Agile
Principles and Practices

Agile Development Processes

Eric Knauss

Announcements

- Challenge lab
- Optional workshop on Git / Github / Android on Thursday, 2016-04-21?
  - Please indicate interest (now, email, or via course representatives)
- Any Roadblocks?

Overview

- What is Software Engineering?
- Repetition: Principles and Practices
  - XP
  - Scrum
  - Kanban (you did watch the video, right?)
- (Lean software development)

Some philosophy

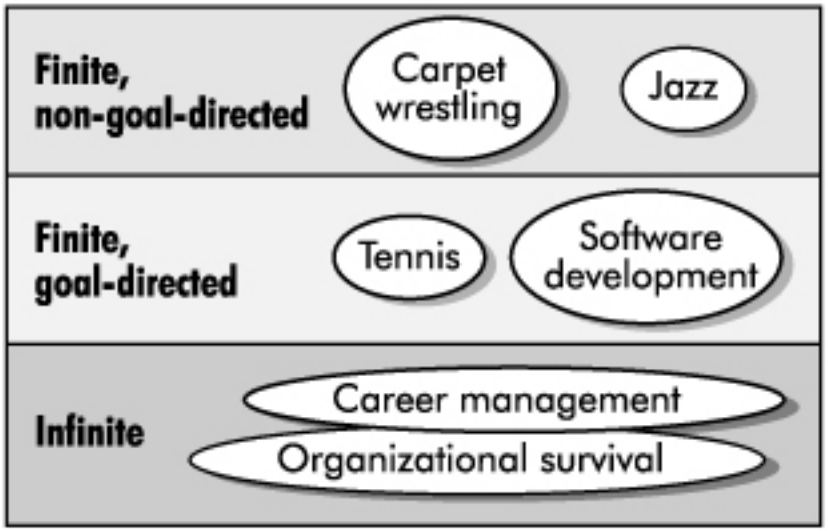
WHAT IS SOFTWARE ENGINEERING?

Some comparisons

- Production?
- Engineering?
- Design?
- Craft?
- Art?
- Theory building?
- Collaborative game of invention and communication?
  - Rock-Climbing
  - Swamp-Game

Software Development as a Game

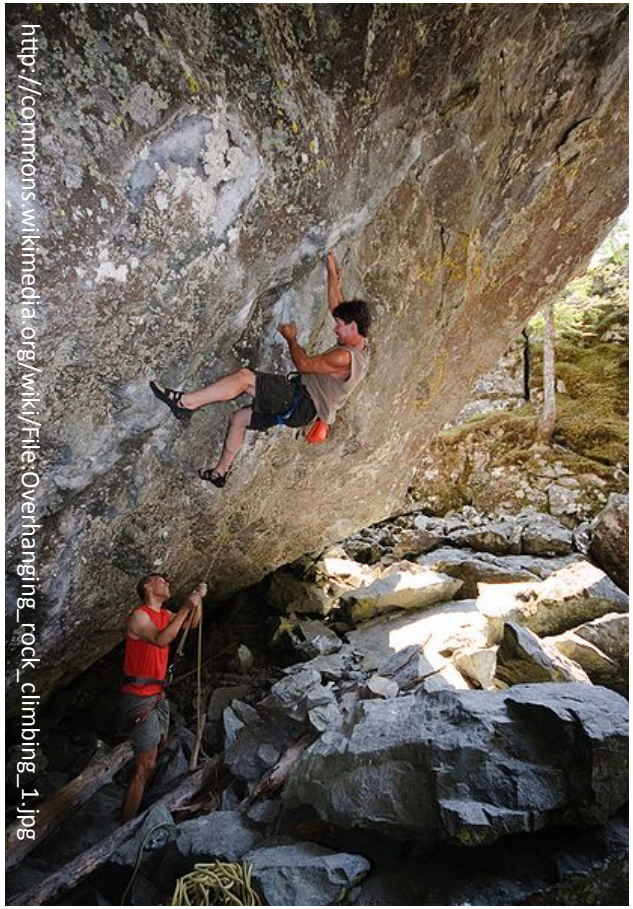
- Zero-sum game?
  - SW-Dev: No, more than one winner
- Positional game?
  - SW-Dev: Some play it like this (comprehensive documentation shows current state of game)
- Competitive / Cooperative game?
  - SW-Dev: Cooperative!!!
- Goal directed?
  - SW-Dev: Definitely
- Finite/ infinite?



Cockburn: Agile software development, the cooperative game, 2nd edition

Rock-climbing vs Software Development

- Cooperative and goal seeking
- Load bearing (limited amount of possible moves)
- Team
- Individuals with talent (some are simply better)
- Skill-sensitive (practice helps to improve)
- Training (you need to hone your skllls)
- Tools (get more critical the more ambitious the project is)
- Resource-limited
- Plan
- Improvised
- Fun
- Challenging
- Dangerous



Cockburn: Agile software development, the cooperative game, 2nd edition

Software Development as Manufacturing

	Manufacturing Mass Production	Waterfall Model
Hidden Cost	• Transportation • Managing inventory and storage • Capital costs of inventory	• Handoffs • Lots of open items; can lead to overwhelming the workforce • Cost of training people to build software
Risks	• Building things you don't need because production goes on after needs go away • Inventory becoming obsolete Huge latency if an error occurs	• Building things you don't need because requirements aren't clear or customers change their minds • Knowledge degrading quickly If a line is discontinued, all WIP wasted • Errors in requirements discovered late in the process • Errors in completed code discovered late in testing

Lean-Agile Software Development: Achieving Enterprise Agility, by Alan Shalloway, Guy Beaver, and Jim Trott. Chapter 1.

Agile Principles and Practices

REPETITION

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

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
- Refactoring
- Pair programming
- Collective ownership
- Continuous integration
- 40h week
- On-site customer
- Coding standards

