Kanban - Crossing the line, pushing the limit or rediscovering the agile vision?

Jesper Boeg, Agile Coach, Developer
jbo@trifork.com

March 11, 2010

In general

- Feel free to ask questions
  - I much prefer an enthusiastic discussion over missing a few slides
  - Might also keep you from taking your lunch time nap ☺
- I am not a PowerPoint black belt so please bear with my less than fancy slides

Agenda

- Kanban origins
- What is software Kanban?
- How is software Kanban different from other agile methods and which problems might it help us solve?
- Disadvantages
- Software Kanban and team maturity
- Last notes

KANBAN IN MANUFACTORING

TRIFORK.

Kanban in manufactoring

- Kanban is Japanese and means “visual card,” “signboard,” or “billboard.”
- Used to limit the amount of inventory tied up in “work in progress” in Lean manufacturing
- Excess inventory is regarded as waste and so is the time spent producing it
- Kanban cards act as a “work permit” representing how much WIP is allowed in a system.
- Typically a color coded plastic card

A simple example of a Kanban pull system

- New paper is ordered when the limit prescribed by the kanban is reached
- When paper arrives the kanban is returned along with the paper

KANBAN IN SOFTWARE

TRIFORK.

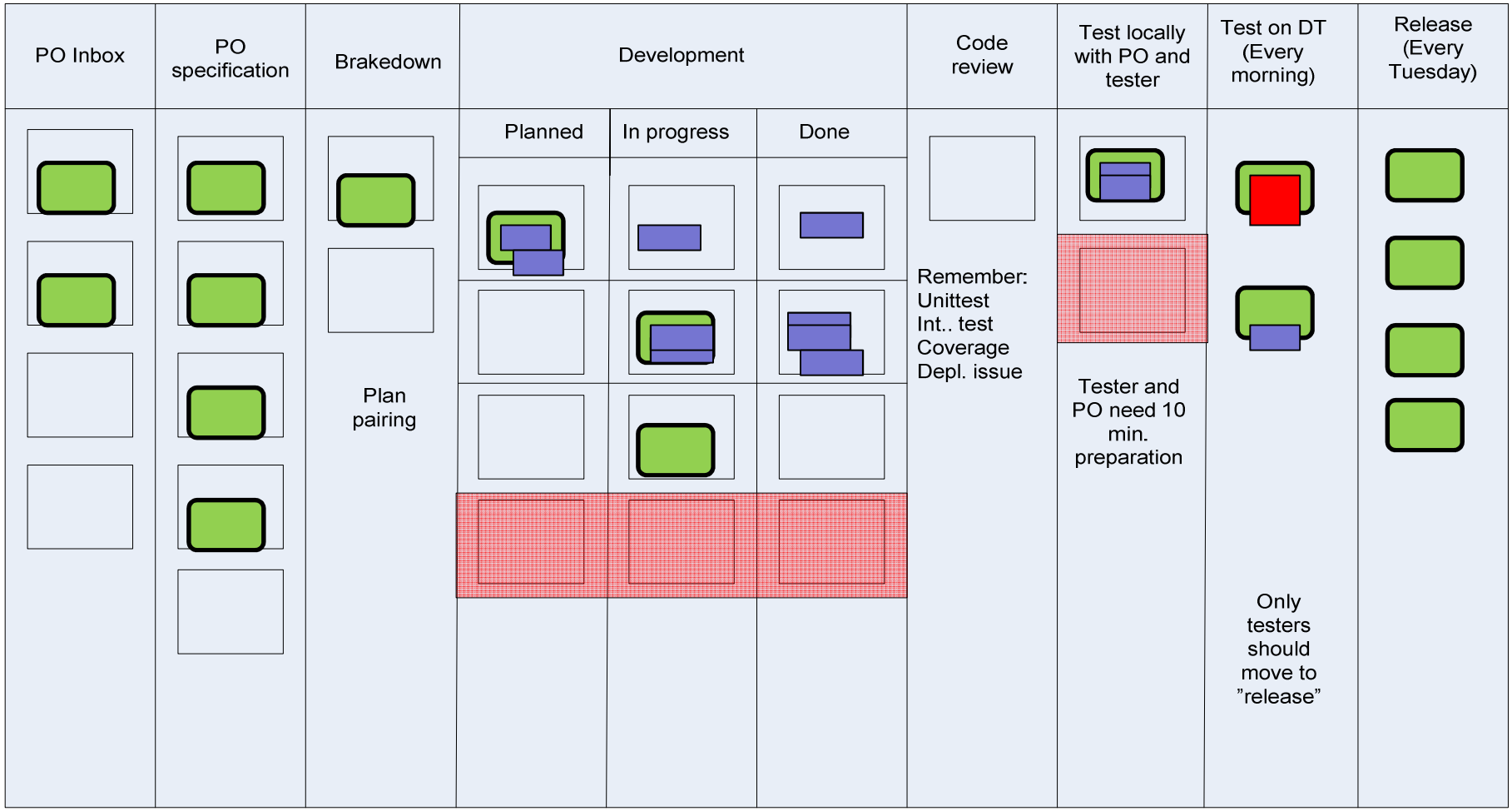
Software Kanban uses a broader Lean perspective

