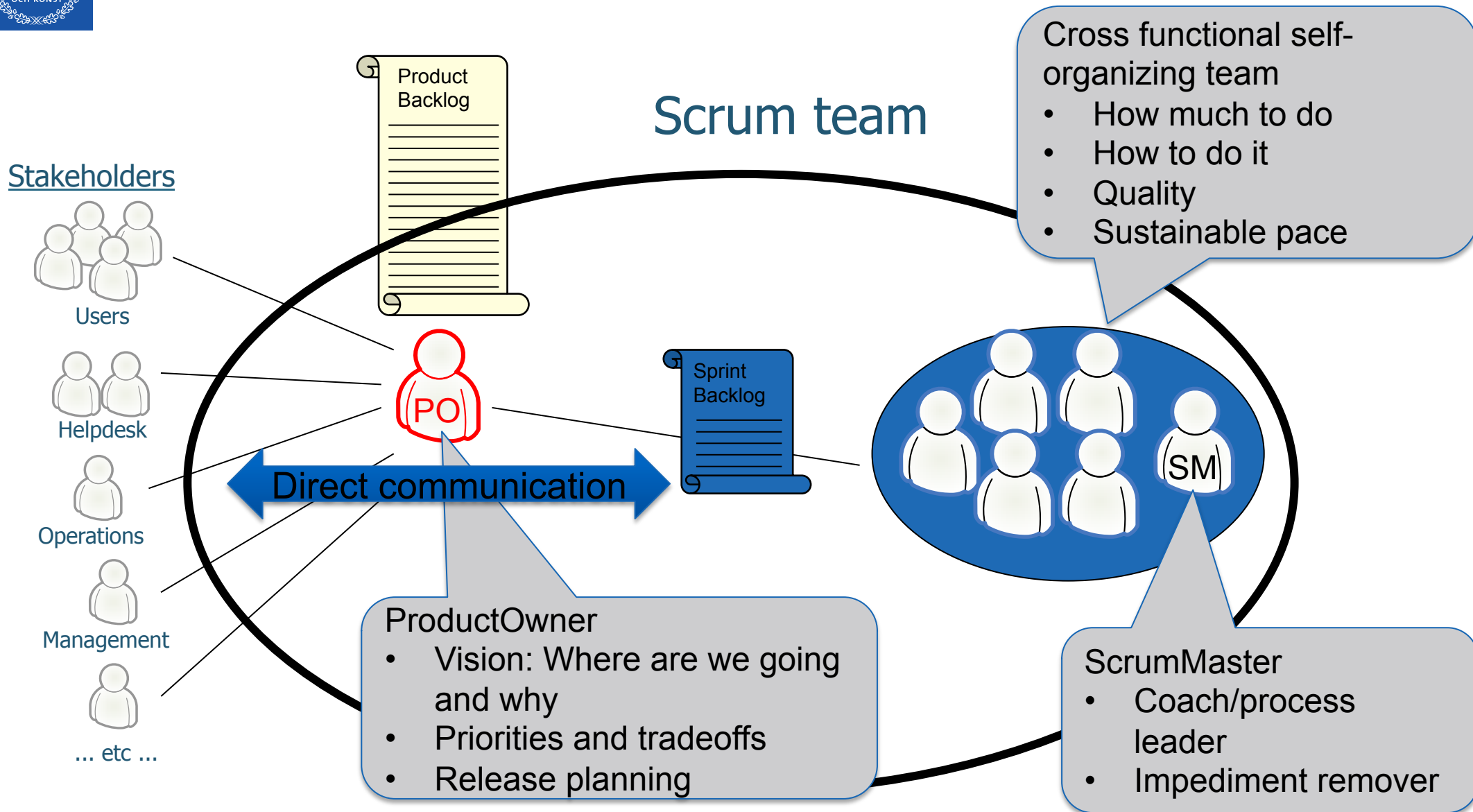
**Security**  
*As a Architect I need the system to ensure that all data transfer and storage is handled securely using transfer protocol*  
*So that no data integrity is compromised*

**Design constraints**  
*As a Product Owner I need the system to be useable by customers with at least the following minimum specification: 1) 800 x 600 pixel ...*  
*So that Client Software Resolutions can be managed that exceeds 800 x 600 pixels.*

**Client Communication Requirement**  
*As a Product Owner I want the system to be useable by customers with a 56kbs modem or faster*  
*So that low level broad band connections should not rule out any potential "buys"*

...

# Scrum overview



# Focus on the group

Requirements

5 – 8 people full-time

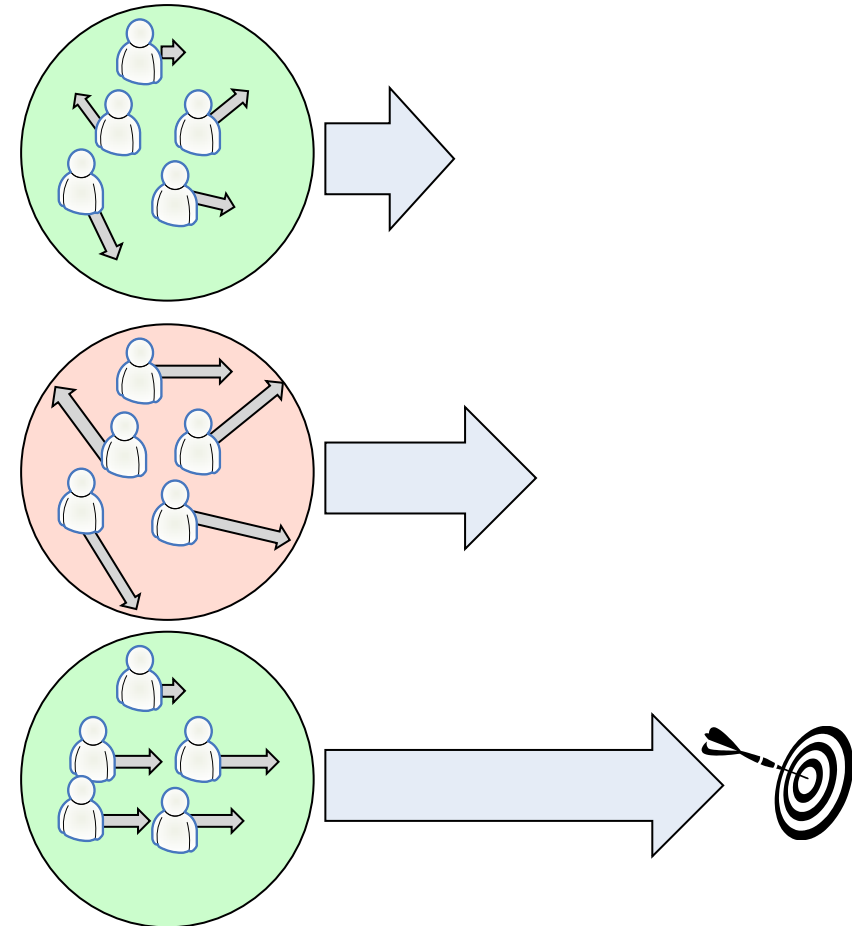
Cross-functional

Sitting together

Share responsibility

Organize themselves

Own the backlog of the sprint







# Roles – Product owner

Product Owner (PO) på Scrumspråk

Represent ALL stakeholders

Decides where the team should go

- Not how they get there
- Not how fast (the pace)