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

Kanban principles

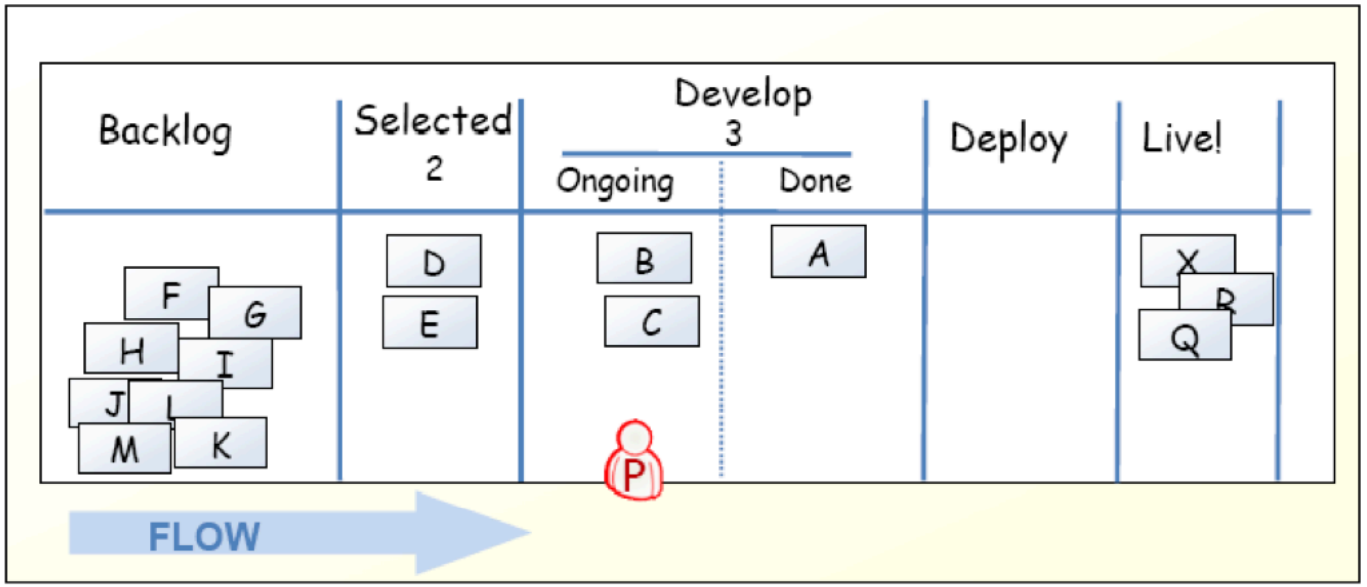
- Start with what you do now
- Agree to pursue incremental, evolutionary change
- Respect the current process, roles, responsibilities and titles
- Leadership at all levels

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

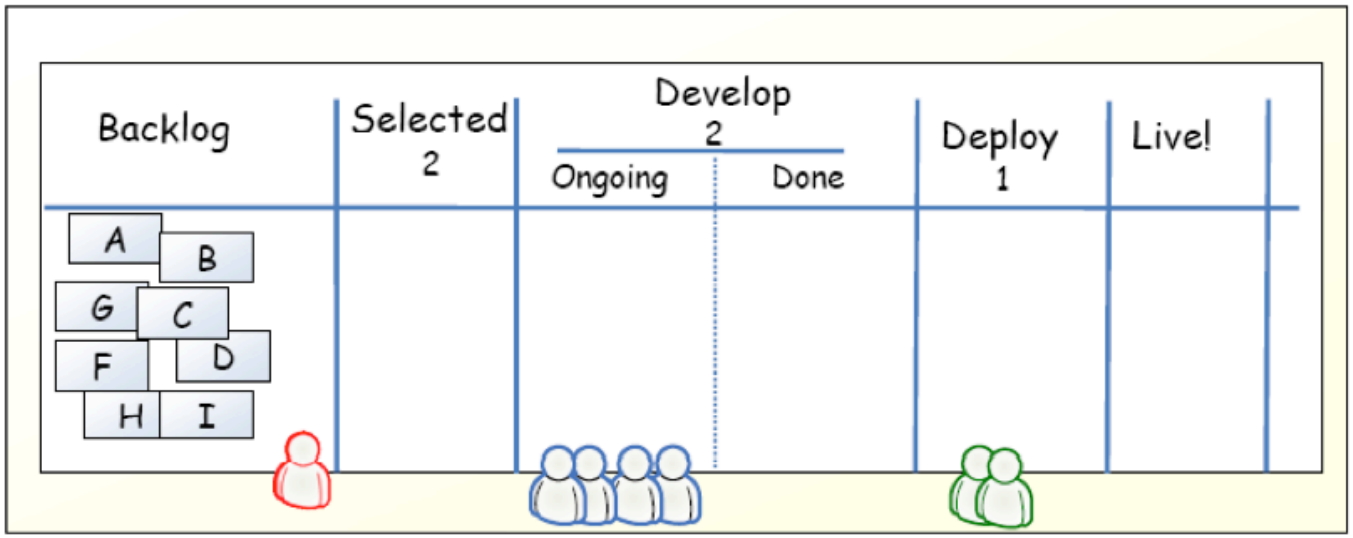
Idle Members

- 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

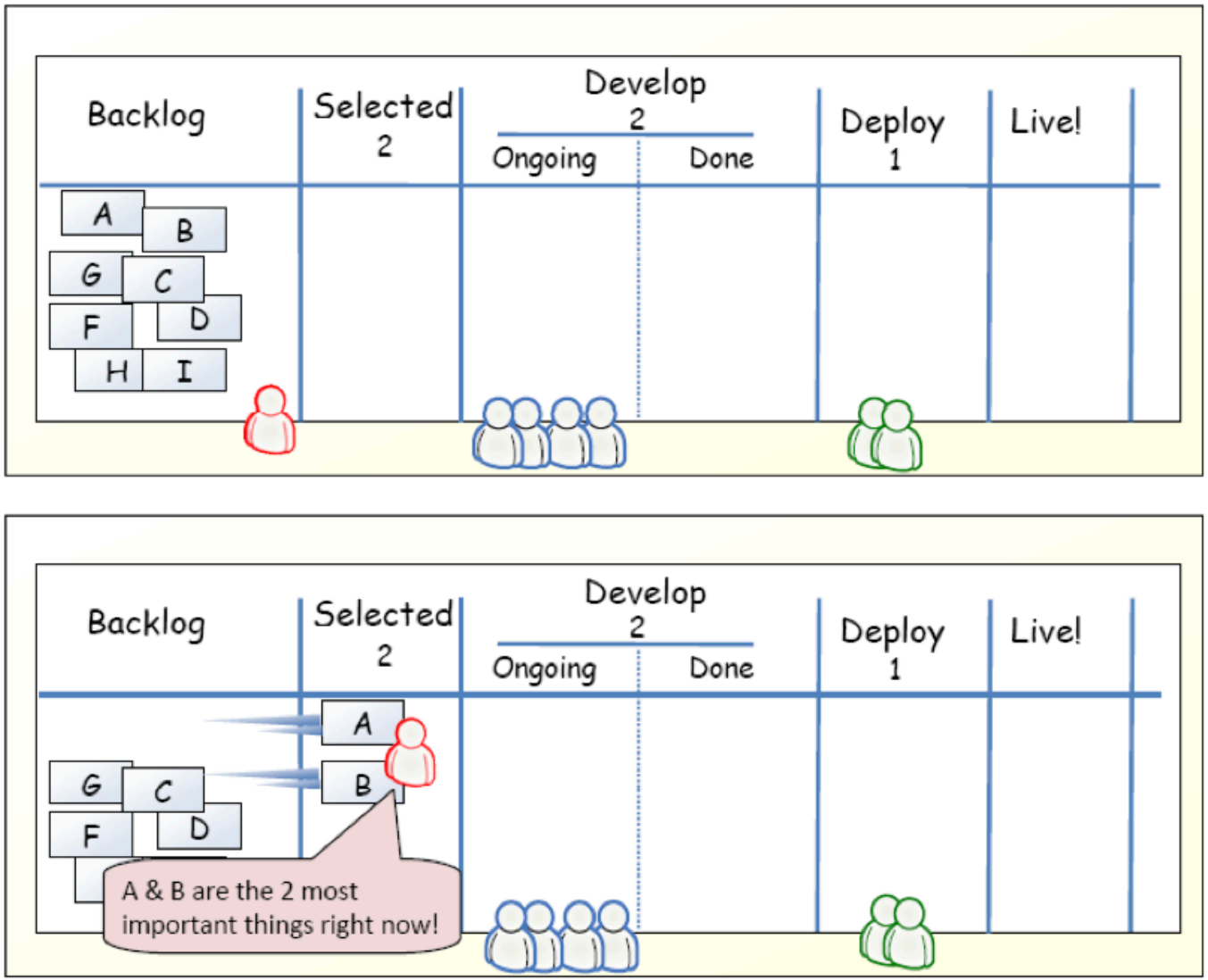
- Stories in progress (SIP)
- When story enters stories queue set entry date (ED)
- When story enters first process step set start processing date (SPD)
- When story is done set finish date (FD)
- Cycle time (CT) = FD – SPD
- Waiting time (WT) = SPD – ED
- Throughput (T) = SIP / CT

