



**CHALMERS**  
UNIVERSITY OF TECHNOLOGY



UNIVERSITY OF GOTHENBURG



A horizontal brushstroke banner in red, yellow, green, and blue spans across the top of the slide.

# Agile Principles and Practices

Agile Development Processes

Eric Knauss

# Announcements



- Some students have joined late. Please register for group work:
  - [https://www.surveymonkey.com/r/EDA\\_397\\_2017](https://www.surveymonkey.com/r/EDA_397_2017)
- Any Roadblocks?



# Overview (might take 2 lectures)



- Principles
- Artefacts and Tools
- Practices
  - XP
  - Scrum
  - Kanban (you did watch the video, right?)
- Later: Lean software development

# Course Objectives

Knowledge and understanding	Skills and ability	Judgement and approach
Compare agile and traditional softw. dev,	Forming a team organically	Explain: people/commun. centric dev.
Relate lean and agile development	Collaborate in small software dev. teams	Apply fact: people drive project success
Contrast different agile methodologies	Interact and show progress continuously	Describe: No single methodology fits all
Use the agile manifest and its accompanying principles	Develop SW using small and frequent iterations	Discuss: methodology needs to adopt to culture
Discuss what is different when leading an agile team	Use test-driven dev. and automated tests	
Sprint 2	Refactor a program/design	
	Be member of agile team	
	Incremental planning using user stories	
Sprint 3		



# User story



- Widely accepted template: As a <role> I want to <feature> so that <customer value>
- Promise to discuss this user goal further
- Keep track of what the team is doing
- A lot of value in physical artifact
- Usually good idea to add: ID, Story Points, List of sub-tasks, Indication of customer value, notes on the backside

# Agile Principles – Revised list

(according to [Mey2014])

## Organizational

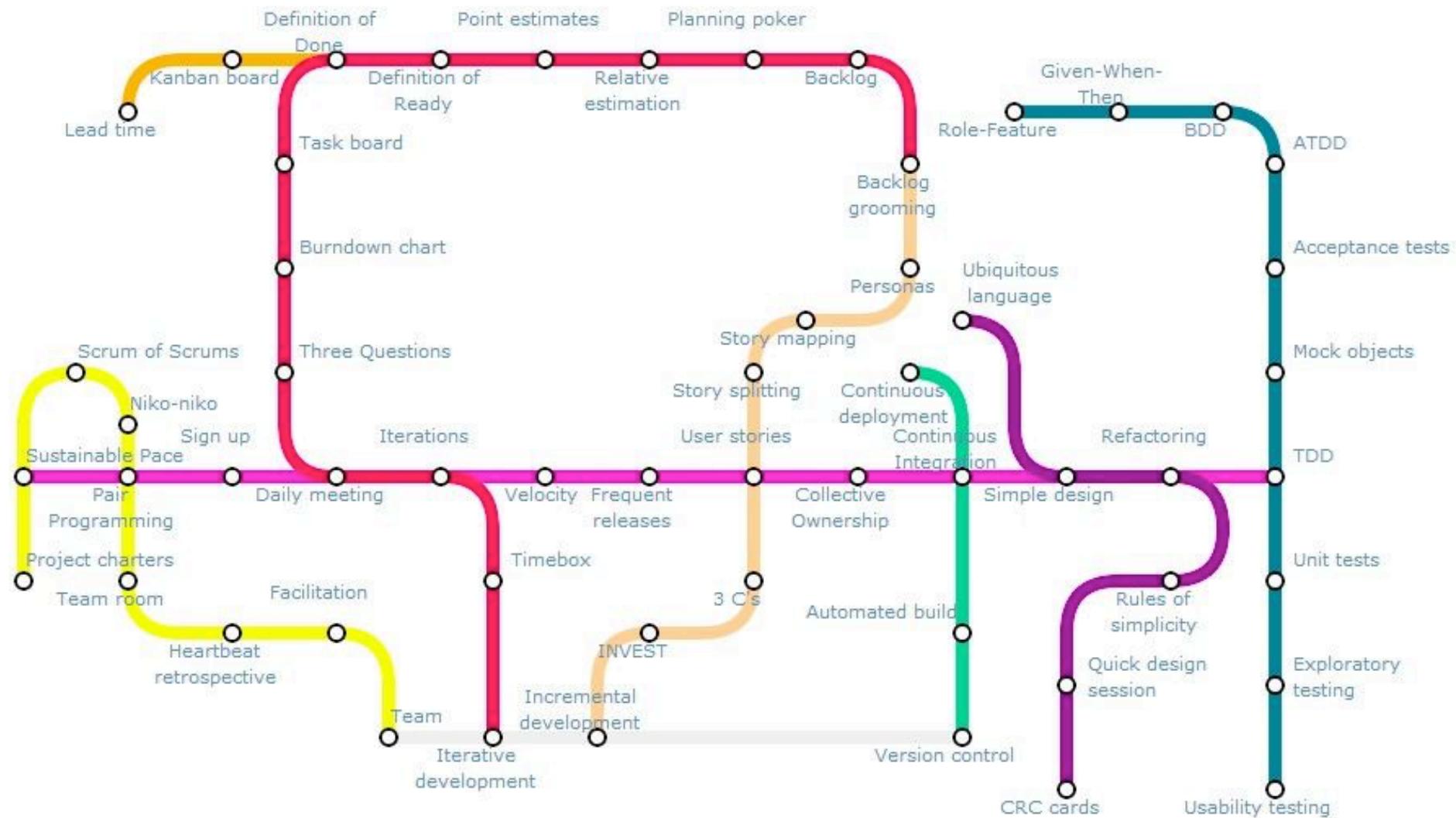
1. Put the customer at the center.
2. Let the team self-organize.
3. Work at a sustainable pace.
4. Develop minimal software:
  1. Produce minimal functionality.
  2. Produce only the product requested.
  3. Develop only code and tests.
5. Accept Change

## Technical

1. Develop iteratively:
  1. Produce frequent working iterations.
  2. Freeze requirements during iterations.
2. Treat tests as a key resource:
  1. Do not start any new development until all tests pass.
  2. Test first.
3. Express requirements through scenarios.

**6. Reflect regularly and improve continuously!**

# Agile Practices



Lines represent practices from the various Agile "tribes" or areas of concern:

<https://techblog.betclicgroup.com/wp-content/uploads/2013/12/agileSubwayMap.pdf>

Also check: <http://guide.agilealliance.org>

Extreme Programming  
Teams  
Scrum

Scrum  
Product management  
Design

Design  
Testing  
Fundamentals

# XP Principles

## Core

- Rapid feedback
- Assume simplicity
- Incremental change
- Embracing change
- Quality work

## Less central

- Teach learning
- Small initial investment
- Play to win
- Concrete requirements
- Open, honest communication
- Work with people's instincts, not against them
- Accepted responsibility
- Local adaptation
- Travel light
- Hones measurement

# XP Practices



- Planning game
- Small releases
- Metaphor
- Simple design
- Testing  
(dedicated lecture)
- Refactoring
- Pair programming
- Collective ownership
- Continuous integration  
(dedicated lecture)
- 40h week
- On-site customer
- Coding standards

# Planning Game

- Basic idea
  - Create a rough iteration plan
  - Plan next iteration in detail
- Players
  - Onsite Customer
  - Team
  - (Coach)
- Moves
  - Write User stories (OC + Team)
  - [do some filtering]
  - OC ranks US by business value [assign Fibonacci numbers]
  - Team assigns story points [Planning poker]
  - OC put story cards in strictly prioritized order
  - Team selects the top n stories to match their velocity



# Pair Programming

- Basic idea
  - Two people on one computer
  - Take **one** user story **with them**
  - Review each other
  - Share knowledge
- Related concepts
  - User story as vertical increment
  - Truck factor
- Sentiment in research
  - Studies with contradictory results, clearly leaning towards rejecting the idea of using pair programming in general
  - But Long term aspects of knowledge management not/insufficiently covered
  - Good results for specific use cases (exploration, difficult task, knowledge sharing, ...)



# Scrum Principles



- Reflection
  - Stop and review product & process
- Self-correction
  - Based on reflection
- Visibility
  - Everything is visible (=known) for all stakeholders,  
e.g. plans, schedules, issues, ...

# Scrum practices

- Product backlog vs. Sprint backlog
- Sprint planning
  - Planning poker: estimate cost
  - ROI (Return on Investment): cost vs. benefit
- Retrospective
- Fixed sprint length
- Burn-down charts
- Daily scrum: no longer than 15min





[https://en.wikipedia.org/wiki/Scrum\\_\(rugby\)](https://en.wikipedia.org/wiki/Scrum_(rugby))

## Overview: XP and Scrum

# SCRUM Practices

Hints

## • SCRUM Meetings

- SCRUM Master minds the time (2-3 min./Person)
  - Standup-meeting: faster
  - Replace status meetings – safe time
- Important: Always same time and place!
  - (not important: where)
  - Daily / frequent meetings avoid long/quiet crisis
- Content
  - What was done since the last meeting?
  - What is planned to be done before the next meeting?
  - Found obstacles? Write on whiteboard!
- Useful: Share information and facilitate social aspects
- Schedule further meetings to follow up on things (e.g. obstacles)

