emergn

Value, Flow, Quality®

看板

KANBAN

FLOW

This publication forms part of Value, Flow, Quality® Education. Details of this and other Emergn courses can be obtained from Emergn, 20 Harcourt Street, Dublin, D02 H364, Ireland or Emergn, One International Place, Suite 1400, Boston, MA 02110, USA.

Alternatively, you may visit the Emergn website at http://www.emergn.com/education where you can learn more about the range of courses on offer.

To purchase Emergn's Value, Flow, Quality® courseware visit http://www.valueflowquality.com, or contact us for a brochure - tel. +44 (0)808 189 2043; email valueflowquality@emergn.com

Emergn Ltd.
20 Harcourt Street
Dublin, D02 H364
Ireland

Emergn Inc.
One International Place, Suite 1400
Boston, MA 02110
USA

First published 2012, revised October 2013, February 2014 - printed 3 March 2016 (version 1.10)

Printed and bound in the United Kingdom by Apple Capital Print.

Contents

INTRODUCTION

Kanban is a change management method that has been gaining many adherents in recent years. We discussed it briefly in our Work In Progress session – here we will go into more detail, as well as discussing the principles and practices associated with the method.

Kanban was first developed in the 1950s in Toyota by Taiichi Ohno as a means to assist 'just in time' production. The basic idea was to prevent overproduction; the waste represented by having large piles of inventory lying around before there was any need for them.

Previously, factories had ordered bulk quantities of particular parts in order to take advantage of the lowest wholesale price. They would then store the parts and use them as required.

Although it seemed as if the factory was getting a better deal, paying less than if it bought in small numbers, in fact they were carrying a higher amount of risk. Suppose the design changed or the finished car did not sell well. The parts would then have to be written off. While in storage, parts could be damaged or lost – more waste.

Instead Toyota decided to 'pull' in parts only when there was an actual and immediate need for them – collecting just enough for the number of cars to be made on that particular day. That was the theory, in practice factory lines needed to operate with a small buffer. So the line would produce enough parts to fill a designated space, and then stop. Only when this space had been emptied by the next activity in the process, could the line be re-started. The problem was that managers didn't really trust the process. What happened if they were left without parts? There was a tendency to sneak an extra buffer, meaning they would collect parts before they really needed them – an attitude that was less 'just in time' and more 'just in case'.

Taiichi Ohno hated people doing this. To stop it, he insisted that no-one could pull parts without authorisation. And this authorisation was simply the piece of paper or sign that stated the space was now empty. This piece of paper was called the 'kanban'. Mr. Ohno once caught someone he knew was about to pull his materials too soon and thundered: "Who are you and where did you come from?! What makes you think you have any right to this material? Show me your kanban!!" Thus the principle of pull, just-in-time production and strict WIP limits became inextricably linked to the humble sign or kanban.

Gradually the idea expanded as this sign or kanban seemed symbolic of the principles Taiichi Ohno was trying to instil. Ensuring that everyone can see what's going on, knows exactly where work is and can then take the appropriate action engenders trust in the system. Although, as Taiichi Ohno always insisted, kanban was just a tool – the ideal was one piece flow.

And having explained all that, you can now forget the history completely. The Kanban (spelt with a capital) Method as practised within software development has left its antecedents far behind. And, in doing so, now appears in many other knowledge working environments.

By the end of this session you will understand:

1. The practices and principles of the Kanban Method.
2. Ways to:
    - Visualise your work
    - Decide how you might limit your WIP
    - Take measurements of the workflow and manage the results
    - Communicate work standards using explicit process policies
    - Implement feedback loops
    - Improve collaboratively using models
    - Use models to recognise improvement opportunities.
3. How to use classes of service.
4. The separation of input and delivery cadence.
5. How to judge where and how to implement Queue Replenishment Meetings, Improvement Katas and Operations Review Meetings.
6. The ideas of evolutionary change and how to overlay the principles with existing processes.
7. How to analyse your progress of improvement.

