Kanban ~~vs~~ and Scrum

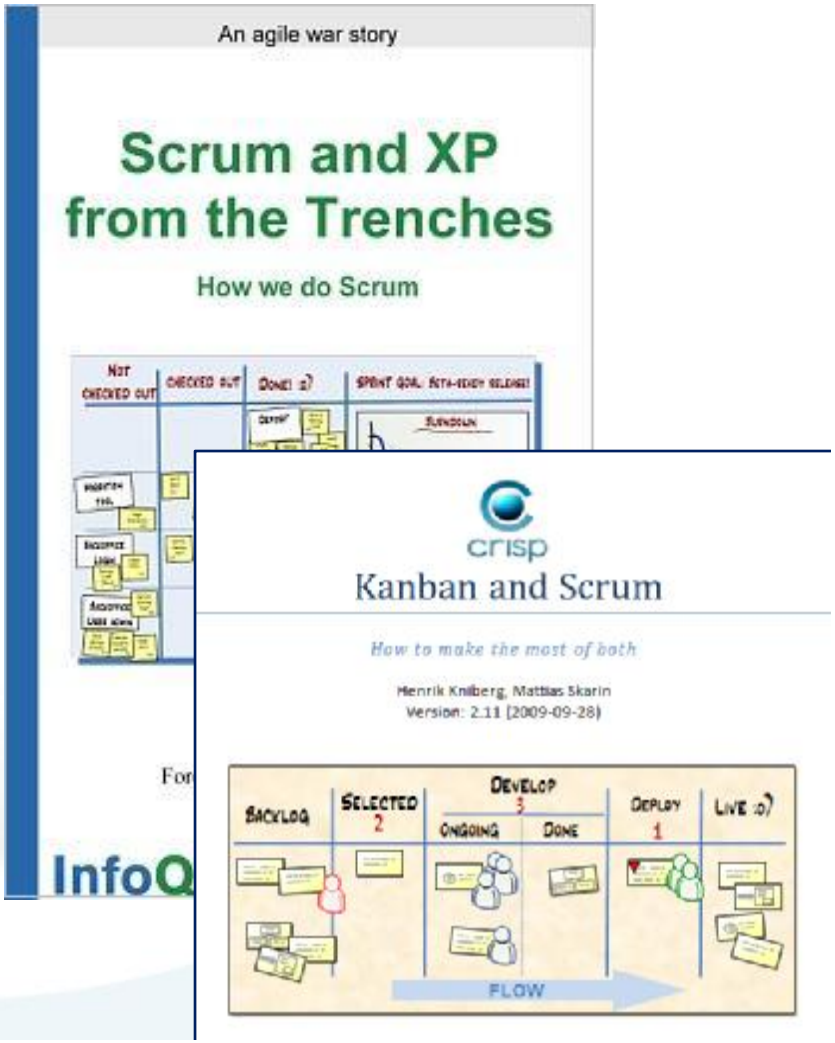
Making the most of both

JAOO, Aarhus
Oct 6, 2009

Henrik Kniberg
Agile/Lean coach @ Crisp, Stockholm

Agile Alliance Board of directors

CERTIFIED Scrum Trainer



henrik.kniberg@crisp.se
+46 70 4925284

crisp

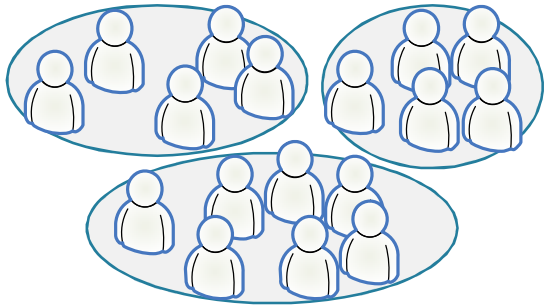
Purpose of this presentation

To clarify Kanban and Scrum by comparing them

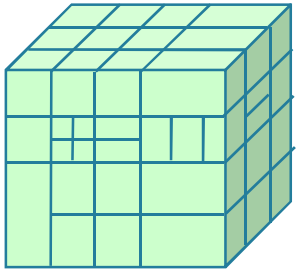
...so you can figure out
how these may come to use
in your context.

Scrum in a nutshell

Split your organization

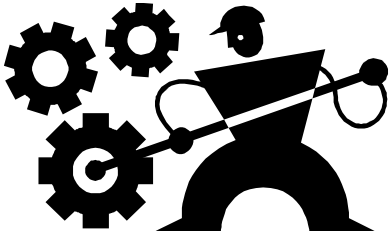


Split your product

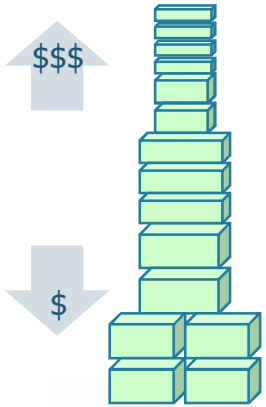


~~Large group~~ spending ~~a long time~~ building a ~~huge thing~~
Small team spending a little time building a small thing
... but integrating regularly to see the whole

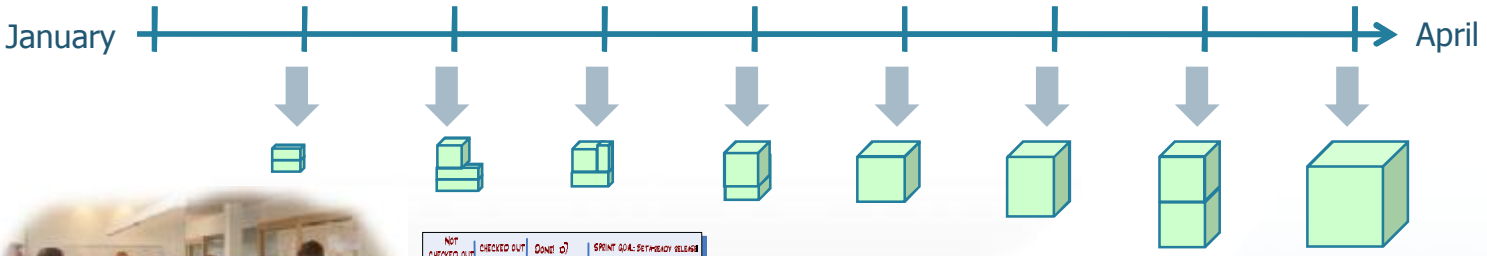
Optimize process

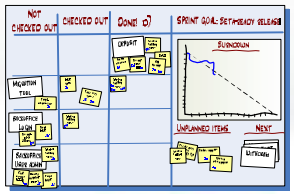


Optimize business value



Split time





The bottom line

If you achieve these you can ignore the rest of the checklist. Your process is fine.

- [ ] Delivering **working, tested software** every 4 weeks or less
- [ ] Delivering what the **business needs** most
- [ ] Process is **continuously improving**

- [ ] Clearly defined **product owner (PO)**
  - [ ] PO is **empowered** to prioritize
  - [ ] PO has **knowledge** to prioritize
  - [ ] PO has direct **contact with team**
  - [ ] PO has direct **contact with stakeholders**
  - [ ] PO speaks **with one voice** (in case PO is a team)
- [ ] Team has a **sprint backlog**
  - [ ] Highly **visible**
  - [ ] **Updated** daily
  - [ ] Owned exclusively by the **team**
- [ ] **Daily Scrum** happens
  - [ ] Whole team participates
  - [ ] Problems & impediments are surfaced
- [ ] **Demo** happens after every sprint
  - [ ] Shows **working, tested software**
  - [ ] **Feedback** received from stakeholders & PO
- [ ] Have **Definition of Done (DoD)**
  - [ ] DoD **achievable** within each iteration
  - [ ] Team **respects** DoD

Core Scrum

These are central to Scrum. Without these you probably shouldn't call it Scrum.

- [ ] **Retrospective** happens after every sprint
  - [ ] Results in concrete improvement **proposals**
  - [ ] Some proposals actually get **implemented**
  - [ ] **Whole team + PO** participates
- [ ] PO has a **product backlog (PBL)**
  - [ ] Top items are **prioritized** by business value
  - [ ] Top items are **estimated**
  - [ ] **Estimates written by the team**
  - [ ] Top items in **PBL small enough to fit** in a sprint
  - [ ] PO understands **purpose** of all backlog items
- [ ] Have **sprint planning meetings**
  - [ ] **PO participates**
  - [ ] PO brings **up-to-date PBL**
  - [ ] **Whole team** participates
  - [ ] Results in a **sprint plan**
  - [ ] Whole team believes plan is **achievable**
  - [ ] PO **satisfied with priorities**
- [ ] Timeboxed **iterations**
  - [ ] Iteration length **4 weeks or less**
  - [ ] Always **end on time**
  - [ ] Team **not disrupted or controlled** by outsiders
  - [ ] Team usually **delivers what they committed to**
- [ ] Team members **sit together**
  - [ ] **Max 9 people** per team

the unofficial Scrum Checklist

crisp

Henrik Kniberg

Recommended but not always necessary

Most of these will usually be needed, but not always all of them. Experiment!

- [ ] Team **has all skills** needed to bring backlog items to Done
- [ ] Team members **not locked into specific roles**
- [ ] Iterations that are **doomed to fail** are terminated early
- [ ] PO has **product vision** that is in sync with PBL
- [ ] PBL and product vision is **highly visible**
- [ ] Everyone on the **team participates in estimating**
- [ ] **PO available** when team is estimating
- [ ] Estimate **relative size** (story points) rather than time
- [ ] Whole team knows top 1-3 **impediments**
  - [ ] **SM has strategy** for how to fix top impediment
  - [ ] **SM focusing** on removing impediments
  - [ ] **Escalated to management** when team can't solve
- [ ] Team has a **Scrum Master (SM)**
  - [ ] SM **sits with the team**
- [ ] PBL items **are broken into tasks** within a sprint
  - [ ] Sprint tasks are **estimated**
  - [ ] Estimates for ongoing tasks are **updated daily**
- [ ] **Velocity** is measured
  - [ ] All items in sprint plan have an **estimate**
  - [ ] PO uses velocity for **release planning**
  - [ ] Velocity only includes items that are **Done**
- [ ] Team has a **sprint burndown chart**
  - [ ] Highly **visible**
  - [ ] **Updated** daily
- [ ] **Daily Scrum** is every day, same time & place
  - [ ] **PO participates** at least a few times per week
  - [ ] Max **15 minutes**
  - [ ] Each team member **knows what the others are doing**

Scaling

These are pretty fundamental to any Scrum scaling effort.

- [ ] You have a **Chief Product Owner** (if many POs)
- [ ] Dependent teams do **Scrum of Scrums**
- [ ] Dependent teams **integrate within each sprint**

Positive indicators

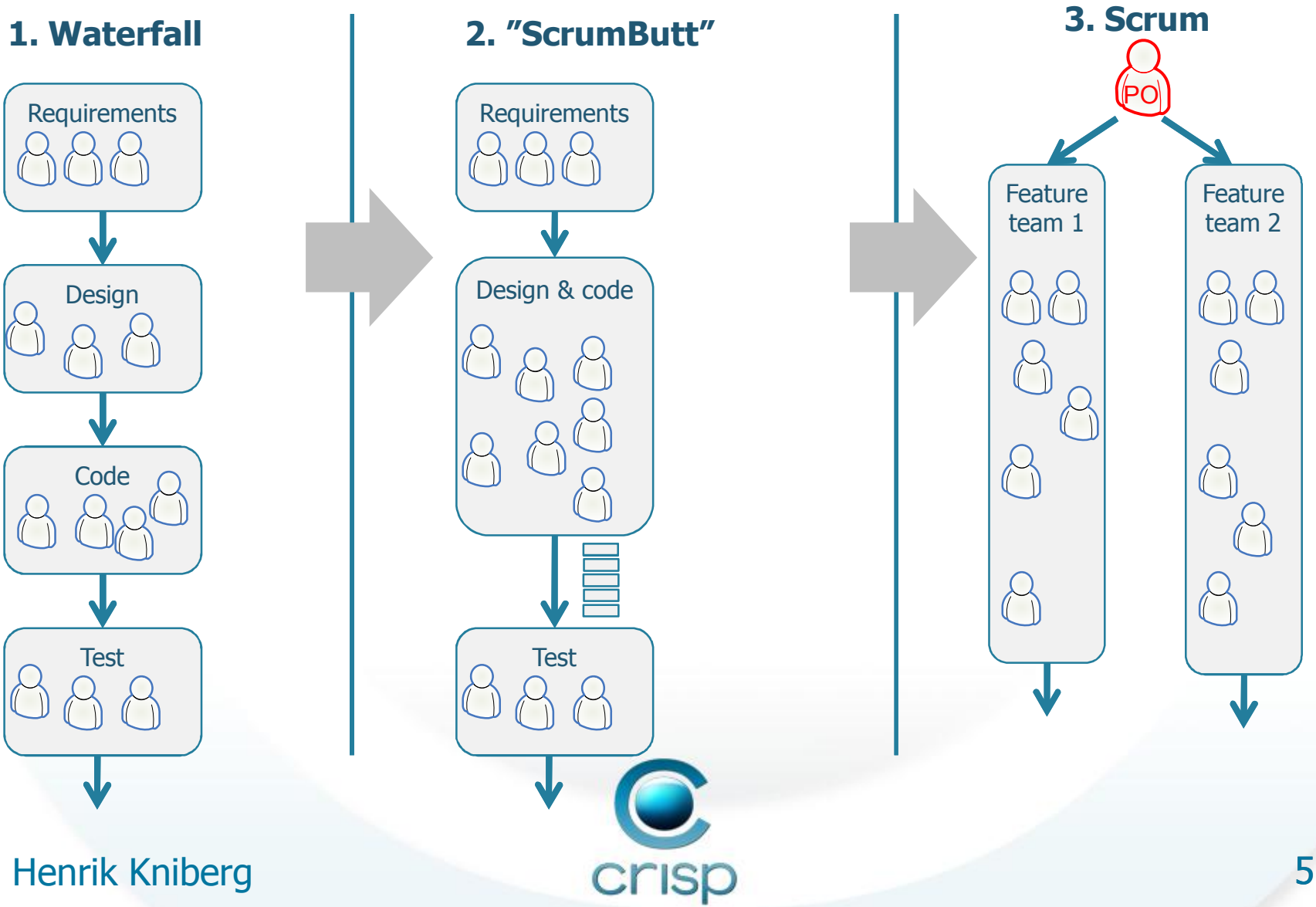
Leading indicators of a good Scrum implementation.