# SCRUM Practices

## Hints

- **Sprint**

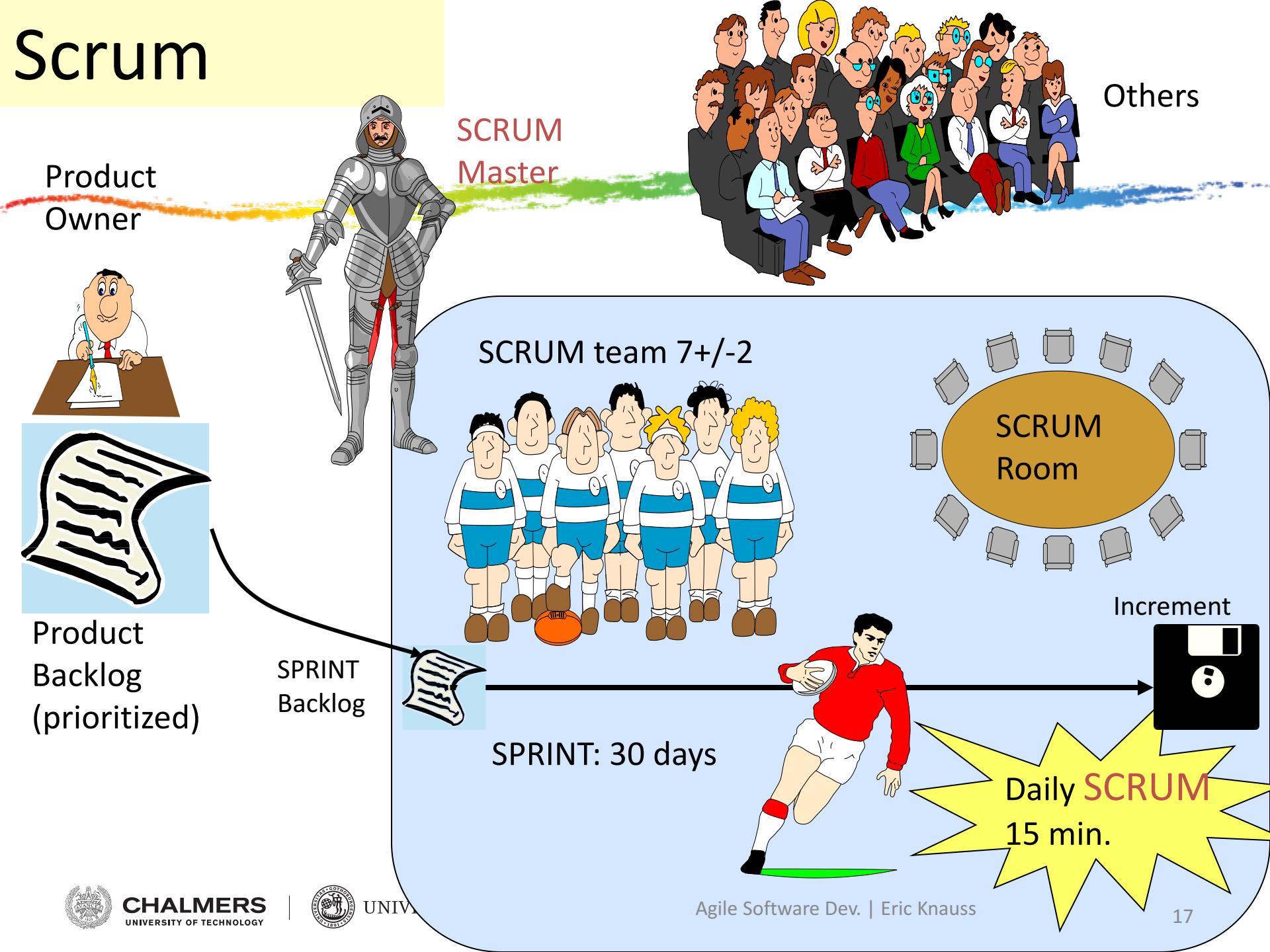
- During sprint: Autonomous team: *Pioneers*
  - No new requirements / no changes
  - No external influences
  - Only sprint goal
    - Fixed: Time (approx. 30 Tage), Cost (Developers etc.), Quality
    - Variable: Functionality
      - » Team can adjust details and scope of functionality based on the time-cost-quality frame and with respect to the sprint goal

- Sprint can be cancelled
- After Sprint: 4h Sprint-Meeting
  - Avoid long preparation (max. 2h)
  - Avoid slides
  - Often very informal

- **Adjust SCRUM (longer Sprints, other Meetings...)**

- Okay, after being successful with the traditional setup
- Only based on experiences – never without experience

# Scrum



# Scrum: organization

Project-goals: Project Backlog

Coord. of  
multiple  
Teams

Multiple SCRUM-Teams sprint in parallel. Master-SCRUMs

Project of  
SPRINTS

SPRINT im  
30 SCRUM-Takt

A workday:  
SCRUM to SCRUM

SPRINT  
With Backlog and Goal

SPRINT creates  
a Product-Increment

SPRINT  
Review 3h

Planning



Workday

Daily SCRUM Post-  
Build (15 min.) discuss.

Or alternatively:



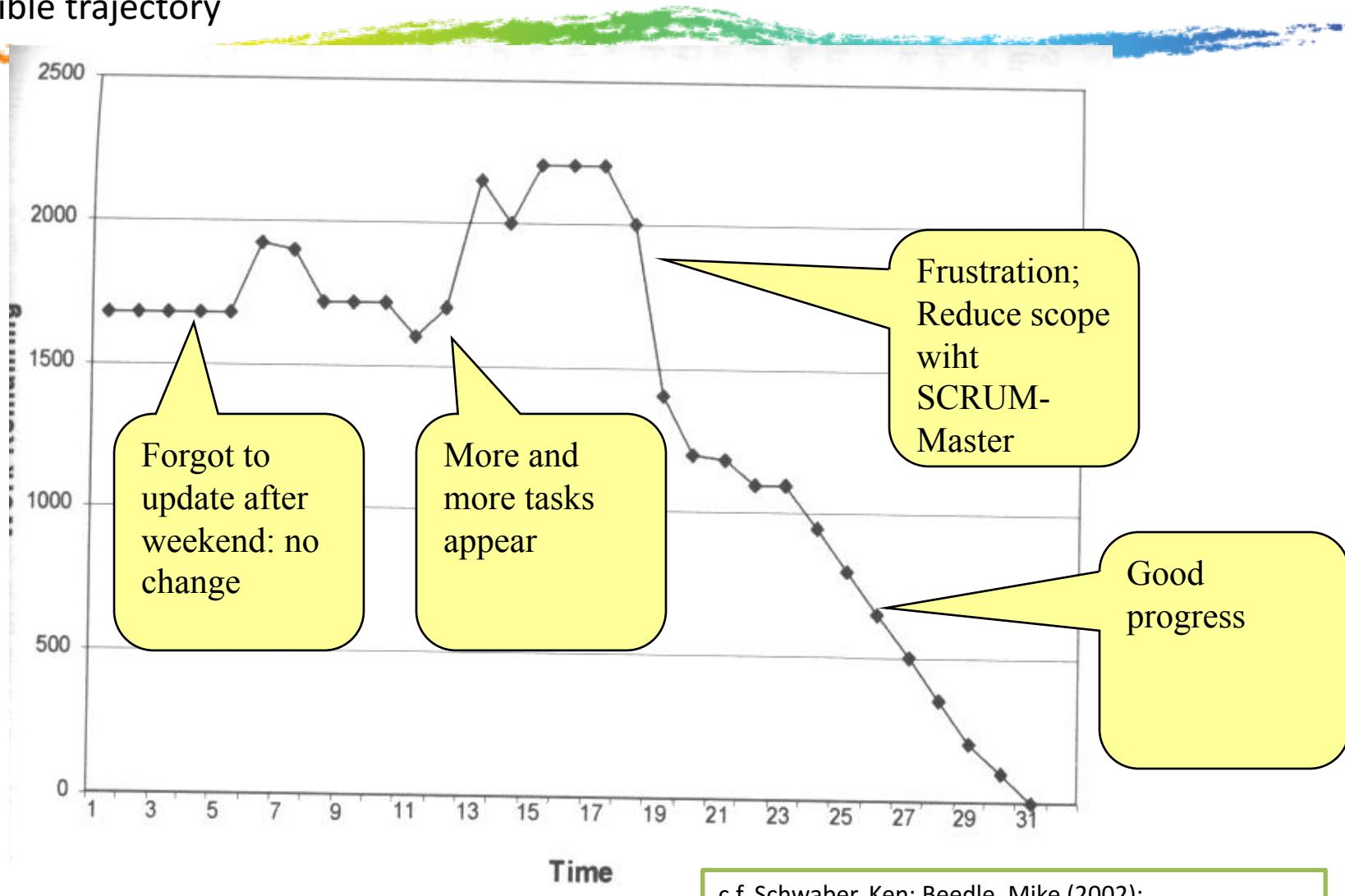
# SCRUM vs. XP



- XP is often hard to introduce
- SCRUM is easy to introduce (according to Schwaber)
- Best-practice: Combine!
  - SCRUM organizational shell: *Day-to-day management*
  - XP method of implementation
  - Shared values with XP
    - Quickly generate executable code
    - Facilitate communication