1 THE PRACTICES OF KANBAN

In an online video called *Introduction to Kanban in under 5 minutes*, Kanban is described as a fantastic way of getting things done, a lean scheduling mechanism, simple and efficient, and most importantly, a way you keep delivering features to your customers.

In any description of the method, two points that remain constant from the manufacturing kanban are:

- Kanban is a 'pull' system
- Visibility remains crucial

A pull system – as we described in our Work In Progress session – means that an upstream process does not begin work until a downstream process has signalled that it is ready. Or to paraphrase Corey Ladas: 'Don't specify more than you can build. Don't build more than you can test. Don't test more than you can deploy.'

The Kanban Method, as applied to software development, is credited to David Anderson who ran the first implementation in 2004. It continues to evolve. There are six core practices in the Kanban Method:

- Visualise workflow
- Limit work in progress (WIP)
- Measure and manage flow
- Make process policies explicit
- Implement feedback loops
- Improve collaboratively, evolve experimentally (using models and the scientific method)

You'll notice that none of these practices are earth-shatteringly revolutionary. A fair comment on Kanban is that its practices ought to be common-sense thinking that are implemented everywhere. This simplicity is a great virtue – Kanban can be implemented easily and quickly by teams.

The Kanban Method also has four principles that demonstrate roots from both Lean and Theory of Constraints. Both of these approaches focus on the idea of continuous improvement to systematically identify constraints, create strategies to remove them and move on to identify the next challenge.

From a Kanban perspective, you:

1. Start with what you do now.
2. Agree to pursue incremental, evolutionary change.
3. Respect the current process, roles, responsibilities and titles.
4. Encourage acts of leadership at all levels.

This very powerful philosophy means that the Kanban approach is very different from other methodologies and frameworks – it doesn't require the imposition of an entirely new system. Scrum and Agile methodologies frequently meet with resistance, whether from within a team or from other departments or managers. Applying the principles and practices of Kanban, by contrast, exposes the way your existing process works. Any changes you make to this tend to be incremental, and due to the visualisation process, better understood – this lessens resistance. As Anderson puts it, 'Job descriptions are the same. Activities are the same. Handoffs are the same. Artefacts are the same.'

This doesn't just apply to starting out from more traditional processes, but can also be applied to teams who have already started out on an Agile journey. As Henrik Kniberg describes in his book ***Lean from the trenches: Managing Large Scale Projects with Kanban***: 'I see many Kanban teams that gradually discover (or sometimes rediscover) the value of many of the Scrum practices. In fact, sometimes Kanban teams start doing Kanban because they didn't like Scrum and then later discover that Scrum was actually pretty good and their problem had been exposed by Scrum, not caused by it. Their real problem was that they had been doing Scrum too much by the book instead of inspecting and adapting it to their context.'

1.1. Visualise workflow

You'll remember that kanban means 'sign' in Japanese. This is an area that differs significantly between manufacturing and creative, knowledge work. Development does not involve a physical sign in a bin to signal that more parts are required. But, the principle of using a visual system to enable everyone on the team to know the work status remains fundamental. So fundamental, in fact, that the practice of work visualisation arises in pretty much all Agile implementations and most modern management methods.

A kanban board emphasises workflow – how a request is passed from one group to another. If at any point a task becomes blocked, the knock-on effects of this are also clear. Like a broken-down car, traffic queues up in the lane behind the blocked task. This encourages the team to prioritise blocked or faulty items in order to get work flowing again.

If one group or team has a WIP limit of three, then one slot being blocked by a task with incomplete information creates a sense of urgency. The impact of one third of the work being stuck is clearly huge for the processes both up and down stream. The pressure to solve the issue becomes something for the team to deal with, rather than an individual sending an email to a project manager for more details, and then forgetting that no reply has been received because he's working on something else. It encourages swarming, when many different functions work together to crack the problem, and makes clear when spare capacity on one function can assist another.

