Kanban
A Lean approach to Agile software development

JFokus
January 26, 2010

Henrik Kniberg
Agile/Lean coach
www.crisp.se

henrik.kniberg@crisp.se
070 4925284
Goals of this tutorial

- Basic understanding of Kanban
  - Where it came from
  - What problem it tries to solve
  - Underlying Lean principles
  - How it looks like in practice
  - How to get started
Flow
Watch the baton, not the runner!
Cycle time

**Cycle time / lead time**
= The reliable, repeatable time from customer need until that need is satisfied (≈ time to market)

**Examples:**
- Time to get my hamburger
- Time to resolve support request
- Time to get home from work
- Time to deliver a feature

**Not to be confused with Iteration/Sprint length**
= release frequency / cadence length
Minimize cycle time

- Competitive advantage
- Detect defects earlier
- Less time for change to happen
Problem: Invisible baton

Managers who don’t know how to measure what they want settle for wanting what they can measure

Russel Ackoff

Henrik Kniberg
Optimizing resource utilization = suboptimizing cycle time

Henrik Kniberg

100% runner utilization => Lose the race

100% Road utilization => Long time to get home

100% Server utilization => Long response time
Queuing theory: Little’s law

Smaller batch size => faster cycle time

Little’s law

Total Cycle Time = \frac{\text{Number of Things in Process}}{\text{Average completion rate}}

Henrik Kniberg
Minimize batch size
Knowledge is perishable

1 fresh banana per day

<table>
<thead>
<tr>
<th>Monday</th>
<th>Tuesday</th>
<th>Wednesday</th>
<th>Thursday</th>
<th>Friday</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Banana" /></td>
<td><img src="image2.png" alt="Banana" /></td>
<td><img src="image3.png" alt="Banana" /></td>
<td><img src="image4.png" alt="Banana" /></td>
<td><img src="image5.png" alt="Banana" /></td>
</tr>
</tbody>
</table>

... or 5 rotten bananas per week?

<table>
<thead>
<tr>
<th>Monday</th>
<th>Tuesday</th>
<th>Wednesday</th>
<th>Thursday</th>
<th>Friday</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Banana" /></td>
<td><img src="image2.png" alt="Banana" /></td>
<td><img src="image3.png" alt="Banana" /></td>
<td><img src="image4.png" alt="Banana" /></td>
<td><img src="image5.png" alt="Rotten Banana" /></td>
</tr>
</tbody>
</table>
Little’s law in action

**Focusing**

- Short cycle time
- 1 task in progress
- Task A
  - Timeline
  - Total Cycle Time = \( \frac{\text{Number of Things in Process}}{\text{Average completion rate}} \)

**Multitasking**

- Long cycle time
- 3 tasks in progress
- A₁, A₂, A₃
- B₁, B₂, B₃
- C₁, C₂, C₃

Henrik Kniberg
Q: What causes multitasking?
A: Optimizing for max resource utilization
Value stream mapping

Game backlog

Sam 2d → Concept pres. 1m → Lisa assigns resources 6m → Graphics design 1d → Sound design 1m → Dev 3w → Design-ready games 6m → Integr. & deploy 3m → Production-ready games 3w

3 m value added time
25 m cycle time = 12% Process cycle efficiency

Cross-functional game team
3-4 m cycle time = 6-8x faster

Game team (graphics, sound, dev, integrate)
3-4 months

Henrik Kniberg
Optimize the whole

I’m fast!

6 months

Joe → Dave → Lisa

We’re slow!

Done!

3 months

We’re alot faster!

I’m a bit slower

Done!

Henrik Kniberg
Problem: Invisible baton

Team 1

Team 2

Managers who don’t know how to measure what they want settle for wanting what they can measure

Russel Ackoff

Henrik Kniberg
It’s hard to optimize something that you can’t see
First step: Visualize the baton!

Team 1

Todo | Doing | Done this week
--- | --- | ---

Avg lead time: **20** days

Team 2

Todo | Doing | Done this week
--- | --- | ---

Avg lead time: **3** days

Henrik Kniberg
Kaizen
Both are empirical

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

Kanban is HIGHLY configurable

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

Great! More options! Oh no, more decisions!

Henrik Kniberg
Almost always a good idea

- Visualize the workflow
- Limit WIP (work in process)
- Short feedback loops
- Focus on quality
- Cadences
Pull
Limit work to capacity

Input: 3 customers / hour

Output capacity: 2 customers / hour
Pull vs Push

Assigning tasks = push
Taking tasks = pull

Push

Pull

Henrik Kniberg
Example: Supermarket pull
Example: Feature pull

You are currently working in Flaky Connection Mode

Adding attachments while in Offline or Flaky mode is currently not supported.