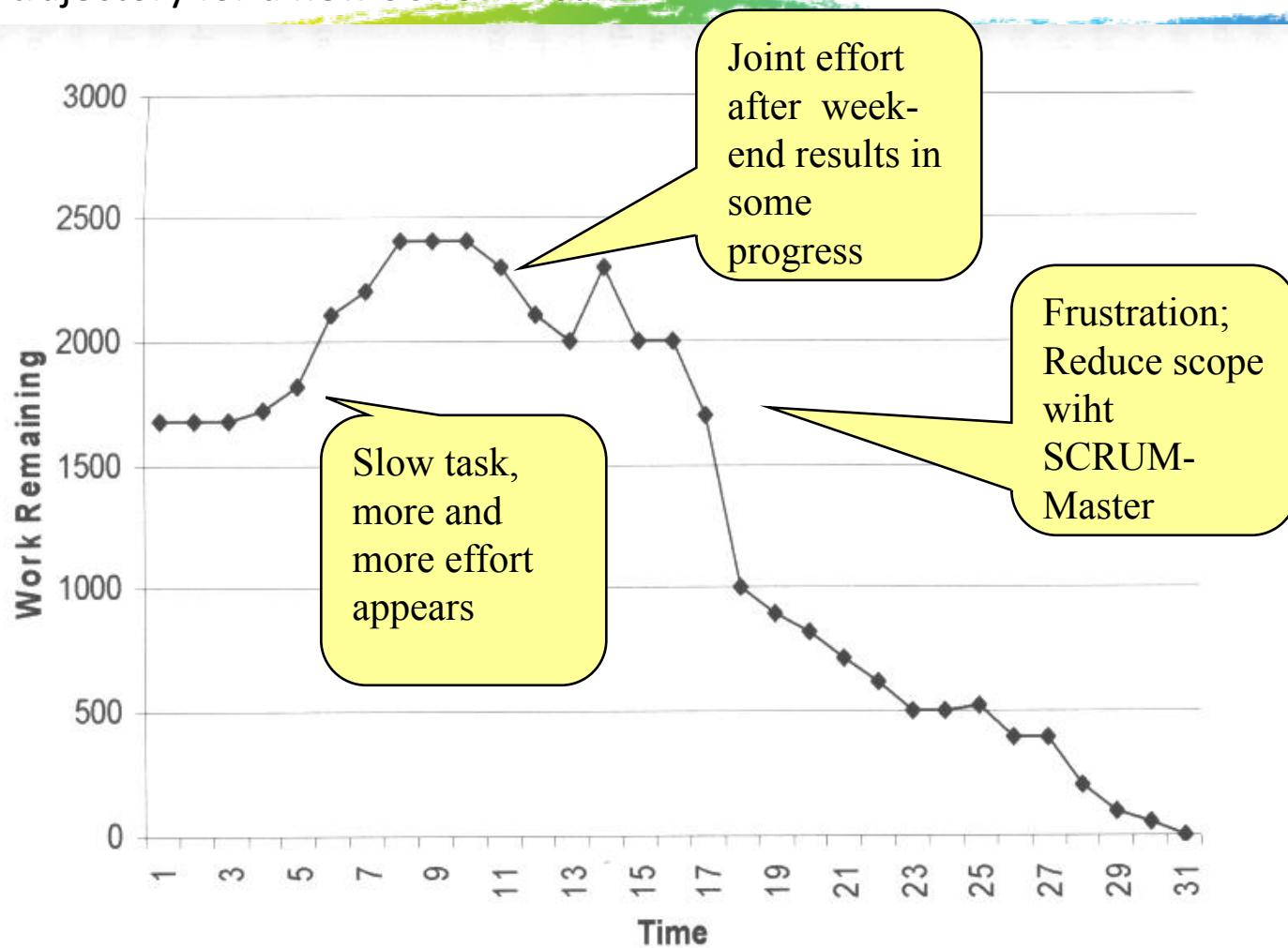
# Assess Sprint Progress

A possible trajectory



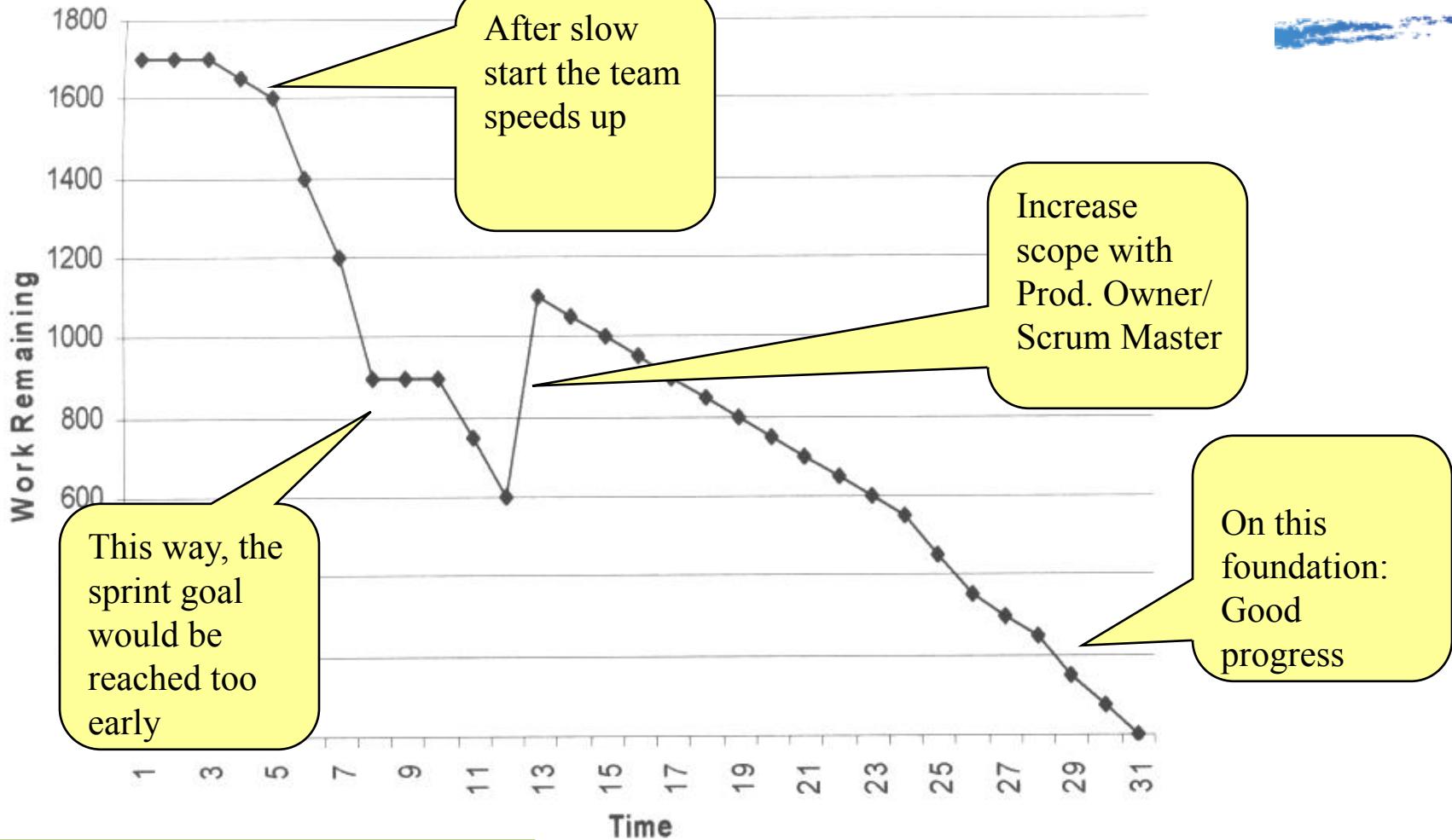
# Assess Sprint Progress

a typical trajectory for a new SCRUM-Team



# Assess Sprint Progress

an



C.f. Schwaber, Ken; Beedle, Mike (2002):  
Agile Software Development with Scrum. Prentice Hall.

# Why does SCRUM work?

- Integrated instability
  - Not too smoothly
- Self-organizing teams
  - Take ownership
- Multi-Learning
  - Between functions
  - Between group, organization, and individual
- Subtle control
- Constant learning
  - Experienced developers in new teams

## *Risk management*

- Risk: Customer unhappy
  - Show working system often
- Risk: Incomplete feature set
  - Prioritize: If something is missing, it is not important
- Risk: Bad estimation
  - Daily updates during SCRUM
- Risk: Lack of experience with Development cycle
  - Test early and execute repeatedly
- Risk: Changes in performance estimation
  - No impact on Sprint

c.f. Schwaber, Ken; Beedle, Mike (2002):  
Agile Software Development with Scrum. Prentice Hall.

# Summary SCRUM



- SCRUM is a management shell
  - Around XP
  - Or other approach: Even waterfall possible
- Overlap with XP, but differences exist
  - Similar values
  - Different practices
  - Partly complement each other
- Not as much impact as XP, easier to introduce
- Strength
  - Information flows not only in one direction
  - Multiple feedback cycles stabilize system



# Kanban principles



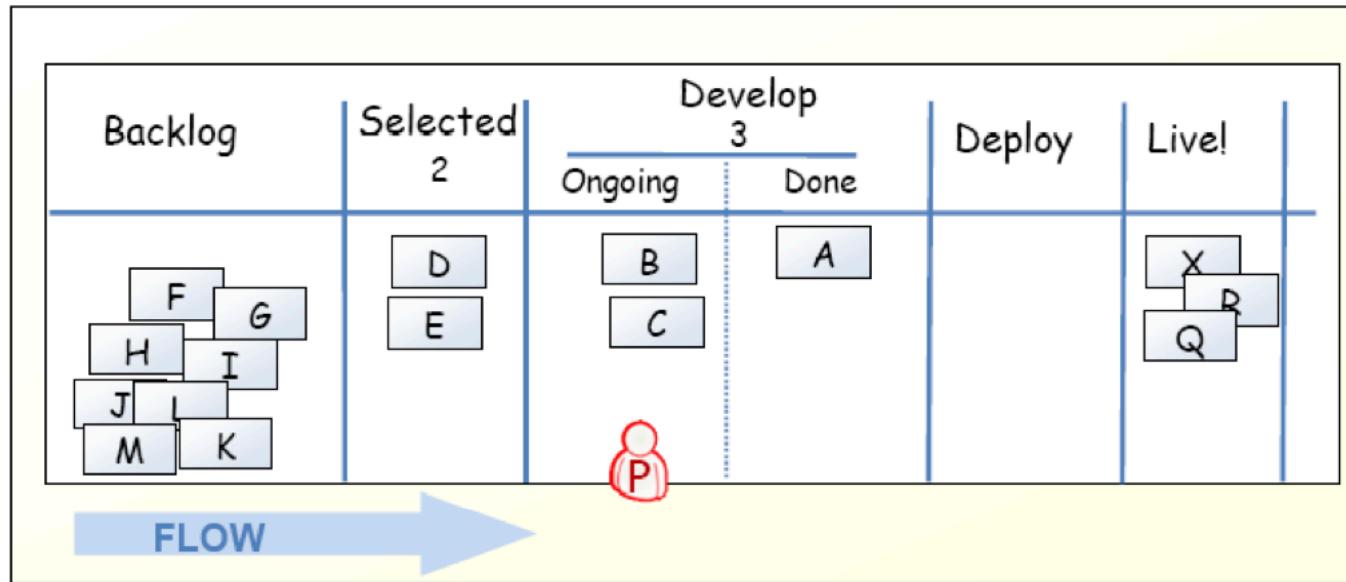
- Start with what you do now
- Agree to pursue incremental, evolutionary change
- Respect the current process, roles, responsibilities and titles
- Leadership at all levels