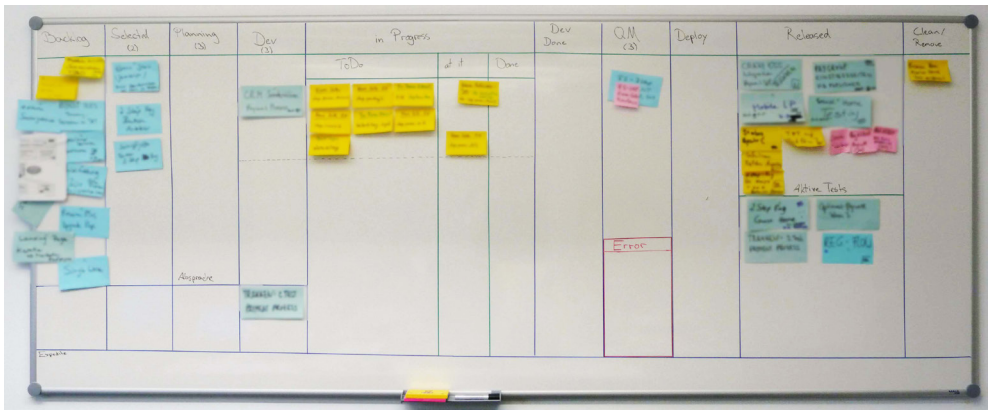


Figure 1. A sample kanban board

Most teams use a physical board with coloured sticky notes to represent tasks and the flow of work. Kniberg suggests a couple of key reasons why physical boards are essential. The first is that Kanban encourages processes to evolve. It is constantly changing to adapt to context and learning. More so, early on. Many of the electronic tools don't cope well with the changes.

The second key point is a physical board increases collaboration and ownership of the work because the information is always available and visible. The frequent interaction with the board increases the sense of progress.

The final point, and one which is cited in many stories of Agile adoption, is that the boards help with the culture shift. Changing the working environment is a clear signal of doing things differently and the visibility of the work helps increase the whole team's accountability for it and getting things completed.

This is not to say that virtual taskboards or software cannot be helpful – especially with distributed teams. The key is to keep things as simple as you can.

Case study – BBC Worldwide, visualising the work

BBC Worldwide is the commercial arm of the BBC, acquiring, creating and developing media content around the world in order to maximise value for the UK licence payer. In 2008, one of its software development teams that was working on a project, Digital Hub, began to implement Lean practices, including Kanban, in an effort to improve flow and cut waste. Specifically the team was looking to improve lead time and reduce defects.

A key measure to permit self-organisation and facilitate pull is visualising the work. The team pushed this further than a traditional taskboard, organising several information radiators and boards and thinking hard about the layout around the room.

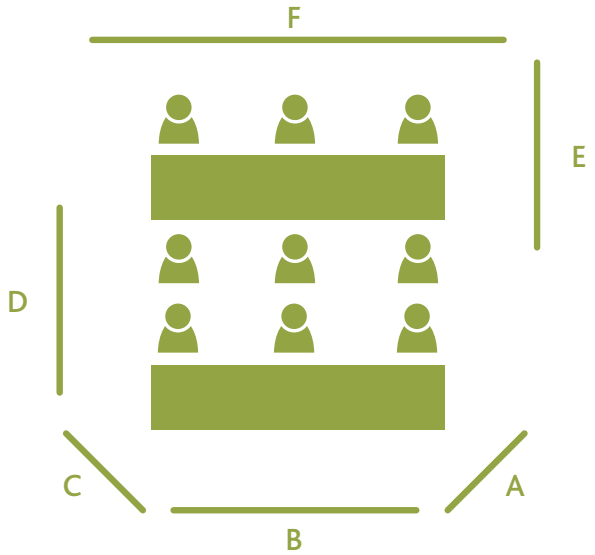


Figure 2. Schematic of the team's work space

Key

A: Ideation pipeline kanban board

For some teams, the information on this board might be referred to as the backlog, but for the Digital Hub team, the concept was more nebulous. Any ideas were captured here – even ones which were not yet approved or detailed. This often triggered further suggestions and clarifying questions from the team and the ideas could be broken down into smaller deliverable units and prioritised. These items then entered a 'ready' queue that could be pulled into development as and when there was capacity.