Example



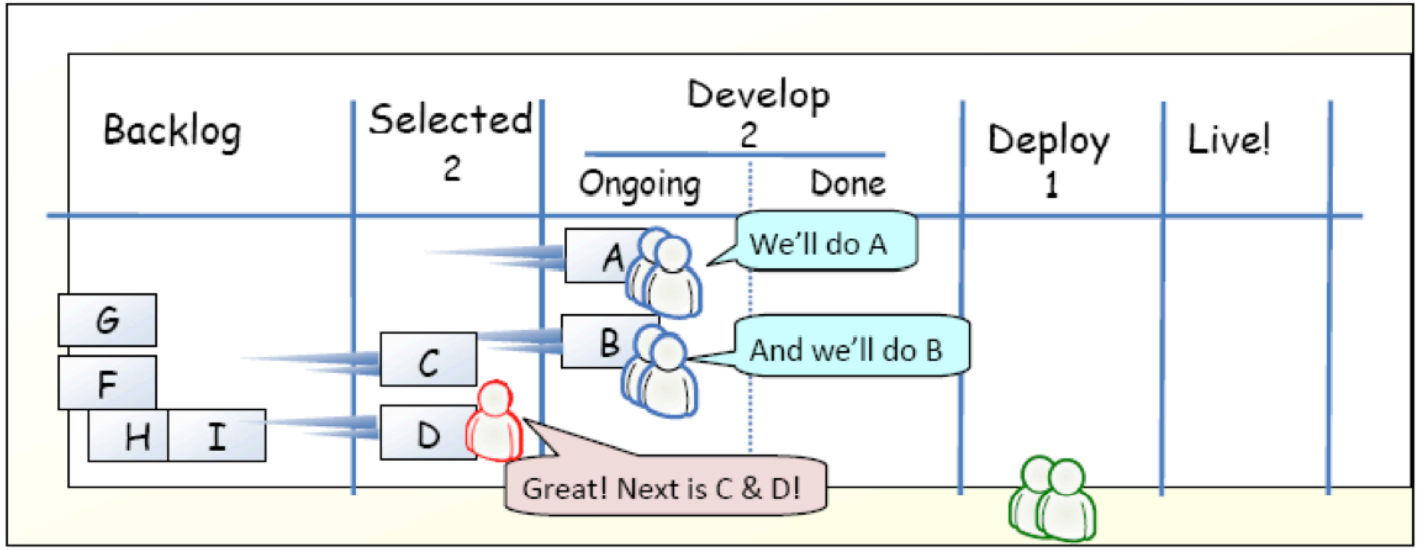
[KS2009]

Example



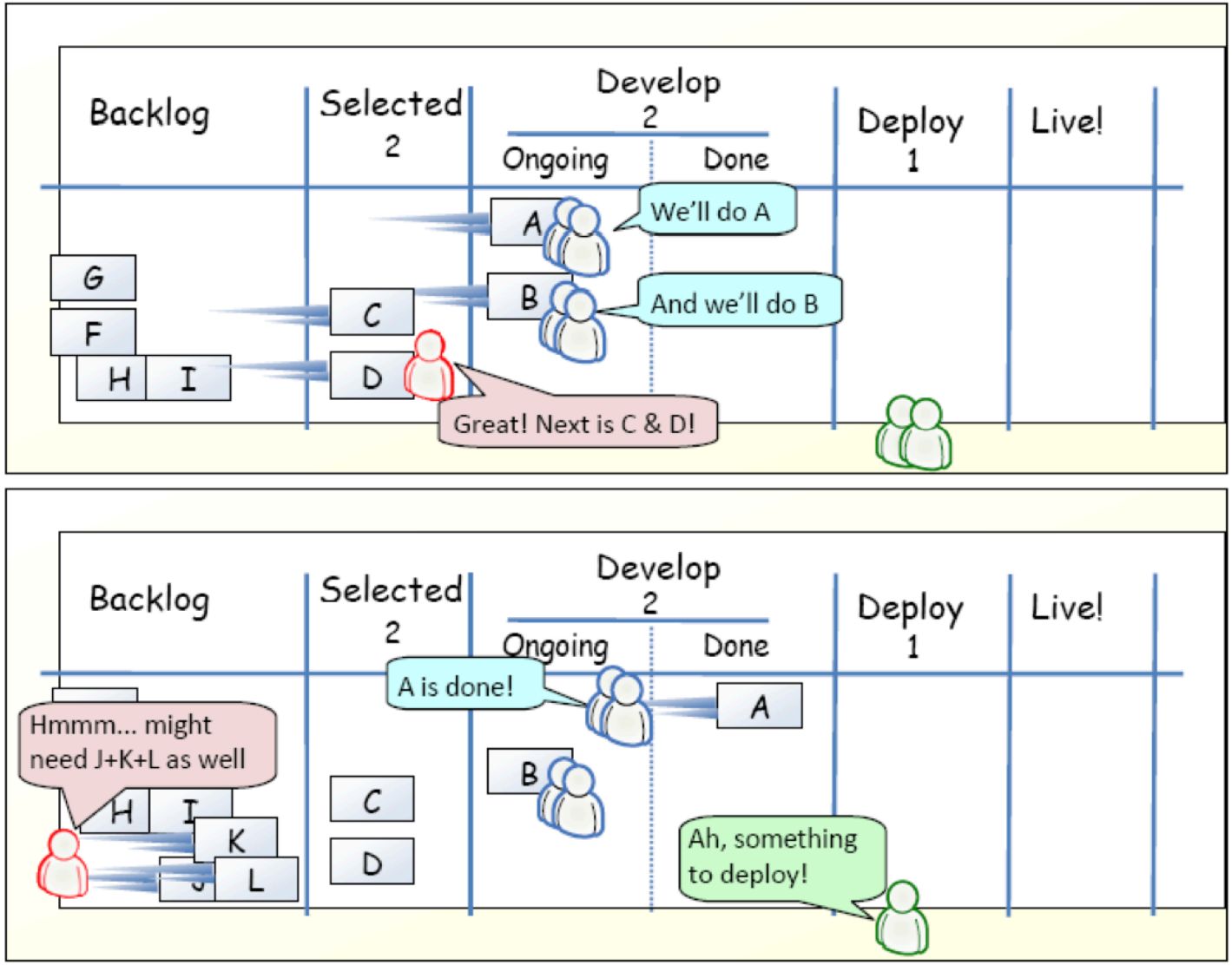
[KS2009]