- [ ] **Having fun!** High energy level.
- [ ] **Overtime work is rare** and happens voluntarily
- [ ] Discussing, criticizing, and **experimenting** with the process

PO = Product owner **SM** = Scrum Master **PBL** = Product Backlog **DoD** = Definition of Done

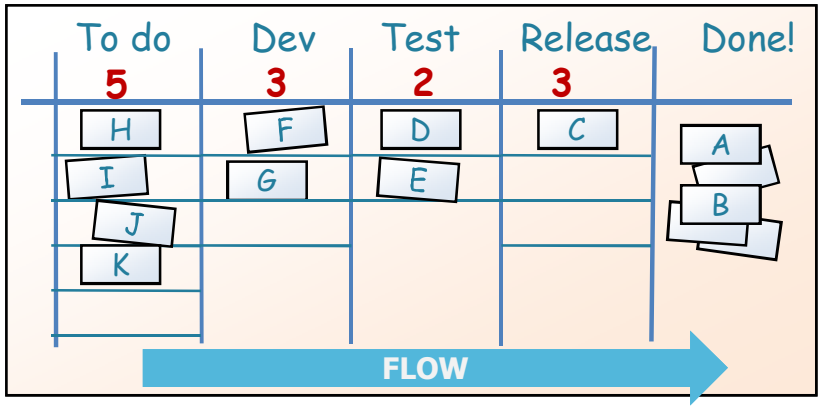
http://www.crisp.se/scrum/checklist | Version 2.1 (2009-08-17)

Typical waterfall => Scrum evolution



Kanban in a nutshell

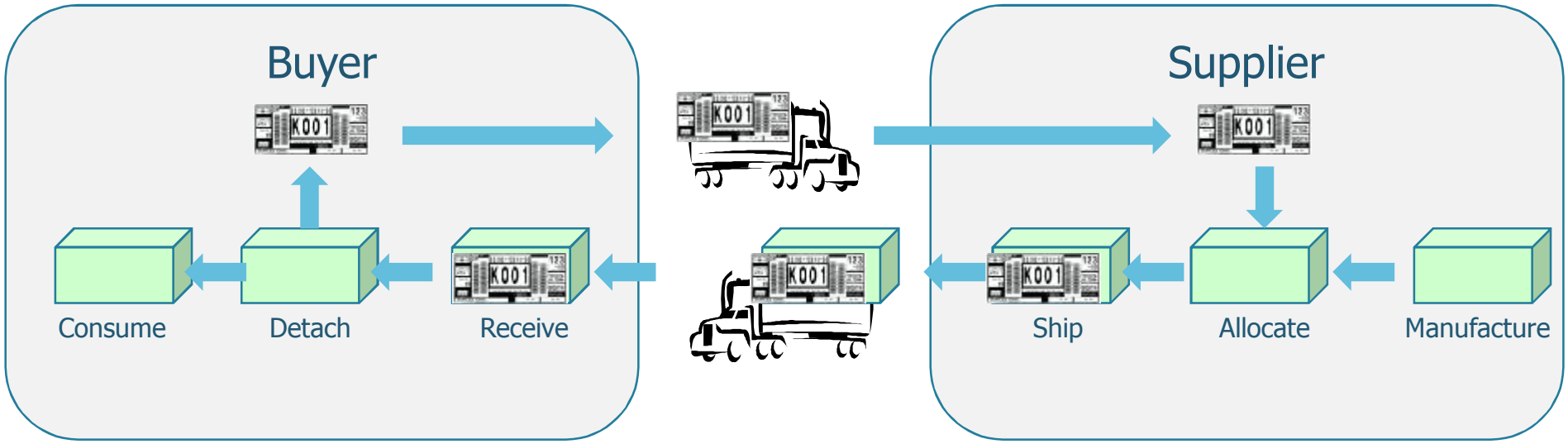
- **Visualize the workflow**
- **Limit WIP (work in progress)**
- **Measure & optimize flow**



Useful starting point for more info:
http://www.limitedwipsociety.org

Roots of Kanban (Toyota)

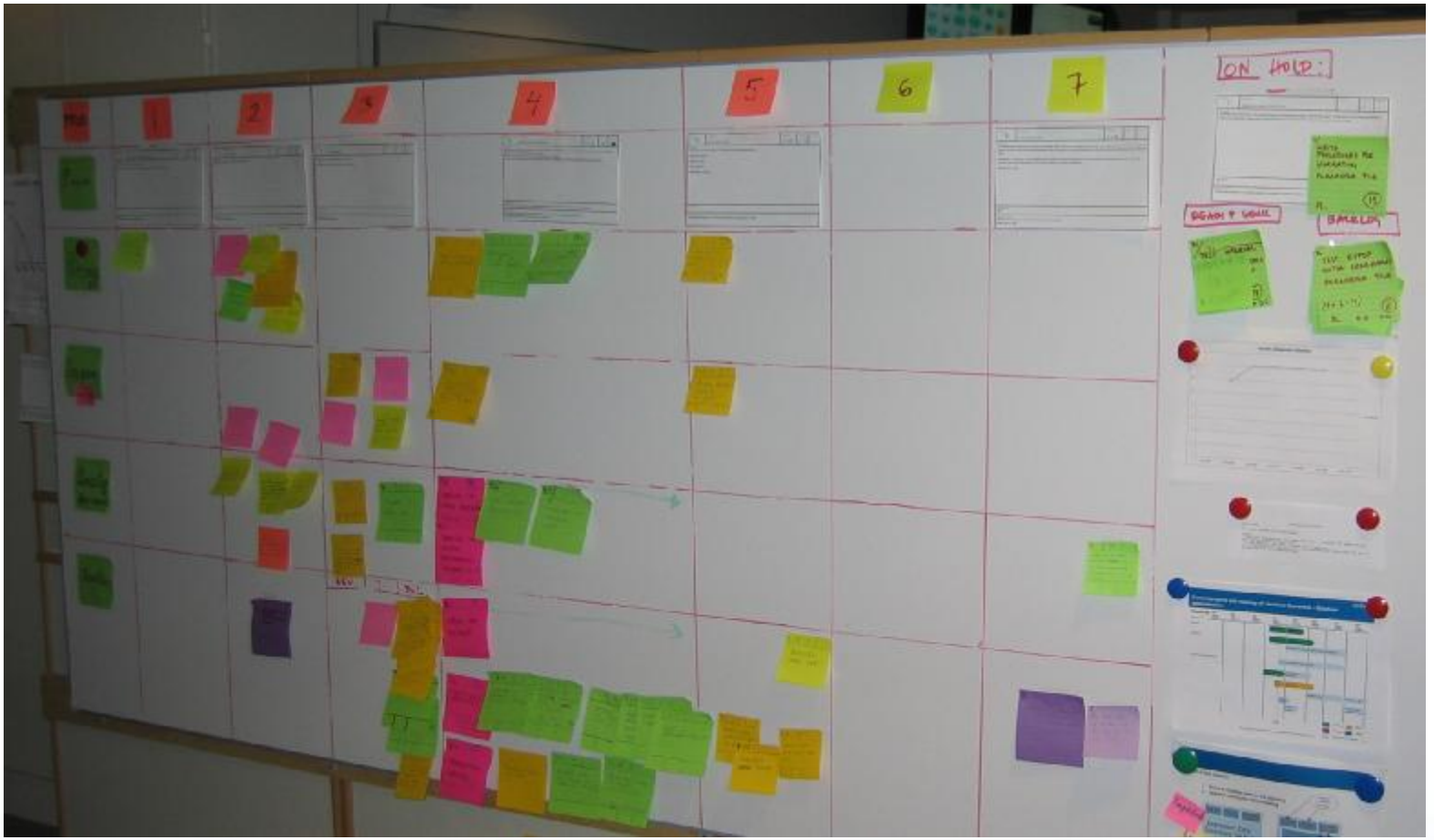
看板
Kan Ban
"Visual Card"



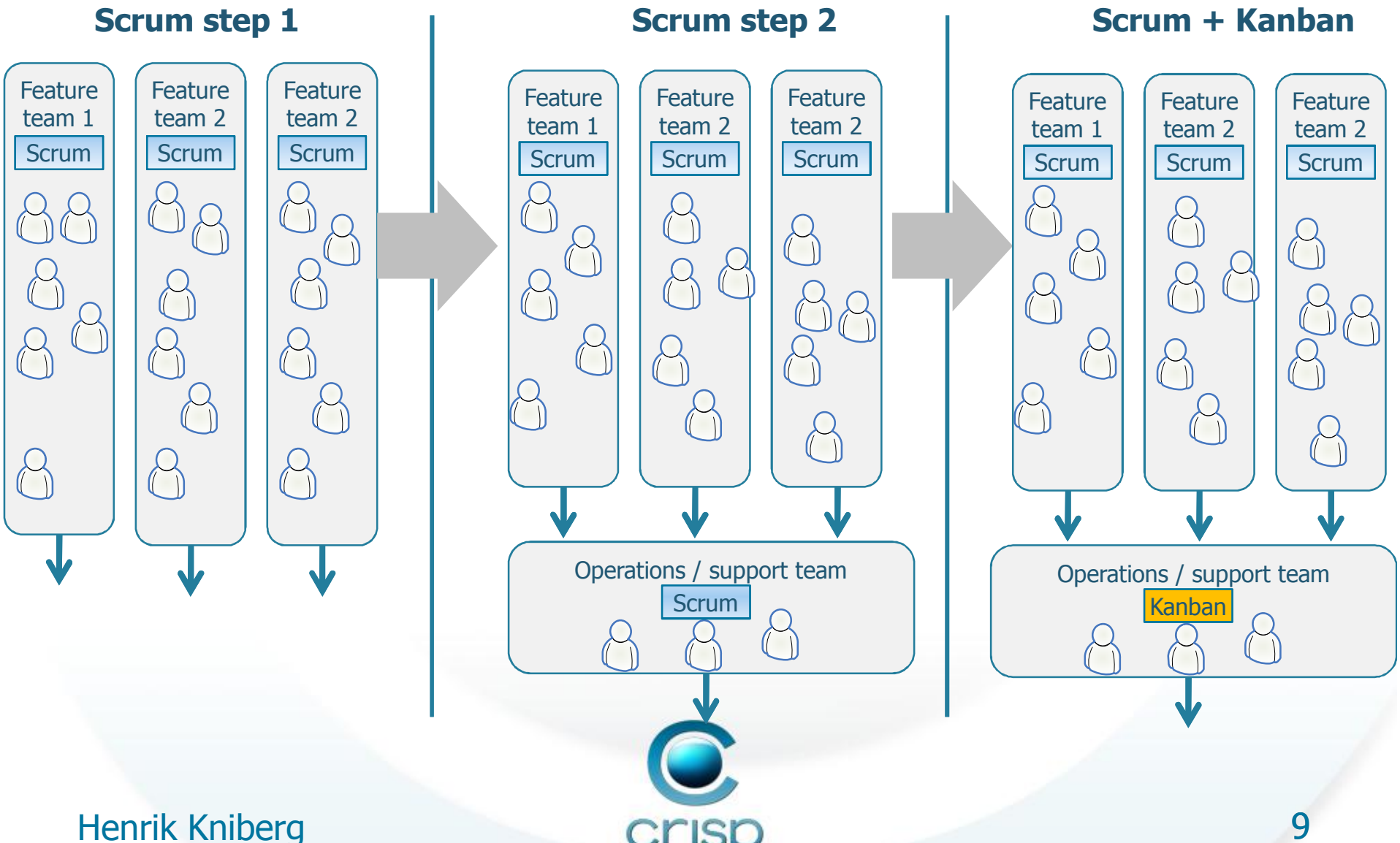
The two pillars of the Toyota production system are just-in-time and automation with a human touch, or autonomation.
The tool used to operate the system is kanban.

Taiichi Ohno
Father of the Toyota Production System

Kanban in software development



Typical Scrum => Kanban evolution



Can we compare Kanban and Scrum?

Should we?

Tool

"anything used as a means of accomplishing a task or purpose."
- dictionary.com

Thinking tools

a.k.a. "mindsets" or "philosophies"

Lean Agile Systems Thinking Theory of Constraints

Toolkits

a.k.a. "frameworks"

Scrum RUP XP Kanban

Physical tools

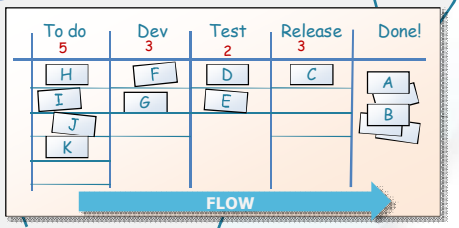
Process tools

a.k.a. "organizational patterns"

Product Owner role

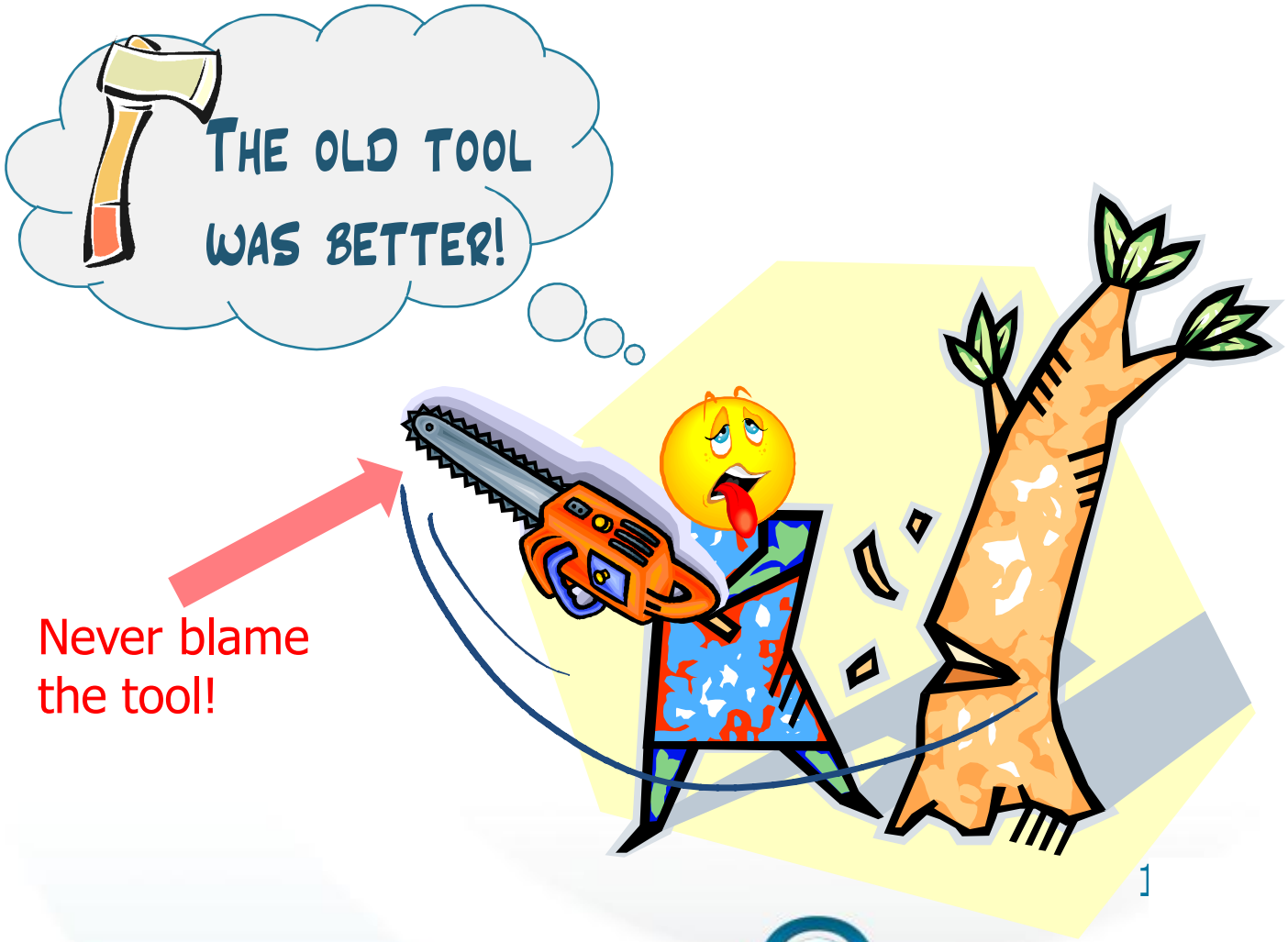
Pair programming

Visualize the workflow



crisp

Any tool can be misused and will



Never blame the tool!

Compare for understanding, not judgement

More prescriptive

More adaptive

RUP (120+)

- Architecture Reviewer
- Business Designer
- Business-Model Reviewer
- Business-Process Analyst
- Capsule Designer
- Change Control Manager
- Code Reviewer
- Configuration Manager
- Course Developer
- Database Designer
- Deployment Manager
- Design Reviewer
- Designer
- Graphic Artist
- Implementer
- Integrator
- Process Engineer
- Project Manager
- Project Reviewer
- Requirements Reviewer
- Requirements Specifier
- Software Architect
- Stakeholder
- System Administrator
- System Analyst
- Technical Writer
- Test Analyst
- Test Designer
- Test Manager
- Tester
- Tool Specialist
- User-Interface Designer
- Architectural analysis
- Assess Viability of architectural proof-of-concept
- Capsule design
- Class design
- Construct architectural proof-of-concept
- Database design
- Describe distribution
- Describe the run-time architecture
- Design test packages and classes
- Develop design guidelines
- Develop programming guidelines
- Identify design elements
- Identify design mechanisms
- Incorporate design elements
- Prioritize use cases
- Review the architecture
- Review the design
- Structure the implementation model
- Subsystem design
- Use-case analysis
- Use-case design
- Analysis model
- Architectural proof-of-concept
- Bill of materials
- Business architecture document
- Business case
- Business glossary
- Business modeling guidelines
- Business object model
- Business rules
- Business use case
- Business use case realization
- Business use-case model
- Business vision
- Change request
- Configuration audit findings
- Configuration management plan
- Data model
- Deployment model
- Deployment plan
- Design guidelines
- Design model
- Development case
- Development-organization assessment
- End-user support mateirla
- Glossary
- Implementation model
- Installation artifacts
- Integration build plan
- Issues list
- Iteration assessment
- Iteration plan
- Manual styleguide
- Programming guidelines
- Quality assurance plan
- Reference architecture
- Release notes
- Requirements attributes
- Requirements management plan
- Review record
- Risk list
- Risk management plan
- Software architecture document
- Software development plan
- Software requirements specification
- Stakeholder requests
- Status assessment
- Supplementary business specification
- Supplementary specification
- Target organization assessment
- Test automation architecture
- Test cases
- Test environment configuration
- Test evaluation summary
- Test guidelines
- Test ideas list
- Test interface specification
- Test plan
- Test suite
- Tool guidelines
- Training materials
- Use case model
- Use case package
- Use-case modeling guidelines
- Use-case realization
- Use-case storyboard
- User-interface guidelines
- User-interface prototype
- Vision
- Work order
- Workload analysis model

XP (13)

- Whole team
- Coding standard
- TDD
- Collective ownership
- Customer tests
- Pair programming
- Refactoring
- Planning game
- Continuous integration
- Simple design
- Sustainable pace
- Metaphor
- Small releases

Scrum (9)

