Kanban vs Scrum
Making the most of both
QCon, San Francisco
Nov 18, 2009

Henrik Kniberg
Agile/Lean coach @ Crisp, Stockholm
http://www.crisp.se/henrik.kniberg

Background: developer, manager, entreprenuer
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

Split time

Optimize business value

Optimize process

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

Henrik Kniberg
## Typical Scrumboard

<table>
<thead>
<tr>
<th>Not Checked Out</th>
<th>Checked Out</th>
<th>Done! :D</th>
<th>Sprint Goal: Beta-ready release!</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migration Tool</td>
<td>Deposit</td>
<td>Write Failing test</td>
<td>Burndown</td>
</tr>
<tr>
<td>Impl migration</td>
<td>GU spec</td>
<td>Test GUI</td>
<td>Unplanned items</td>
</tr>
<tr>
<td>Impl GUI</td>
<td>Tapestry spike</td>
<td>Write failing test</td>
<td>Next</td>
</tr>
<tr>
<td>Impl GUI</td>
<td>Write failing test</td>
<td>Fix memory leak</td>
<td>Withdraw</td>
</tr>
<tr>
<td>Impl GUI</td>
<td>Clarify requirements</td>
<td>Sales support</td>
<td></td>
</tr>
<tr>
<td>Impl GUI</td>
<td>GUI design (CSS)</td>
<td>Write whitepaper</td>
<td></td>
</tr>
<tr>
<td>Impl GUI</td>
<td>GUI design</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Impl GUI</td>
<td>GUI design</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Impl GUI</td>
<td>GUI design</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Backoffice
- Login
  - Impl GUI
  - GUI design
  - GUI design
  - GUI design

- User Admin
  - GUI design (CSS)
  - GUI design
  - GUI design
  - GUI design

---

Henrik Kniberg
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

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 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

Clear definition 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

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
- PO understands purpose of all backlog items

- Top items in PBL small enough to fit in a sprint
- PO understands purpose of all backlog items

- PO has a product backlog (PBL)
- Top items are prioritized by business value
- Top items are estimated
- Estimates written by the team
- PO understands purpose of all backlog items

- PO speaks with one voice (in case PO is a team)

Scrum Checklist

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

the unofficial
Henrik Kniberg
Typical waterfall => Scrum evolution

1. Waterfall
   - Requirements
   - Design
   - Code
   - Test

2. "Scrum But"
   - Requirements
   - Design & code
   - Test

3. Scrum
   - PO
   - Feature team 1
   - Feature team 2

Henrik Kniberg
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
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

Henrik Kniberg
Kanban in software development
Typical Scrum => Kanban evolution

Scrum step 1

Feature team 1
Scrum

Feature team 2
Scrum

Feature team 2
Scrum

Scrum step 2

Feature team 1
Scrum

Feature team 2
Scrum

Feature team 2
Scrum

Scrum + Kanban

Feature team 1
Scrum

Feature team 2
Scrum

Feature team 2
Scrum

Operations / support team
Scrum

Operations / support team
Kanban

Henrik Kniberg
**Tool**

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

**Physical tools**

** Thinking tools**
a.k.a. “mindsets” or “philosophies”
- Lean
- Agile
- Systems Thinking
- Theory of Constraints

**Process tools**
a.k.a. “organizational patterns”
- Pair programming
- Visualize the workflow

**Toolkits**
a.k.a. “frameworks”
- Scrum
- RUP
- Kanban
- XP

**Product Owner role**

- Henrik Kniberg

**Tool**

“anything used as a means of accomplishing a task or purpose.”
- dictionary.com
Can we compare Kanban and Scrum?

Should we?
Any tool can be misused

Never blame the tool!
Compare for understanding, not judgement

More prescriptive

RUP (120+)

• Architectural frameworks
• Business use case model
• Business rules
• Business architecture document
• Business case
• Business glossary
• Business modeling guidelines
• Business object model
• Business roles
• Business use case

More adaptive

XP (13)

• Whole team
• Coding standard
• Test plan
• Collective ownership
• Customer tests
• Pair programming
• Refactoring
• Planning game
• Continuous integration
• Simple design
• Sustainable pace
• Pair coding
• Small releases

Scrum (9)

• Sprint Backlog
• Product Owner
• Team
• Sprint planning meeting
• Daily Scrum
• Sprint review
• Sprint planning
• Sprint backlog

Kanban (3)

• Visualization
• Team
• Visualize the workflow
• Lean WIP
• Measure and optimize lead time

Do Whatever (0)

• Do Whatever

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

Miyamoto Musashi
17th century samurai
Distinguish the tool itself from specific usage techniques

Specific patterns, techniques, "best practices", etc

Scrum core

Kanban core
Scrum prescribes 3 roles

Product owner

Team Scrum Master
Scrum prescribes timeboxed iterations

Scrum team

Kanban team 1

Kanban team 2

Kanban team 3

Henrik Kniberg
Both limit WIP, but in different ways

Scrum board

Kanban board

WIP limited per unit of time (iteration)

WIP limited per workflow state

Henrik Kniberg
Both are empirical

- Capacity (aka velocity)
- Lead time (aka cycle time)
- Quality (defect rate, etc)
- Predictability (SLA fulfillment, etc)

Kanban is more configurable

Great! More options!
Oh no, more decisions!

