# Kanban kick-start (v2)

By Tomas Björkholm at Crisp, October 2011

## Introduction

For everyone who asks the question, how do we get started with Kanban. Here comes one way of kick starting a Kanban team. Let's first describe what a kanban system is. Kanban is Japanese and means signal card and is the driver behind Toyota's just-in-time concept. These signal cards are used to limit the number of parts ordered from the manufacturer. Every part has a kanban attached to it, which is released when the part is consumed. Toyota orders new parts by sending the physical, just released, cards back to the manufacturer. By limiting the number of kanbans they limit the number of parts. The idea of just-in-time is to have minimum or no stocks of parts. Actually, the word Lean comes from the fact that Toyota made cars with minimum parts in stock.
If you are not familiar with car manufacturing you may still have been involved in a kanban system, even though it wasn't as visual as Toyota's. You have probably waited outside a nightclub where the bouncer only lets a person in when someone comes out. He or she is behaving as if there were a kanban system in place limiting the number of people inside.

The idea of limiting parts, people or why not work in progress inspired David J Anderson to come up with the idea of the Kanban method.

The method Kanban has five main rules:

1. Visualize your workflow. When you visualize, it's easier to see your bottlenecks and what you see is what you can fix.

2. Limit the things you work on. This is to make sure you focus on a few things at a time. The rule of thumb is to focus on finishing things instead of starting. It also helps you stop queues from forming within your system.
3. Measure and manage flow and optimize on cycle time, that is the time it takes from when you start working on something until it's done, released and you start earning money from it
4. Make process policies explicit. Find out, write down and make sure everyone follows good behaviour that make your process flow with optimal speed.
5. Use models to recognize improvement opportunities. Use your knowledge from areas like Lean, queuing theory and "Theory of constraints" to improve your flow.

Those are the five rules, it's up to you to decide all the other rules that are needed to get your Kanban team to work. That's because your organisation is special and not like anything else. In this paper I present an approach and some questions to ask yourself. My hope is that the answers will give you a good start and that my suggestions inspire you to find what works for your organization. I would like to point out that Kanban is meant to be an evolution and not a revolution. This is to minimize resistance from the organisation.

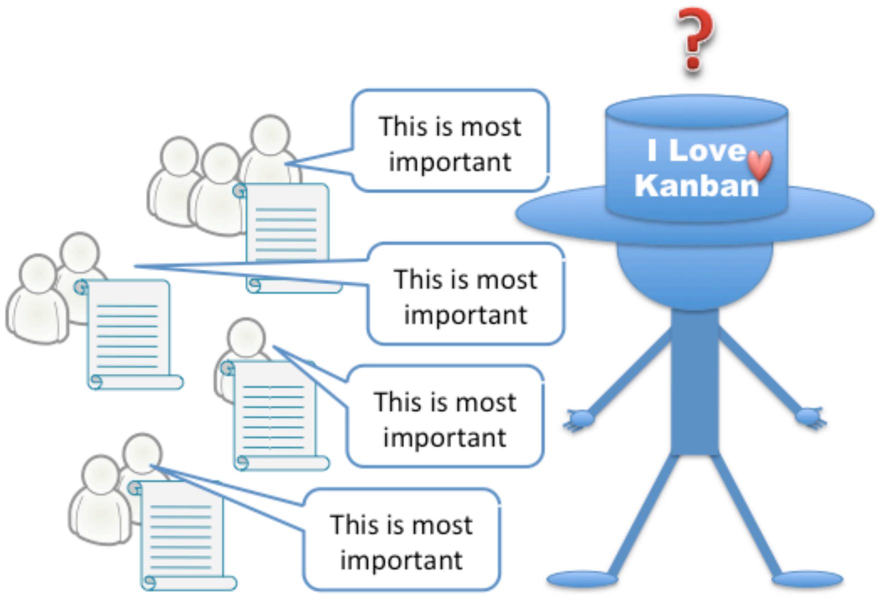
Kanban works well with the Lean culture and one of the best descriptions of Lean for software development I have seen comes from Mary Poppendieck. She said that Lean is to "Deliver continually increasing customer value - In the shortest possible timeframe - Expending continually decreasing effort - With the highest possible quality". Lean is not something you just implement and then you are done. It's a never-ending continuous improvement. More about that later.