Example



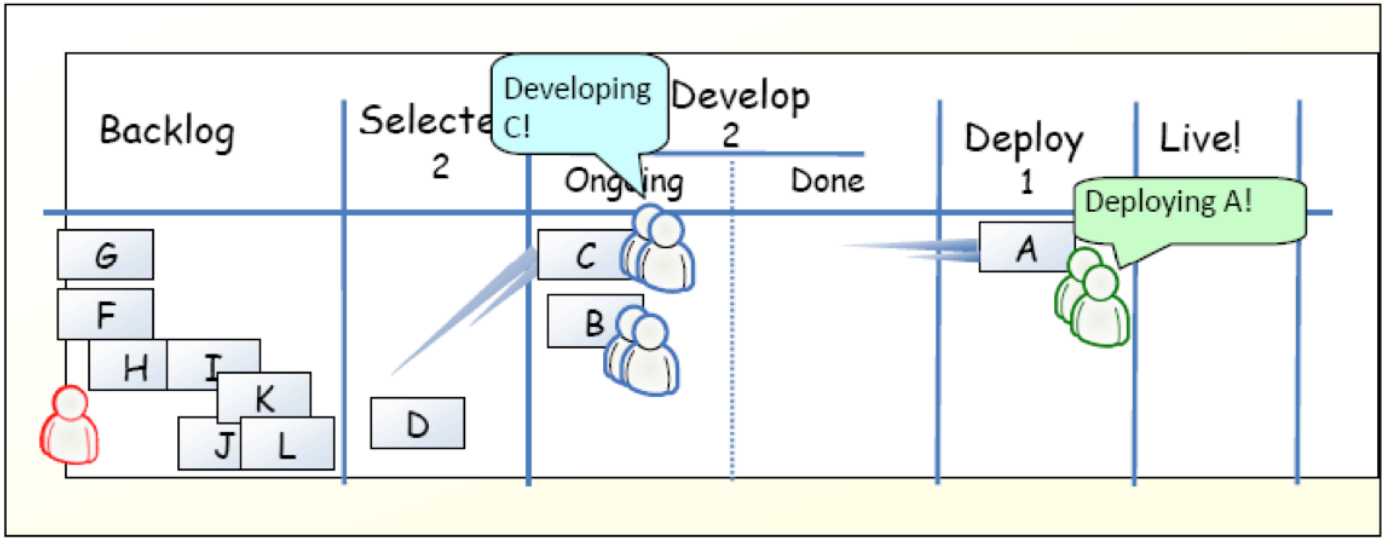
[KS2009]

Example



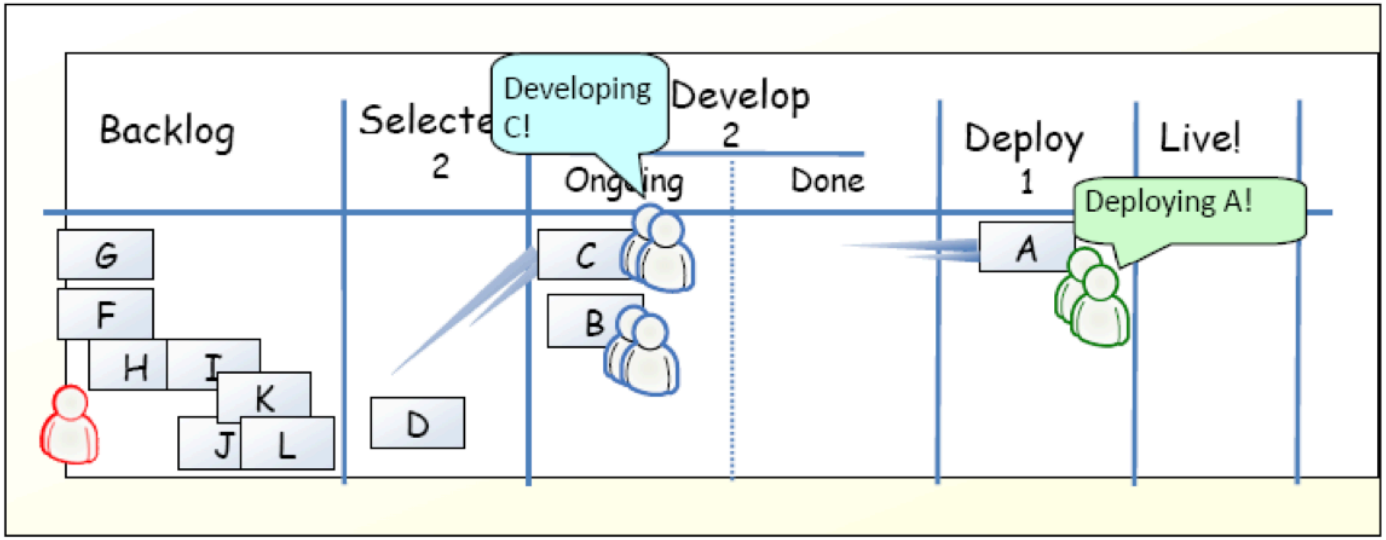
[KS2009]