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 ...

Henrik Kniberg
Scrum discourages change in mid-iteration

Scrum

To do | Ongoing | Done :o)
-----|---------|---------
C     | A       |         
D     | B       |         

Kanban

To do | Ongoing | Done :o)
-----|---------|---------
2     | 2       |         
C     | A       |         
D     | B       |         

I’d like to have E!
Wait until next sprint!
Wait until a To Do slot becomes available! Or swap out C or D!
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 team
Kanban team 1

Cross-functional team

Kanban team 2

Specialist
Cross-functional team
Specialist team

Henrik Kniberg
Scrum backlog items must fit in a sprint

**Scrum**

Sprint 1 | Sprint 2 | Sprint 3 | Sprint 4
---|---|---|---

**Kanban**

WIP limit = 3

Long running task
Scrum prescribes estimation and velocity

Likely velocity: 8 per sprint
(sustainable pace?)
Both allow working on multiple products simultaneously.

Scrum example 1
Green Product
Green team
Yellow Product
Yellow team

Scrum example 2
All products
Cross-product team

Scrum example 3
All products
Cross-product team

Kanban example 1
Color-coded tasks

Kanban example 2
Color-coded swimlanes
Both are Lean and Agile

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 Consensus, Thoroughly Considering All Options; Implement Decisions Rapidly
14. Become a Learning Organization Through Relentless Reflection (Hansei) and Continuous Improvement (Kaizen)

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
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 maintenance 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.
Evolve your own unique board!

Some of these photos courtesy of David Anderson, Mattias Skarin, and various other people.
**Feature / story**

- Date when added to board
- Hard deadline (if applicable)

**Task / defect**

- = task
- = defect
- = completed
- = blocked
- = who is doing this right now

**What to pull first**

1. Panic features ★★★
   (should be swarmed and kept moving. Interrupt other work and break WIP limits as necessary)
2. Priority features ★
3. Hard deadline features (only if deadline is at risk)
4. Oldest features

**Definition of Done**

- **Analysis**
  - Customer accepted
  - Ready for production

- **Development**
  - Code clean & checked in on trunk
  - Integrated & regression tested
  - Running on UAT environment

- **Acceptance**
  - Customer accepted
  - Ready for production

**Next**

- 2

**Analysis**

- 3

**Development**

- 3

**Acceptance**

- 2

**Prod**

- 2

**Feature / story**

- Date when added to board
- Hard deadline (if applicable)

**Task / defect**

- = task
- = defect
- = completed
- = blocked
- = who is doing this right now

**What to pull first**

1. Panic features ★★★
   (should be swarmed and kept moving. Interrupt other work and break WIP limits as necessary)
2. Priority features ★
3. Hard deadline features (only if deadline is at risk)
4. Oldest features

**Definition of Done**

- **Analysis**
  - Customer accepted
  - Ready for production

- **Development**
  - Code clean & checked in on trunk
  - Integrated & regression tested
  - Running on UAT environment

- **Acceptance**
  - Customer accepted
  - Ready for production

**Next**

- 2

**Analysis**

- 3

**Development**

- 3

**Acceptance**

- 2

**Prod**

- 2
Comparison: Typical Scrum board & Kanban board

Henrik Kniberg
Scenario 1 – one piece flow

<table>
<thead>
<tr>
<th>Backlog</th>
<th>Next 2</th>
<th>Dev 3</th>
<th>In production :o)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>H</td>
<td>I</td>
<td>J</td>
</tr>
<tr>
<td>J</td>
<td>L</td>
<td>K</td>
<td></td>
</tr>
<tr>
<td>K</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Henrik Kniberg
Scenario 1 – one piece flow

<table>
<thead>
<tr>
<th>Backlog</th>
<th>Next</th>
<th>Dev</th>
<th>In production :o)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>Ongoing</td>
<td>Done</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>K</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Scenario 1 – one piece flow

<table>
<thead>
<tr>
<th>Backlog</th>
<th>Next 2</th>
<th>Dev 3</th>
<th>In production :o)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Ongoing</td>
<td>Done</td>
</tr>
<tr>
<td>G</td>
<td>C</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>D</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>J</td>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>K</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Henrik Kniberg
## Scenario 1 – one piece flow

<table>
<thead>
<tr>
<th>Backlog</th>
<th>Next</th>
<th>Dev</th>
<th>In production :o)</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td></td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>H</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J</td>
<td></td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>K</td>
<td></td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Scenario 1 – one piece flow

Henrik Kniberg
Scenario 1 – one piece flow.