The following is based on [KS2009]

# Kanban core practices

- Visualize
  - Visualization of workflow allows to understand and improve it
- Limit Work-in-Progress
  - Limit the amount of workitems for each step
  - Introduce a pull-system
- Manage flow
  - Measure how workitems flow through the process and understand, if a change improves the situation
- Make policies explicit
- Implement feedback loops
  - Understand (as a team) how good the process is working
- Improve collaboratively, evolve experimentally
  - Whole team needs to share a theory on why (small) change helps

# Kanban-Board



[KS2009]

# Limit work in progress

- Prevent context switching
  - Reduce multi-tasking
  - performing tasks sequentially yields results sooner
- Maximize throughput
- Enhance teamwork
  - working together to make things done
  - increase cross-functionality

# WIP Strategy



- Start with some initial value
  - Small constant (1-3)
  - number of developers
  - number of testers
- Measure the cycle time
  - average time of one piece full cycle flow
- Change limit to decrease cycle time

# Idle Members



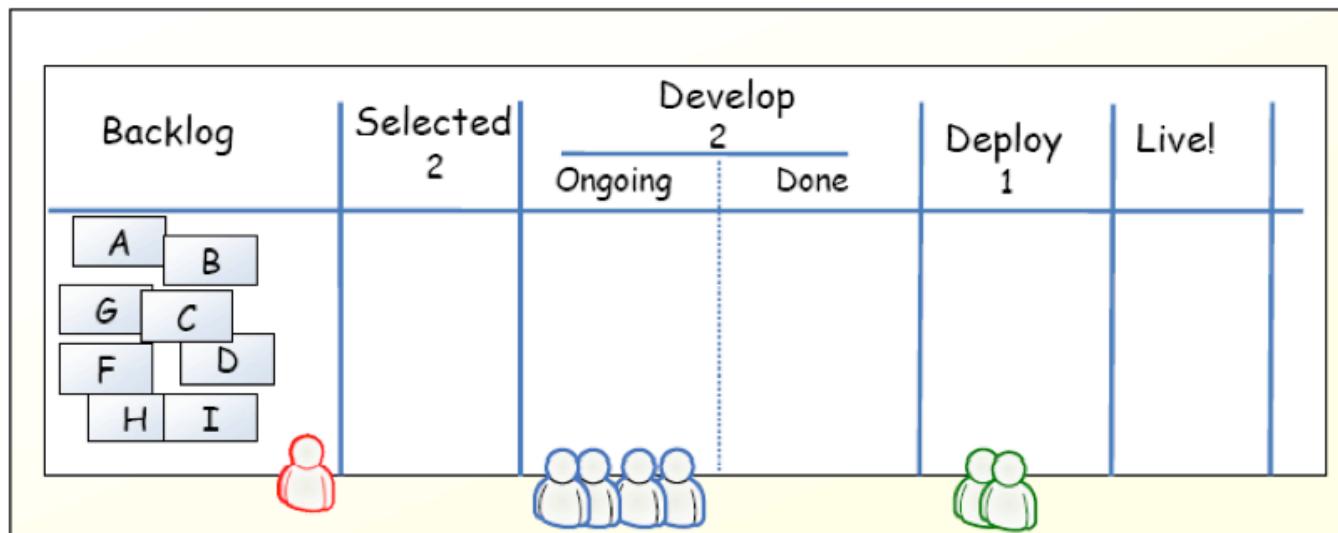
- Can you help progress an existing kanban? – Work on that.
- Don't have the right skills? – Find the bottleneck and work to release it.
- Don't have the right skills? – Pull in work from the queue.
- Can't start anything in the queue? – Check if there is any lower priority to start investigating.
- There is nothing lower priority? – Find other interesting work (refactoring, tool automation, innovation).

# Metrics



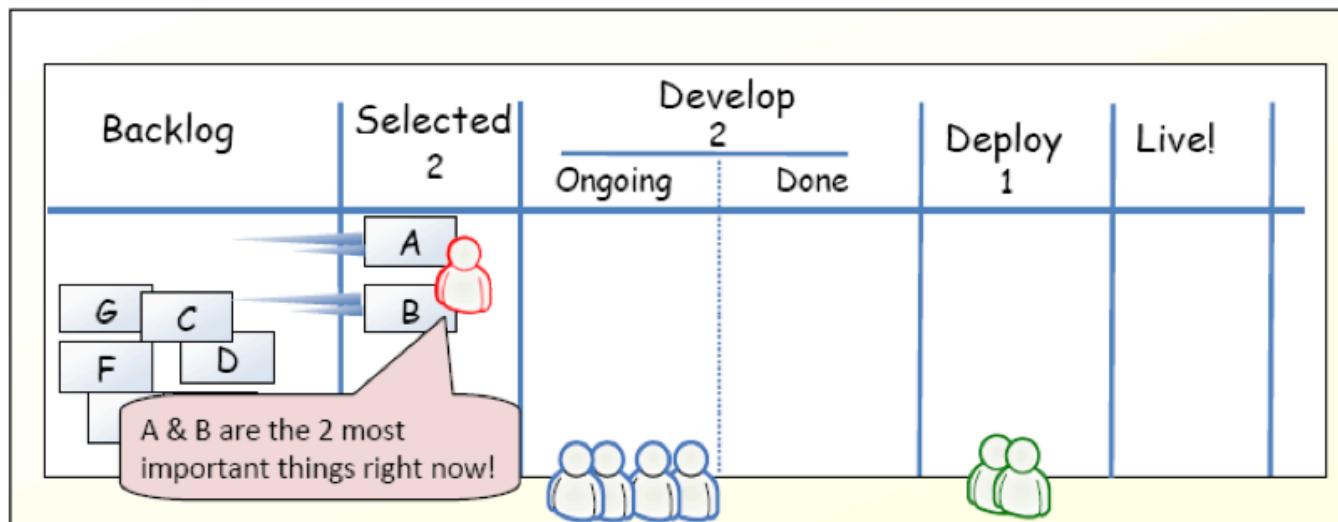
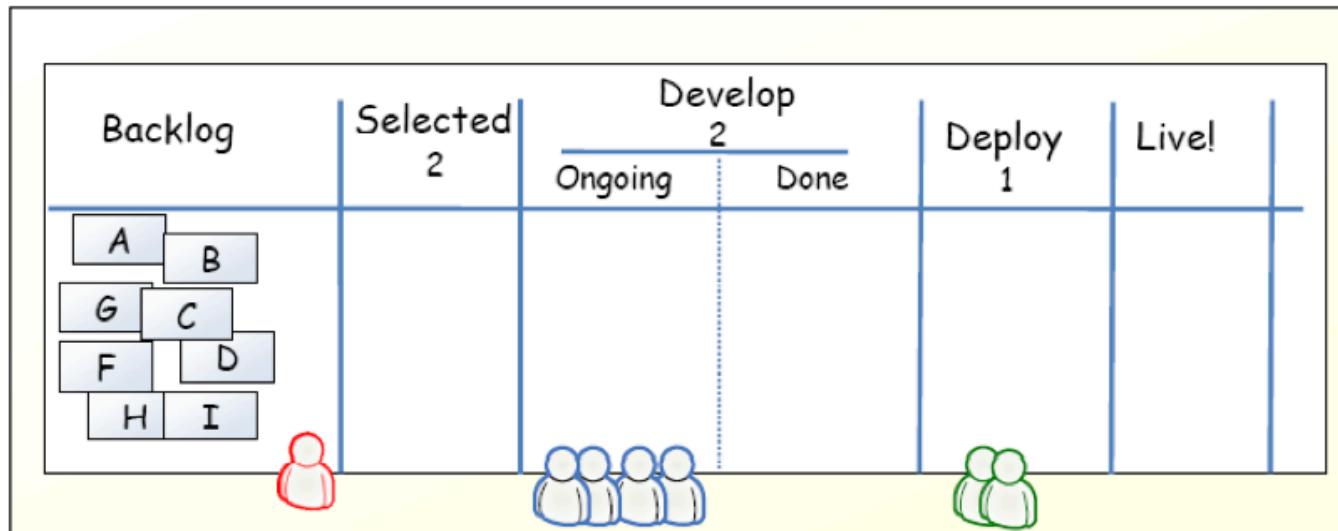
- Stories in progress (SIP)
- When story enters stories queue set entry date (ED)
- When story enters first process step set start processing date (SPD)
- When story is done set finish date (FD)
- Cycle time (CT) = FD – SPD
- Waiting time (WT) = SPD – ED
- Throughput (T) = SIP / CT