B: Main task kanban board

This board tracked all development work and the daily stand-up took place in front of it. Clusters of cards indicated bottlenecks which needed to be resolved; cards which did not move were 'blocked' – an indication of an issue which the team needed to work on together by exploring alternative solutions or perhaps escalating the problem to a line manager.

C: Structure of daily stand-up information radiator

A process board, this was intended to keep the team focused on running Kanban properly. First, team members checked their own work status was correctly displayed, and anyone blocked should then report the problem. Next, bottlenecks were noted and discussed, and finally the work itself was reviewed to see if priorities had changed or flow could be improved. An 'expedite' class existed and any urgent or late changes in this would be brought up at the daily stand-up.

D: Release notification and daily support process information radiator

Any scheduled releases or support tasks would be noted here. It kept the team focused on the more mundane tasks that otherwise tended to be ignored.

E: Architecture, estimating and breaking down projects information radiator

This board recorded decisions about architecture for the software, estimates and how the work had been broken down into MMFs (Minimum Marketable Features – more about this can be found in our Requirements session). By keeping the decision-making process visible it ensured the team remembered why certain decisions had been taken, allowing them to revisit or clarify them without unnecessary rework.

F: Kaizen board and technical debt information radiator

This board was focused on identifying the areas of poor legacy code that were impeding the speed of current or future work. The team then voted on which items they would work on to improve or modify. Although the work was invisible to customers outside the team, explicitly making these tasks visible was an important part of continuous improvement.

For the team, the boards revealed indicators of issues to be addressed, helping them look both up and down stream to facilitate the flow of work. The data was seen as a source of empowerment for the team, not as a control tool for management, because the team had the responsibility to take action based on the problems highlighted by the boards. The way the boards were laid out evolved over the course of the project, but the team noted how their use encouraged collaboration. Because of the limits and visibility there was no capacity for 'cherry-picking', instead everyone needed to work to eliminate bottlenecks or blocks. Overall, lead times improved by 37% and reported defects fell by 24%.

Practicalities

Begin by deciding the boundaries of your process visualisation. This is not the place to map the full value chain for the company; it is about where you might apply controls to your process. Those up stream and down stream of this region may not need to make changes and apply things like WIP limits themselves, for example, although they may need to deal with you differently in the future.

Now look at the types of work – the tasks or requests – that are likely to arrive. These could range from bugs and defects to user stories or features. It helps if you can also identify where they come from – sales, strategic request, regulatory requirement.

Sketch out the process that will happen to the work from the moment it arrives (input) to when it leaves your bounded area (output). Rather than using roles (developer, analyst, tester, etc.), use the process instead (analysis, development, testing). This helps to clarify that it is the work that matters, rather than rigid definitions of who's job it is.

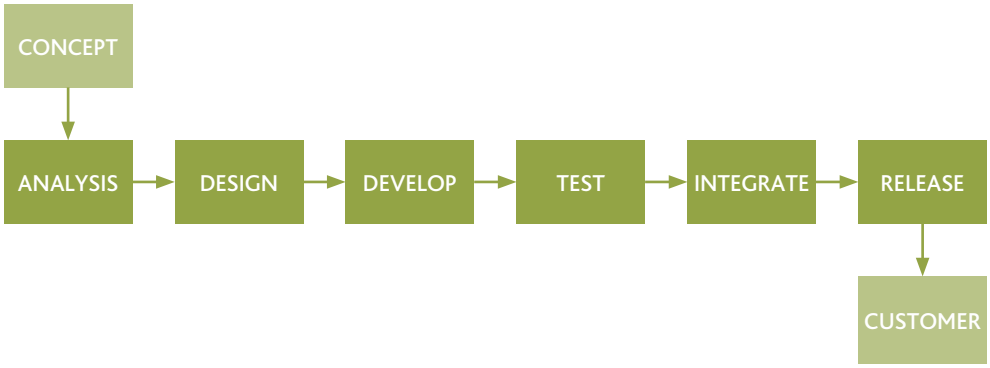


Figure 3. A sample value map

Then draw up the process as a 'card wall', the chart which will visualise the work. At the start is your input queue, followed by the downstream processes that will lead to final output. Each of these processes needs two columns: in progress and completed work. The completed column acts as a buffer to even out unpredictable variation of delivery.