- Limit work in progress.
  - Focus on flow
  - Deliver often
- Focus on quality
  - stop the line
- Balance demand and throughput
  - Getting people home at night
  - Finding the right bottleneck
  - Having free time on your hands
  - Optimizing the whole
- Continuous improvement
  - Keep getting better
- Prioritize
  - Focus on business value and minimal marketable feature set

To achieve this

- Start by mapping the value stream and track work on a white board

Define WIP limits for each stage



Pick the low hanging fruits

- You will be surprised how much you can achieve by
  - Limiting work in progress.
  - Balancing demand and throughput

How does that fit with current Agile best practices?

- You can do fixed iterations or not
  - As long as you deliver often
- You can use iteration retrospectives or not
  - As long as you focus on continuous improvement
- You can use estimation or not
  - As long as you are able to do necessary planning
- You can leave out iteration retrospectives
  - If you replace them with spontaneous quality circles or a better way to continuously improve

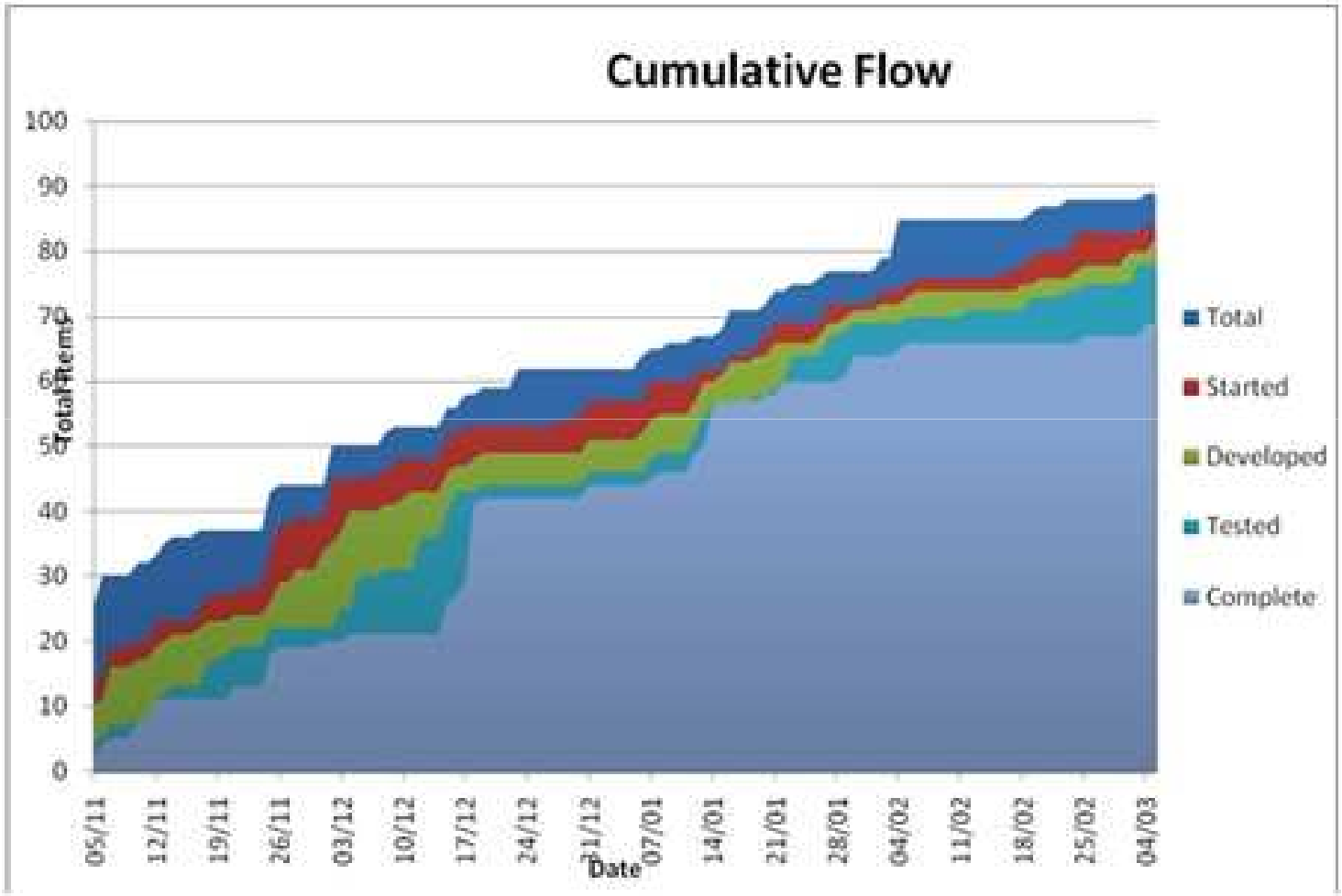
But that does not mean

- It is illegal to do iterations
  - If doing iterations will increase flow
- It is illegal to estimate
  - If estimation provides valuable information to stakeholders and motivates developers
- It is not possible to do release planning
  - Release planning can be done on other metrics e.g. cycletime or average number of items completed
- You are not focusing on improving the way you work

Typical measurements

- Cycle time
  - Measured from when you started working on it
- Lead time
  - Measured form when the customer ordered
- Quality
  - Time spend bugfixing per iteration
- WIP
  - Average number of “stories” in progress
- Throughput
  - Number of “stories” completed per iteration (when using fixed iterations)

Use Cumulative Flow diagrams



http://leanandkanban.files.wordpress.com/2009/04/cfd-example.jpg

Focusing on value sets instead of practices

- Using Kanban focus is no longer on specific practices
  - Choose practices that will help you use resources at hand most effectively in your context
- You might end up doing Scrum ☺
  - If Scrum practices are the perfect way to limit WIP, build quality in, level throughput and demand and prioritize according to business value in your context

But that is not my practice!!

David Anderson:

“I don’t care about your practices”