# Example



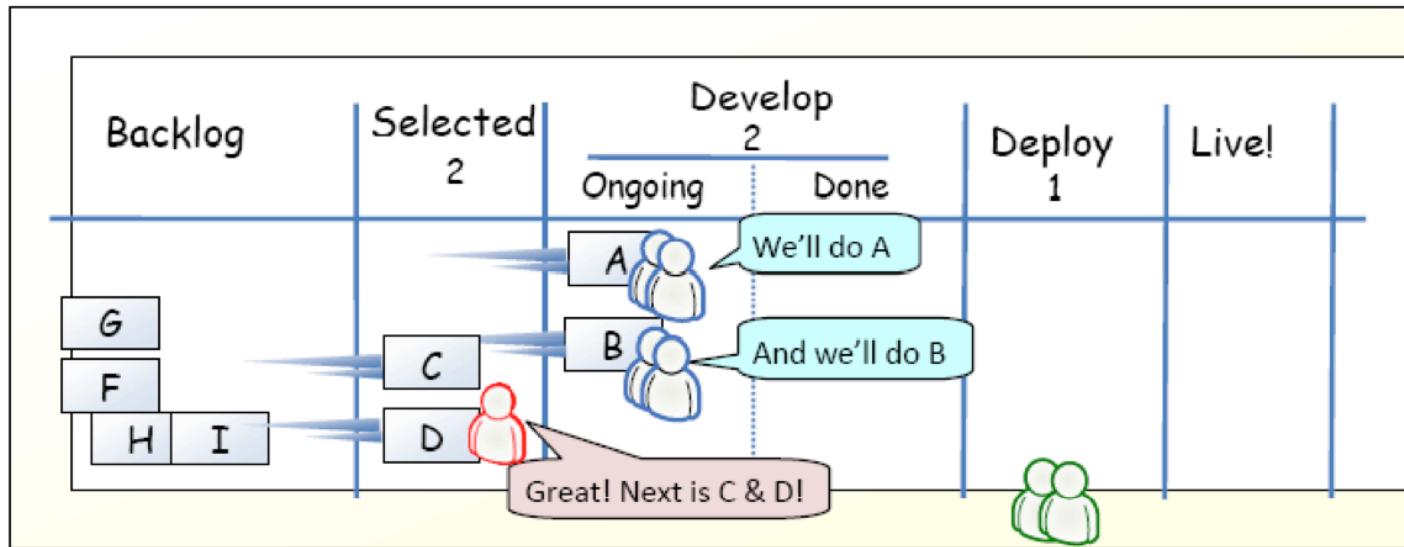
[KS2009]

# Example



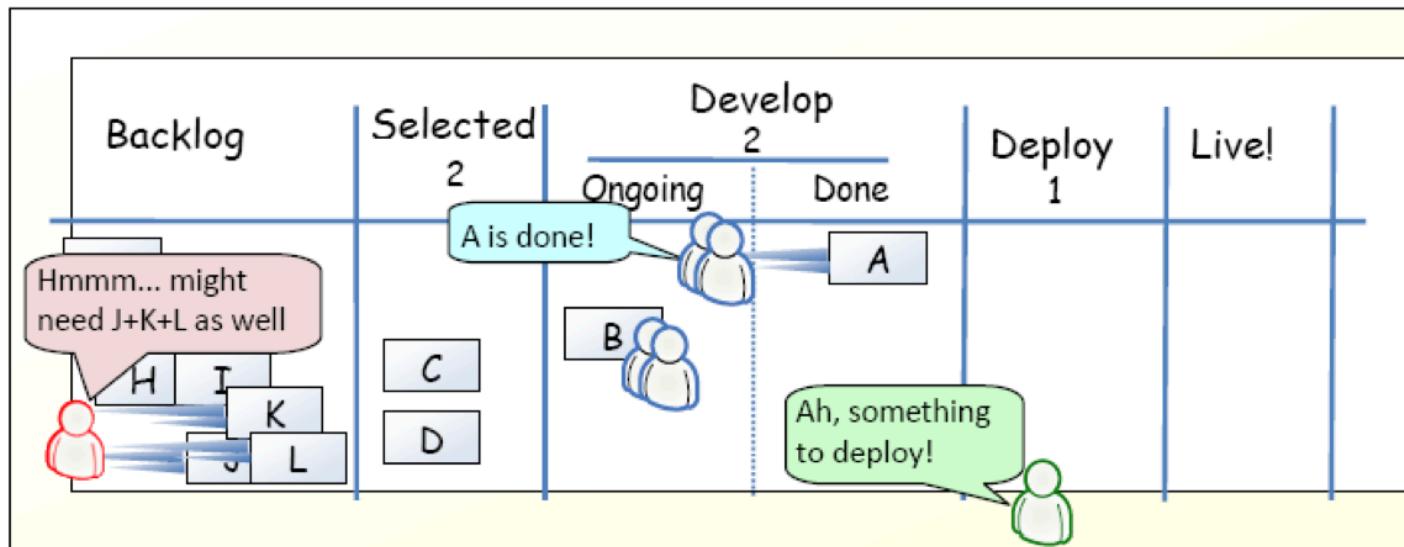
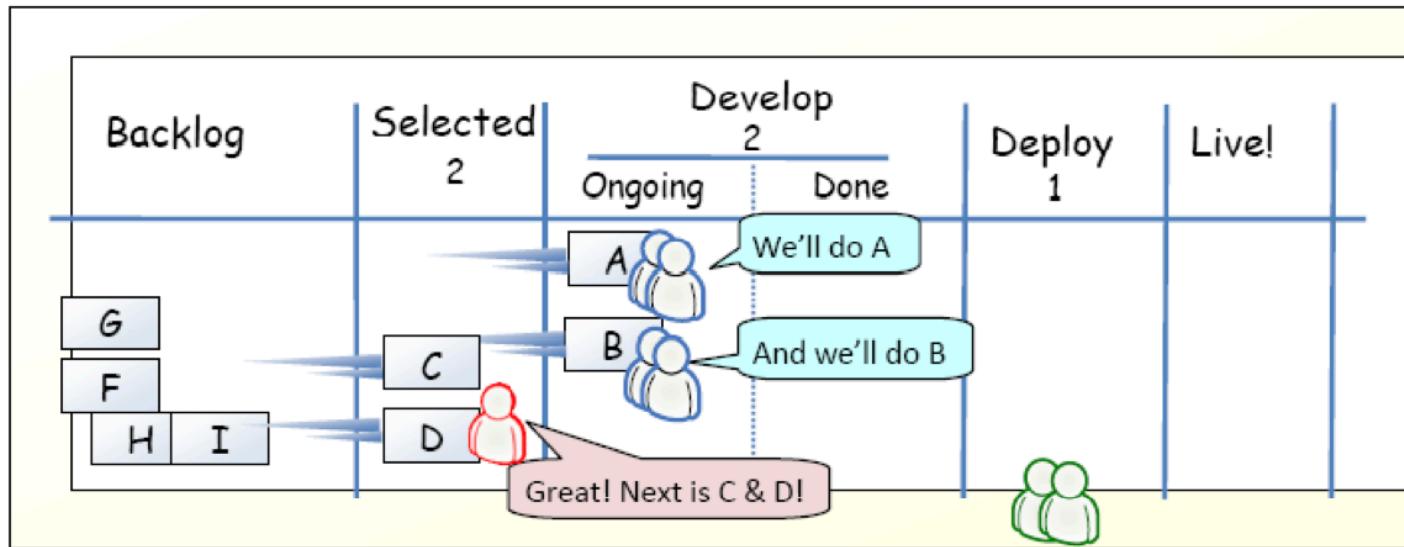
[KS2009]

# Example



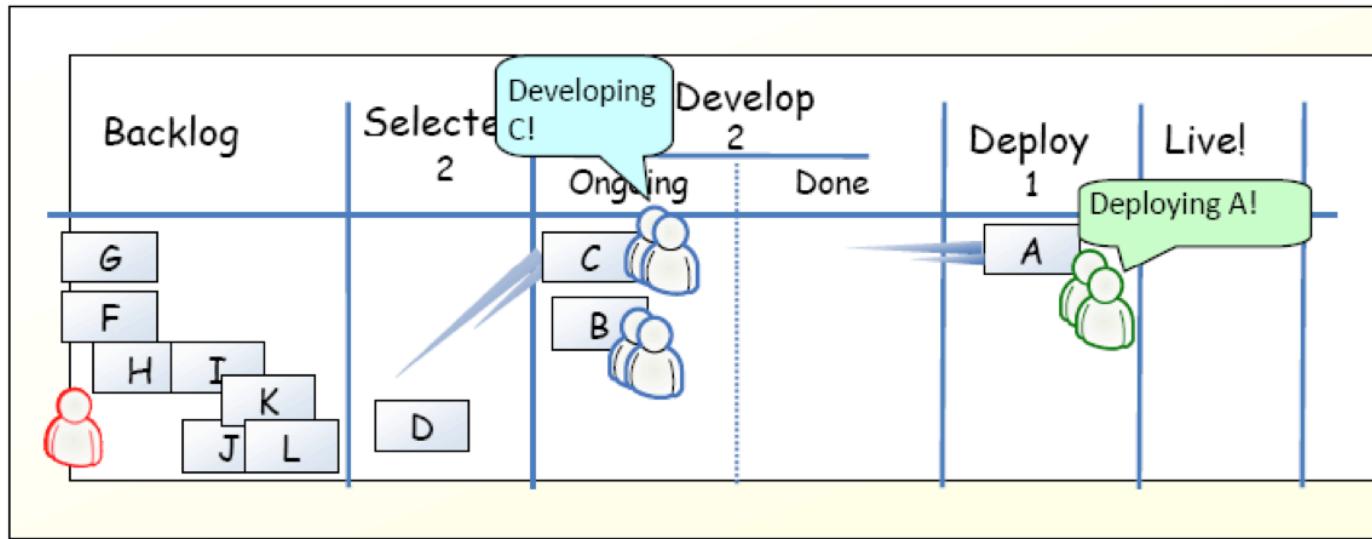
[KS2009]