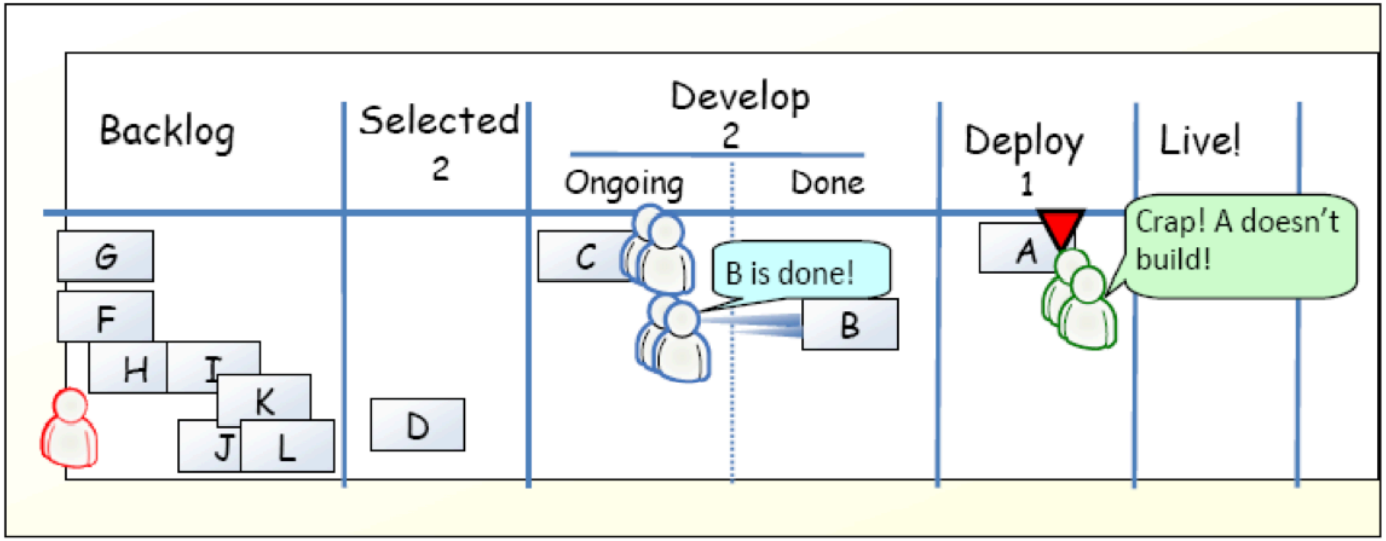
Example



[KS2009]