Define scope

Prioritize

Own product backlog



# Role - ScrumMaster

Responsible for method

- **Coachar** instead of control and demand

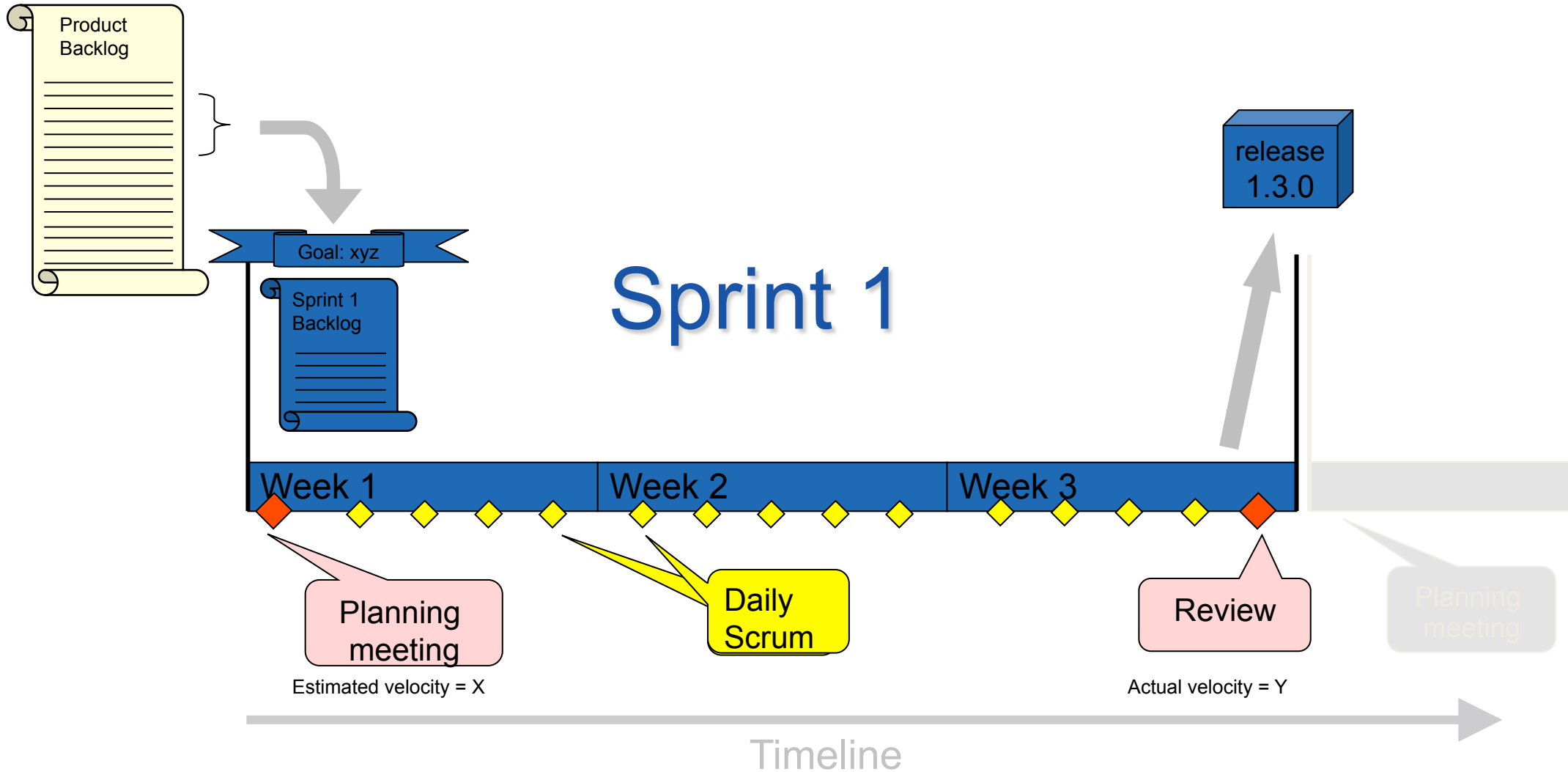
Removes impediments

Part of team

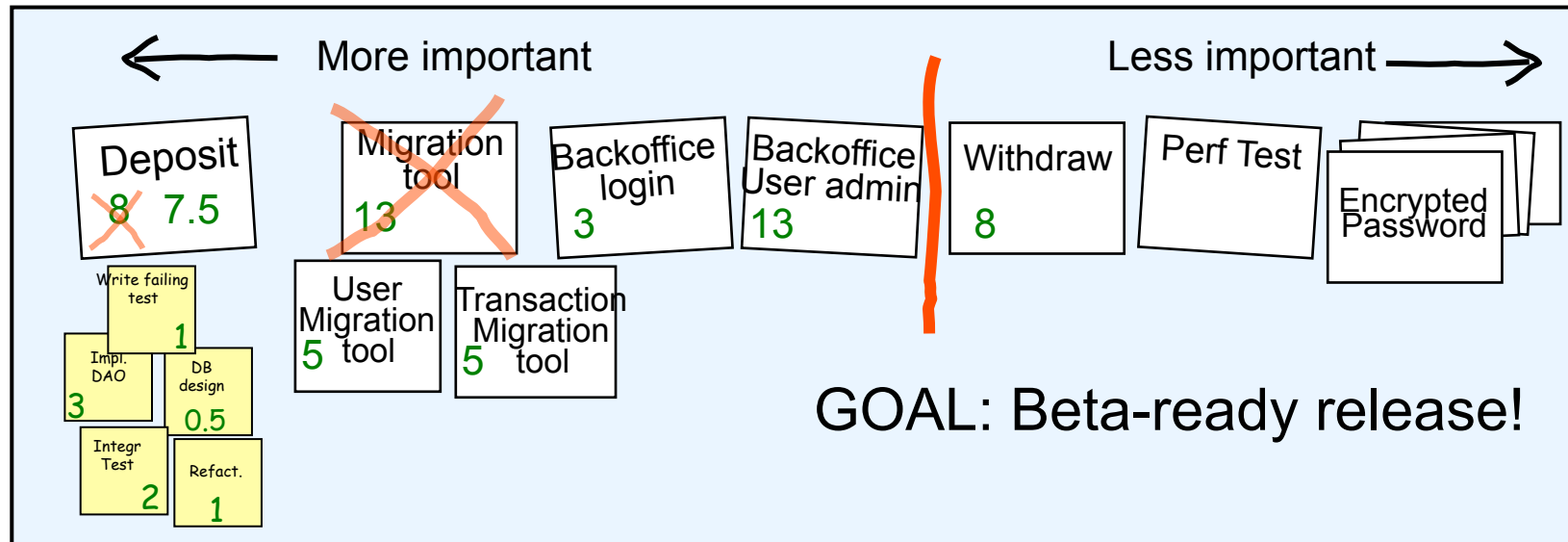
Is not:

- Line manager
- Team leader
- Technology guru
- Not a full-time position

# Process

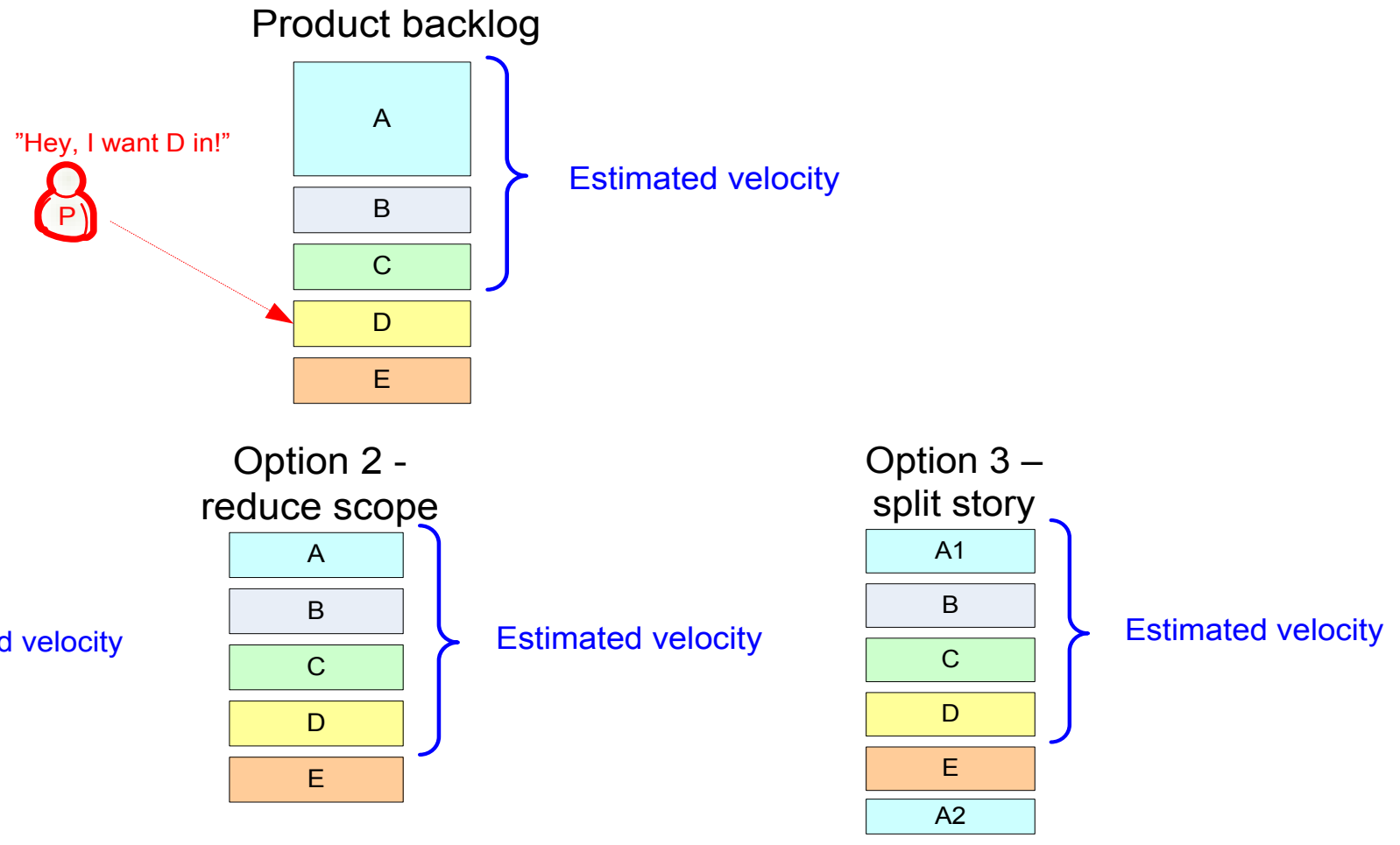


# Planning in Scrum

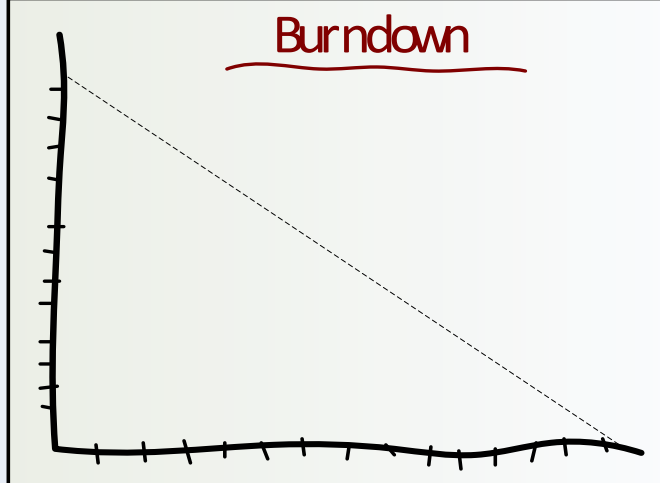


- Go through goals
- Present backlog
- Estimate times
- Re-priorize, re-estimate, break down task (stories), merge tasks (stories)
- Remove tasks
- Estimate what can be done in the next sprint, draw a line

# Dilemma for product owner



# Sprint backlog – day 0

Not checked out	checked out	Done! :o)	SPRINT GOAL: Bet a ready release!
<p><b>Deposit</b></p> <ul style="list-style-type: none"> <li>Code cleanup 1d</li> <li>Integr. test 2d</li> <li>DB design 1d</li> <li>DAO 3d</li> <li>Write failing test 2d</li> </ul>			<p><b>Burndown</b></p>  <p><u>Unplanned items</u></p> <p><u>Next</u></p> <p>Withdraw</p>
<p><b>Migration tool</b></p> <ul style="list-style-type: none"> <li>Impl. migration 8d</li> <li>Tapestry spike 2d</li> <li>Write failing test 2d</li> <li>GUI spec 2d</li> </ul>			
<p><b>Backoffice Login</b></p> <ul style="list-style-type: none"> <li>Integr. with JBoss 2d</li> <li>Impl. GUI 1d</li> <li>Write failing test</li> </ul>			
<p><b>Backoffice User admin</b></p> <ul style="list-style-type: none"> <li>Write failing test 3d</li> <li>GUI design (CSS) 1d</li> <li>Clarify requirements 2d</li> <li>Impl. GUI 6d</li> </ul>			



## Daily meetings

Short stand up meetings at a fixed time ever day  
(15 minutes)

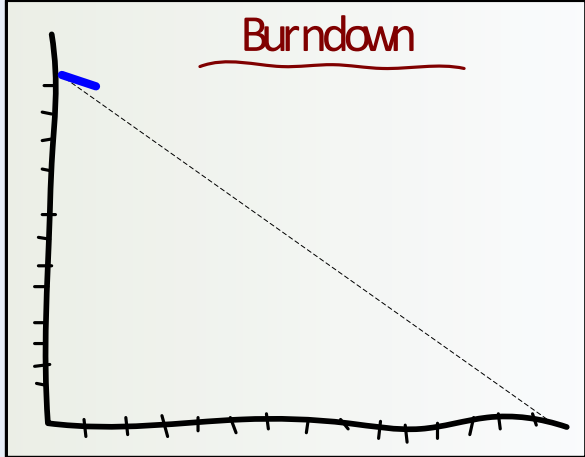
Everyone shares:

What did I do yesterday?

What will I do today?