We know this is totally lame. But don’t forget, Gmail didn’t have basic stuff like drafts when it launched. We’ll keep working on it.

OK Leave Flaky Connection Mode
Why feature pull is important

Features and functions used in a typical system:

Half of the features we build are never used!

Always 7%

Often 13%

Sometimes 16%

Rarely 19%

Never 45%

Source: Standish Group Study Reported at XP2002 by Jim Johnson, Chairman
Kanban
Kanban

- Signaling system
- Visual
- Limited in supply

"Visual Card"
Kanban @ Imperial Palace Gardens
Kanban in your wallet

Darn. Forgot to limit.
The tool used to operate the [Toyota Production] system is kanban.
Kanban in SW development

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

**FLOW**

Avg lead time: **12 days**

Pioneered by David Anderson in 2004
Evolve your own board!

Some of these photos courtesy of David Anderson, Mattias Skarin, and various other people
One day in Kanban land
"One day in Kanban land"

http://blog.crisp.se/henrikkniberg/tags/kanban/
## 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>A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>3</td>
<td>Ongoing</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td>Done</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>H</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>J</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>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>
<tr>
<td>N</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z</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 2</th>
<th>Dev 3</th>
<th>In production :o)</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</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>M</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>K</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Scenario 1 – one piece flow

Henrik Kniberg

In production :o)
Scenario 1 – one piece flow

![Diagram](image)

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></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>B</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>E</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **G**
- **F**
- **H**
- **I**
- **J**
- **L**
- **M**
- **K**
- **E**
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>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>
</tbody>
</table>

Henrik Kniberg
Scenario 2 – Deployment problem

Backlog

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
</table>

Next

| 2 |

Dev

| 3 |

In production :o)

Ongoing | Done

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></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</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>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>E</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Diagram showing a process with tasks and states]
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>G</td>
<td>C</td>
<td>Ongoing</td>
<td></td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>Done</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

POI
Scenario 2 – Deployment problem

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

Henrik Kniberg
Scenario 2 – Deployment problem

Backlog | Next | Dev | In production :o)
---|---|---|---
| | 2 | 3 | |

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

- D

- Ongoing

- C
- !?

- Done

- A
- B

Henrik Kniberg
# Scenario 2 – Deployment problem

## Diagram

<table>
<thead>
<tr>
<th>Backlog</th>
<th>Nexet</th>
<th>Dev</th>
<th>In production :o)</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="POI" /></td>
<td><img src="image" alt="G" /></td>
<td><img src="image" alt="D" /></td>
<td><img src="image" alt="A" /></td>
</tr>
<tr>
<td><img src="image" alt="F" /></td>
<td><img src="image" alt="E" /></td>
<td><img src="image" alt="B" /></td>
<td><img src="image" alt="C" /></td>
</tr>
<tr>
<td><img src="image" alt="H" /></td>
<td><img src="image" alt="I" /></td>
<td><img src="image" alt="J" /></td>
<td><img src="image" alt="L" /></td>
</tr>
<tr>
<td><img src="image" alt="M" /></td>
<td><img src="image" alt="K" /></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Backlog** (POI)
- **Nexet** (2)
- **Dev** (3)
- **In production :o)**
Scenario 2 – Deployment problem

![Diagram]

Henrik Kniberg
Scenario 2 – Deployment problem

Backlog | Next 2 | Dev 3 | In production :o)
---|---|---|---
G | | | A
F | D | E | B
H | I | | |
J | L | | |
M | K | | |
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></td>
<td></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>PO</td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ongoing</td>
<td>Done</td>
<td></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>

Henrik Kniberg
Bottlenecks
Theory of constraints – Smooth Flow

Goal

Reality

Strategy

1. Reduce intake
2. Fix bottleneck
3. Increase intake
4. Fix next
Theory of constraints

- All systems have a bottleneck
- When you fix it, it jumps to somewhere else.
- The whole system performs only as well as its bottleneck
- Ensure you have slack before & after the bottleneck
  - Happens automatically with pull scheduling
- Use the slack to fix the bottleneck
How to find your bottleneck

- Busy at bottleneck
- Slack downstream
- Queue upstream

Without WIP limits

With WIP limits

Henrik Kniberg
Goal: “T-shaped” people

<table>
<thead>
<tr>
<th></th>
<th>Test</th>
<th>DB</th>
<th>WEB</th>
<th>JAVA</th>
<th>Domain</th>
<th>CM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lisa</td>
<td></td>
<td></td>
<td></td>
<td>⭐</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Joe</td>
<td></td>
<td>⭐</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fred</td>
<td></td>
<td></td>
<td>⭐</td>
<td>⭐</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jenny</td>
<td></td>
<td></td>
<td>⭐</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>David</td>
<td>⭐</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Erik</td>
<td>⭐</td>
<td></td>
<td>⭐</td>
<td>⭐</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

I'm good at Java!