- Keep your eye on the ball
  - We are hopefully using best practices because we believe they help us deliver business value to our customers – not because somebody told us to
- Once practices become faith based and cargo cult we risk loosing sight of the goal
  - Remember Alistair Cockburn’s: Shu, Ha, Ri

SO HOW DOES THIS MAKE A DIFFERENCE?

Traditional agile methods have challenges

- Development items small enough to fit a 2 week iteration are often too small to deliver real business value and obtain real feedback
- Traditional iterations have consequences:
  - Requirements may suffer as product owners rush to prepare for upcoming cycles
  - Development may suffer when busy product owners, testers and users are unable to inspect software or answer questions during development
  - Functional quality may suffer as testers race to complete work late in the development time-box
  - Code quality may suffer when developers prioritize finishing a set of features over refactoring, TDD and pair programming

Keeping a sustainable pace

- Sustainable pace is a core value in agile – tech wise and people wise
  - But many “agile” projects exhibit anything but sustainable pace
  - Both in terms of stressed out people and a low quality code base

Accept that most traditional agile methods are feature driven and therefore require more measures than delivering working software to keep a sustainable code base

We need to allow more than one cadence

David Anderson: *“Concept that input cadence, output cadence and cycle time should be synchronous e.g. 2 week iteration, will be seen as edge case 5 years from now”*

- I don’t know if that will be true but it does seem reasonable to decouple prioritization, delivery and cycle time to wary naturally according to the context and transaction costs
  - Actually one of the main reasons kanbans are used in manufacturing

Why do we readily accept agile overhead?

- Stopping the development team for 1-2 days to do sprint planning
- Low quality feedback because functionality is to small to provide business value
- Stressing the real bottleneck/constraint by protecting the development team from external interruptions
- …….

Immediate results

- More pair programming
- Better functional quality
- Better code coverage
- More refactoring
- Closer collaboration and Team feeling across teams
- Immediate focus on the “real” bottleneck
  - which turned out to be PO specification

Rediscovering the Agile vision?