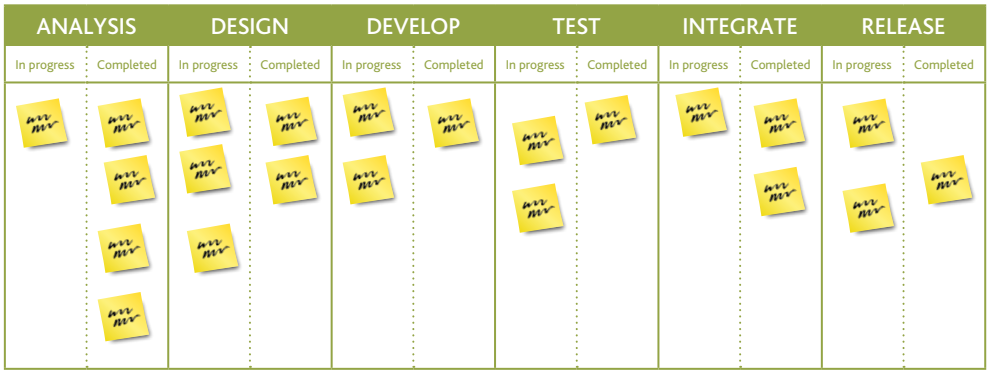


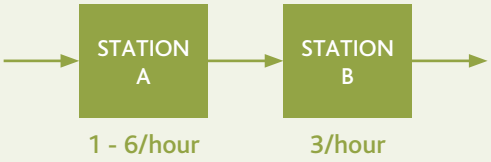
Figure 4. A schematic of a card wall

Activity 1 – Using buffers to smooth flow

This activity will require 20 minutes of your time. Before you can play it you'll need to find yourself a dice – if you already have that, great, let's get going.

We will be looking at an imaginary process which includes 2 workstations, which we've simply labelled Station A and Station B.

Station A is able to produce up to 6 units an hour, but this process is subject to a high level of variability meaning that it won't always achieve its maximum. Station B is able to consistently process 3 units per hour.

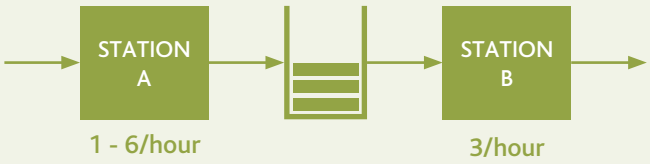


Round 1:

In Round 1 you are going to simulate progress of items through the process over 20 hours; one hour at a time. A dice roll will simulate the variation at Station A.

1. Roll the dice to establish how many items of work are at Station A. Write the number down in the table provided overleaf.
2. Next, write down how many work items would be carried through by Station B given that it can only process 3 work items. Any items above the 3 that can be carried through should be disregarded (i.e. if you previously rolled a 5 only 3 will progress, whereas if you previously rolled a 2, all of those items can progress).
3. Repeat the above steps until you have completed the table below.
4. Total the amount of items that went through Station A and Station B in the last row of the table.

You will have noticed in Round 1 that there were situations when Station A produced more items than Station B could actually process; there will also have been other times when Station A didn't produce enough for Station B. The dice roll was introducing variation into the process – simulating the variation inherent in product development processes. In Round 2 we're going to improve this situation by introducing a buffer. The buffer will act to level the work flow through the whole system.



Round 2:

1. Roll the dice to establish how many items of work are at Station A. Write the number down in the table below.
2. Next, write down how many work items would be carried through by Station B; Station B should always pull up to its maximum 3 work items from its buffer. Assuming Station A produces fewer than or equal to 3 work items in the first instance all of those will progress to Station B. In the case of Station A producing more than 3 items, the remaining ones are kept in the buffer.
3. Repeat the above steps until you have completed the table below.
4. Total the amount of items that went through Station A and Station B in the last row of the table.