- Scrum Master
- Product Owner
- Team
- Sprint planning meeting
- Daily Scrum
- Sprint review
- Product backlogt
- Sprint backlog
- BUrndown chart

Kanban (3)

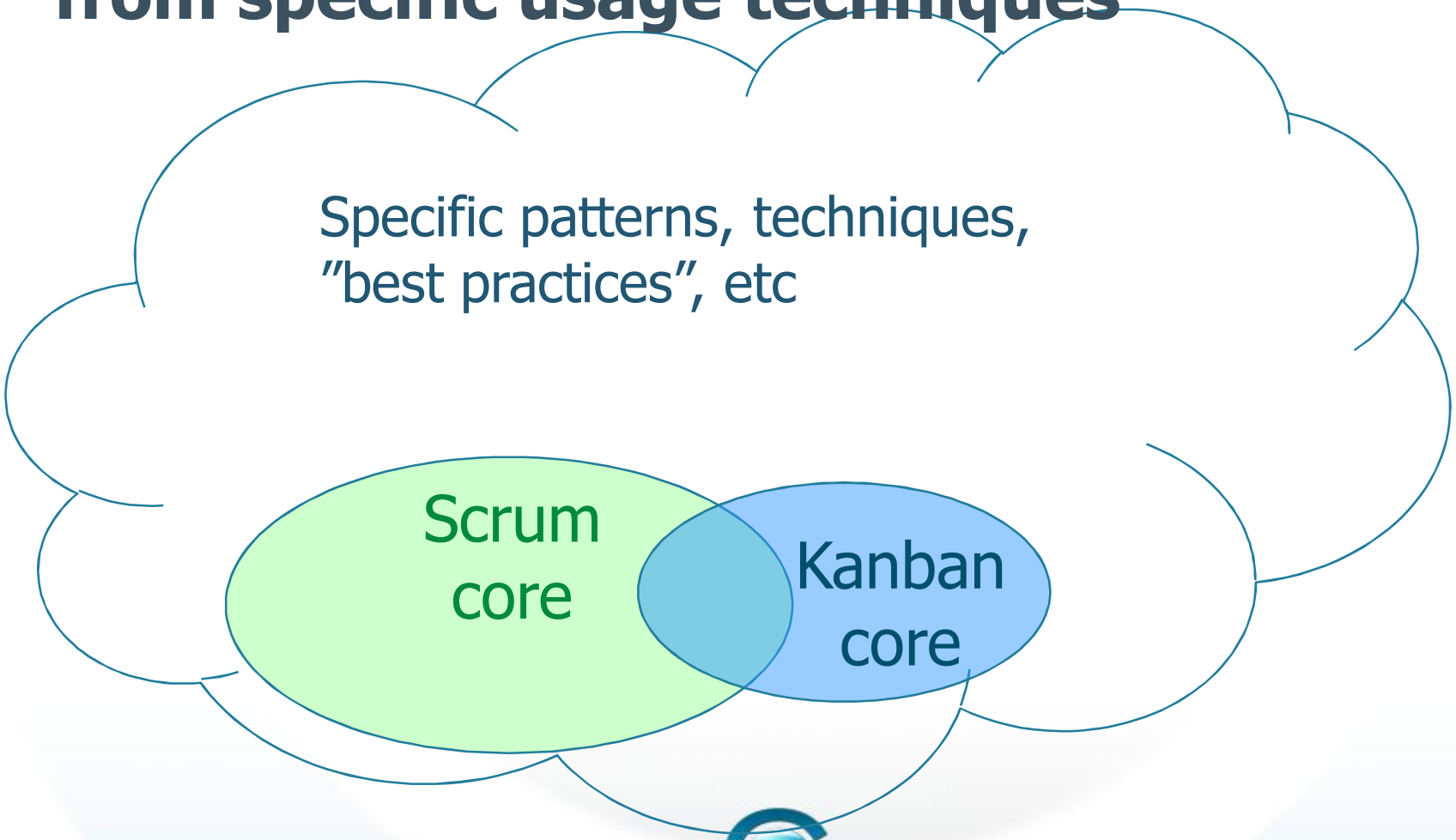
- Visualize the workflow
- Limit WIP
- Measure and optimize lead time

Do Whatever (0)

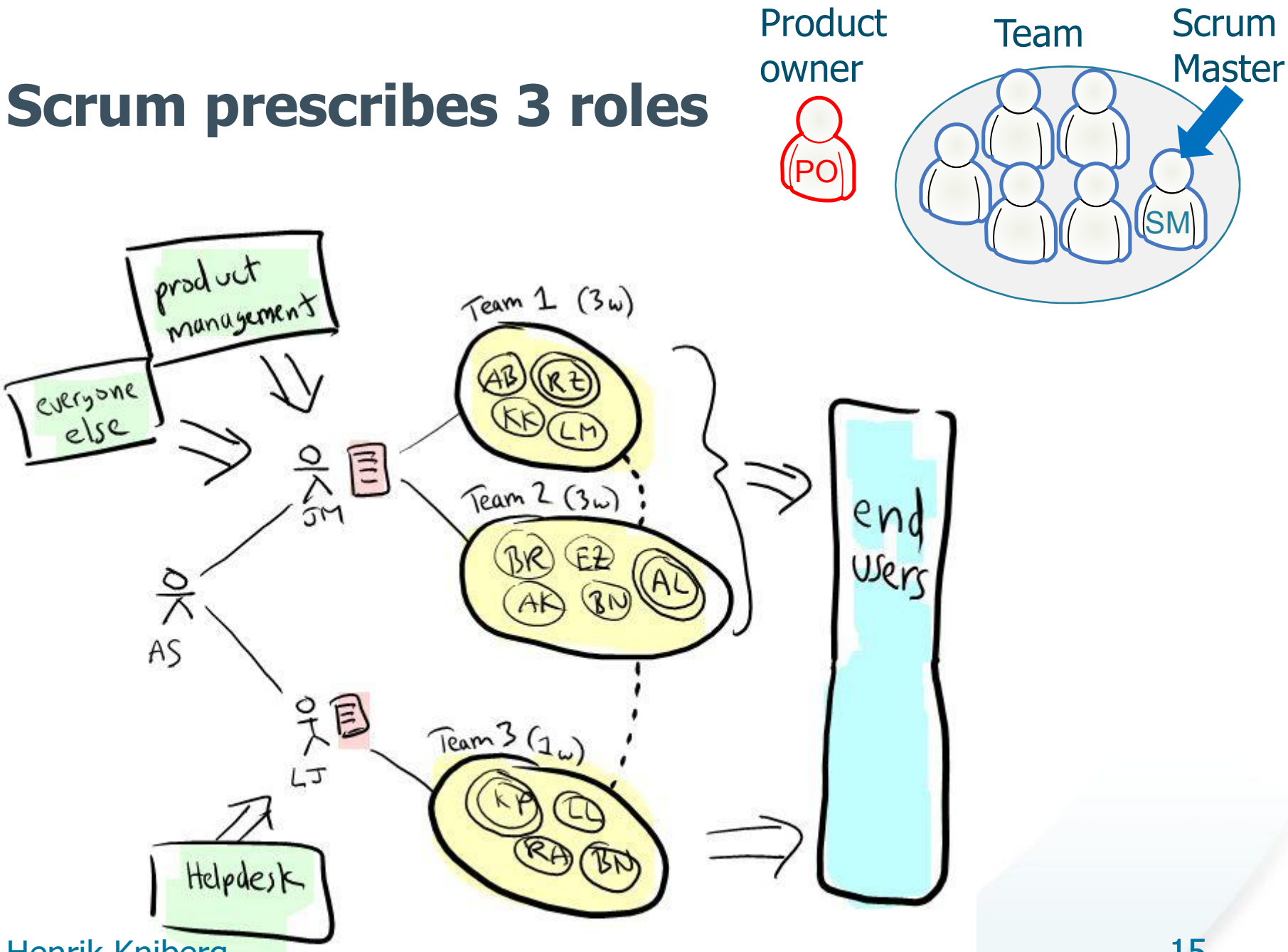
Do not develop an attachment
to any one weapon
or any one school of fighting

Miyamoto Musashi
17th century samurai

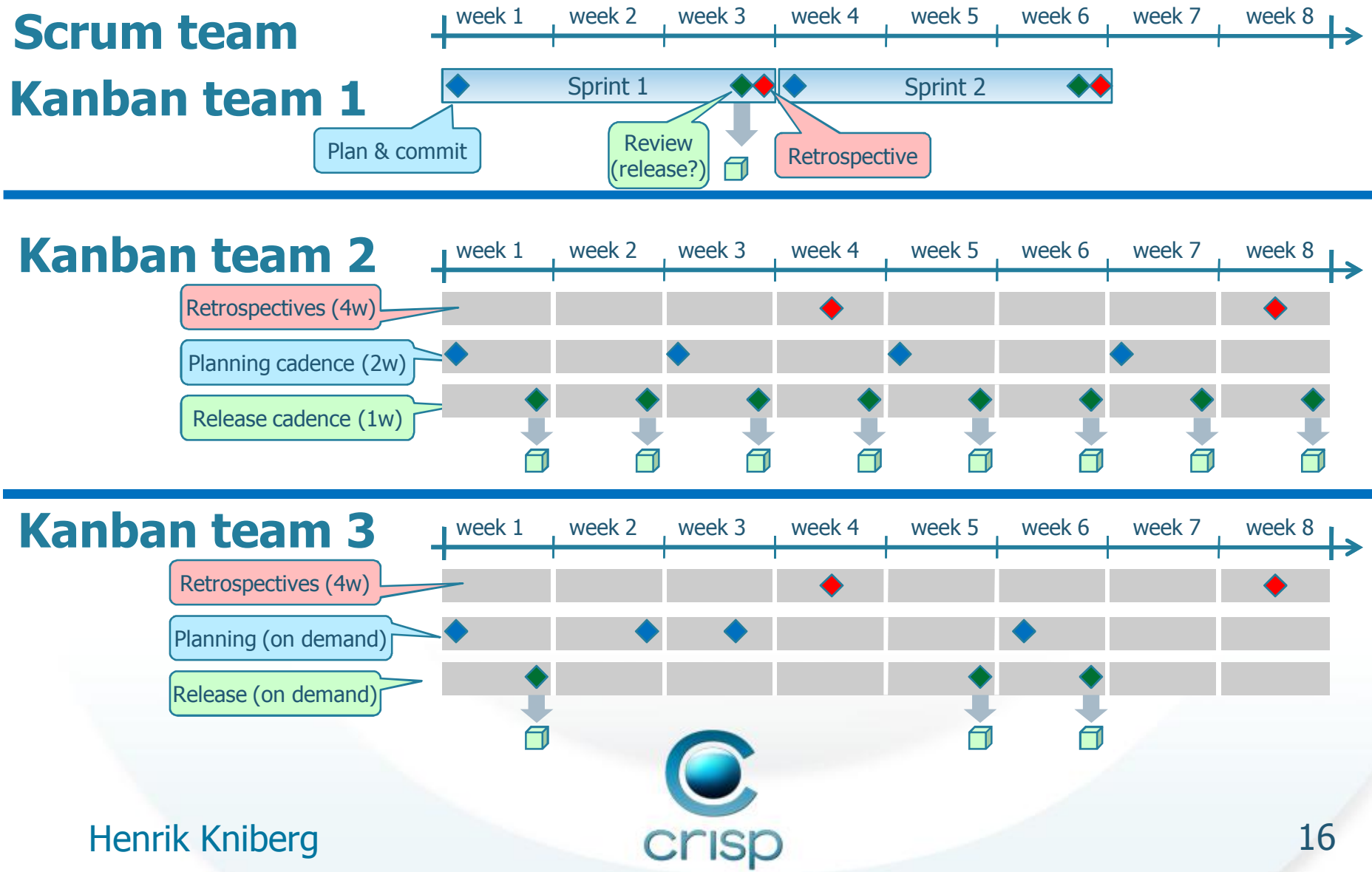
Distinguish between the tool itself from specific usage techniques



Scrum prescribes 3 roles

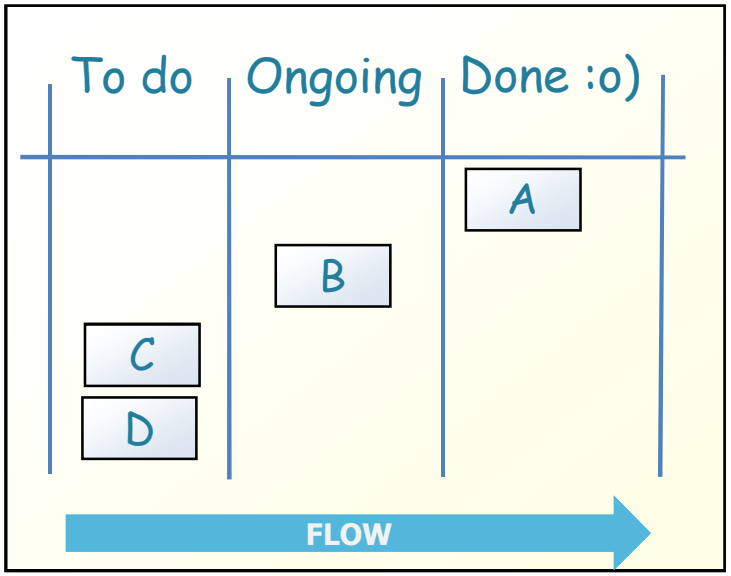


Scrum prescribes timeboxed iterations



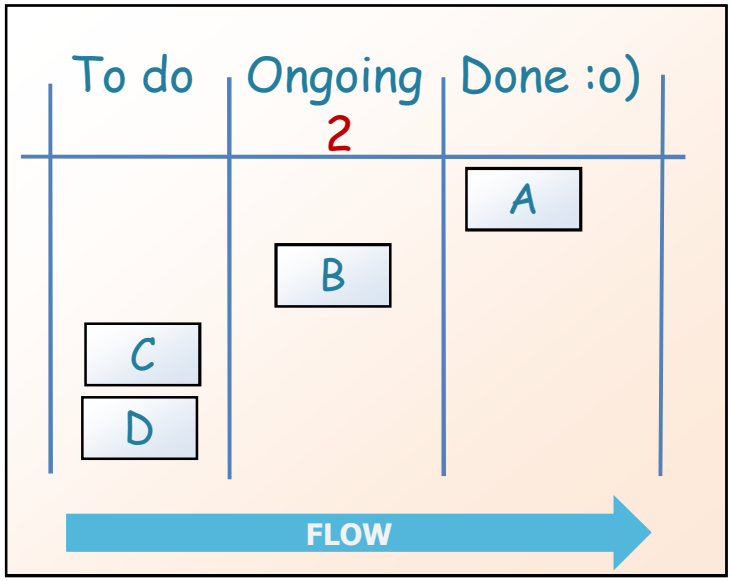
Both limit WIP, but in different ways

Scrum board



WIP limited per unit of time (iteration)

Kanban board



WIP limited per workflow state

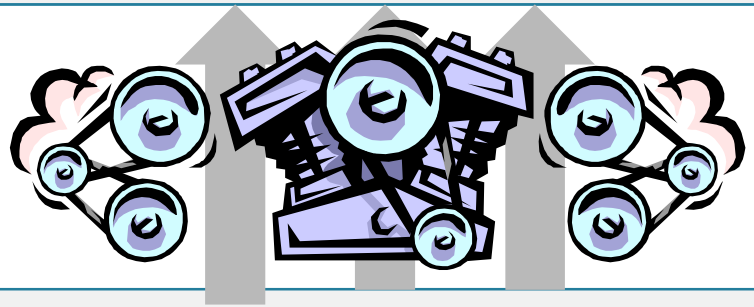
Both are empirical

Capacity
(aka velocity)

Lead time
(aka cycle time)

Quality
(defect rate, etc)

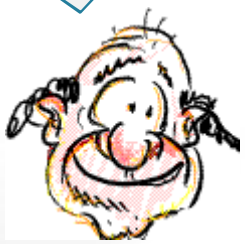
Predictability
(SLA fulfillment, etc)



Many small teams	Few large teams
Low WIP limits	High WIP limits
Few workflow states	Many workflow states
Short iterations	Long iterations
Little planning	Lots of planning
.... etc ...	.... etc ...

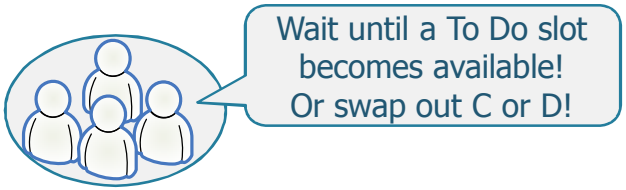
Kanban is more configurable

Great! More options!