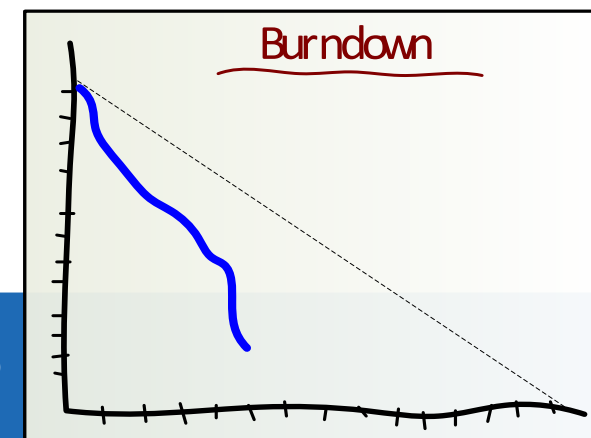
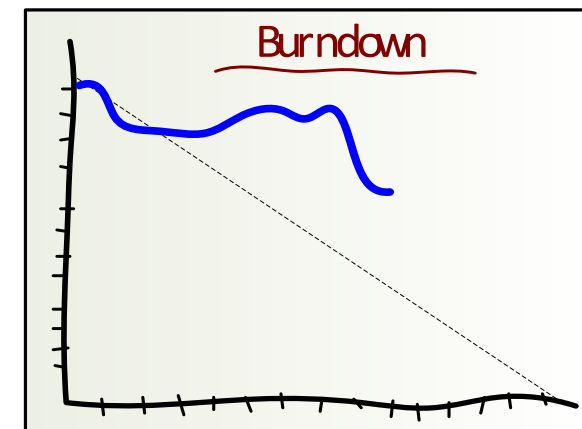
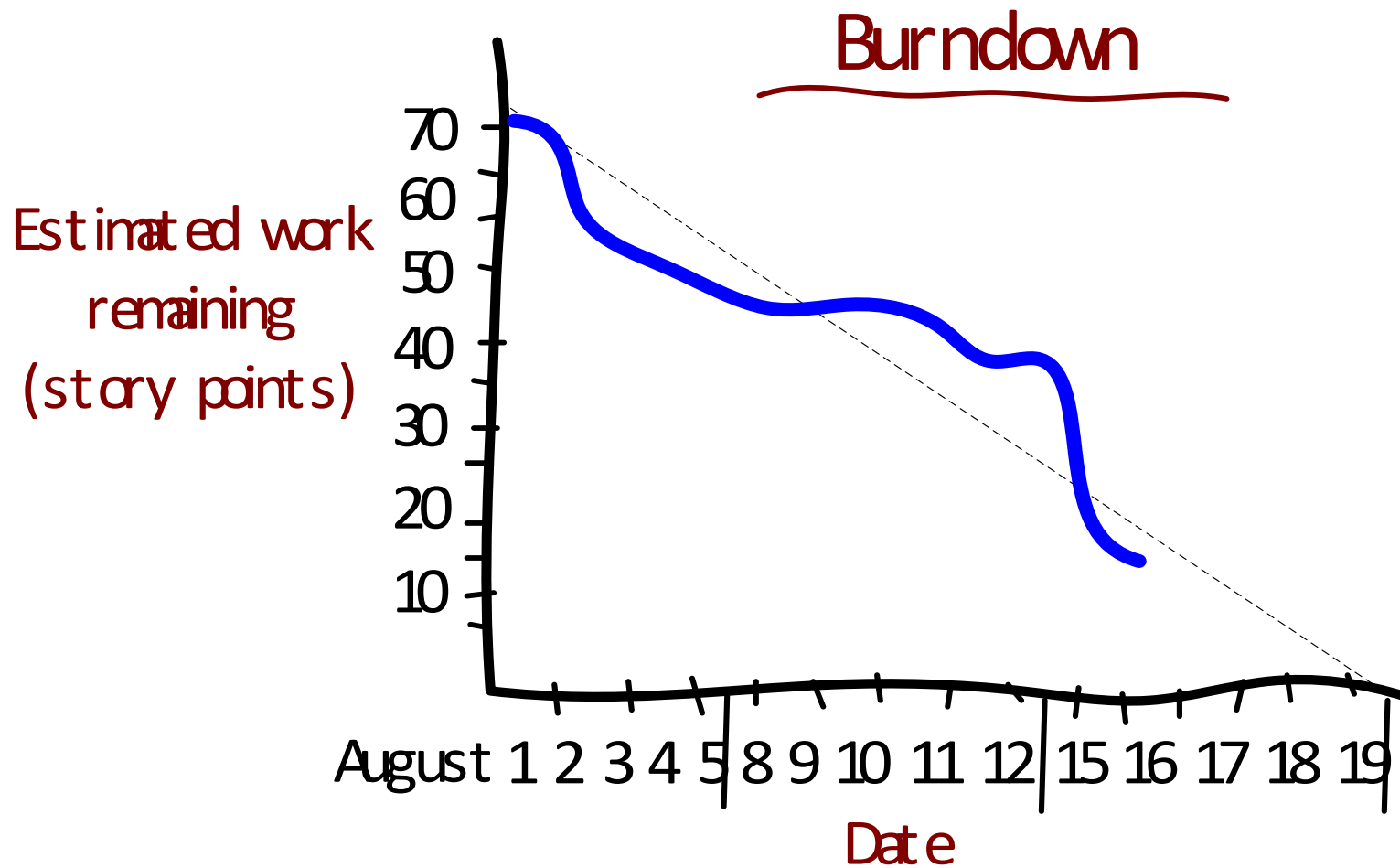
What is hindering me?