## An approach to get started with Kanban

### Step 1 – Get to know your system

With "system" I mean your process, your organisation, your tools, your way of communicating, well, actually everything that happens from the point where you get a request from a customer until you have fulfilled the request. That is your system. To make sure your Kanban system fulfils its purpose, you should look through the eyes of your customers and try to understand what your customers expect from you. How, why and with what frequency does the customer give you things to do and is this the way your customers want it to work? What makes him/her happy? Now that you know what your customer wants, it's time to optimize your system to become better at fulfilling his/her needs.

## Step 2 – Identify your sources and prioritize



Do your tasks come from more than one source? If you work with software sold in many countries or have many users who request adoptions from you, then you probably have more than one source. The different sources can be different customers or country managers. Who prioritizes between them? The same applies when your team works with both support and development, then you need to decide what to do first. Here are some different examples of how your decision rules could look like:

1. You have one owner for all queues who decides the prioritization independent of the source queue. This is similar to Scrum's product owner.
2. You have one owner for each queue and those owners decide the prioritization together. It is important that prioritization disputes are handled outside the team. Teams are not more productive when they have stakeholders putting pressure on them to work their way through the queue.
3. You have one owner for each queue and a predefined percentage of how much work to take from each queue. You also need to decide the time frame in which the percentage should be measured. If one stakeholder is promised 50% of your capacity you need to make sure at the end of the year that's what he gets even if not 50% of the resources are always working for him. You could solve this by taking every other job from that stakeholder's queue.
4. You use an algorithm to decide which task to take next. The rule can be to prioritize according to severity or by cost of delay. Read more about this in my paper "Agile support with Kanban".
http://blog.crisp.se/tomasbjorkholm/2010/02/17/1266404160000.html

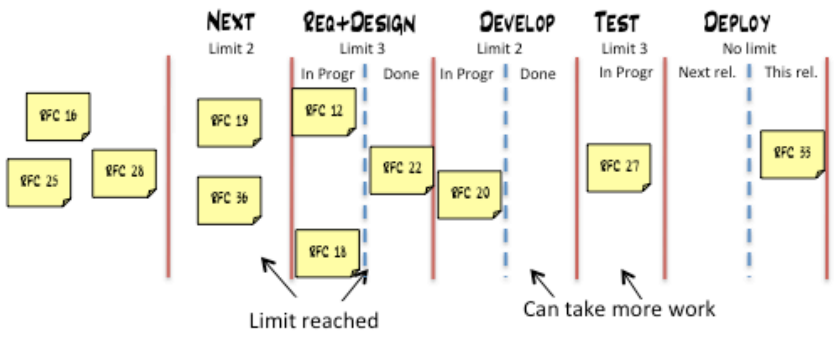
## Step 3 – Find your process

Before you design your workflow board you need to figure out your process. This is because the purpose of the board is to visualize your process. So, what is your process? Imagine that your tasks are batons and try to figure out who will carry them and what work will be done as they make their way from idea to realised feature. Be honest and map how the process really

looks like, not how you want it to look like. Hopefully your reason for implementing Kanban is to improve not to look good. Back to mapping your process. For example your process might start with a product manager who prioritizes the work, even if the work comes from different sources. After that a requirements analyst prepares the task for a developer. After development a tester runs the tests before the delivery team deploys.
Do all your tasks follow this flow? Maybe you have separate flows for different kinds of tasks.