# Example



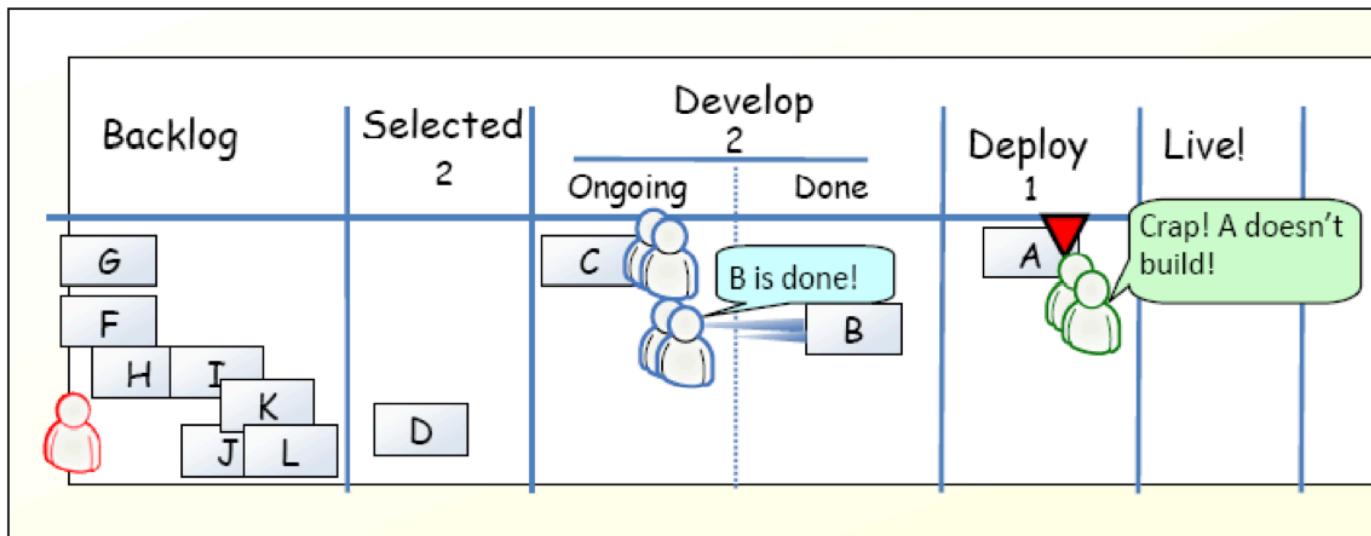
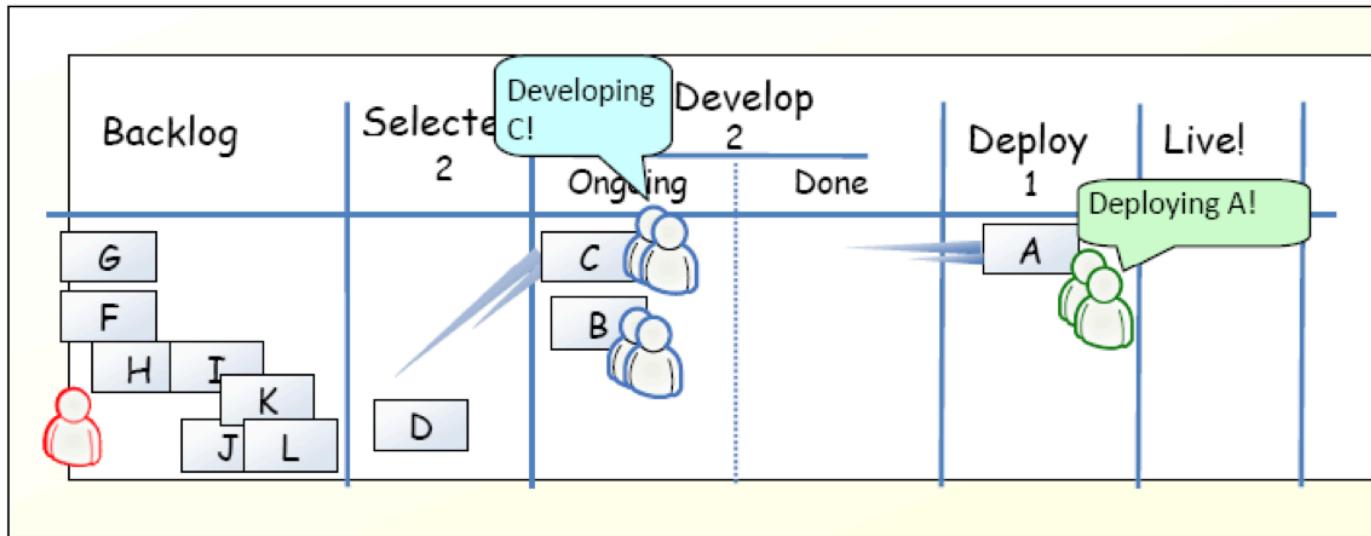
[KS2009]

# Example



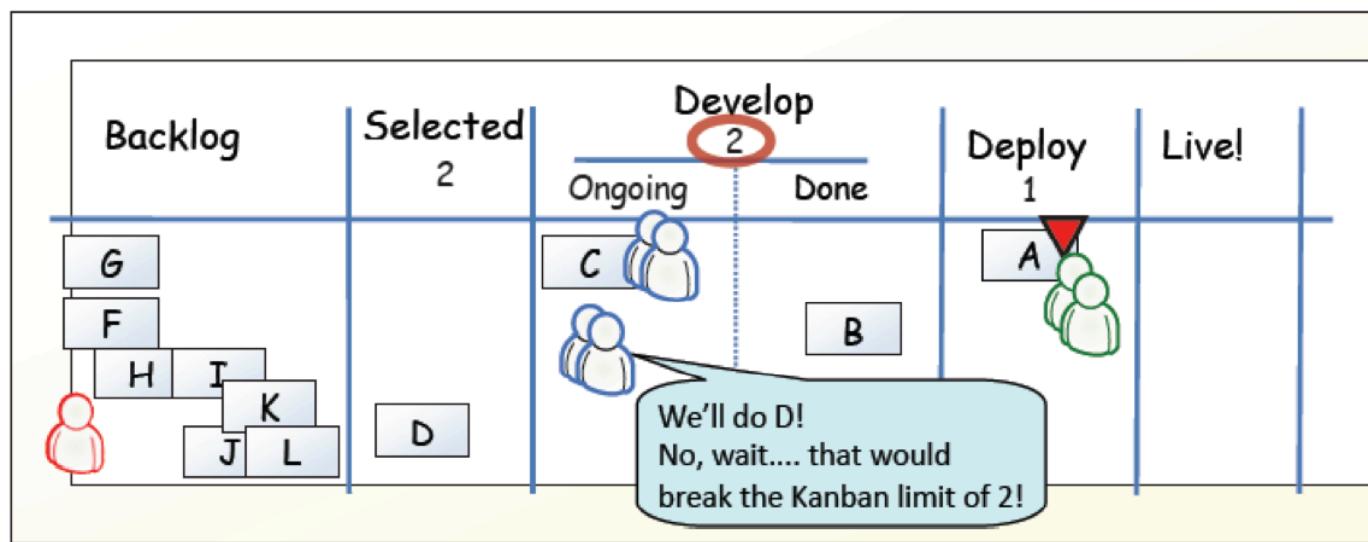
[KS2009]

# Example



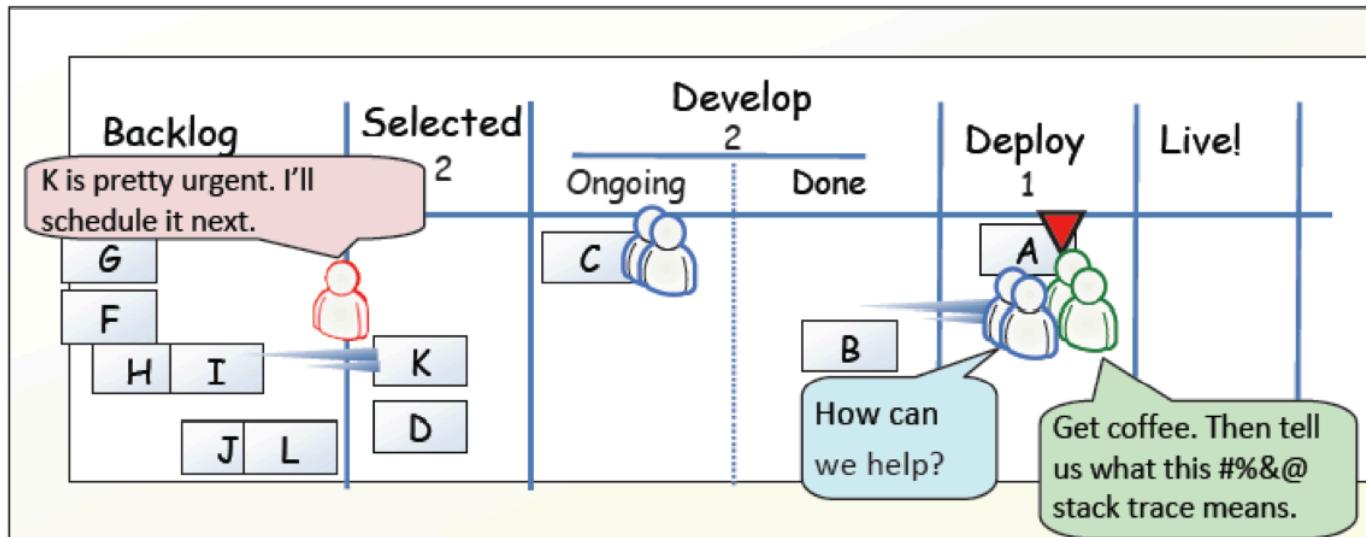
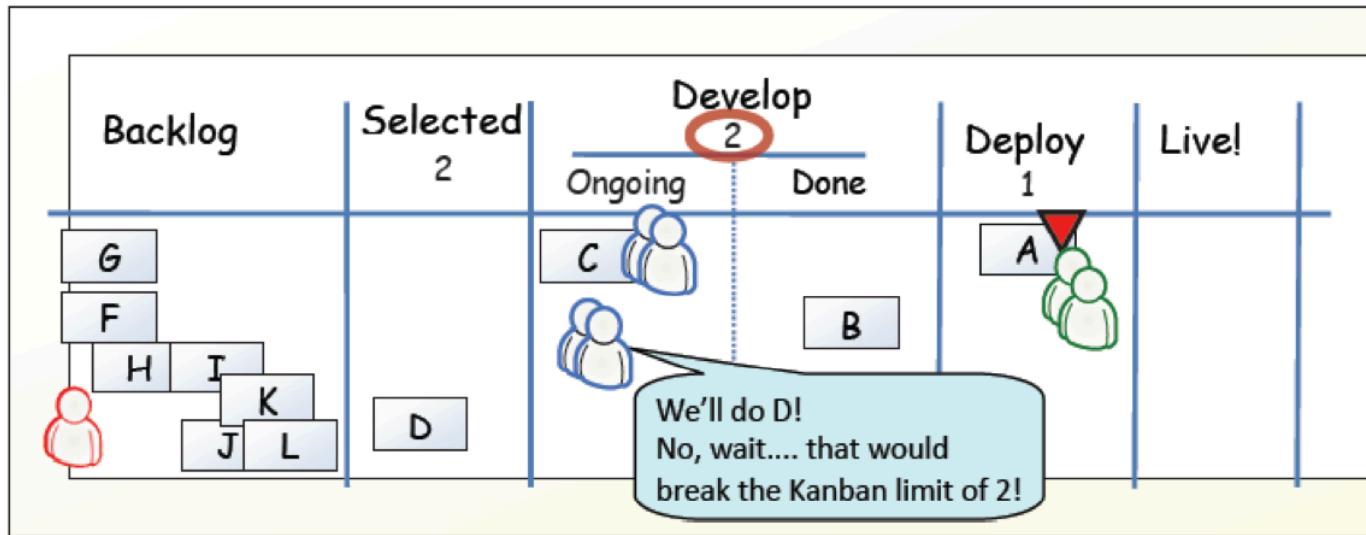
[KS2009]