# Spring backlog – after first meeting

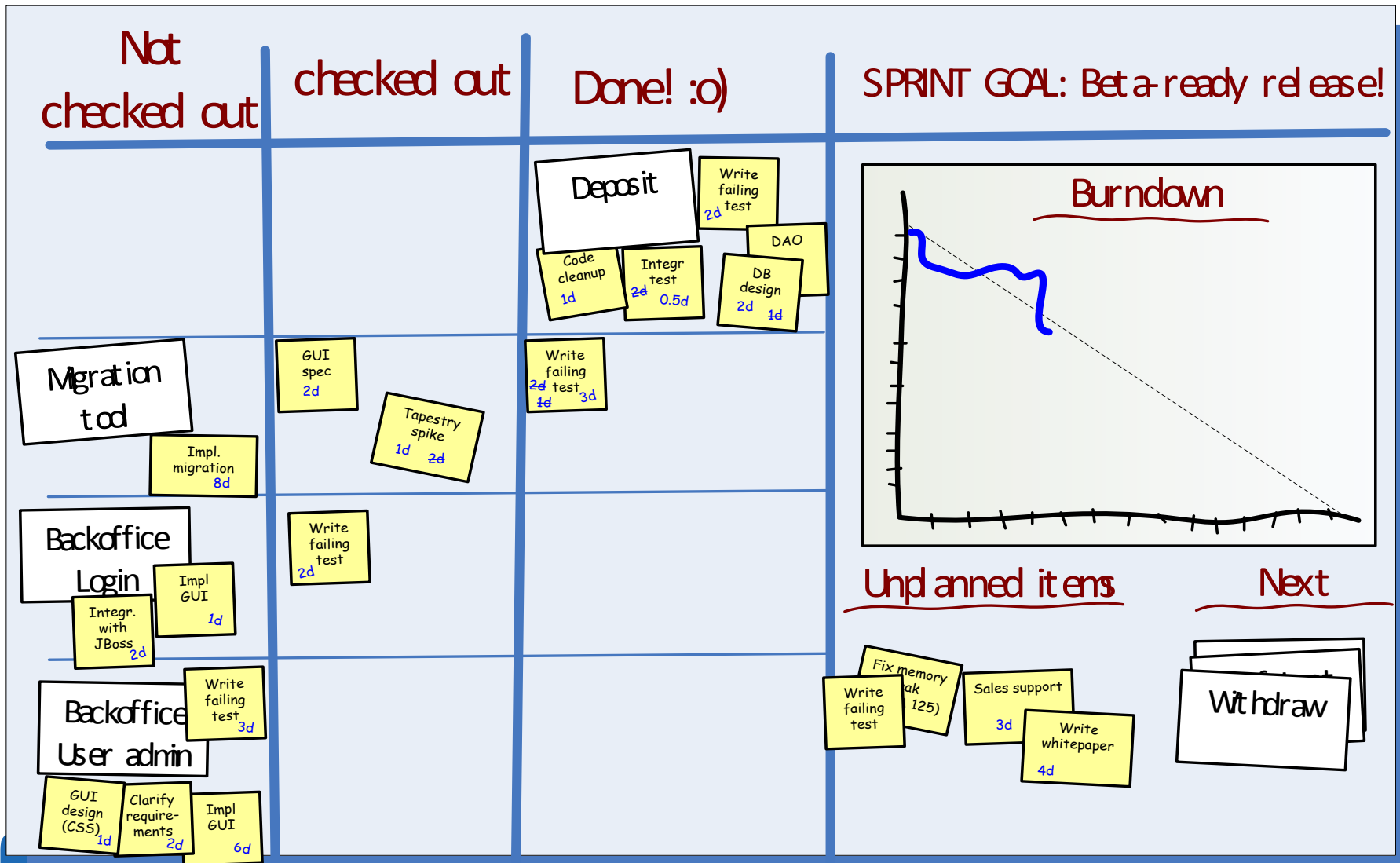
Not checked out	checked out	Done! :o)	SPRINT GOAL: Bet a ready release!
<p><b>Deposit</b></p> <ul style="list-style-type: none"> <li>Code cleanup 1d</li> <li>Integr test 2d</li> <li>DAO 3d</li> </ul>	<ul style="list-style-type: none"> <li>Write failing test 2d</li> <li>DB design 1d</li> </ul>		<p><b>Burndown</b></p>  <p><u>Unplanned items</u></p> <p><u>Next</u></p> <ul style="list-style-type: none"> <li>Withdraw</li> </ul>
<p><b>Migration tool</b></p> <ul style="list-style-type: none"> <li>Impl. migration 8d</li> <li>Tapestry spike 2d</li> <li>GUI spec 2d</li> </ul>	<ul style="list-style-type: none"> <li>Write failing test 2d</li> </ul>		
<p><b>Backoffice Login</b></p> <ul style="list-style-type: none"> <li>Integr. with JBoss 2d</li> <li>Impl GUI 1d</li> <li>Write failing test</li> </ul>			
<p><b>Backoffice User admin</b></p> <ul style="list-style-type: none"> <li>Write failing test 3d</li> <li>GUI design (CSS) 1d</li> <li>Clarify requirements 2d</li> <li>Impl GUI 6d</li> </ul>			