I can test, but I'm not so good at it.

I don't know CM at all. But I'm willing to learn!
Kanban allows both specialists & generalists
Cumulative Flow Diagram (CFD)

Day 4

<table>
<thead>
<tr>
<th>Backlog</th>
<th>Dev</th>
<th>Test</th>
<th>Prod</th>
</tr>
</thead>
</table>

# of items in each column

- 6 days Lead time
- 9 items WIP

Henrik Kniberg
Cadence
Daily standup meeting

Source: Visual management blog
Cadence

Scrum team

Kanban team 1

Kanban team 2

Kanban team 3

Henrik Kniberg
Reduce transaction costs

We need to do big batches because transaction cost is high

We need to minimize transaction cost so we can do smaller batches

Automate init of test environment
Implement hot-reload on server

Economies of Scale
"We need to do big batches because transaction cost is high"

Economies of Flow (Lean)
"We need to minimize transaction cost so we can do smaller batches"
Quality
Quality

Defects per unit of valued work

Value demand

Failure demand

Henrik Kniberg
Lean principle: Stop the Line

The build server is your friend! Really!

Henrik Kniberg
Optimizing the WIP limit
Optimizing the WIP limit

**Too low WIP limit**

- People often idle
- Zzzzzzzzz
- People never idle
- Tasks rarely idle
- Slow flow (end-to-end)

**Just Right WIP limit**

- Tasks often idle
- People sometimes idle (slack)
- Fast flow

**Too high WIP limit**

- People never idle
- Tasks often idle
- Lack of wall space...
Use WIP limits to break vicious cycles

Henrik Kniberg
Evolution of a Kanban board
**Feature / story**
- Date when added to board
- Hard deadline (if applicable)
- Description
- Priority / panic
- Who is analyzing / testing right now

**Task / defect**
- Description
- Task =completed
- Defect
- Why = blocked
- Who is doing this right now

**What to pull first**
- Panic features
- Priority features
- Hard deadline features
- Oldest features

**Definition of Done**
- Next
  - Ongoing
  - Done
- Analysis
  - Ongoing
  - Done
- Development
  - Ongoing
  - Done
- Acceptance
  - Ongoing
  - Done
- Prod
  - Ongoing
  - Done

**Kanban kick-start example**
Henrik Kniberg
version 1.2
www.crisp.se/kanban/example
2009-11-16
Estimation & planning
Estimating is optional in Kanban

**Scrum**

- **Velocity = 11**
  - Sprint 1: 4
  - Sprint 2: 2, 1, 4
  - Sprint 3
  - Sprint 4

**Kanban**

- **WIP limit = 3**
  - Long running task
  - Sprint 1
  - Sprint 2
  - Sprint 3
  - Sprint 4


**Lightweight estimation techniques**

**T-shirt sizing**

- Small (S): Hours?
- Medium (M): Days?
- Large (L): Weeks?

**SLA – service level agreement**

Example: SLA is 12 days
- Can we do this item within 12 days?
- If yes: just pull it in.
- If no: reject it, or break it down, or estimate/negotiate
Example: estimation paradox

Estimation is sometimes used to manage a problem that was caused by the estimation itself.

- **8 days work**
- **5 months cycle time**

\[
\text{Process cycle efficiency} = \frac{8 \text{ days work}}{5 \text{ months cycle time}} = 9\%
\]

- **Prestudies, estimation, resource allocation, approval**
- **Develop, test, deploy**

- **5 month cycle time** → Scared of commitment
- **2 week cycle time** → Little fear of commitment
- **Lots of estimation & prestudy** → WIP limits
- **Little or no estimation & prestudy**

Henrik Kniberg
Using Kanban to visualize the problem
Metrics
Metrics

- Cycle time: Medium-sized MMFs take < 8 days, 95% of the time.
- Item #25 took 7 days.
- Average capacity during Q1 was 6 story points per week.
- If we lower WIP limit from 6 to 2 in test we can probably halve the cycle time!
- Last week we released 8 things.