## Step 4 – Design your workflow board

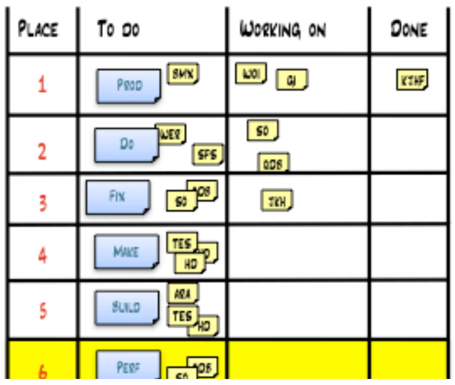
When you know your process, it's time to design your board. It should visualize your process so include all the necessary steps. In the example above, that means one column each for analyse (requirement+design), develop, test and deploy. Do you need one column for "Next", a column for the product manager to put the next things to work on? It depends on how available the product manager is. If he/she can react quickly when new work is needed, then the need for a "Next" column is little. There is also no need for a "Next" column when selection is made using an algorithm.



You can find more examples of different Kanban boards on Mattias Skarin's blog.
http://blog.crisp.se/mattiasskarin/2010/12/03/1291361993216.html

## Step 5 – Set the limits

Once you have your columns, it's time to set your limits. I'm not aware of a good rule of thumb to suggest here so I can only recommend that you try and improve. When you have a good limit you get good cycle time without too much idle time. A large part of Lean is the art of finding balance. In this case it's the balance between cost of delay and cost of idle labour.



Whatever your limit is, I like to think of the rows at the bottom of the board as warning signals. If you have a limit of 7 then filling places 6 and 7 should trigger a discussion so that you make sure you don't exceed your limit. I like this way of limiting since it's better to have this discussion when there are still options. If you need to take urgent jobs you can have special emergency tracks to easily bypass your normal flow.

For example, one track can have the policy: Start working with this when you have a natural break. That means very soon but not now. Another emergency track can have the policy: drop everything else you're doing, now.

## Step 6 – Decide the roles

Do you need roles? Kanban does not prescribe any roles. It's up to every company and team to decide. Kanban however does prescribe minimizing the cycle time, so if adding a role helps minimize the cycle time then it's good to add that role. If it makes the process slower, then the role should not be there. If the cost of the role is higher than the value of improved cycle time, then it's unnecessary overhead.

## Step 7 – Decide your meetings

Kanban does not prescribe any meetings but I recommend having some anyway. My recommendations are close to what Scrum suggests.

### Planning / story start meeting

Since there are no iterations you don't need to plan for a whole period so it's up to you to decide how often you need to plan. You can meet regularly, like once a week, or just in time whenever more work is needed. I like something I call a story start meeting. It's a meeting where you bring all the people you need to clarify the requirements and the technical conditions. The meeting is for the team to ask question, discuss and gain a good understanding of what to do and how to do it. This meeting usually builds up a good amount of positive energy, which makes the development fun and quick.

### Daily sync

Just like the daily Scrum it's a good idea to have a daily, short meeting to synchronize. But since the work is visible on the board there is little need for the questions, what did you do yesterday and what do you plan for today. Instead you can concentrate on the flow and discuss and solve things that are stopping the flow. Good questions are:

1. Is the board updated? If not postpone the meeting 5 minutes to get it updated.
2. Is something stopping the flow and how can we fix it as a team?
3. Do we all have something to do? If not, bring in new tasks. After the meeting the team can analyse how to best develop or solve the new task(s) preferably in a story start meeting.

### Retrospective – the improvement meeting

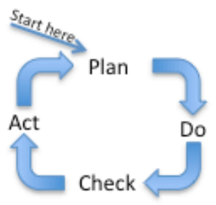
Kanban is Lean and Lean has a great focus on continuous improvement. In Scrum, this meeting is called a retrospective so why not use the same term in Kanban. Even though you can hold a retrospective whenever it feels necessary my recommendation is to have it on a periodic predefined basis. Just to make sure you don't forget it when you are stressed. The purpose of the retrospective is to reflect on things that make us less productive and solve those issues. Even though it's good to have regular retrospectives, the improvement you do immediately when a problem occurs is just as important. Don't wait for the next retrospective, fix the problem now.