ROUND 1		ROUND 2		
Station A	Station B	Station A	Buffer	Station B

Commentary:

In the second round we showed you that introducing a buffer means that a resource that is a bottleneck won't become starved of work. This is quite counter-intuitive when first considered: we're suggesting that a constrained resource – a bottleneck – needs to be kept busy – most assume that a bottleneck is always busy. That's not the case in a system with variation. Introducing a buffer smoothed flow which in turn led to improved throughput.

There is another point we should make. In order to simplify this activity we didn't worry about the amount of items the buffer could hold, but in reality you should be careful with buffers – choose where you use them wisely! Storing work in a queue will increase its lead time and cause disruption to other items too.

Each task should have its own card (sticky note – or whatever works for your team), perhaps colour-coded by work type or person requesting it. There's no need to fit the entire feature description on the card – 'new log-in' or 'unsubscribe fix' for example, will do. As the card moves across the board, it reveals its own progress through the system. You can time how long it takes to process as a whole, how long it spends being actively worked on, and how much time it spends queuing. To do this, most teams give each task a number which is also tracked in an electronic system. This can be written on the corner of the card, or simply matched to its title. Finally, teams can find a way of indicating who has been assigned to work on each task. This ranges from scribbling a name on the whiteboard to a fancy magnetic character badge to indicate whose responsibility it is at the moment.

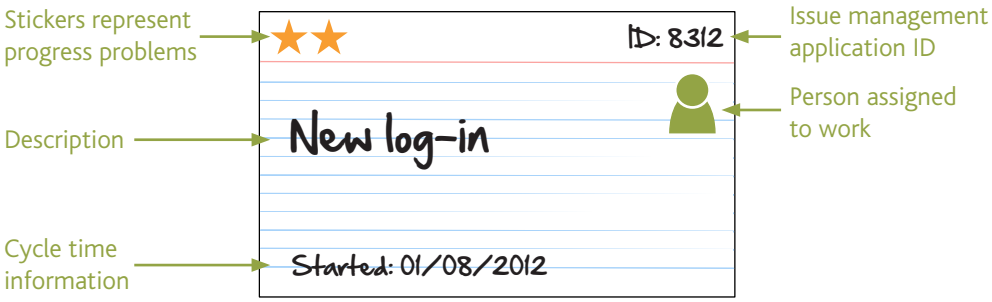


Figure 5. A schematic task card

If there is a standard time of three days in development and an item has sat there for five, then a star sticker might be attached to it. If an issue requiring management input has arisen on the item, then it's common to attach a second luminous sticky to the first – it acts as a red flag to signal that help is needed.

Rather than going through every possible option, we hope it's obvious that each team must decide for itself exactly what is needed on the card or not. There's no set recipe dictating which colours you should use or exactly what you should put on there. The method is not a rigid one, your team must change and adapt it as your team requirements dictate. This can mean that different teams working in the same office all have very individual kanban boards. As long as they are understood by the teams using them and sufficiently intuitive that a visitor could work them out, then this individualisation is probably to be encouraged.

Activity 2 – Design a kanban card

This activity will take 30 minutes and is split into 2 parts. It is designed to benefit your whole team, so you should gather as many of those people together as possible. For particularly large groups, you could split into teams and then present the designs at the end and compare them.

A kanban card is meant to be a simple token to represent work. In most software teams Post-its or index cards are used. In this activity, we will guide you through the process of designing a kanban card that caters to your team's needs.

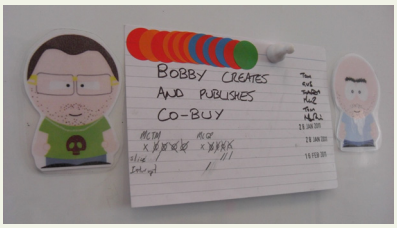


Figure 6. A sample kanban card

The purpose of a kanban card in software is to visualise the work in progress. At a very basic level, the kanban card itself should allow somebody to understand the work entailed in the feature it describes. Often the information provided is extended to cover things such as: the type of work; a number to identify the task for tracking purposes; who is currently working on the feature; its relative priority. You may have other pieces of information that are important to your team.