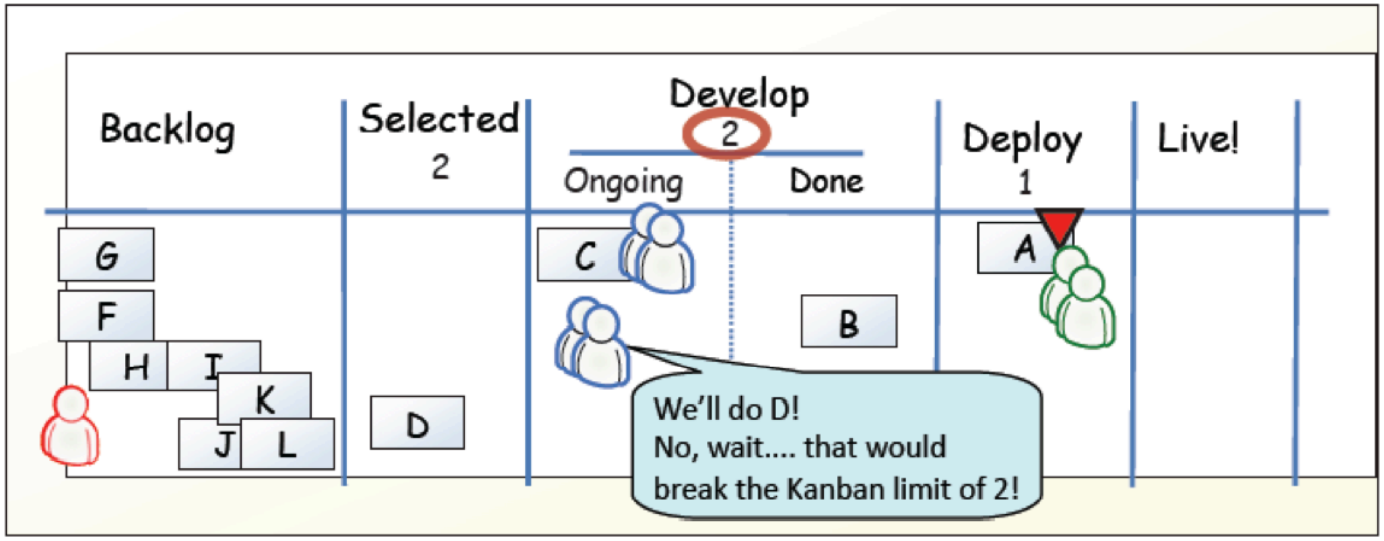
Example





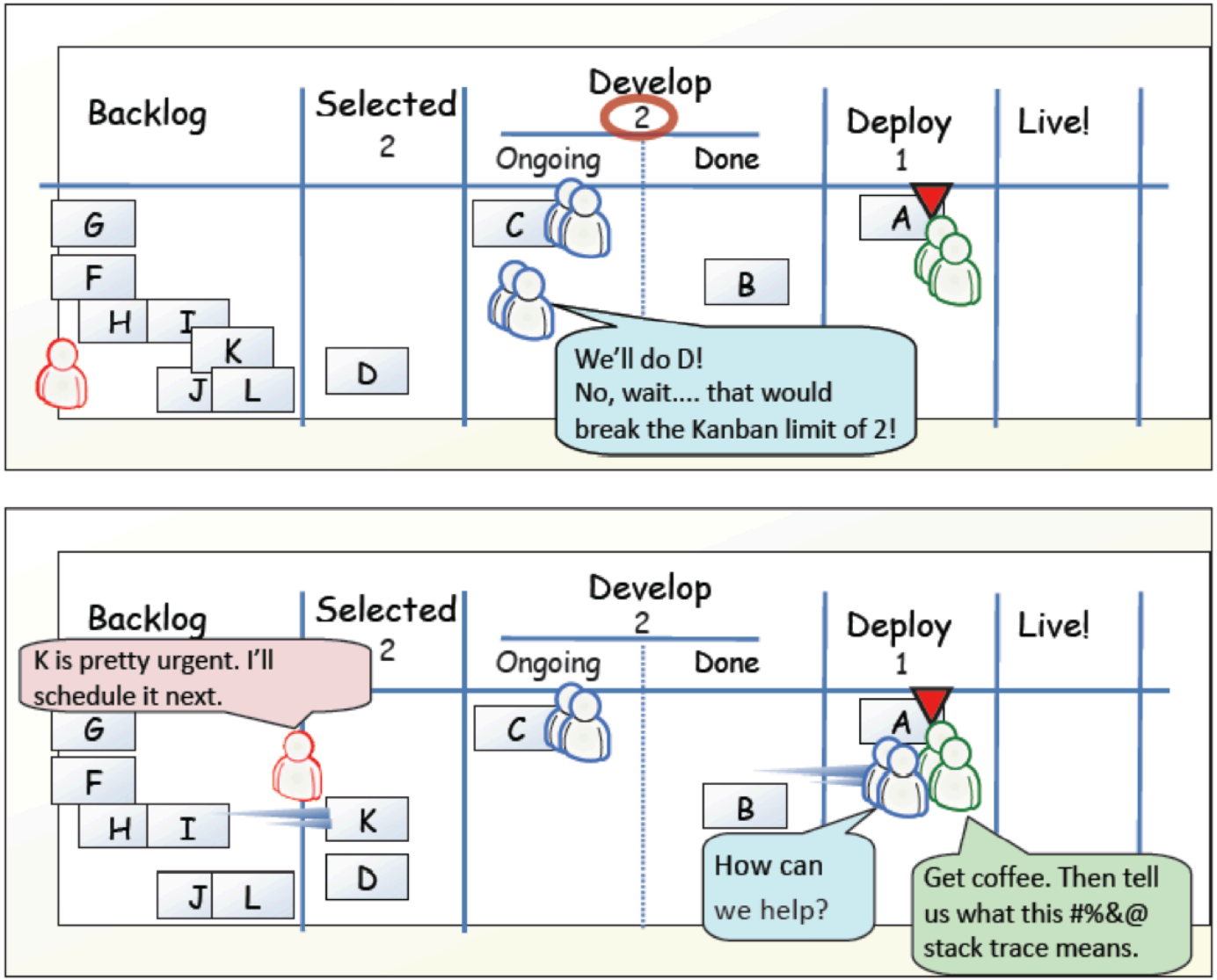
[KS2009]

Example



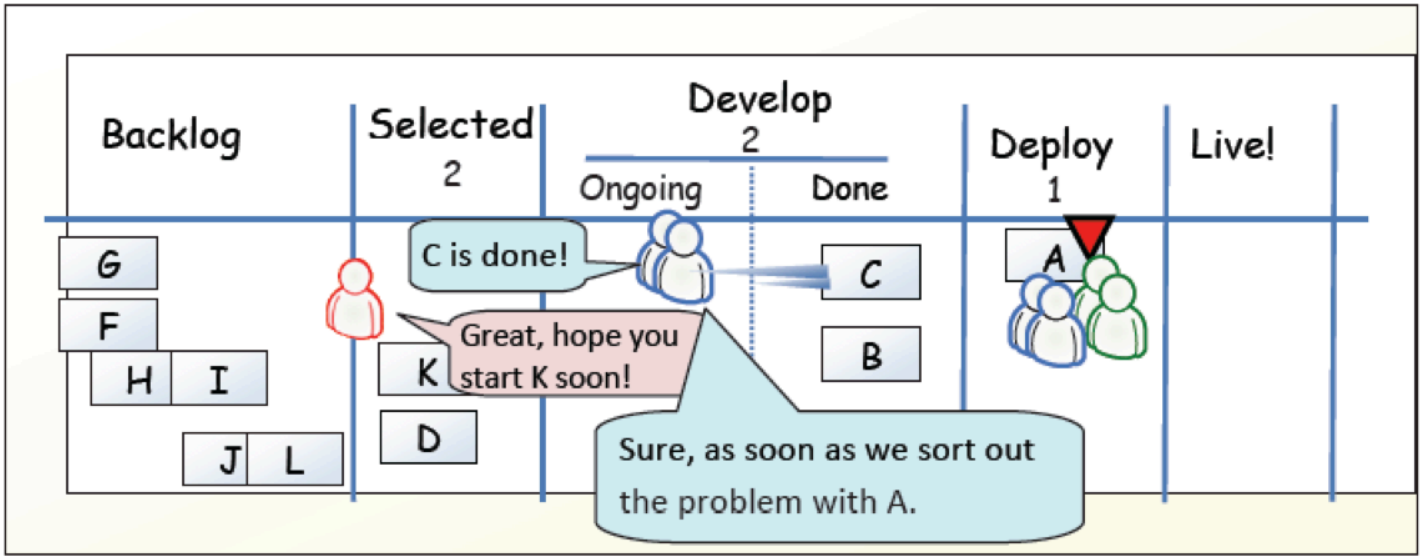
[KS2009]