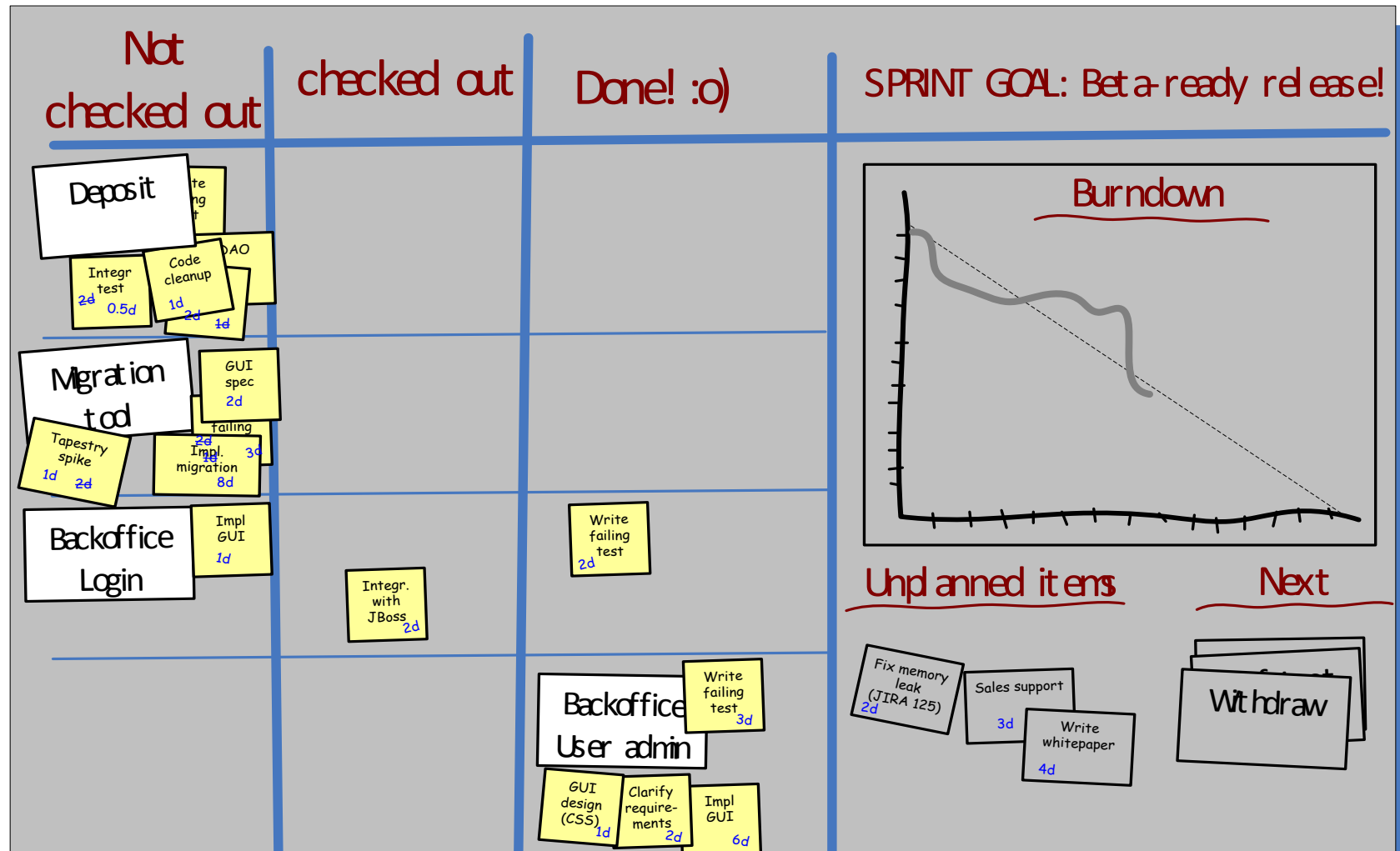
# Sprint burndown chart



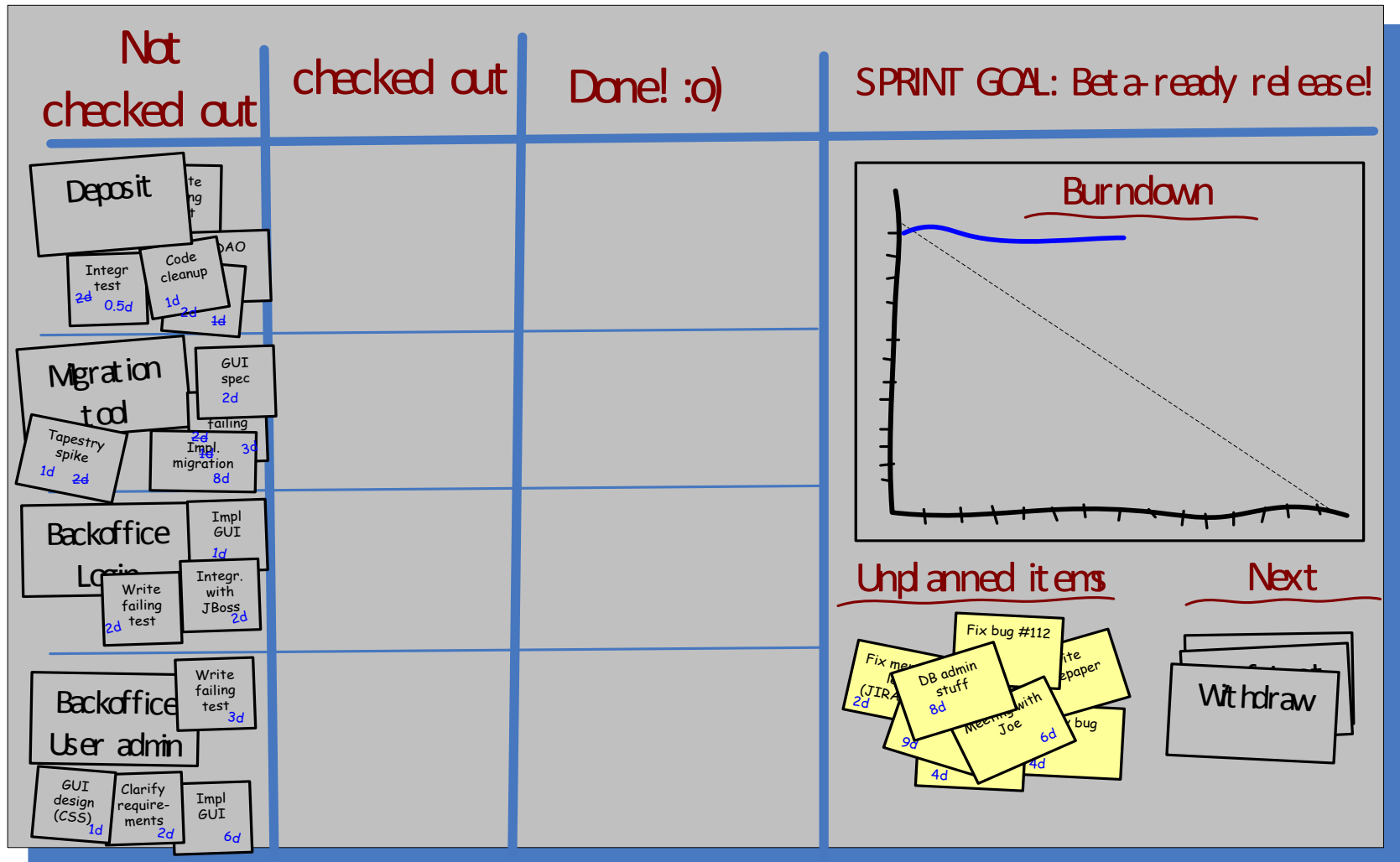
# Sprint backlog – day X



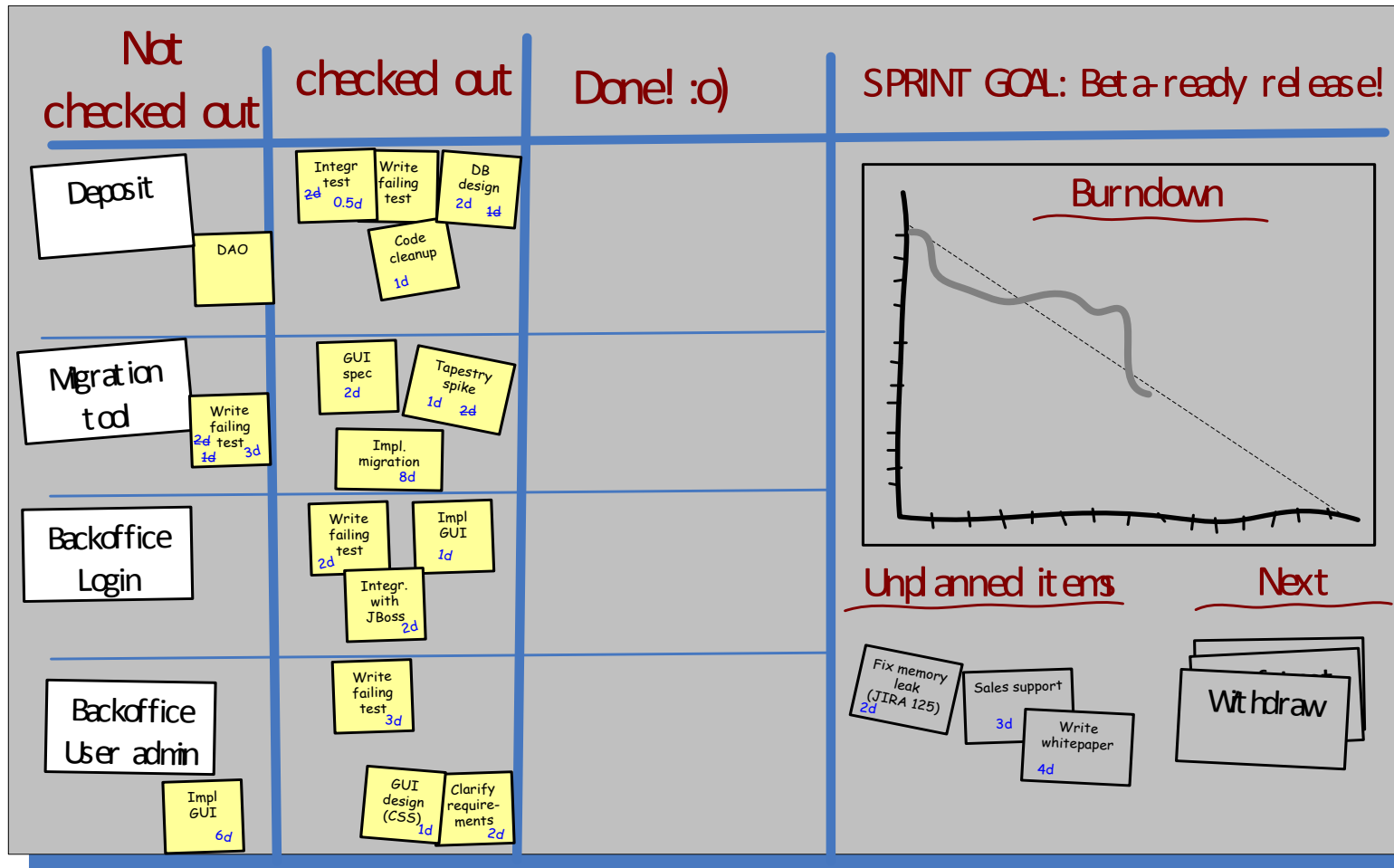
# Warning signals



# Warning signals



# Warning signals



Hur har man räknat på sin burndown?