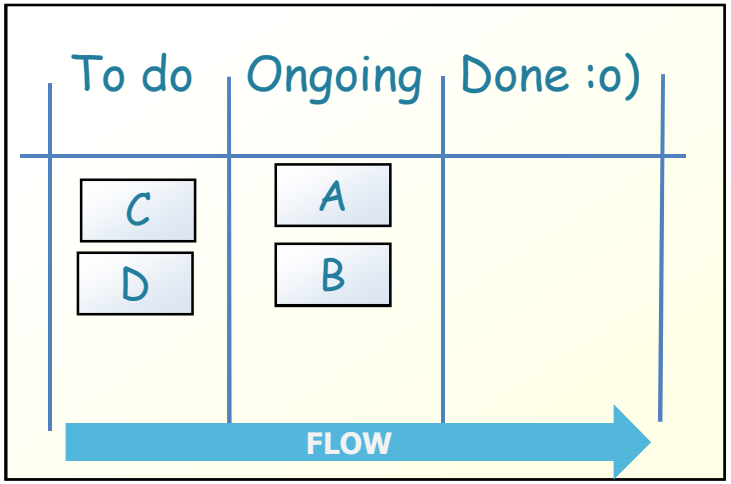


Oh no, more decisions!

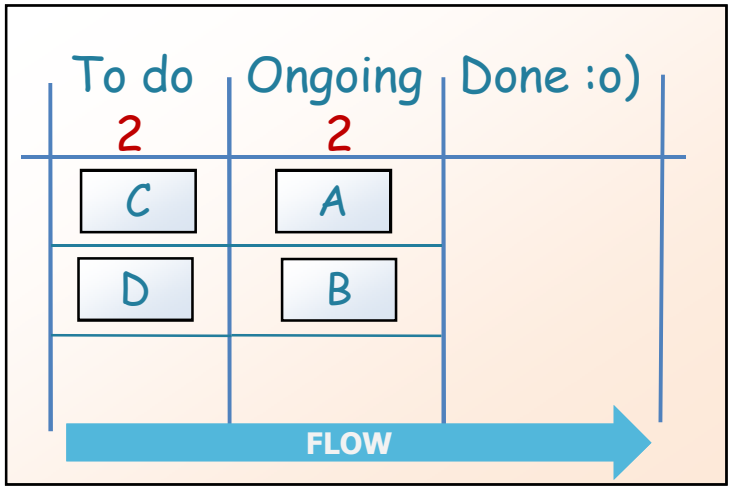
Scrum discourages change in mid-iteration



Scrum



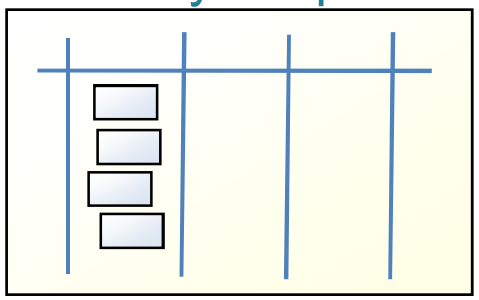
Kanban



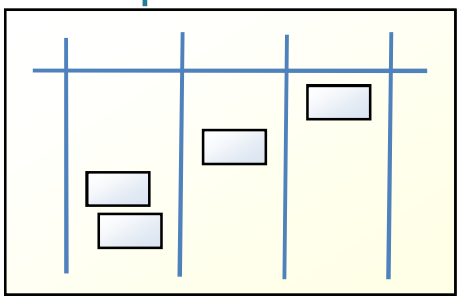
Scrum board is reset between each iteration

Scrum

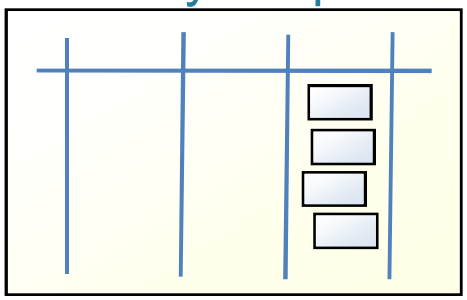
First day of sprint



Mid-sprint

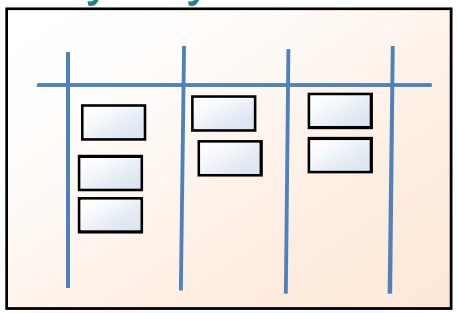


Last day of sprint

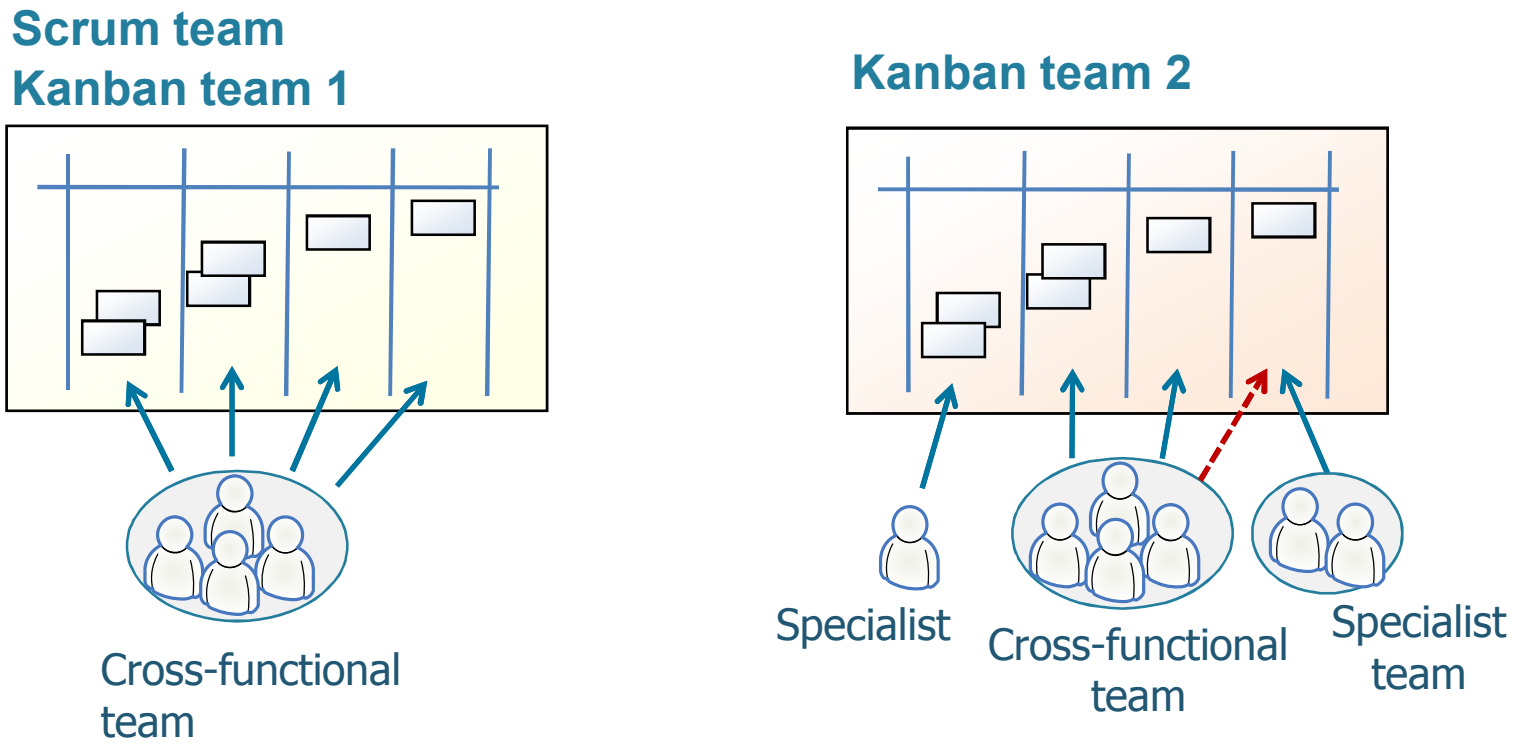


Kanban

Any day

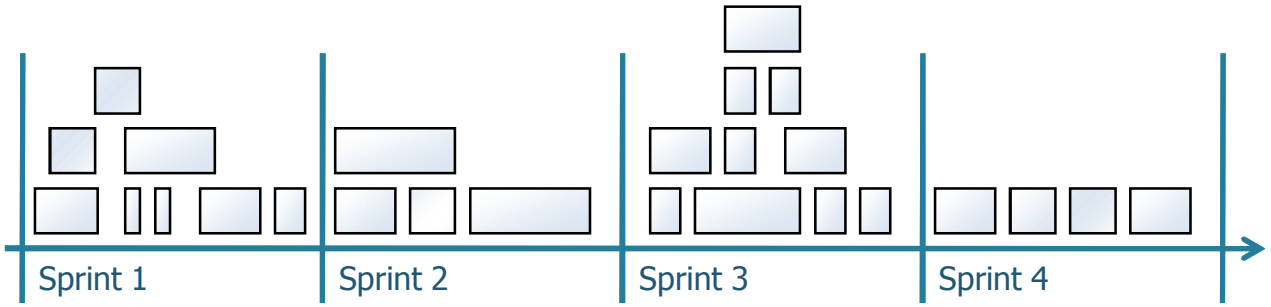


Scrum prescribes cross-functional teams

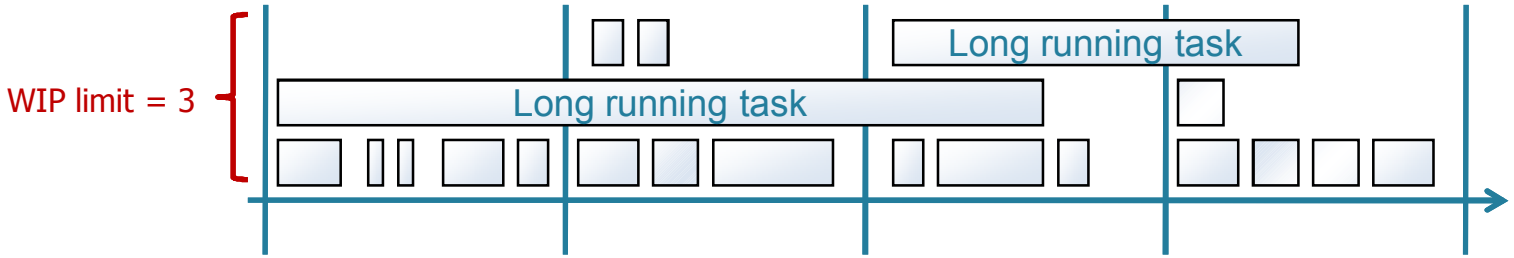


Scrum backlog items must fit in a sprint

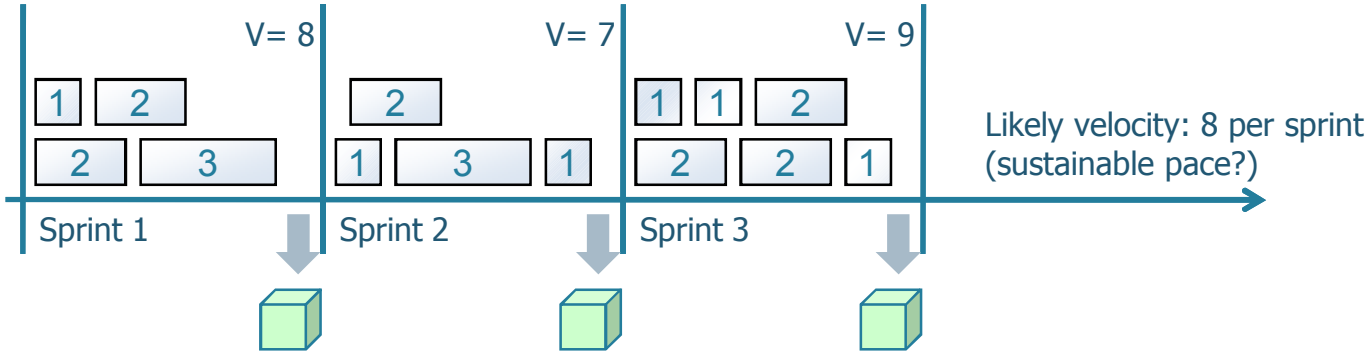
Scrum



Kanban

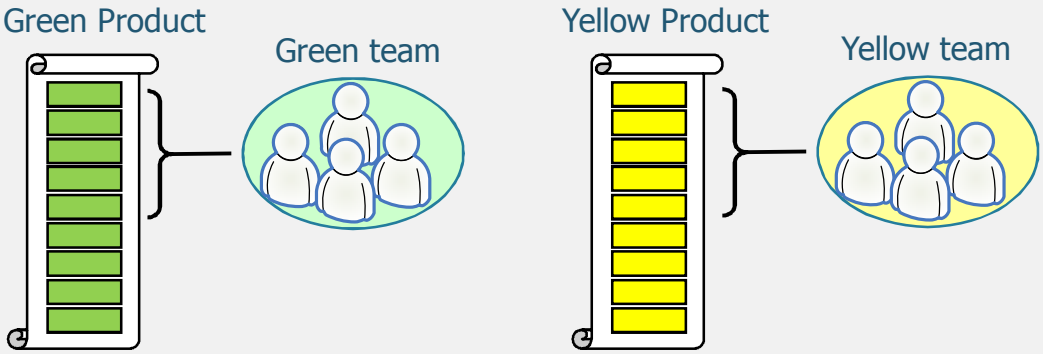


Scrum prescribes estimation and velocity



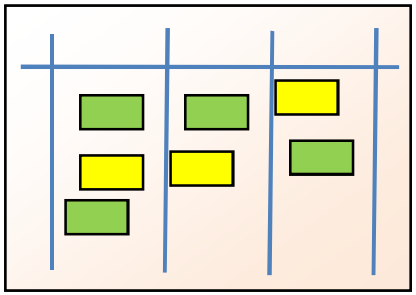
Both allow working on multiple products simultaneously

Scrum example 1

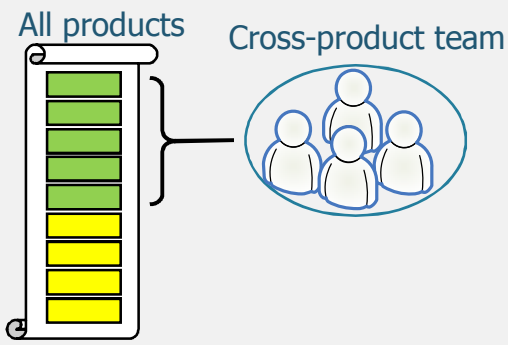


Kanban example 1

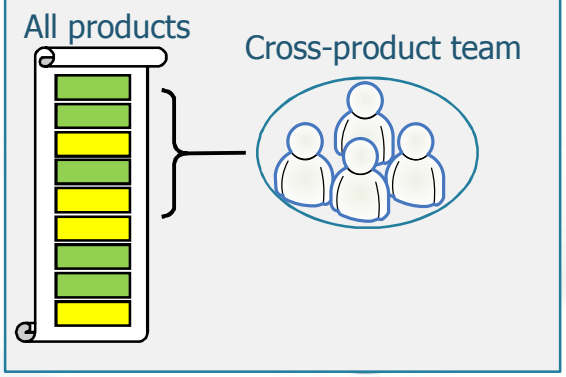
Color-coded tasks



Scrum example 2

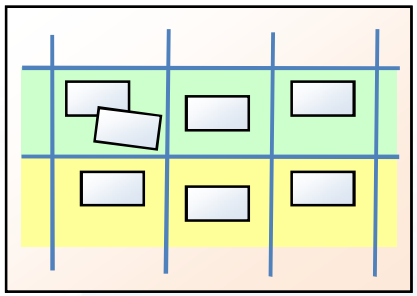


Scrum example 3



Kanban example 2

Color-coded swimlanes



more or less

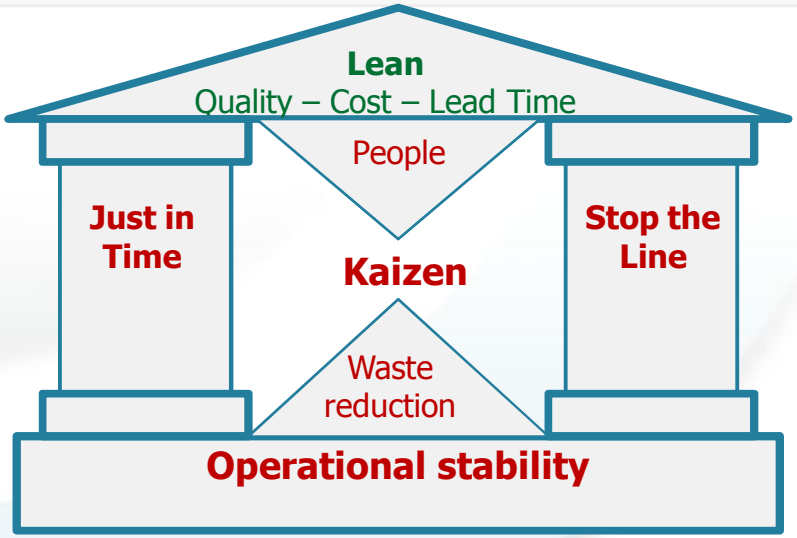
Both are Lean and Agile

Agile Manifesto

1. Individuals and Interactions over Processes and Tools
2. Working Software over Comprehensive Documentation
3. Customer Collaboration over Contract Negotiation
4. Responding to Change over Following a Plan