- It actually kind of felt that way. Back to the basics of
  - Flow
  - Feedback
  - Quality built in
  - Close communication and collaboration across the entire value chain
  - Continuous improvement
- Valuing people over processes and tools
  - That goes for Agile processes and tools as well
- Though for a moment I must admit I did feel quite lost without my Scrum safety blanket ☺

Kanban is not the only way

- I am 100 percent sure you can find ways to achieve similar results using traditional agile methods
  - But it might take you longer to get there
  - So keep an open mind

BUT THERE ARE NO FREE MEALS

Difficulties

- It has become increasingly hard to protect the team from all sorts of interruptions
  - A hard deadline is easy for everyone to understand
  - Both within the team and people outside the project
- We have to spend more time discussing plans and long term goal
  - Since people are no longer as focused on the short term goal

Difficulties

- We are using considerably more time explaining why we are doing things the way we are to management
  - Who for the most part had bought the Scrum silver bullet
- We have experienced that people react very differently to the new structure
  - Some find it very hard to stay focused while others are taking on more responsibility and are becoming true craftsmen
- What I still consider good “Scrum habits” have to be reinforced
  - Daily standup, division of responsibility (PO/team)

Difficulties

- Reprioritizing flexibility escalated to the point where the PO would try to reprioritize work in progress
- Problem with understanding that though I helped you out this time, it does not automatically become my responsibility
- New people on the team using longer to get adjusted to the way we work

Difficulties

- Many more will probably come since we have yet to see the long term effect

GETTING STARTED: 2 WAYS OF LOOKING AT KANBAN AND TEAM MATURITY

Kanban requires high team maturity

- Since Kanban is based on Lean value sets and Agile principles it requires high maturity to adopt the right practices
  - Requires a large toolbox
  - Ability to distinguish between practices that are effective but difficult to implement and practices that does not fit the context
  - Ability to focus on the individual “story” and avoid unnecessary interruptions
  - Use the added flexibility to find practices that deliver more business value faster – not to compensate for poor requirements and failed iterations

Kanban is a good way to start

- Since Kanban does not include specific practices you can start with your current process and improve it one step at a time
  - Visualize your current value chain and remove one bottleneck at a time
  - Implement one practice at a time and gradually improve your process

The jury is still out on that one

- At least for me personally
- But I think I am leaning towards high maturity

SOME LAST NOTES

Kanban is just a process

- Sometimes one process will work better than another and sometimes they will be equally good.
  - Understand your problem before trying to solve it.
  - Expand your toolkit.
  - My tool is better than yours attitude won’t get you anywhere
  - Compare processes to understand them not for judgment.

Kanban is just a process

- You NEED good practices
  - Agile product management principles do not work well without good practices to support them
  - Quality built in is not just well tested. It is also good architecture and good coding practices
- If you haven‘t got the technical practices in place it doesn’t matter what process you are using,
  - It won’t get you anywhere in the long run.
  - But a good process will help you focus on having good technical practices.

Look at the entire value stream

- Start by acknowledging that development is not always the bottleneck
- In cases where this is true you would rather want developers doing nothing than stressing the real bottleneck further
  - Ideally developers are of cause helping relieving the real bottleneck
- In traditional Agile methods, development is almost by definition regarded as the bottleneck
  - Keeps you from exposing the real bottleneck
  - Keeps you from taking the right actions do improve your process
  - It took a switch from Scrum to Kanban for us to realice this

Kanban is “Leaner” than traditional Agile Methods

- Lean thinking done right can provide you with a wealth of opportunities for improvement
  - Exposing bottlenecks, visualizing flow, optimizing the whole…..
- Even Toyota forgot the fundamentals - everyone gets caught up in the new sexy stuff and technology
- But remember to distinguish between Lean manufacturing and Lean product development
  - You cannot eliminate variability without eliminating value added in LPD
  - Cost of delay in manufacturing is often the same

Look at your process from a true Lean perspective

- Don’t try to make a process seem Lean just because it’s a popular word
- A team pulling items from a backlog does not make it a pull system
  - It only means that you have a pull mechanism within your system
  - It doesn’t keep you from delivering more functionality than the customer needs or is able to adopt.
- A true pull system is based on the entire value stream and making sure it is closely aligned with the needs and capabilities of the customer
  - A software Kanban system should represent such value stream since balancing throughput and demand is a core value

Ask yourself

- Are you environment driven or environment driving?
  - Methods
  - Organization
  - People
  - Technology
- That could very well be your biggest impediment since it stops continuous improvement

Keep time-boxed product and process inspection

- Keep regular time-boxes in your process as a cue for **product** inspection:
  - Evaluate the quality of the growing product from a functional, engineering, and user experience perspective
- Evaluate your **pace** of development:
  - Look at the number of development items completed relative to goals
  - Look at the average cycle time per development item
  - Adjust your development plan as necessary
- Evaluate and adjust the **process** you're using
  - Use a process reflection session to identify changes you could make to improve your product or pace

Ending cycles right: http://www.stickyminds.com/s.asp?F=S14865_COL_2

QUESTIONS?