Example



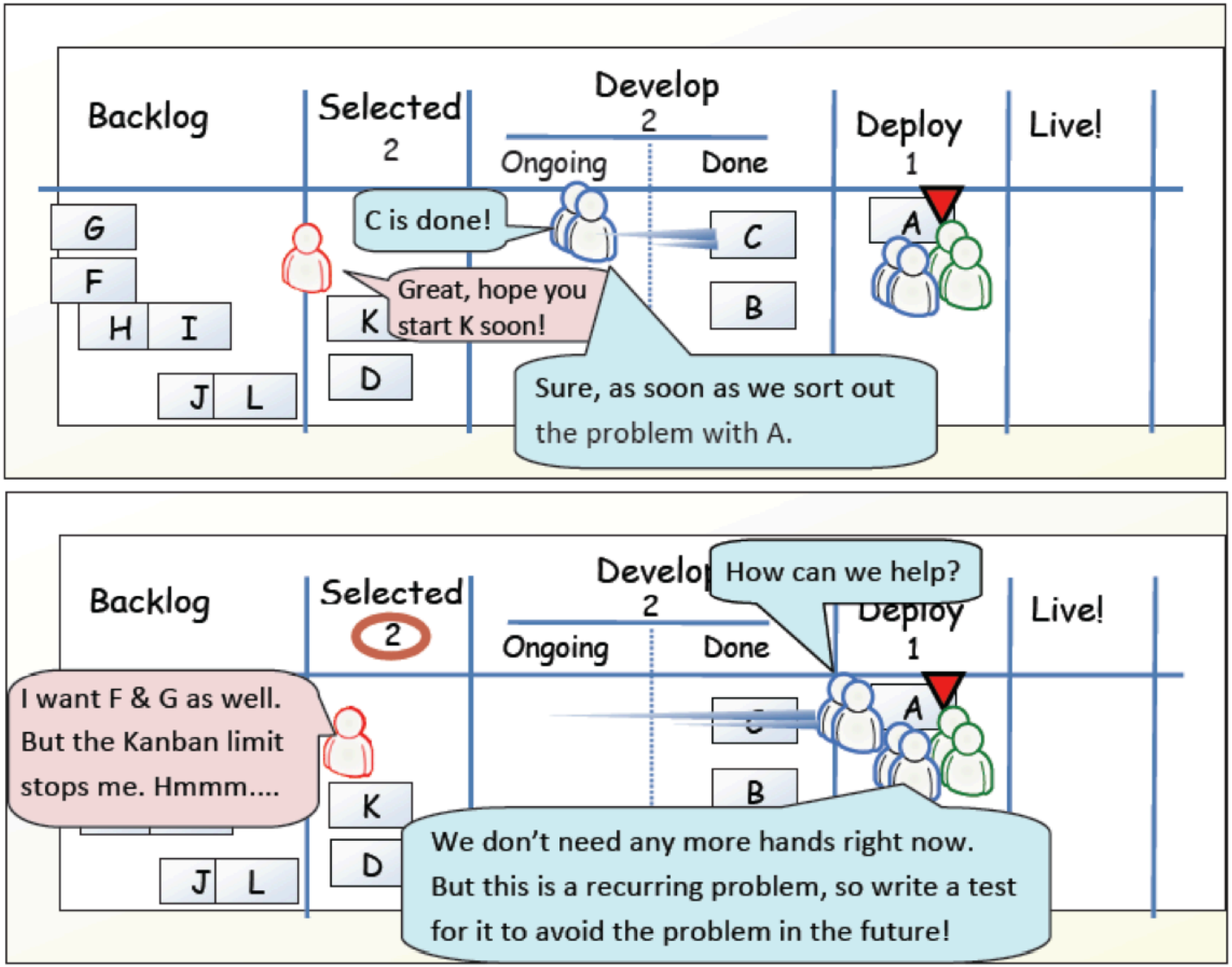
[KS2009]

Example



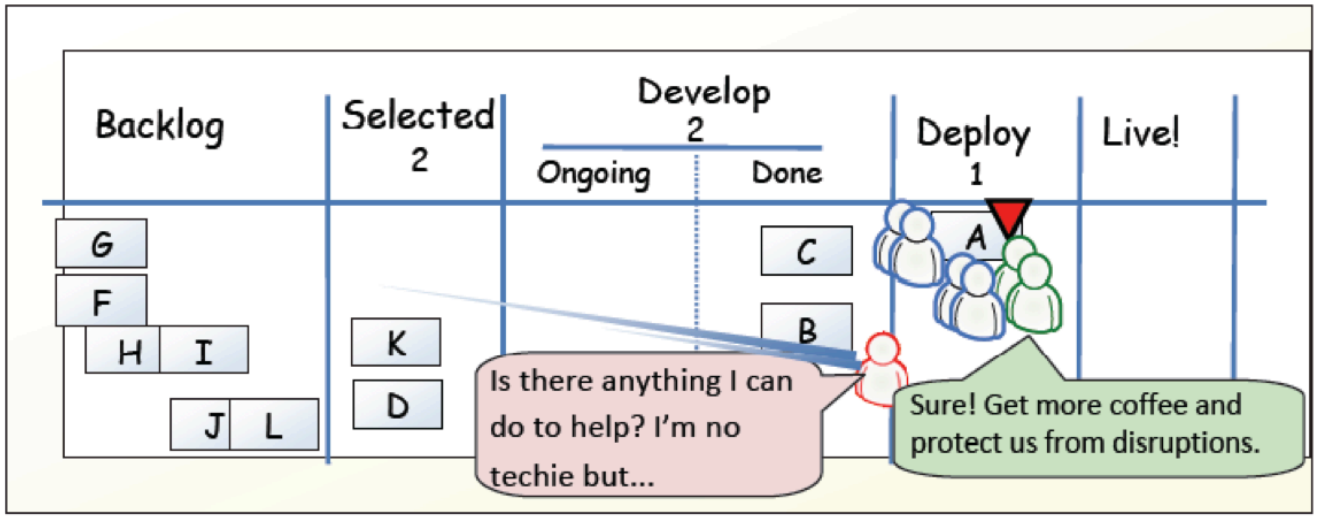
[KS2009]

Example



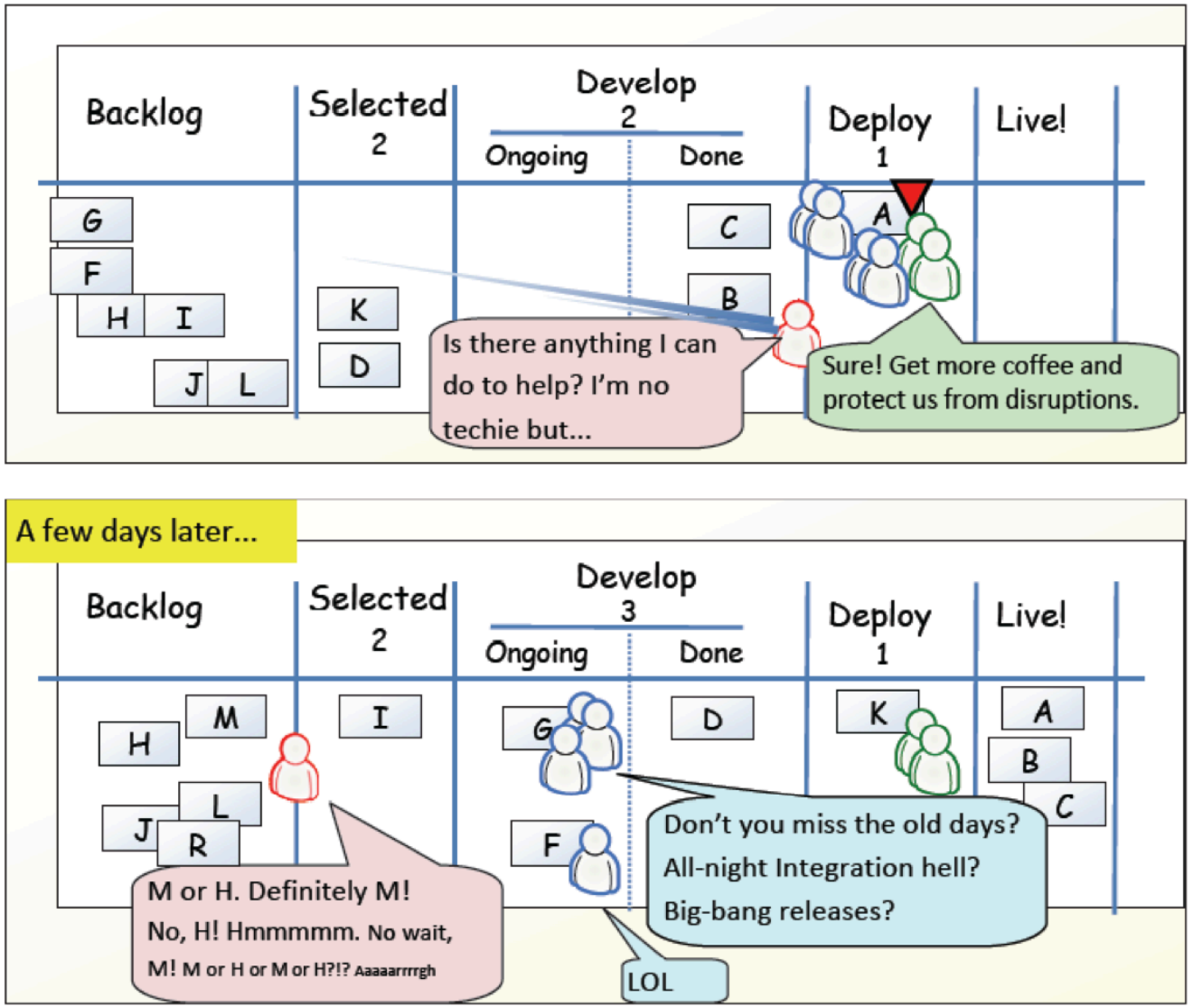
[KS2009]