The Toyota Way

1. Base your management decisions on a Long-Term Philosophy, Even at the Expense of Short-Term Financial Goals
2. Create Continuous Process Flow to Bring Problems to the Surface
3. Use Pull Systems to Avoid Overproduction
4. Level Out the Workload (Heijunka)
5. Build a Culture of Stopping to Fix Problems, to Get Quality Right the First Time
6. Standardized Tasks are the Foundation for Continuous Improvement and Employee Empowerment
7. Use Visual Controls So No Problems are Hidden
8. Use Only Reliable, Thoroughly Tested Technology That Serves Your People and Processes
9. Grow Leaders Who Thoroughly Understand the Work, Live the Philosophy, and Teach It to Others
10. Develop Exceptional People and Teams Who Follow Your Company's Philosophy
11. Respect Your Extended Network of Partners and Suppliers by Challenging Them and Helping Them Improve
12. Go and See for Yourself to Thoroughly Understand the Situation (Genchi Genbutsu)
13. Make Decisions Slowly by Concensus, Thoroughly Considering All Options; Implement Decisions Rapidly
14. Become a Learning Organization Through Relentless Reflection (Hansei) and Continuous Improvement (Kaizen)



Henrik Kniberg

crisp

Minor difference:
Scrum prescribes a prioritized product backlog

Scrum:

- Product backlog must exist
- Changes to product backlog take effect next sprint (not current sprint)
- Product backlog must be sorted by "business value"

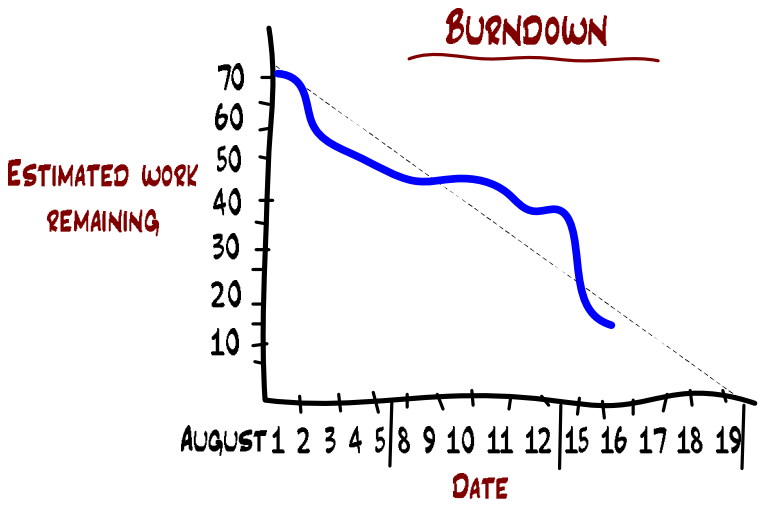
Kanban:

- Product backlog is optional
- Changes to product backlog take effect as soon as capacity becomes available
- Any prioritization scheme can be used. For example:
  - Take any item
  - Always take the top item
  - Always take the oldest item
  - 20% on maintainance items, 80% on new features
  - Split capacity evenly between product A and product B
  - Always take red items first

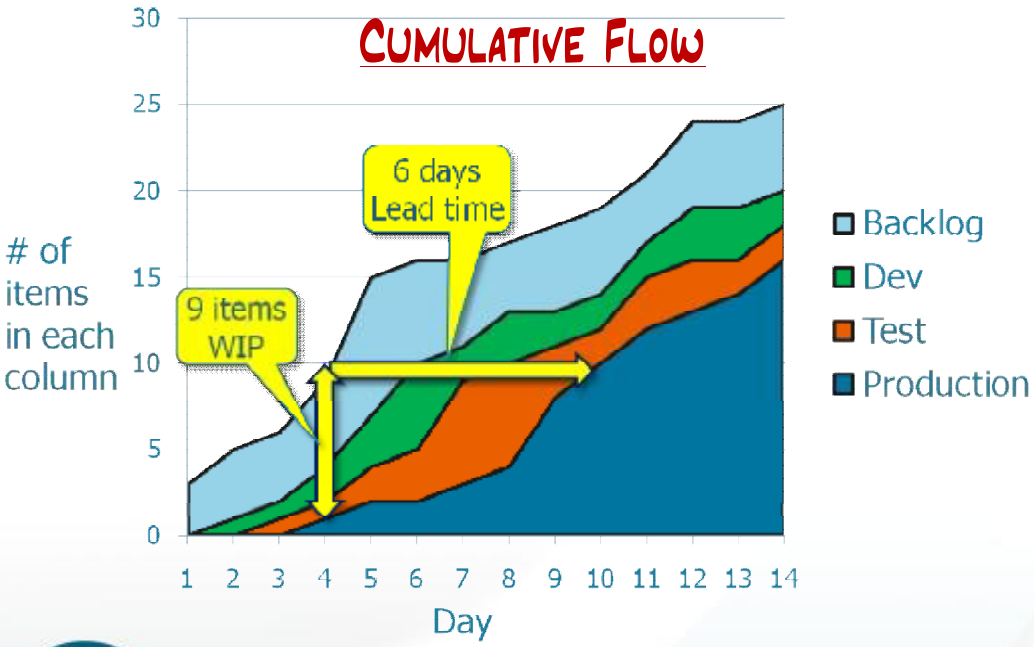
Minor difference:
Scrum prescribes daily meetings

... but many Kanban teams do that anyway.

Minor difference:
In Scrum, burndown charts are prescribed

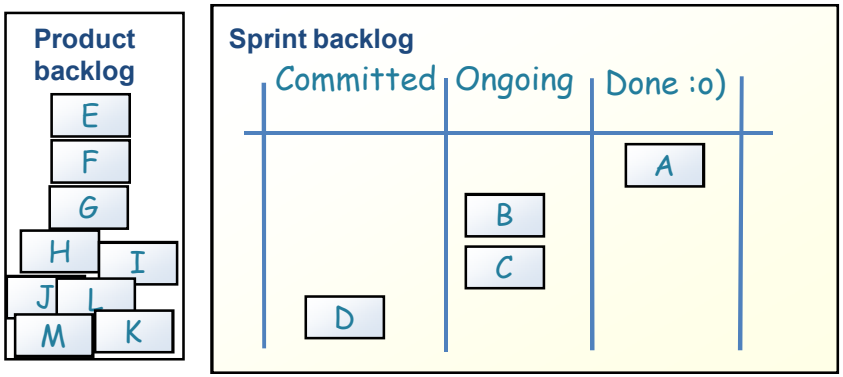


No specific types of diagrams prescribed in Kanban. Teams use whatever they need.

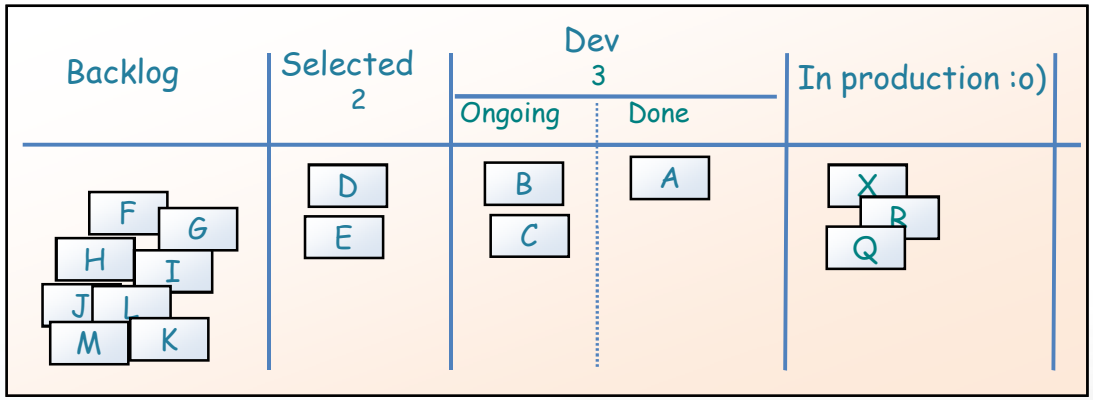


Example: Scrum board vs Kanban board

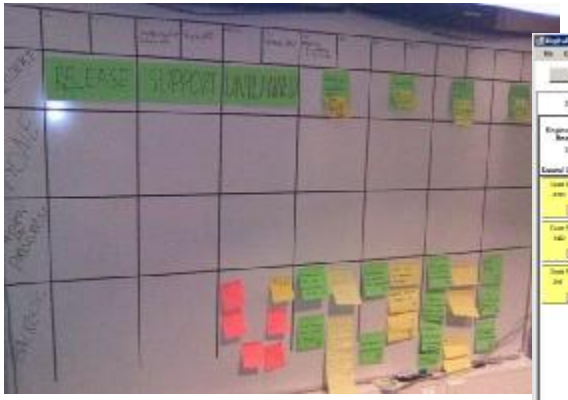
Scrum

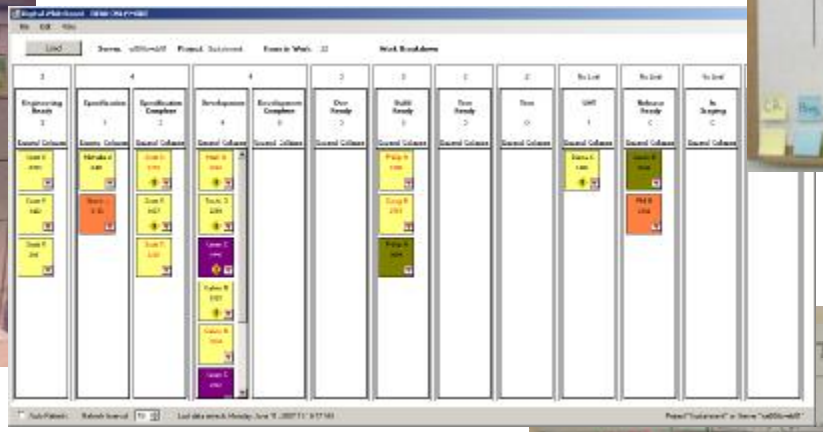


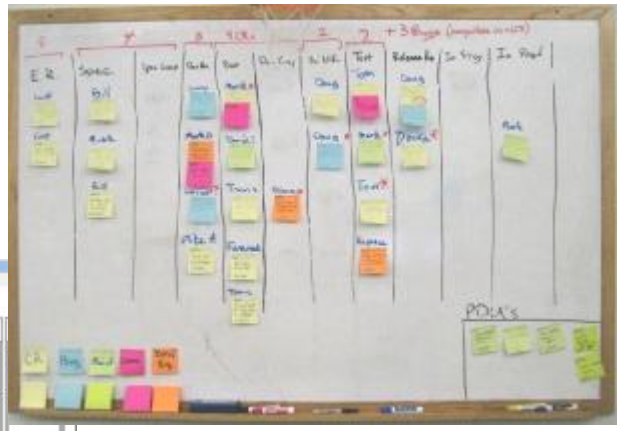
Kanban

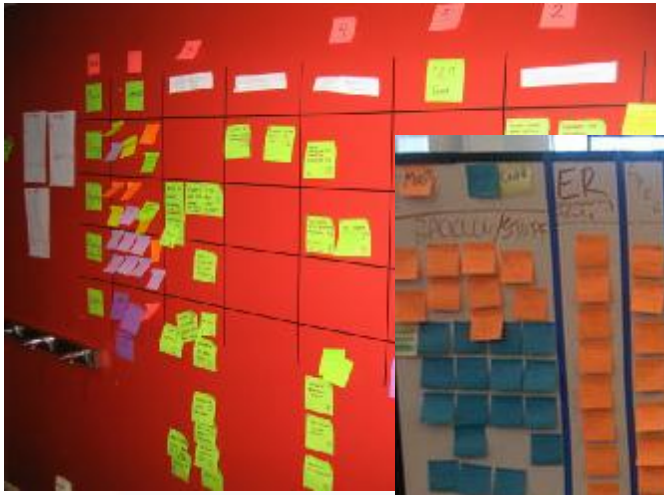


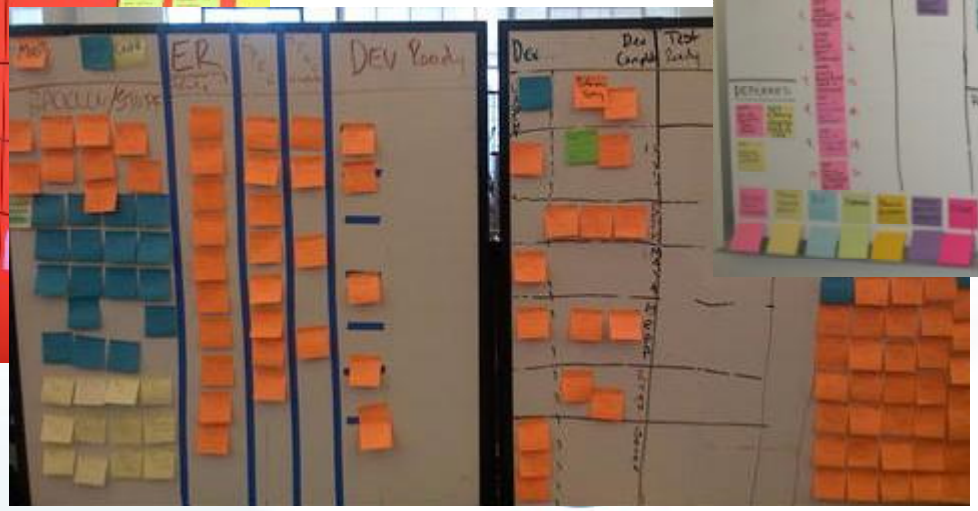
Evolve your own unique board!