# Finish a sprint

What did we manage to do?

Team demonstrate finished functionalities to stakeholders

Only things that is 100% finished can be shown

Makes it possible to get feedback from stakeholders

Feedback is added to product backlog

What have we learned

Before next sprint is started there is an improvement meeting

- What did we do well
- What could have been done better
- Lessons learned



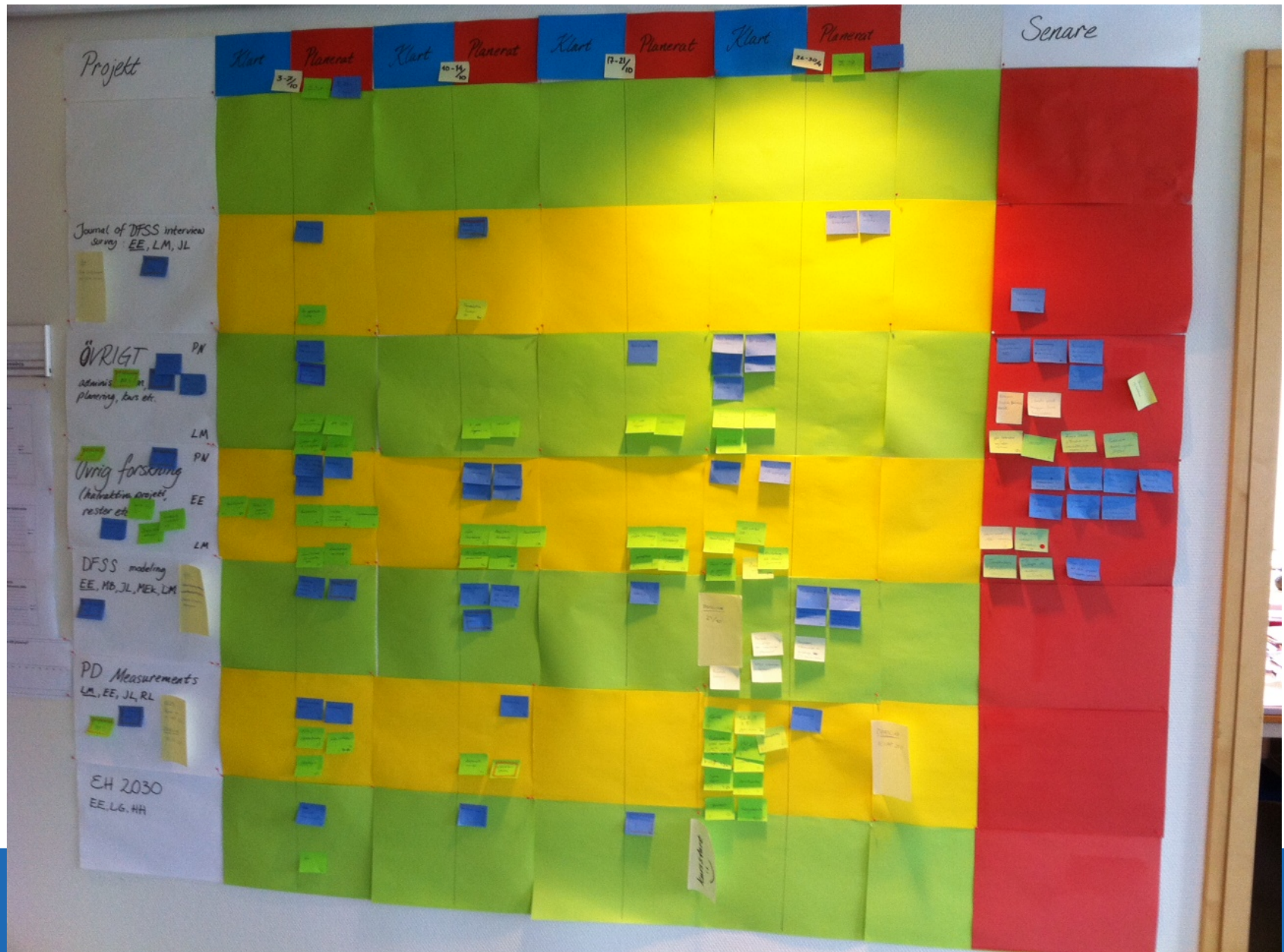
# Strength

Work becomes visual

Focus to solve problems as a team and continuous improvements

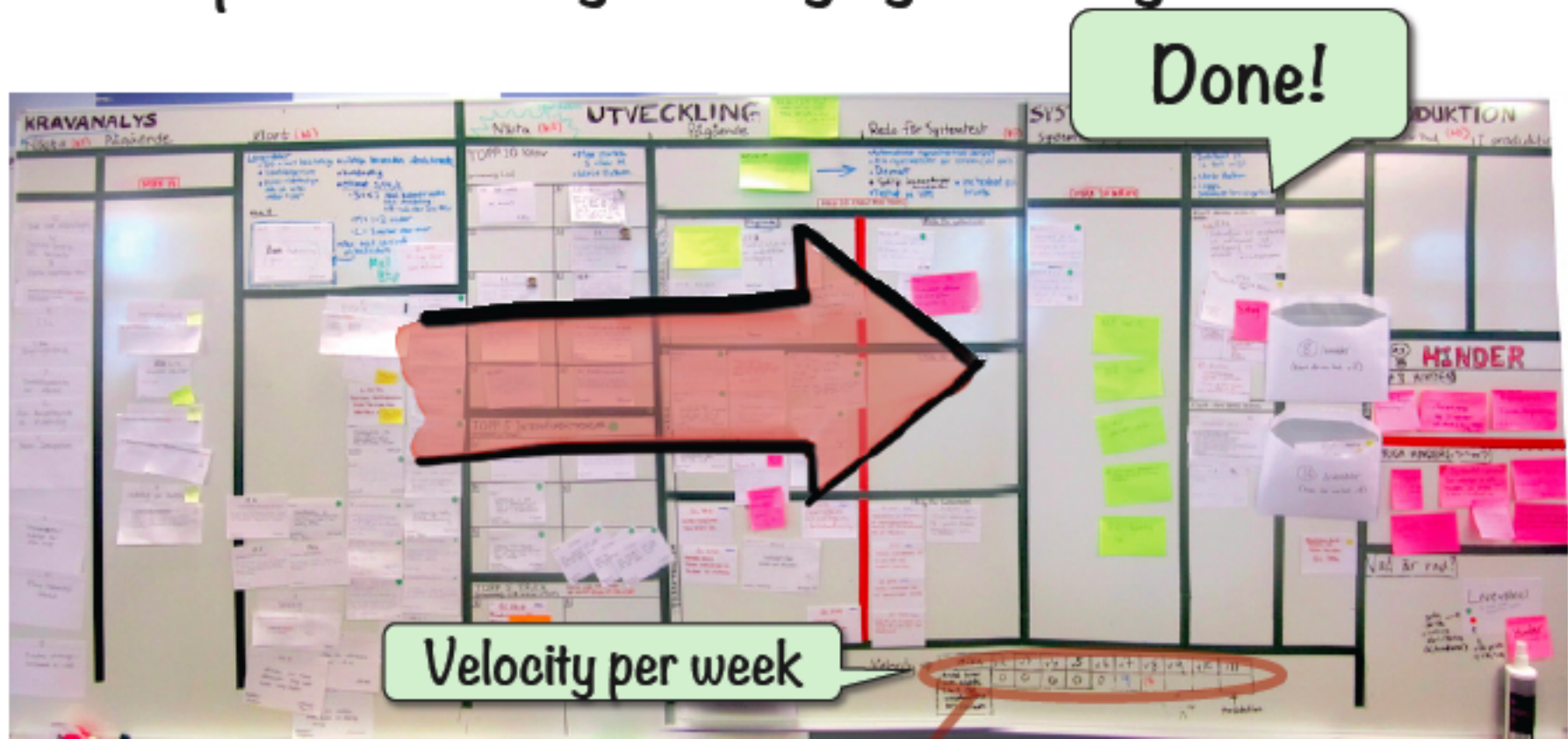
The developers are responsible for their own estimates