### Plan – Do – Check – Act

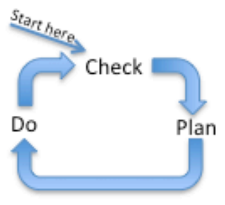
Edwards Deming is one of the influences of Lean. His most famous contribution is the Deming cycle, which helps you improve your work. The idea is that you take a problem, make a plan, execute your plan, check the result and act (decide the next step based on the result). The next step might be to improve the solution for the original problem or to focus on solving another problem. Once you have decided on your next step you make a plan and your cycle starts over again. This improvement cycle goes on forever. Lean is a journey, not a destination.

John Seddon suggests a simpler version of the Deming cycle, which only has three steps: Check (or study) – Plan – do. I especially like John's approach when there is a team and a process (a system) that is already running. Then it makes sense to start by studying how well the existing process works before making plans for changes. It's also common that acting (deciding the next step) is done at the same time as planning for it, so it makes sense to join them into one step (plan).

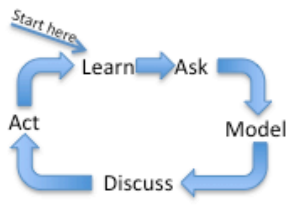
I have also seen a third version of the improvement cycle. It's called LAMDA, which is an abbreviation for Learn, Ask, Model, Discuss and Act. This can be mapped to Seddons cycle where learn and ask are parts of studying while model, discuss and act are parts of planning. I think the people behind LAMBA considered "doing" such an obvious part after planning that it didn't need to be explicitly mentioned in the model.



*Deming cycle*



*Seddons cycle*



*LAMDA cycle*

## Step 8 - Set up your principles and policies

Since the basic idea is that your process will always improve and adapt to new situations it's good to have principles to guide you in a certain direction. Your principles should help you become more aligned with what your customers' need, which I wrote about earlier in this paper. To inspire you I will give you some of my favourite principles.

1. Find and fix failures early. The cost of fixing an error grows exponentially over time so you can save a lot of money by building the right thing with high quality from the start.
2. Keep it small and simple. Cost grows exponentially over growth of complexity. Work in small groups, with small batches and short release cycles. This has been shown to not just improve flow but also improve quality.
3. Everyone is responsible for the flow. Upstream: make sure you get what you need to do your work. Downstream: make sure to help the next step to get a good start.

4. Right from me. What ever you get (from upstream), make sure the work you deliver has good quality.
5. The largest part of a product's total cost is not building it but maintaining it, so build for easy maintenance.
6. Happy customers are cheap customers.
7. Good quality is cheaper than bad quality.

Together with those there are also the principles of Lean, like:

- Optimize the whole, not the sub-parts
- Long term thinking
- Respect people
- Eliminate waste, stress and unevenness

Your principles will then end up in some policies. Here come some examples:

8. Delete everything in the product backlog that is older than 6 month. If it's been there for that long it's probably not so important anymore. If this assumption is wrong it will return again
9. Don't start developing features that take more than 15 days to develop. Ask the stakeholder if it's still worth developing and if it is, break it down into smaller pieces.
10. Don't start developing unless there will be a tester available to test it within 5 days after estimated completion.
11. Fixing a bug has higher priority than developing new functionality

In David J. Andersons book, Kanban – Successful evolutionary change for your technology business, chapter 11, you can read about classes of service which is a great example of explicit policies.

## Final

I hope these eight steps help you get a good start implementing Lean and Kanban. Remember, Lean is a never-ending journey, which starts with your first improvement. There is no template or "Lean out of a box" you can implement, it's just a couple of principles and good practices. Some of the wisest words I've heard on this subject come from Tom Poppendieck. He said, "Whatever you see or hear about Lean implementations is the solution of someone else's problem. Because it solved their problem doesn't mean it will solve yours". Get inspired by others' implementations of Lean but it is how you implement it that matters.

Good Luck!

/Tomas Björkholm
http://www.crisp.se/tomas.bjorkholm

P.S.
Many, many thanks to Yassal Sundman for her help with content and language improvements