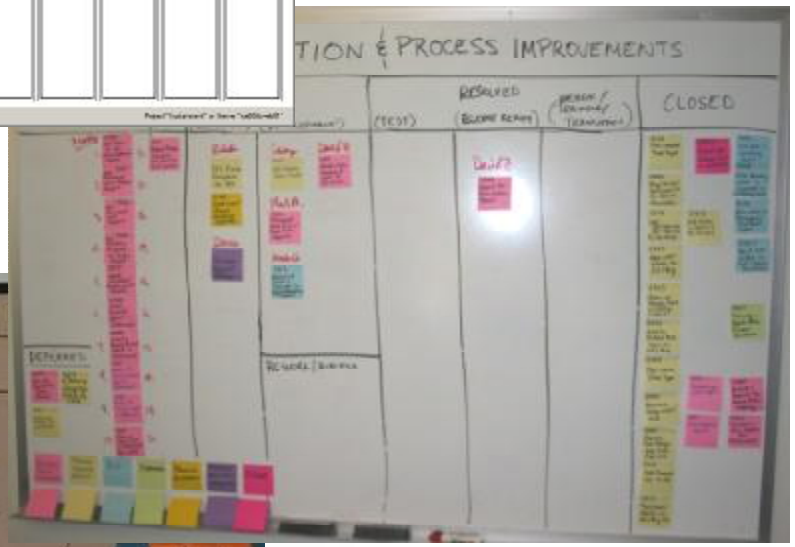






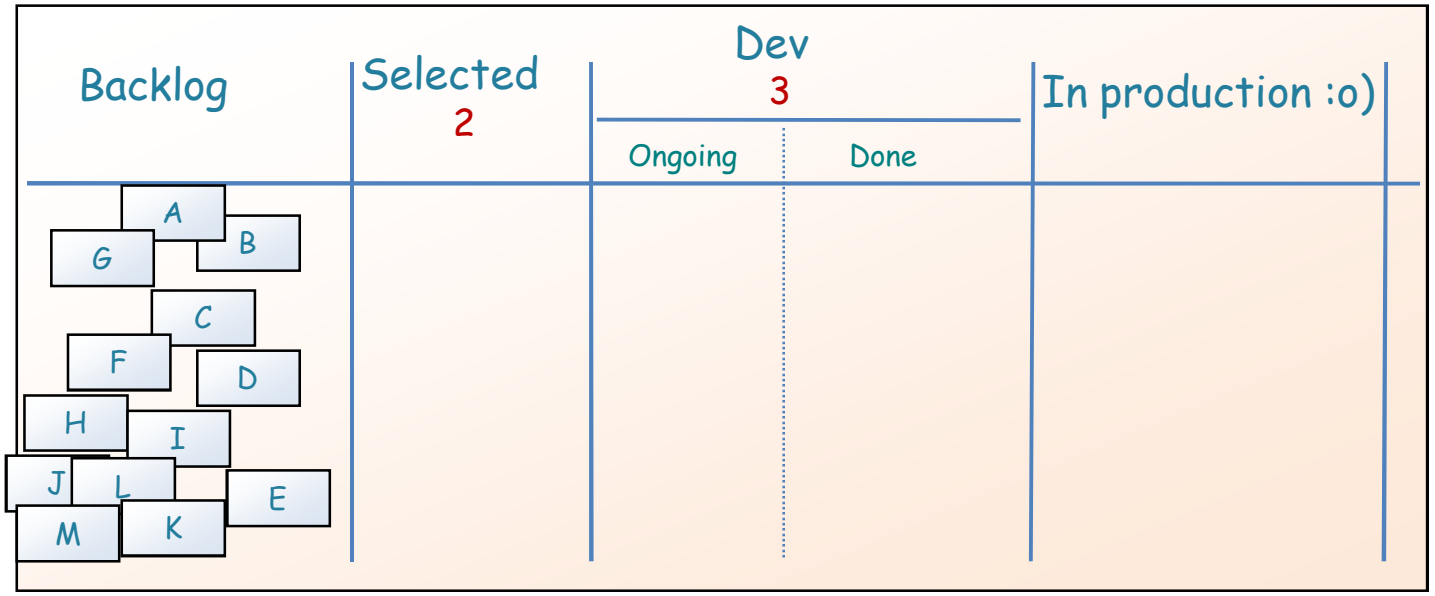




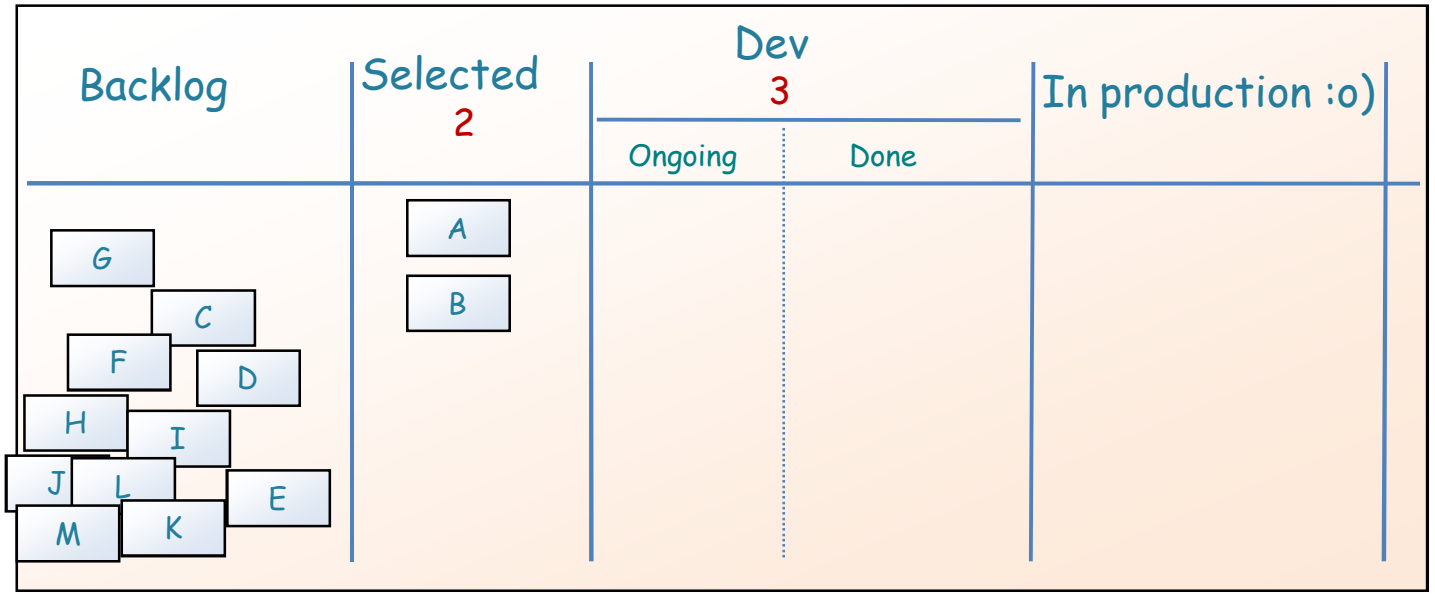


Some of these photos courtesy of David Anderson, Mattias Skarin, and various other people

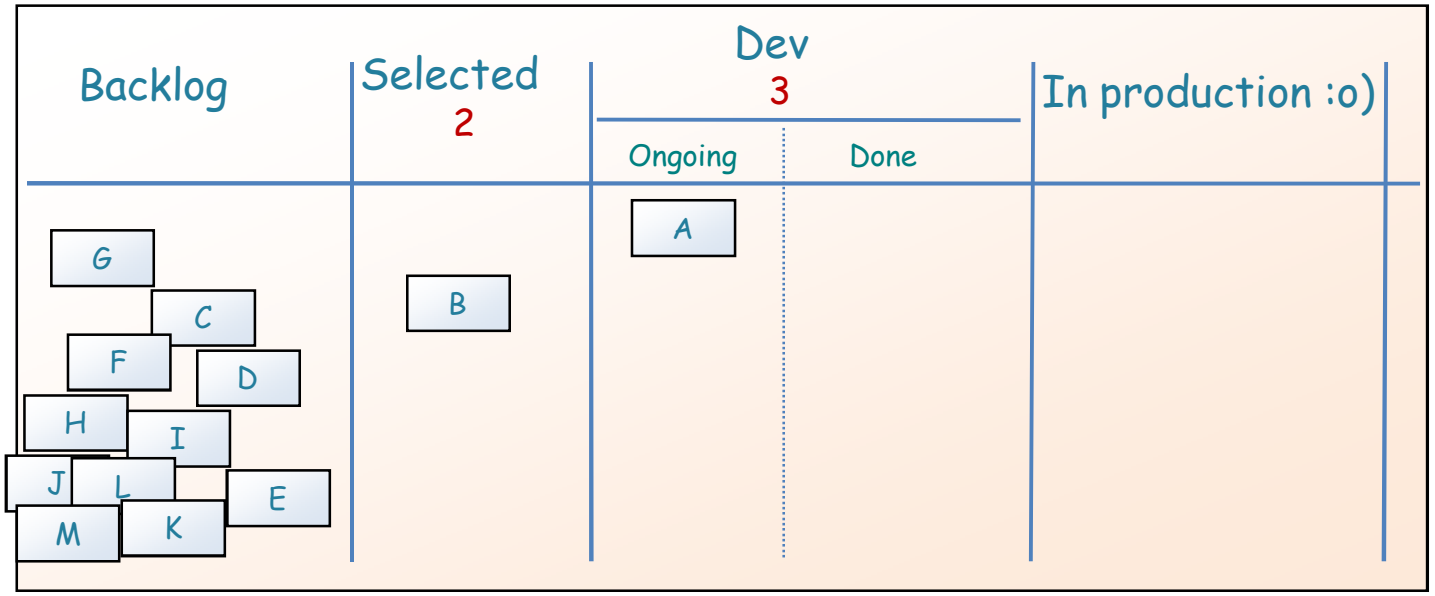
Scenario 1 – one piece flow



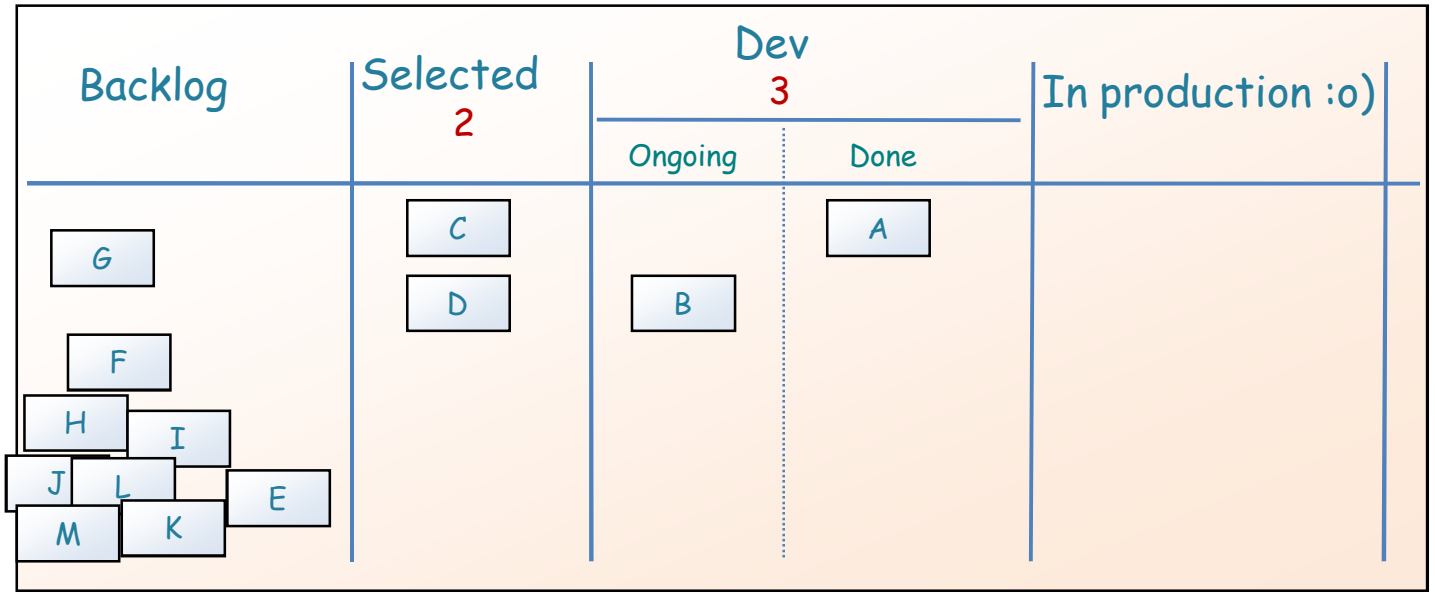
Scenario 1 – one piece flow



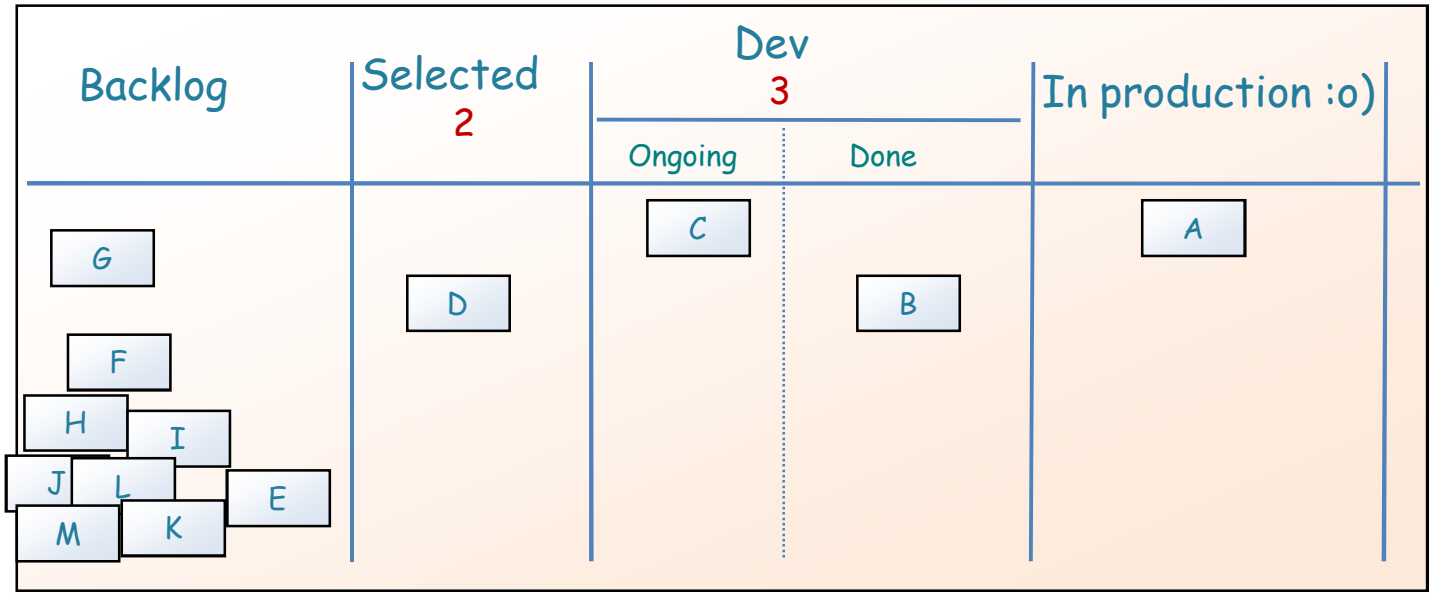
Scenario 1 – one piece flow



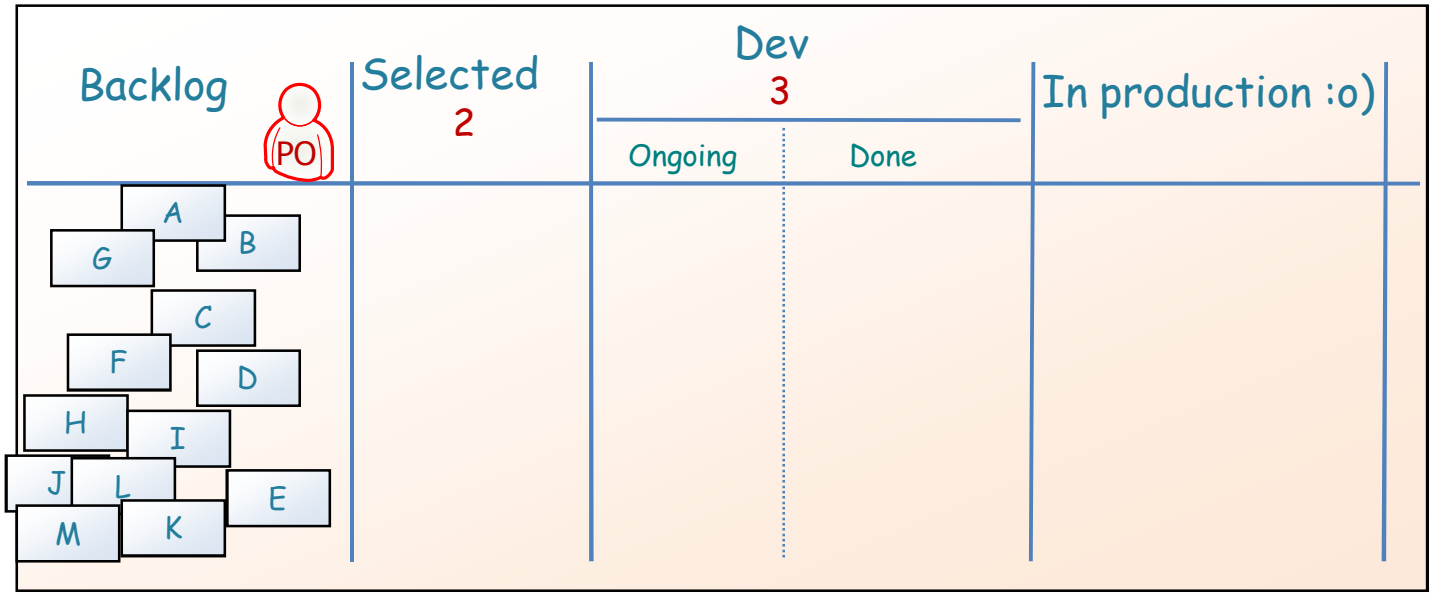
Scenario 1 – one piece flow



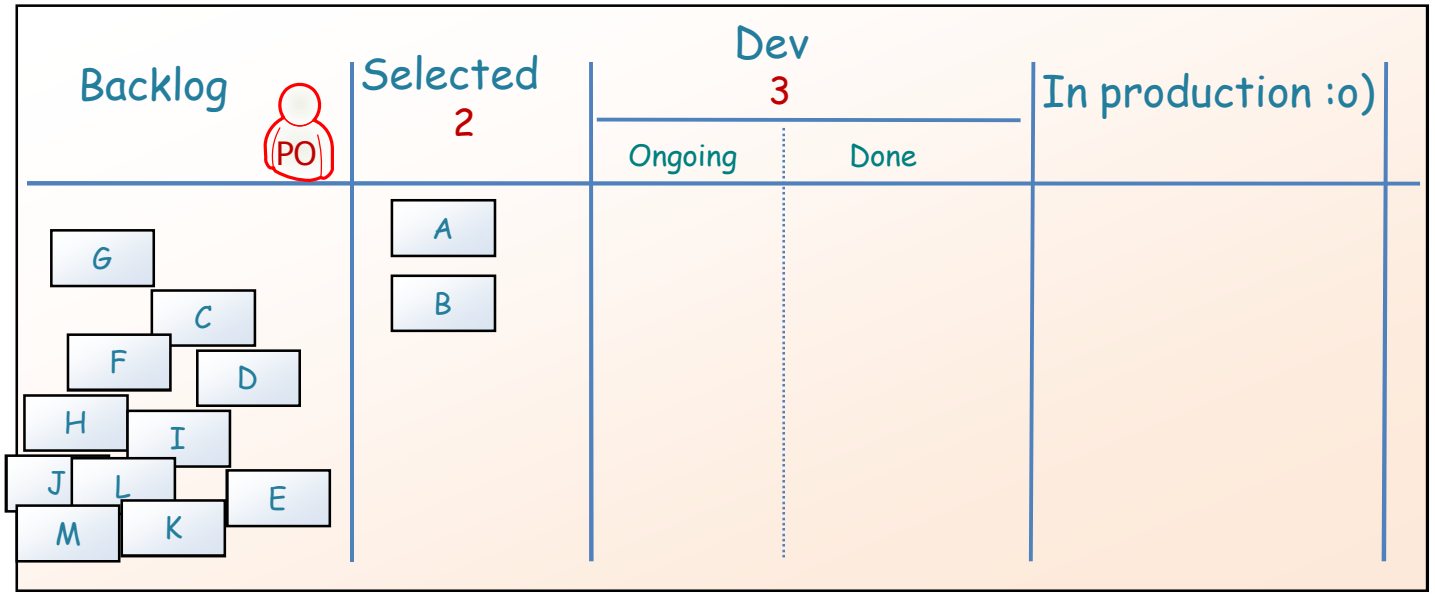
Scenario 1 – one piece flow.