# Ex other approaches to visualisera





# Example: Measuring velocity by counting cards



Vecker	v10	v11	v12	v13	v14	v15	v16	v17	v18	
Antal nya funktioner som nått till 'Redo för Acitest'	4	0	2	4	5	0				
	↑ Prövsättn.						<b>VEL</b>			

Henrik Kniberg

From lecture with Henrik Kniberg @ KTH EH2720

# Visualization

How important is the information?

Toyota is trying to visualize as much as possible:

- *Problem analyzis*
- *Test resultat*
- *Project progress*
- *Etc.*

*"If you see, you can understand. If you understand, you can act"*

Tools:

- *A3 -thinking*
- *Obeya ("projectroom")*
- *Visual planning*



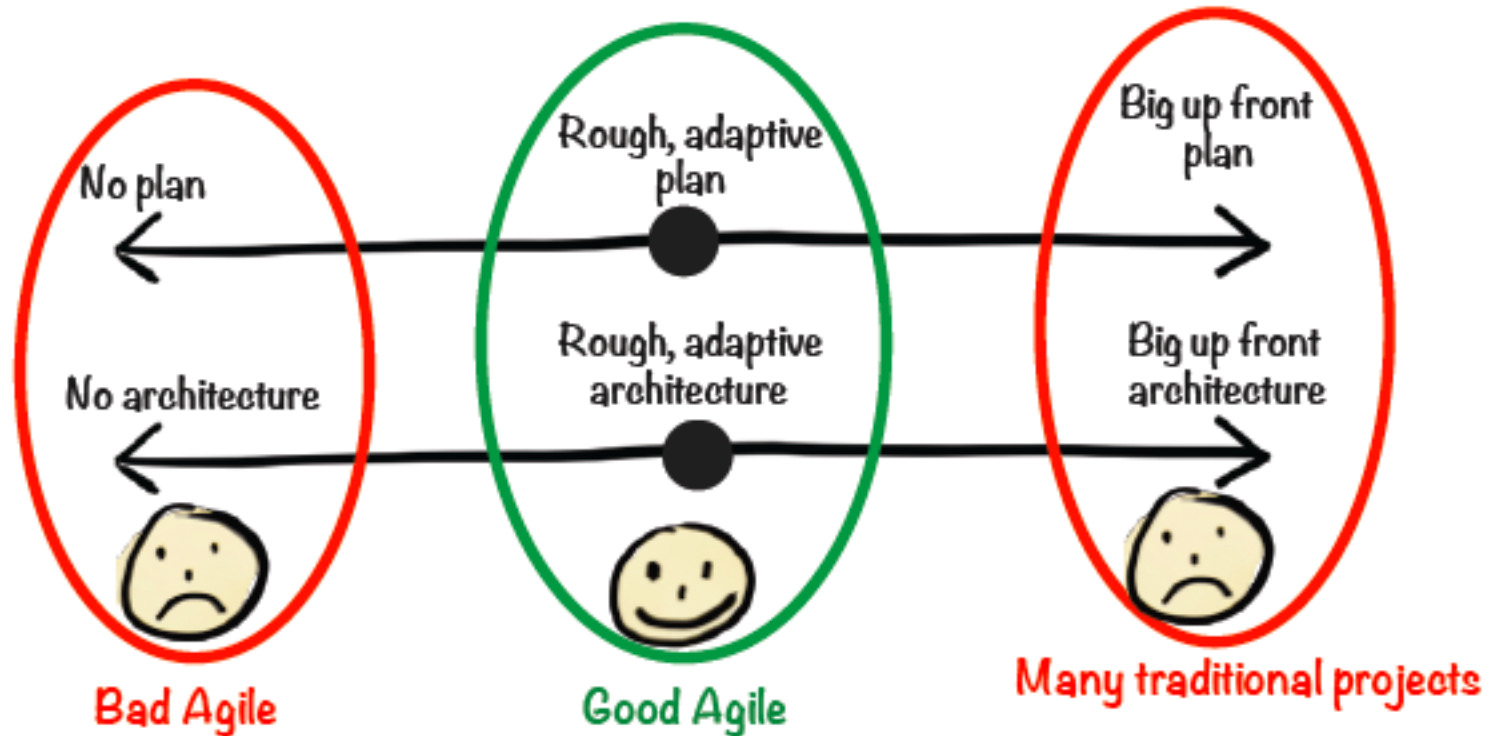
Scania's "pulse room"



# Pros and cons

Applying Scrum visualized in a research environment	
Positive effects	Negative effects
Helps structuring the working days.	Stressing with a lot of remaining activities at the end of the week.
Visualize “time thieves”.	Fully planned weeks foster cheat-working without defining activities on the pulse board
Creates awareness of timescales in the process and its activities.	Easy to over load the weeks with activities.
Provides immediate feedback by clarifying what the working time is used to.	Fires during a sprint are killed without re-planning the sprint and eliminating some activities.
Enables rating between projects.	Difficult to concretize activities and make them all on the same detail level.
Visualize work distribution between projects.	Over-loaded plans foster a stressed realization of activities.
Smoothing of time between participants in and between projects	
Foster increased structure since reflection of internal process between activities is necessary.	

# Don't go overboard with Agile!



# Some advice – for what is it worth

## Agile @ KTH?

How to recognize real agility:

- Work in small, cross-functional, self-organizing teams
- Release often & get real user feedback
- Focus on Value rather than Output/Cost
- Experiment a lot with product & process

- Find common working hours
- Use GIT to share code
- Use Trello or Google drive to visualize progress

Awesome (and free!) online tools:

### Trello



### Google drive



# Read more



Henrik Kniberg

From lecture with Henrik Kniberg @ KTH EH2720



# How to categorize risks in scrum

- Product risks
- Sociala risks
- Technical risks
- Cost/schedule



# Feedback

Give each other feedback after every sprint

Use format: I like – I wish

I like that you always are positive and have something to say

I wish you would contribute more on our meetings since the things you have to say is important and well thought through

(Could be used after each meeting for continuous improvement)





# Interesting links – Spotify engineering culture

## Part 1:

<https://labs.spotify.com/2014/03/27/spotify-engineering-culture-part-1/>

## Part 2

<https://labs.spotify.com/2014/09/20/spotify-engineering-culture-part-2/>