Quality:
- Undecided: 25%
- Value demand: 15%
- Failure demand: 60%

Value demand:

Cumulative Flow Diagram

- Backlog
- Dev
- Test
- Production

Release capacity (velocity)

- Average capacity during Q1 was 6 story points per week.

Analysis Dev Test Deploy

WIP (work in progress)
Misc patterns & Tips & Tricks
Process = collection of policies

- Visible on the board
- Brief
- Continuously renegotiated as needed
- Created on-demand

Examples:
- Definition of Done
- What to do when WIP limit is exceeded
- Who decides priorities
- When do we do planning?
- Which types of work items to we pull first
- Service level agreements
2 tier board with feature swimlanes
## 2 tier board-within-board

![Image of a board with sticky notes and tasks]

**PROJECT: ALCHEMY 2**

**ACTIVE**
- Development 5
  - SLA: 5 days

**PROPOSED**
- Described
  - Close
  - Slowed

**RESOLVED**
- Unresolved
- Closed

**SLA:**
- 5 days

**M1:**
- ???

Source: David Anderson
Stopping dead projects
Manager removing impediments

- 2 slots on manager’s door
- If both are full, team can add a new one if they remove a less important one
- Team decides when issue is solved

Source: Mattias Skarin
Bootstrapping Kanban
Typical Scrum => Kanban evolution

Step 1

Feature team 1
Scrum

Feature team 2
Scrum

Feature team 2
Scrum

Step 2

Feature team 1
Scrum

Feature team 2
Scrum

Feature team 2
Scrum

Step 3

Feature team 1
Scrum

Feature team 2
Scrum

Feature team 2
Scrum

Operations / support team
Scrum

Operations / support team
Kanban

Henrik Kniberg
Example: Kanbanizing a Scrum team

<table>
<thead>
<tr>
<th>Before sprint</th>
<th>Sprint</th>
<th>After sprint</th>
</tr>
</thead>
<tbody>
<tr>
<td>TFS: checked in</td>
<td>TFS: in progress</td>
<td>TFS: done</td>
</tr>
<tr>
<td>Sprint backlog</td>
<td>In progress</td>
<td>CQA testing</td>
</tr>
<tr>
<td>Q 25</td>
<td>Ready for QA</td>
<td>Ready for QA</td>
</tr>
<tr>
<td>Prod backlog</td>
<td>Testing</td>
<td>QA testing</td>
</tr>
<tr>
<td>Q 10</td>
<td>Q 25</td>
<td>Q 25</td>
</tr>
<tr>
<td>Q 20</td>
<td>Deployed to customer</td>
<td>Q 20</td>
</tr>
</tbody>
</table>

![Kanban board example](image)
Kanban bootstrapping

1. Define your value stream
   • Which part do you intend to control?
   • Where is your input & output?

2. Define work item types
   Task, Bug

3. Meet with upstream & downstream stakeholders
   • Policies, WIP limits
   • Input coordination
   • Output coordination
   • Target cycle time (SLA – service level agreement)

4. Create Kanban board
   (+ optionally electronic system)

5. Agree on time for daily standup

6. Agree on when to do retrospectives / operations reviews

7. Go! Iteratively improve.

Henrik Kniberg
Wrapup
Recommendations

- Visualize existing process first, then optimize.
- Put the board in the team room
- Focus on exposing problems
- Keep the board clean & simple (just like your code...)
- Don’t try to get it perfect from start. Let it evolve.
- Don’t overuse metrics
- Involve the team & managers & stakeholders
- Take photos
- Consider getting a coach for Kanban bootstrapping

Henrik Kniberg
The Lean & Agile toolkit

Values & Principles
Lean, Agile, Theory of Constraints, Systems Thinking, etc

Kanban
Scrum
XP

Other lean tools
(Value Stream Mapping, Root Cause Analysis, etc)
Perfection is a direction, not a place

Henrik Kniberg
Expand your toolkit!
www.crisp.se/utbildning

Kanban starting points on the web:
http://www.crisp.se/kanban
http://www.limitedwipsociety.com

Leading Lean Software Development
March 4-5

Tom & Mary Poppendieck & Henrik Kniberg

Certified ScrumMaster & Kanban
Feb 17-19

Kanban Applied
Feb 23

Henrik Kniberg & Mattias Skarin

Kanban coaching workshop
April 29-31

David Anderson