Part 1:

Start by defining the types of work that you see within your team. For example, some teams might identify their types as: content changes, bugs, documentation, new feature, maintenance, etc.

Next consider the last month or so within your team. Have there been any incidents for which knowing more about a task's status would have helped you catch a problem earlier? For example, did a feature spend much longer in analysis than normal, perhaps suggesting that it was more complicated or much larger than originally thought? Note such occasions and consider what information might have helped.

Now think about improvements that your team is looking to make. What data could you capture as a feature progresses through your process that would

help you make these improvements? For example, the kanban card above uses coloured dots to record the history of the card's progress – the blue/red/blue/red/blue/red sequence shows the number of times the task was in development and then went into testing before being returned to development. Such a visual reminder might highlight problems with bugs and rework.

Finally, consider the needs of people outside of your team. Do others have any requirements for data from your team? For example, do people want to know what release candidate a feature has made it into?

Once you have gathered all this information consider how you might want to represent it on the card. You can use different colours, shapes and sizes for the cards.

Create your card and test this design against recently completed tasks to ensure that it works.

Part 2:

Start using this design within your team. Over the course of the next few weeks take time to review whether it has been effective. Make any changes or amendments that seem necessary.

Commentary:

A kanban card can be an effective indicator of progress, highlighting any potential issues as they arise. Some teams delight in a chance to cut out shapes and play with colours; others are rather cynical about the idea. There's no need to lavish hours creating works of art, but you should reassure more sceptical team members that the time you spend doing this is a worthwhile investment. Kanban cards increase clarity for the whole team of what is happening with your work. This assists flow, reduces the number of 'dropped' or 'forgotten' items and supports continuous improvement in your processes.

1.2. Limit WIP

Our Work In Progress session explains the rationale behind WIP limits in detail. Here, we will just remind you that lower WIP limits encourage flow, reduce queues and enable faster delivery.

Practicalities:

Begin by setting a work limit for each process. How many tasks are allowed into development at any one time? These tasks should be assigned by person, pair or small work team, each of which will have their own WIP limit – typically between one and three items. For example, if you have three pairs of build programmers, you might assign them each a WIP limit of two. That means your total WIP limit would be six, but you might also have three items 'engineering ready' in a buffer. Thus if a pair finish a task, they can pull a new one from the buffer, which then has a slot ready to be filled by the upstream process.

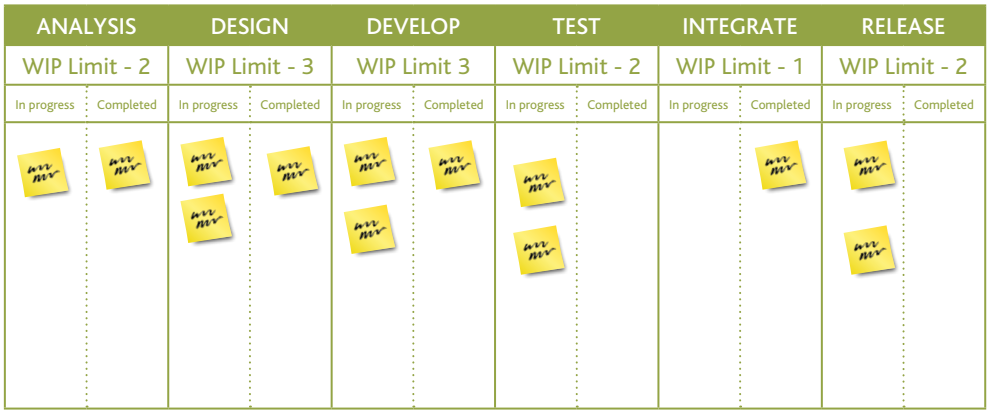


Figure 7. A schematic of a card wall

It is important to limit the amount in the buffers as well, since otherwise they can turn into the old problem of endless queues. Since business sponsors are constantly debating and reprioritising which item should enter the buffer ready to be pulled in, you should ensure that only items of value to the company are to be worked on. It's useful to purge a lengthy queue of work that has not yet entered the system, after a certain time period. If an item is continually not selected for development then it is probably neither especially urgent nor valuable. After six months, for example, that item should be purged. If a business sponsor is adamant it is still required then it can be resubmitted – indeed, this is often how to focus sponsors' attention on an item!