# Example



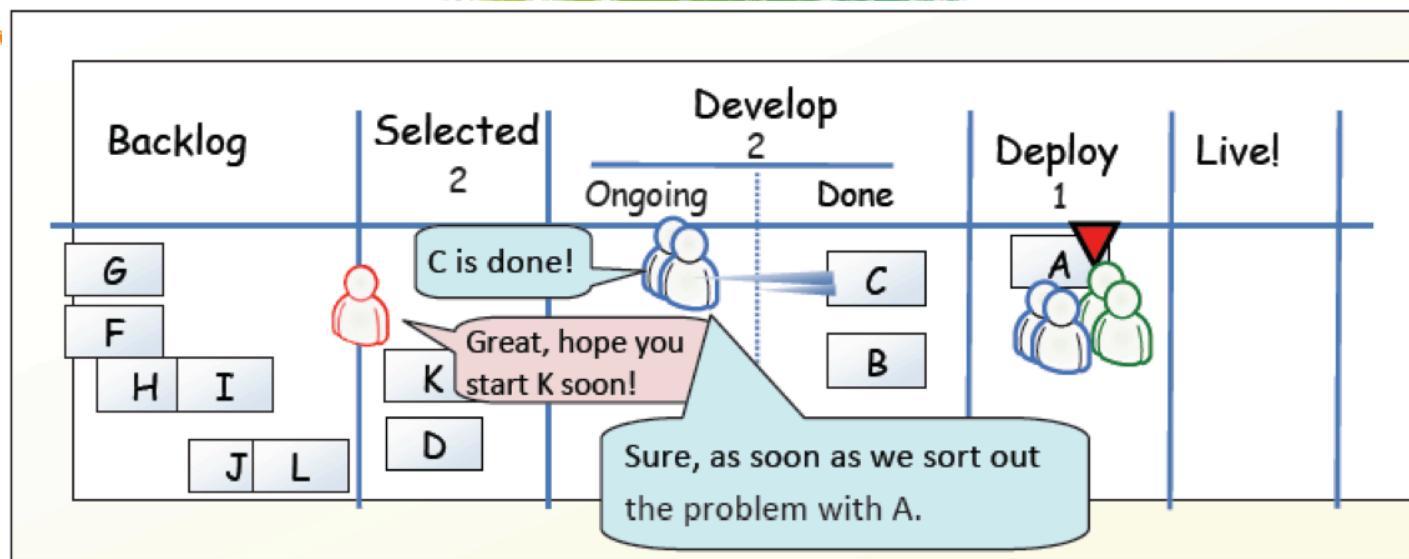
[KS2009]

# Example



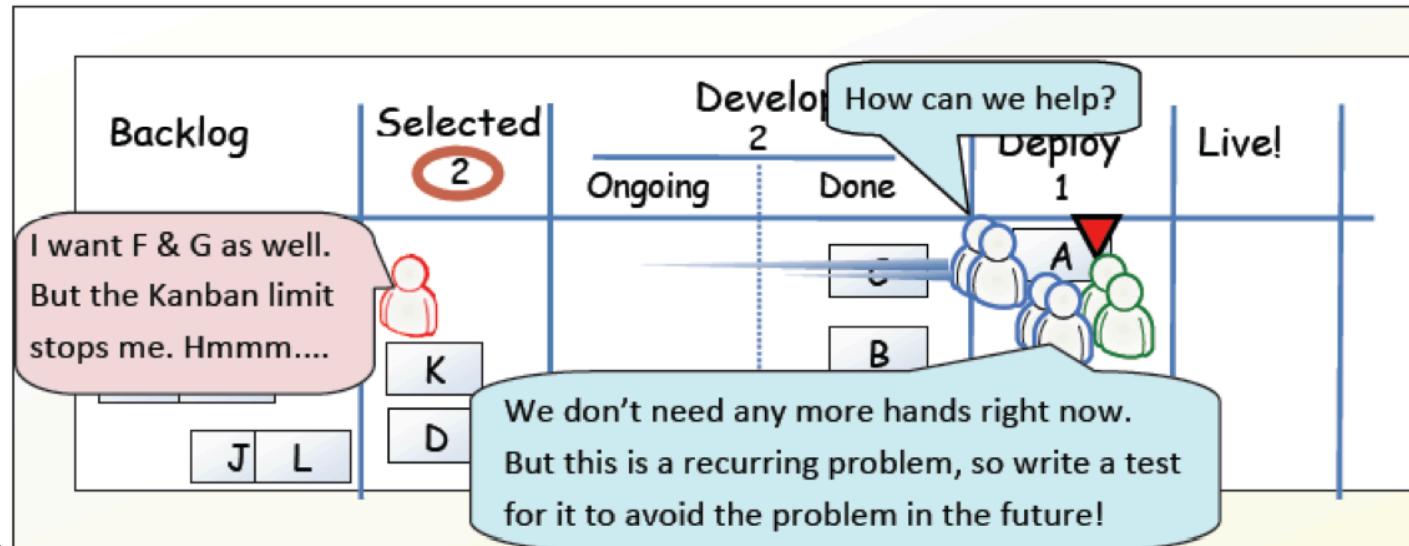
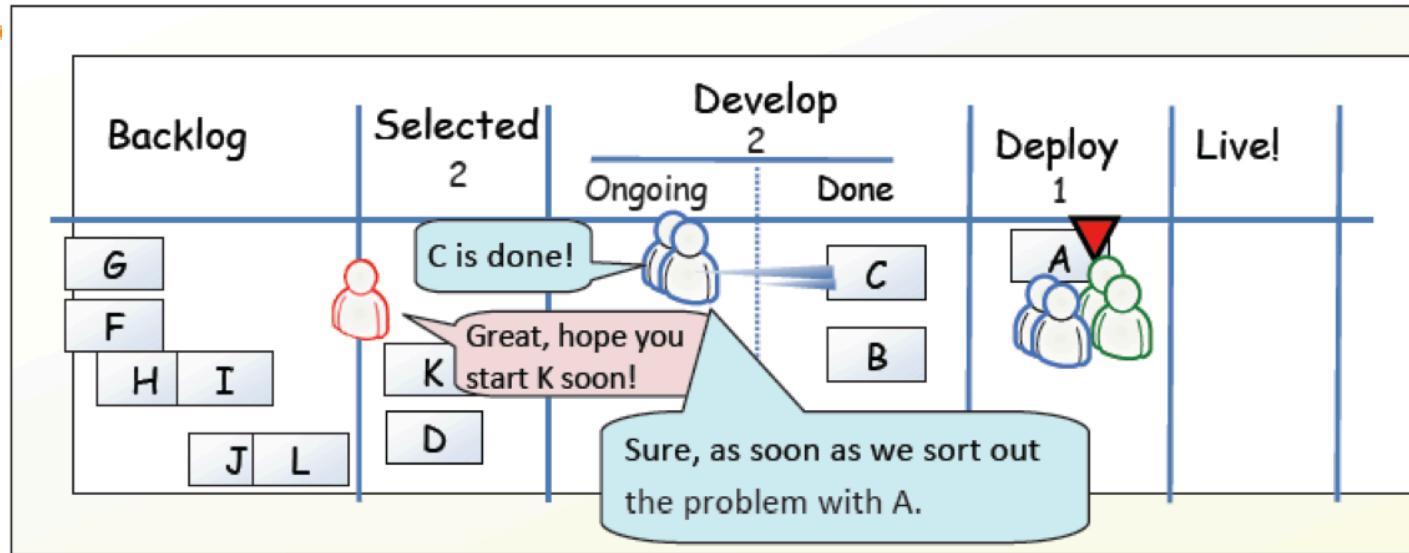
[KS2009]