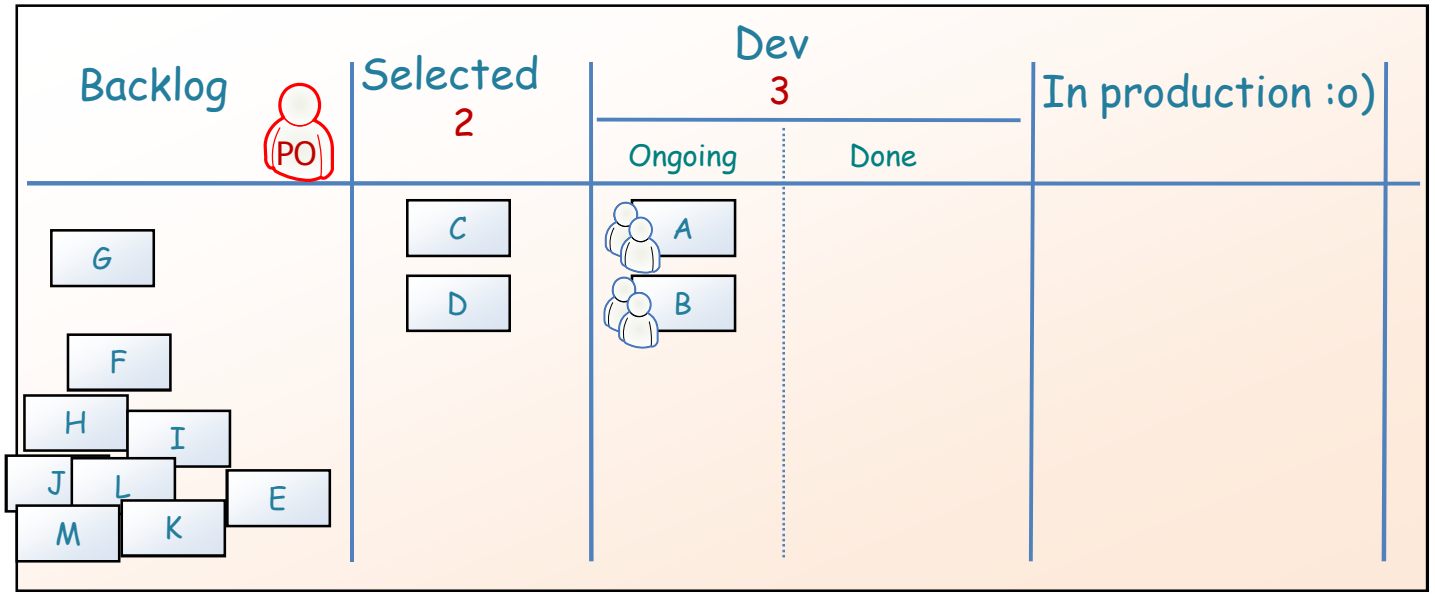
Scenario 2 – Deployment problem



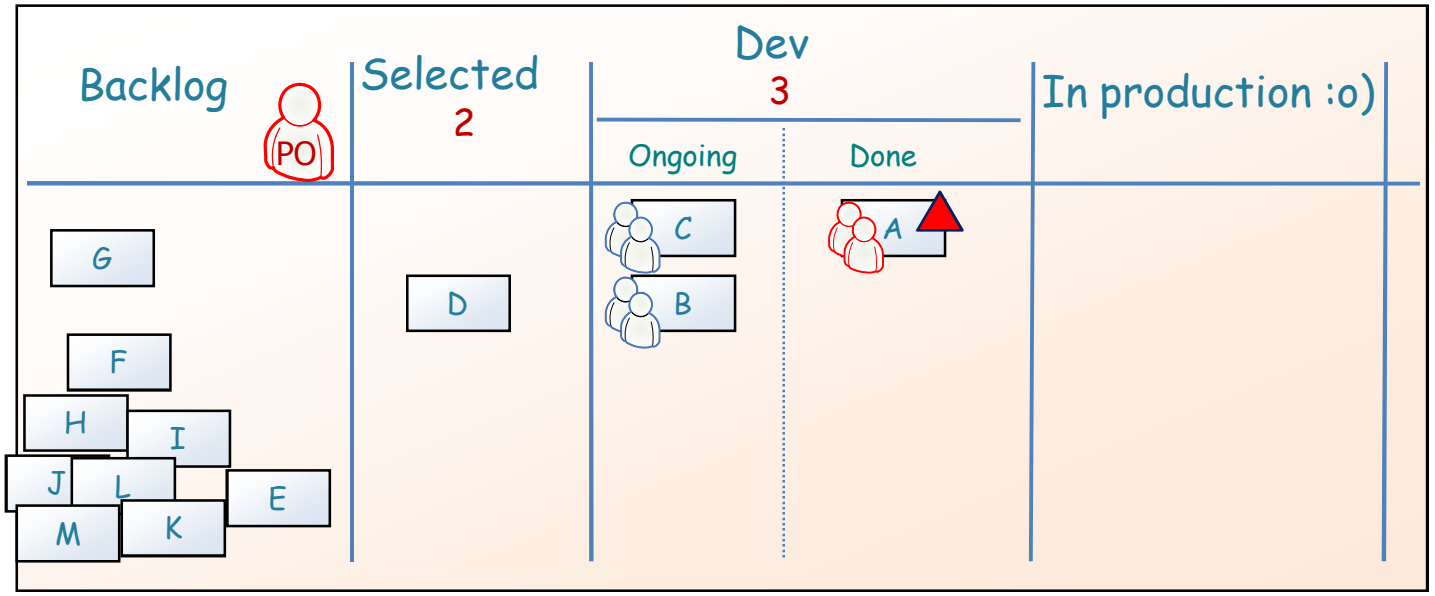
Scenario 2 – Deployment problem



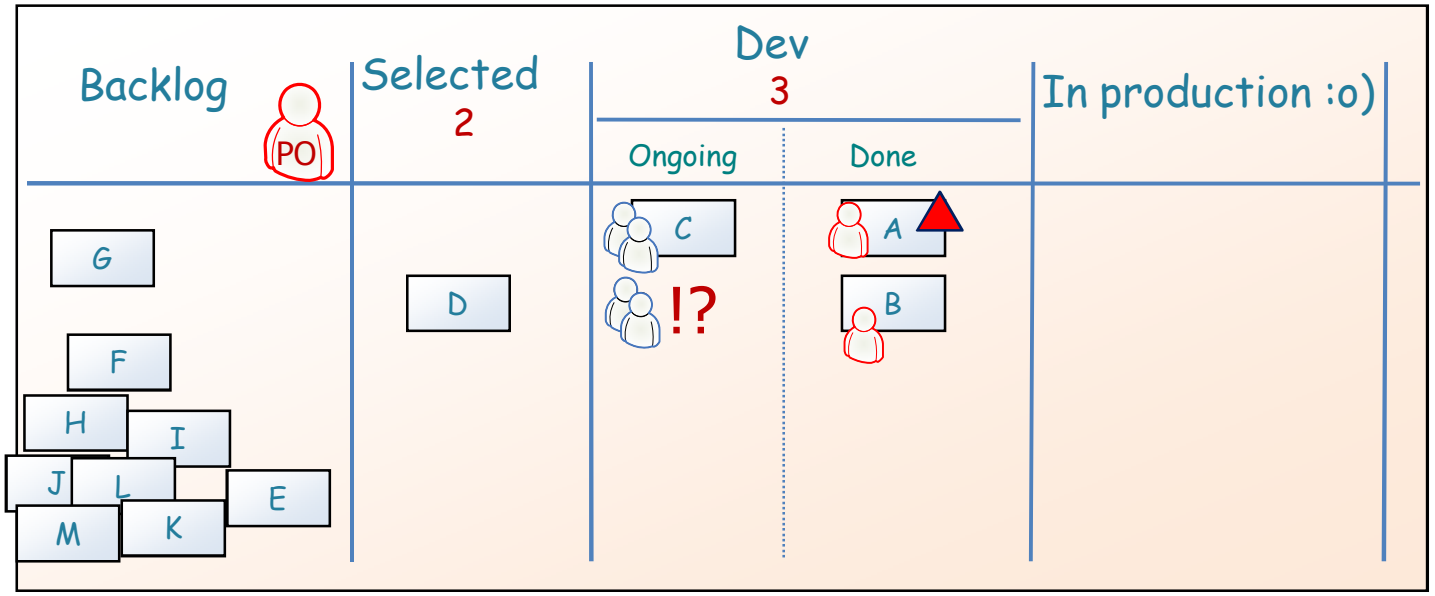
Scenario 2 – Deployment problem



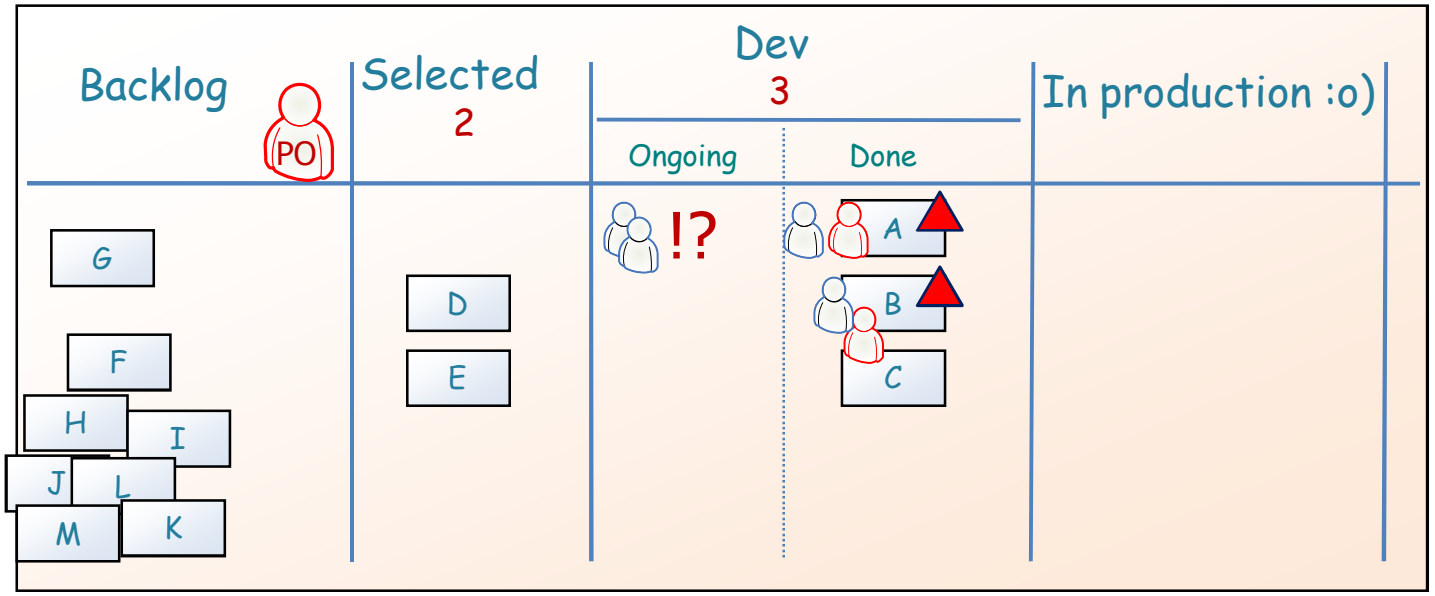
Scenario 2 – Deployment problem



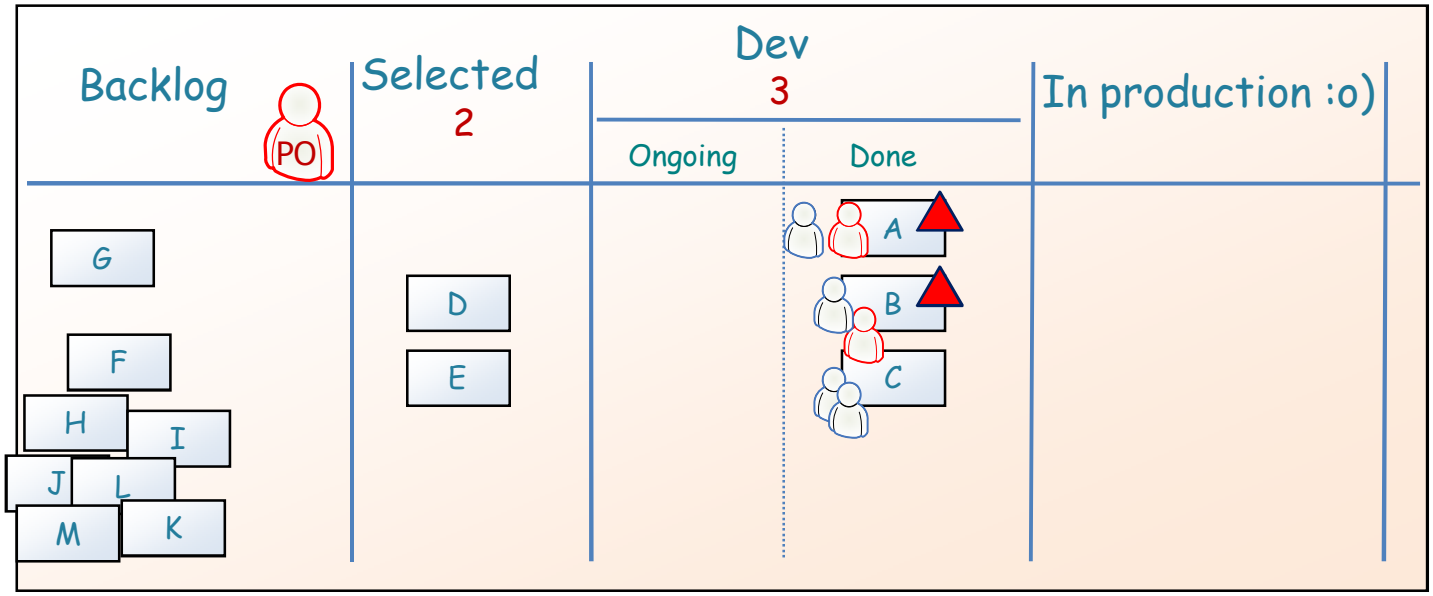
Scenario 2 – Deployment problem



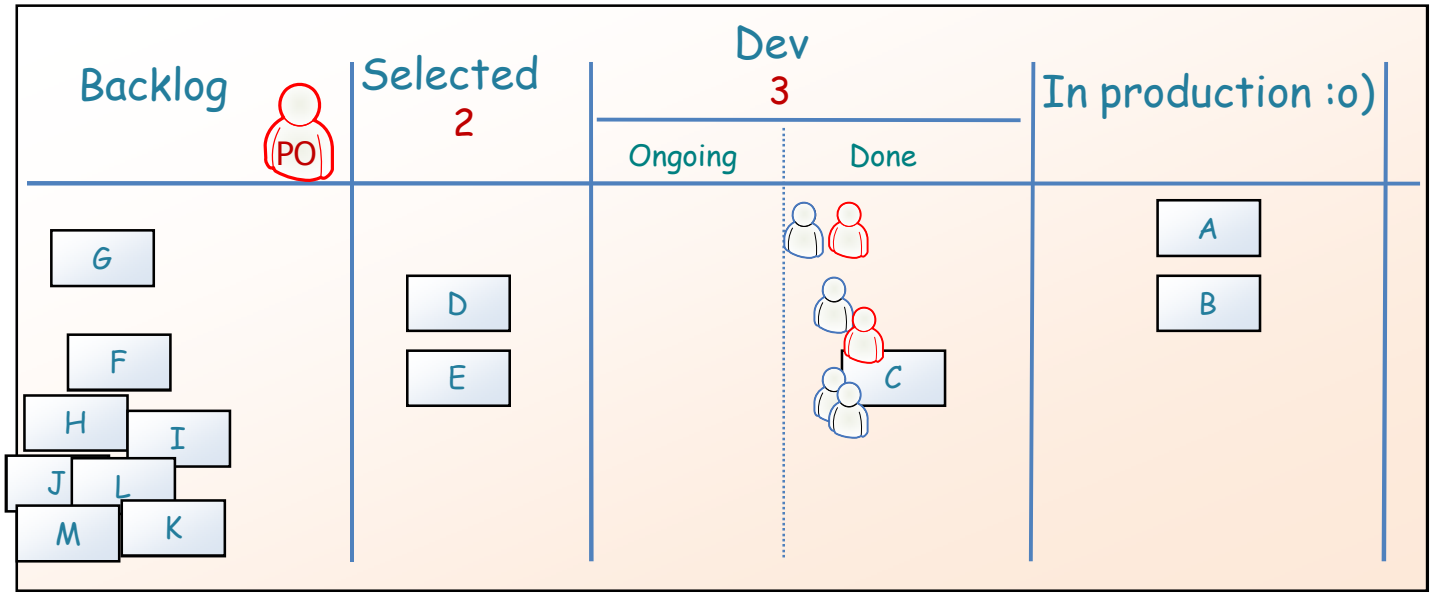
Scenario 2 – Deployment problem



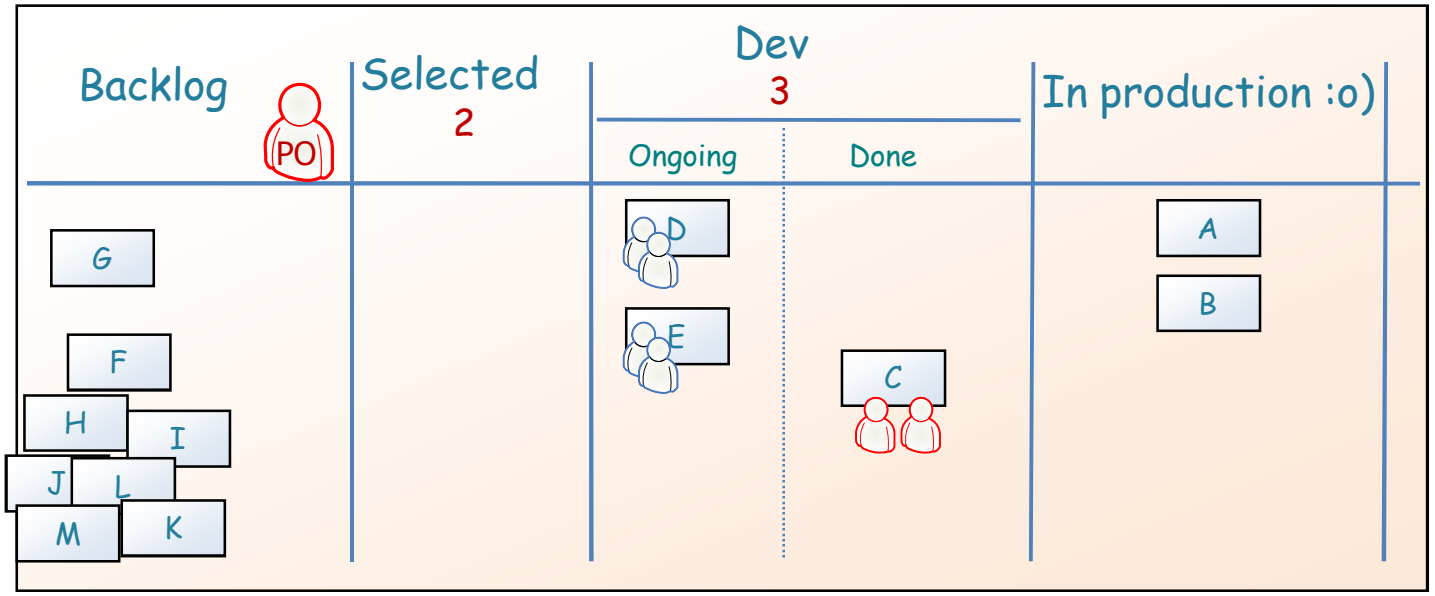
Scenario 2 – Deployment problem



Scenario 2 – Deployment problem

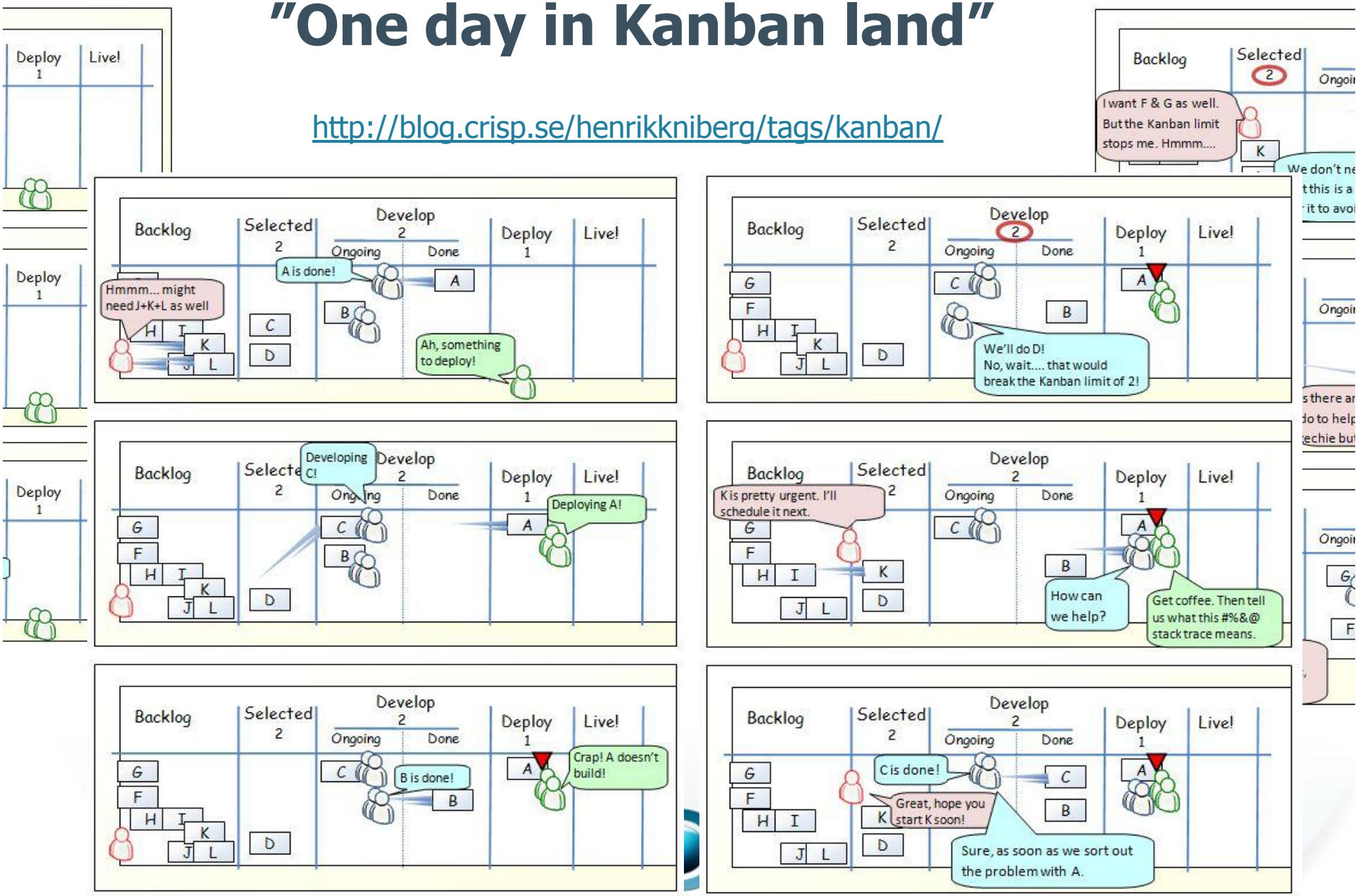


Scenario 2 – Deployment problem



”One day in Kanban land”

http://blog.crisp.se/henrikkniberg/tags/kanban/



Kanban vs Scrum

www.crisp.se/henrik.kniberg/kanban-vs-scrum.pdf

Summary

Similarities

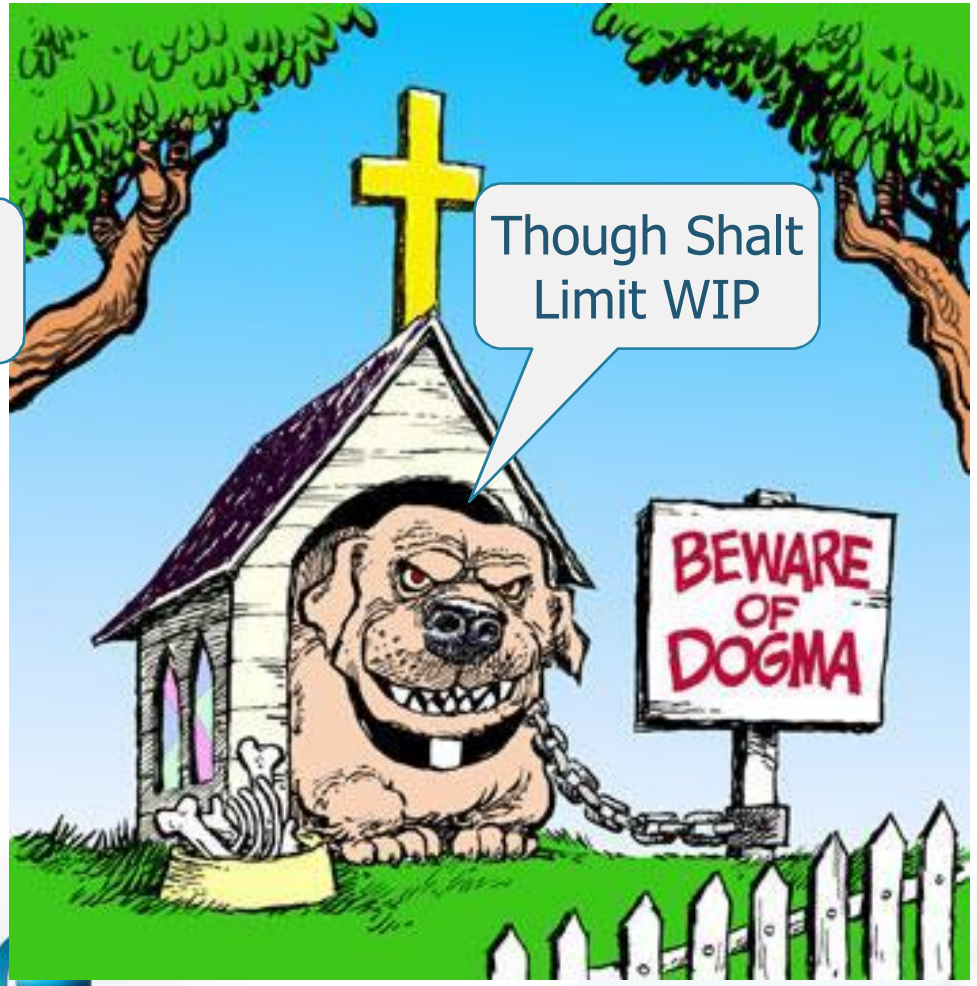
- Both are Lean and Agile
- Both based on pull scheduling
- Both limit WIP
- Both use transparency to drive process improvement
- Both focus on delivering releasable software early and often
- Both are based on self-organizing teams
- Both require breaking the work into pieces
- In both cases the release plan is continuously optimized based on empirical data (velocity / lead time)

Differences

Scrum	Kanban