<table>
<thead>
<tr>
<th>Backlog</th>
<th>Next</th>
<th>Dev</th>
<th>In production :o)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ongoing</td>
<td>Done</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>D</td>
<td>C</td>
<td>A</td>
</tr>
<tr>
<td>F</td>
<td>J</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>J</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>K</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Henrik Kniberg

37
Scenario 2 – Deployment problem

<table>
<thead>
<tr>
<th>Backlog</th>
<th>Next 2</th>
<th>Dev 3</th>
<th>In production :o)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>K</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Henrik Kniberg
### Scenario 2 – Deployment problem

<table>
<thead>
<tr>
<th>Backlog</th>
<th>Next</th>
<th>Dev</th>
<th>In production :o)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>K</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

PO

In production :o)
Scenario 2 – Deployment problem

Backlog  | Next  | Dev  | In production :o)
|--------|-------|------|------------------
| G      | C     | Ongoing | Done
| F      | D     |        |                 
| H      |       |        |                 
| I      |       |        |                 
| J      |       |        |                 
| L      |       |        |                 
| M      |       |        |                 
| K      |       |        |                 

POI
Scenario 2 – Deployment problem

[Diagram with labels and placeholders]

Henrik Kniberg
### Scenario 2 – Deployment problem

<table>
<thead>
<tr>
<th>Backlog</th>
<th>Next</th>
<th>Dev</th>
<th>In production :o)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>D</td>
<td>C</td>
<td>A</td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>J</td>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>K</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>!?</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

POI

In production: 😊

Scenario 2 – Deployment problem
# Scenario 2 – Deployment problem

## Diagram

<table>
<thead>
<tr>
<th>Backlog</th>
<th>Nexet 2</th>
<th>Dev 3</th>
<th>In production :o)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>K</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Backlog
- G
- F
- H
- I
- J
- L
- M
- K

### Nexet 2
- D
- E

### Dev 3
- A
- B
- C

### Status
- Ongoing
- Done

Henrik Kniberg
Scenario 2 – Deployment problem

![Diagram showing the scenario with tasks and people assigned to them.]

Henrik Kniberg
Scenario 2 – Deployment problem

<table>
<thead>
<tr>
<th>Backlog</th>
<th>Next</th>
<th>Dev</th>
<th>In production :o)</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>K</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PO</td>
<td></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Ongoing</td>
<td></td>
<td>Done</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Scenario 2 – Deployment problem

Backlog | Next 2 | Dev 3 | In production :o)
---|---|---|---
G | | | 
F | | | 
H | I | | 
J | L | | 
M | K | | 

Ongoing | Done
---|---
D | E | C | A
B

Henrik Kniberg
"One day in Kanban land"

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

<table>
<thead>
<tr>
<th>Scrum</th>
<th>Kanban</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Timeboxed iterations</strong> prescribed.</td>
<td><strong>Timeboxed iterations</strong> optional.</td>
</tr>
<tr>
<td><strong>Team commits</strong> to a specific amount of work for this iteration.</td>
<td><strong>Commitment</strong> optional.</td>
</tr>
<tr>
<td>Uses <strong>Velocity</strong> as default metric for planning and process improvement.</td>
<td>Uses <strong>Lead time</strong> as default metric for planning and process improvement.</td>
</tr>
<tr>
<td><strong>Cross-functional teams</strong> prescribed.</td>
<td><strong>Cross-functional teams</strong> optional. <strong>Specialist teams allowed.</strong></td>
</tr>
<tr>
<td><strong>Items broken down</strong> so they can be completed within 1 sprint.</td>
<td>No particular item size is prescribed.</td>
</tr>
<tr>
<td><strong>Burndown chart</strong> prescribed.</td>
<td>No particular type of diagram is prescribed</td>
</tr>
<tr>
<td>WIP limited indirectly (per sprint)</td>
<td>WIP limited directly (per workflow state)</td>
</tr>
<tr>
<td>Estimation prescribed</td>
<td>Estimation optional</td>
</tr>
<tr>
<td>Cannot add items to ongoing iteration.</td>
<td>Can add new items whenever capacity is available</td>
</tr>
<tr>
<td>A <strong>sprint backlog</strong> is owned by one specific team</td>
<td>A <strong>kanban board</strong> may be shared by multiple teams or individuals</td>
</tr>
<tr>
<td>Prescribes 3 roles (PO/SM/Team)</td>
<td>Doesn’t prescribe any roles</td>
</tr>
<tr>
<td>A <strong>Scrum board</strong> is reset between each sprint</td>
<td>A <strong>kanban board</strong> is persistent</td>
</tr>
<tr>
<td>Prescribes a prioritized product backlog</td>
<td>Prioritization is optional.</td>
</tr>
</tbody>
</table>
Don’t be dogmatic

Go away! Don’t talk to us! We’re in a Sprint.

Come back in 3 weeks.

Though Shalt Limit WIP

Beware of Dogma
Essential skills needed for both Kanban and Scrum

Splitting the system into deliverable increments

Software craftsmanship

Retrospectives

Root-cause analysis

Henrik Kniberg

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.