# Example



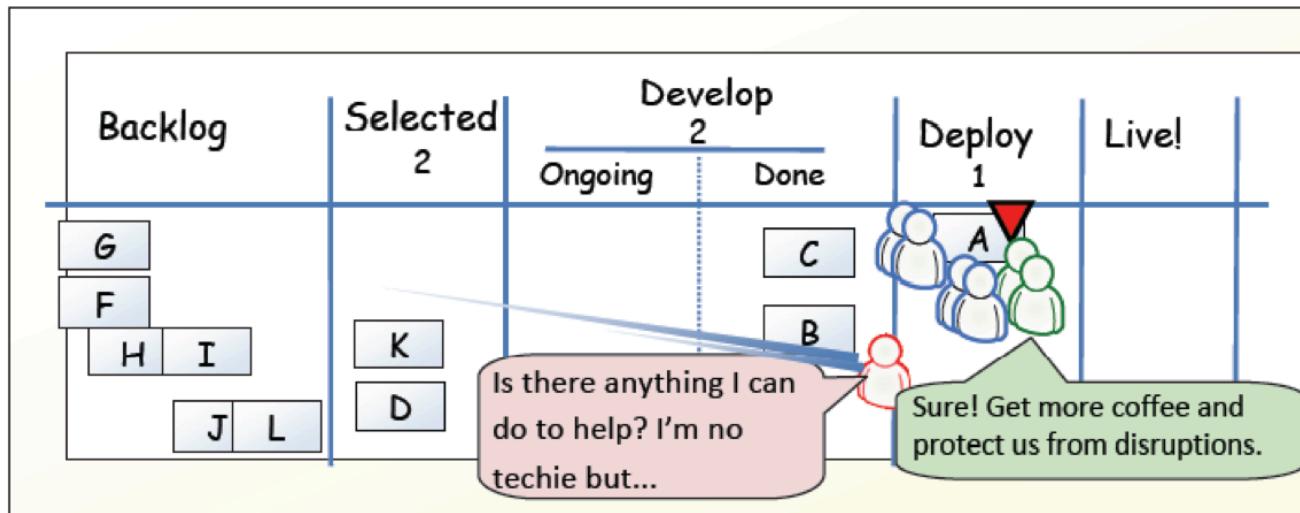
[KS2009]

# Example



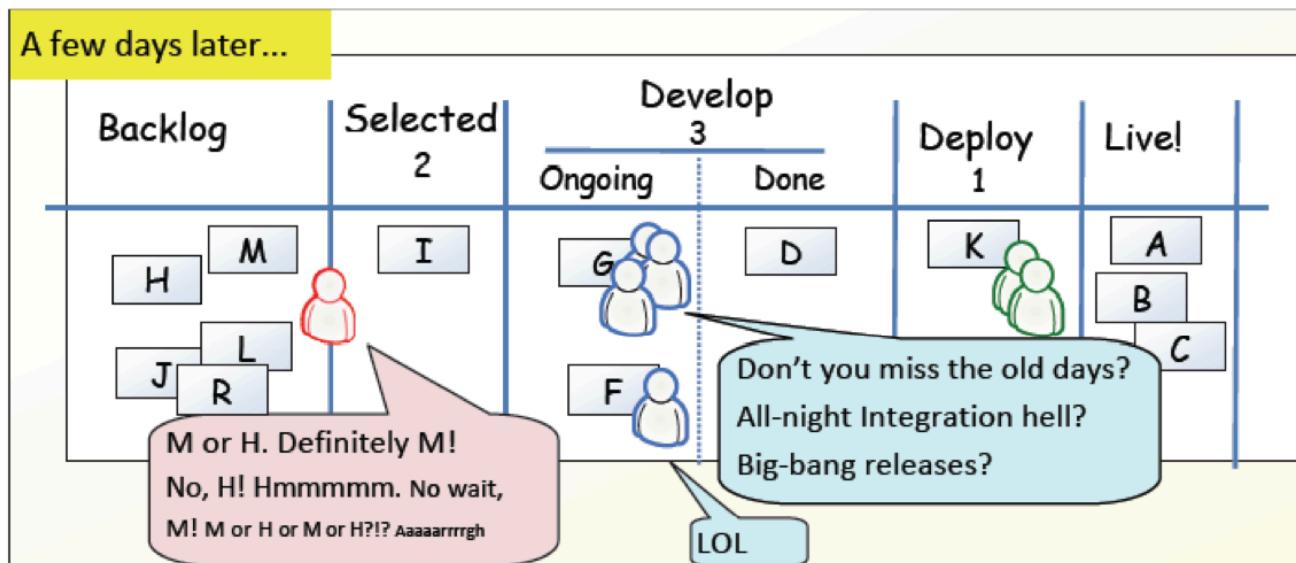
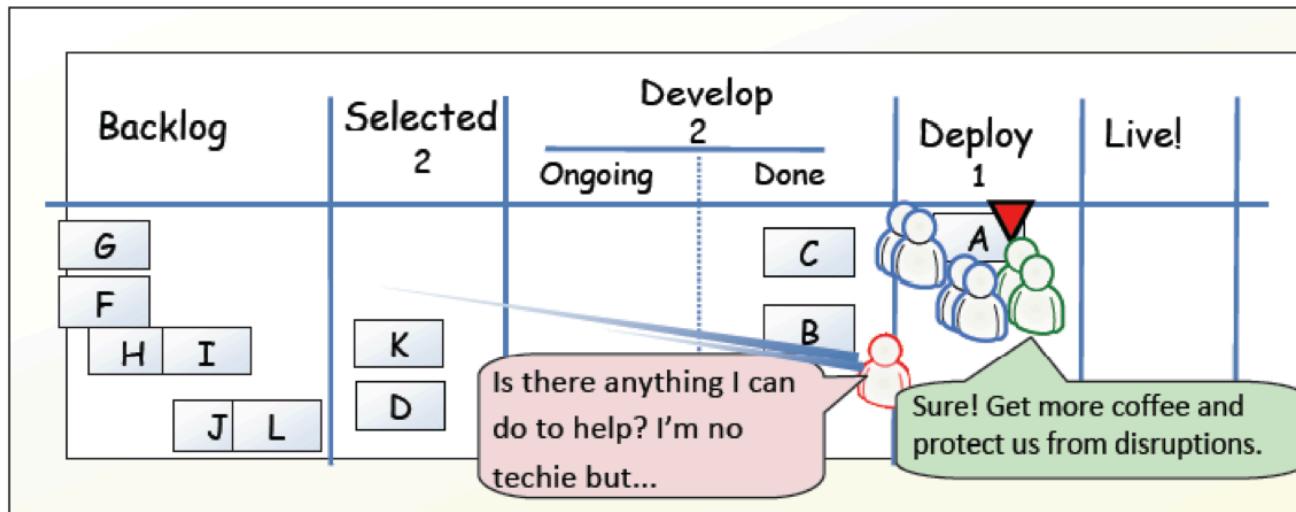
[KS2009]

# Example

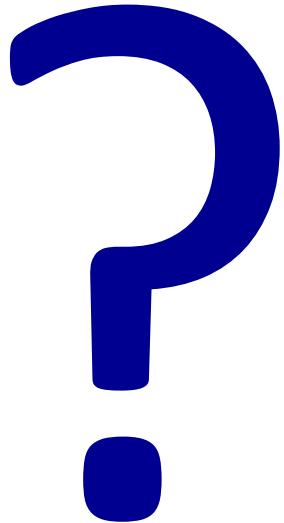


[KS2009]

# Example



[KS2009]



- Compare Kanban with your favorite agile method.
  - What is similar?
  - What is different?
  - Is Kanban agile?

# Scrum vs. Kanban

[KS2009]

Scrum	Kanban
<b>Timeboxed iterations prescribed.</b>	<b>Timeboxed iterations optional.</b> Can have separate cadences for planning, release, and process improvement. Can be event-driven instead of timeboxed.
<b>Team commits</b> to a specific amount of work for this iteration.	<b>Commitment optional.</b>
Uses <b>Velocity</b> as default metric for planning and process improvement.	Uses <b>Lead time</b> as default metric for planning and process improvement.
<b>Cross-functional teams</b> prescribed.	Cross-functional teams optional. <b>Specialist teams allowed.</b>
<b>Items must be broken down</b> so they can be completed within 1 sprint.	No particular item size is prescribed.
<b>Burndown chart prescribed</b>	No particular type of diagram is prescribed

# Scrum vs. Kanban

[KS2009]



Scrum	Kanban
<b>WIP limited indirectly (per sprint)</b>	<b>WIP limited directly (per workflow state)</b>
<b>Estimation prescribed</b>	<b>Estimation optional</b>
<b>Cannot add items to ongoing iteration.</b>	<b>Can add new items whenever capacity is available</b>
<b>A sprint backlog is owned by one specific team</b>	<b>A kanban board may be shared by multiple teams or individuals</b>
<b>Prescribes 3 roles (PO/SM/Team)</b>	<b>Doesn't prescribe any roles</b>
<b>A Scrum board is reset between each sprint</b>	<b>A kanban board is persistent</b>
<b>Prescribes a prioritized product backlog</b>	<b>Prioritization is optional.</b>

# Agile Principles



1. Early and continuous delivery of valuable software
2. Welcome changing requirements, even late
3. Deliver working software frequently
4. Business people and developers must work together
5. Build projects around motivated individuals
6. Face-to-face communication is most effective and efficient
7. Working software is the primary measure of progress
8. Sustainable development
9. Continuous attention to technical excellence and good design
10. Simplicity is essential
11. Self-organizing teams
12. Regular reflection

# Agile Principles – Revised list

(according to [Mey2014])

**Task:** For each of the principles, compare XP, Scrum, and Kanban and discuss differences NOW: Pick two – the others might be a good exercise for exam and report.

## Organizational

1. Put the customer at the center.
2. Let the team self-organize.
3. Work at a sustainable pace.
4. Develop minimal software:
  1. Produce minimal functionality.
  2. Produce only the product requested.
  3. Develop only code and tests.
5. Accept Change

## Technical

1. Develop iteratively:
  1. Produce frequent working iterations.
  2. Freeze requirements during iterations.
2. Treat tests as a key resource:
  1. Do not start any new development until all tests pass.
  2. Test first.
3. Express requirements through scenarios.

6. Reflect regularly and improve continuously!

# References



- [KS2009] Henrik Kniberg and Mattias Skarin: Kanban and Scrum – Making the Most of Both. InfoQ (2009) Available online:<http://infoq.com/minibooks/kanban-scrum-minibook>