Timeboxed iterations prescribed.	**Timeboxed iterations optional.**
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 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
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**.

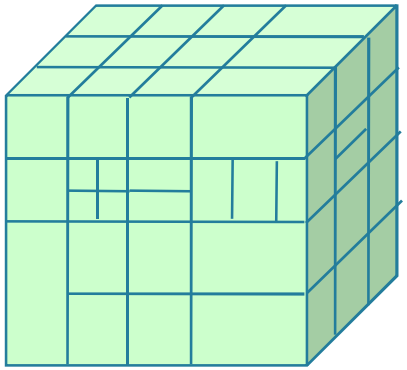
Henrik Kniberg

Don’t be dogmatic



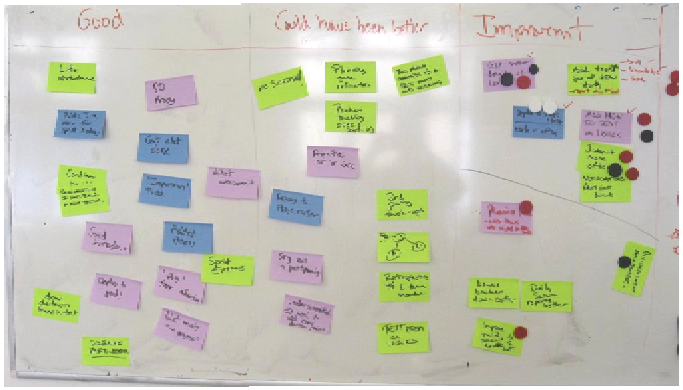
Essential skills needed both Kanban and Scrum

Splitting the system into deliverable increments

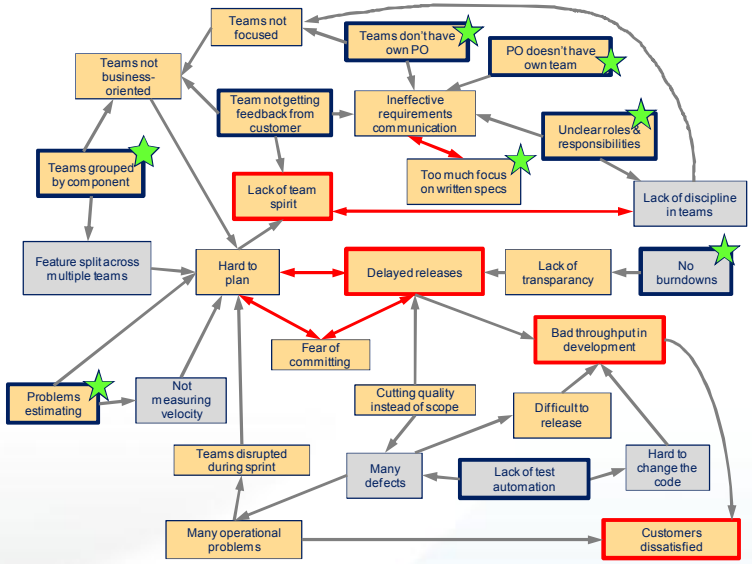


Software craftsmanship

Retrospectives

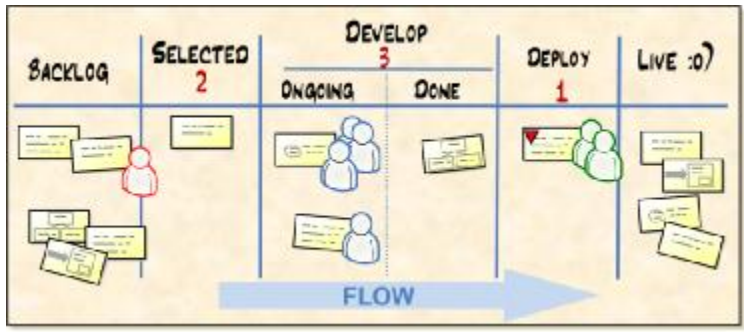


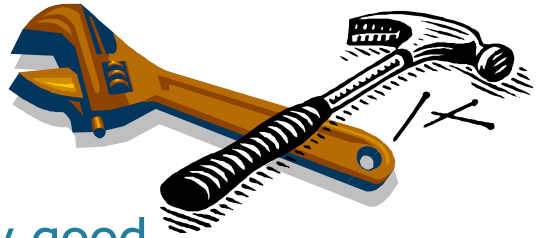
Root-cause analysis



http://www.crisp.se/henrik.kniberg/cause-effect-diagrams.pdf

Take-away points





1. **Know your goal**
   - Hint: Agile/Lean/Kanban/Scrum isn't it.
2. **Never blame the tool**
   - Tools don't fail or succeed. People do.
   - There is no such thing as a good or bad tool. Only good or bad decisions about when, where, how, and why to use which tool.
3. **Don't limit yourself to one tool**
   - Learn as many as possible.
   - Compare for understanding, not judgement.
4. **Experiment & enjoy the ride**
   - Don't worry about getting it right from start.
   - The only real failure is the *failure to learn* from failure.