Example

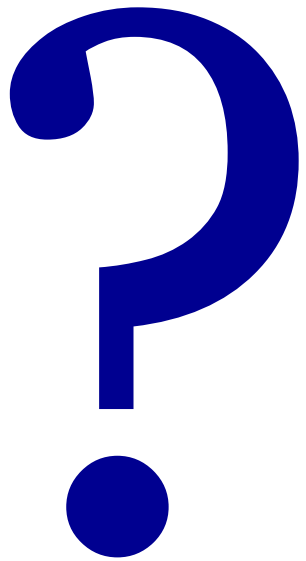


[KS2009]

Example



[KS2009]



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

Scrum vs. Kanban

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

[KS2009]

Scrum vs. Kanban

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

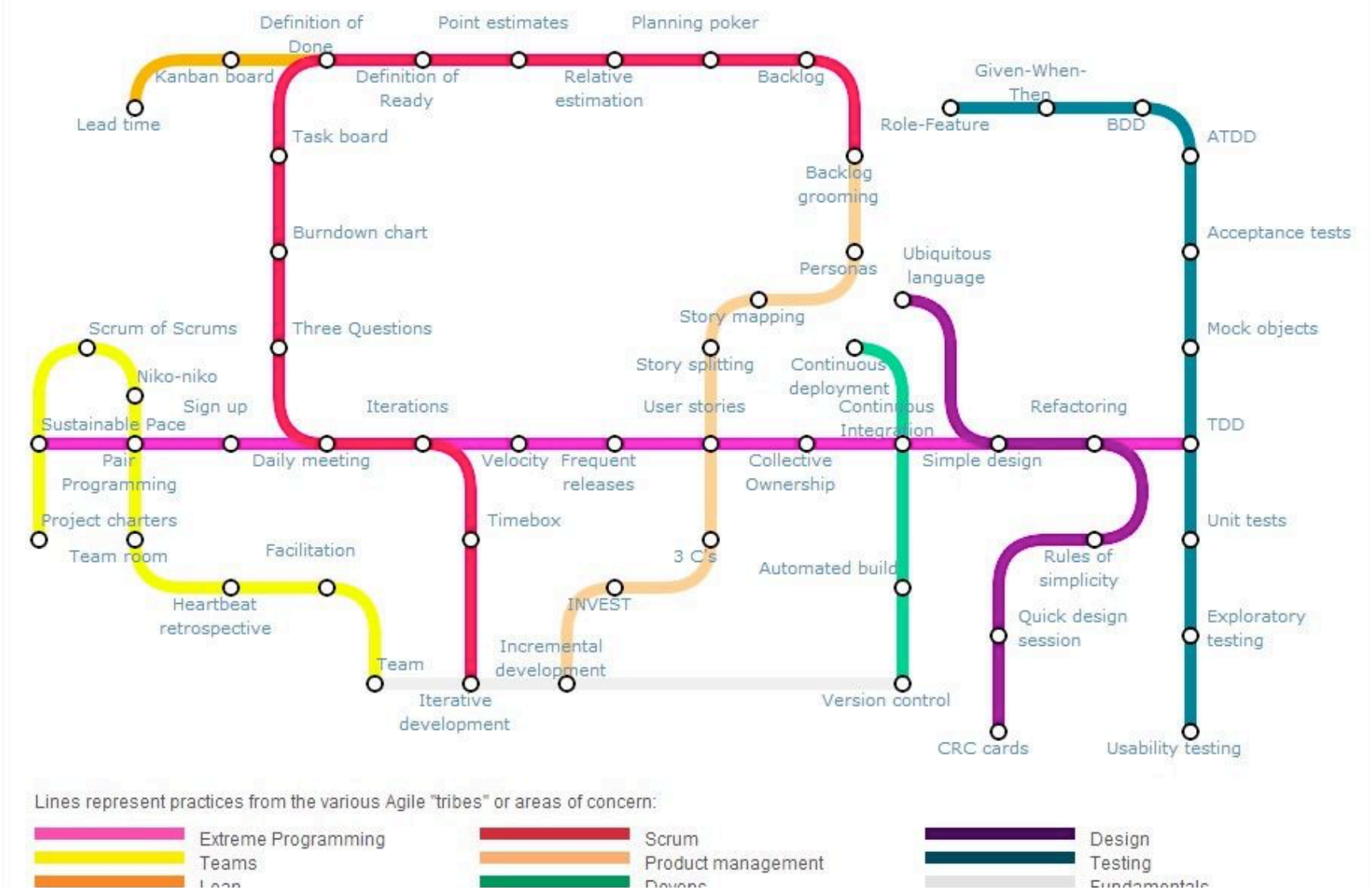
[KS2009]

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

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.

Agile Practice Map



https://techblog.betclicgroup.com/wp-content/uploads/2013/12/agileSubwayMap2.jpg

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

Post-Lecture Additions

- Placeholder for Scrum practices