WIP limits must be adjusted as you work through the process and spot where bottlenecks emerge. This helps you to adjust your capacity accordingly to match the demand. For example, if testers are often sitting idle while waiting for items to come through from development, you need to adjust your head-count allocation to add more capacity into development and possibly less in testing.

1.3. Measure and manage flow

Simply making the WIP visible on a board is the most important step in managing flow. It can be supported by other tools, however. The Cumulative Flow Diagram (CFD), details of which can be found in the Technique Library, can also be useful. Many standard tools can be used to create a CFD – for example, in Microsoft Excel, a stacked area graph would be quite sufficient.

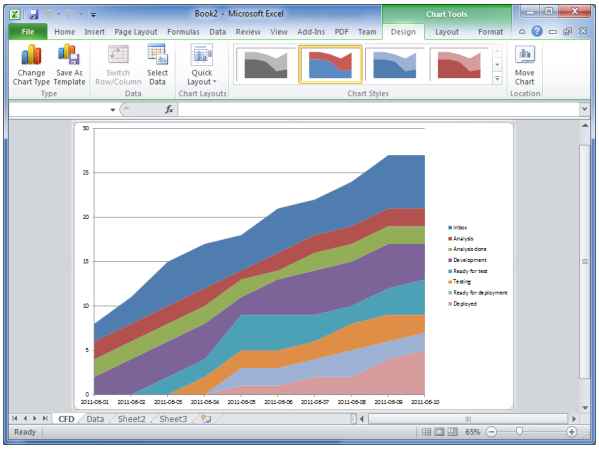


Figure 8. A Cumulative Flow Diagram created using Microsoft Excel

You can track the total number of features to be created for a product, split by those not yet done, those in progress, and those completed overall. This provides rather more information than a simple 'burn chart', and also allows us to track where extra features have been added in. It allows us to observe total lead times and stay on top of likely completion dates. We can do this by measuring the queue lengths (the distance on the x axis that a single item spends in progress) and we can discover our average lead time.

We can become slightly more sophisticated in our use of the CFD, when we break it out by activity and plot the flow of the work.

By doing this, we can spot bottlenecks and pinpoint where they have existed at any given time. Of course, we should be looking out for these simply on our card wall, but the CFD allows us to gain an overall picture of flow through time. Perhaps there is an element of seasonality to requests that causes bottlenecks at a particular time... this is clear because step changes in the CFD indicate batch transfers. Using the CFD we can match our resource accordingly. Most importantly, CFDs can prove a powerful tool to help persuade other departments, whose work impacts on the team, of the importance of small batches and managing flow to avoid bottlenecks.

You can measure lead time simply by attaching a date to each work item and then recording its progress either manually or via an electronic system which automatically generates reports. You can then also use Little's Law with your lead time measurement to predict when each task will be completed.

In either case, lead time is one of the most important metrics. It enables your team to perform against standards, whether your 'standard' is simply an average designed to help you spot troublesome outliers, or because you are using a 'class of service' scheme, and need to deliver against different service level agreements (more on this below).

It also lets you see how much time an item spends in a queue and how much it spends being worked upon. Just to assist with terminology: Kanban refers to the time actually being worked on as 'touch time', while the time spent by an item in the process as a whole, (from input to output), is known as 'lead time'.

Activity 3 – Building a Cumulative Flow Diagram

A Cumulative Flow Diagram (CFD) is a representation of the quantity of work in different states (often representing steps in the development process) over a period of time. It is a visualisation tool that allows a team to track the progress of a delivery stream and identify how the work flows through the system. The diagram synthesises valuable information about work in progress, queues and cycle times.

This activity will take 15 minutes and the output will be a Cumulative Flow Diagram that you will be able to use to analyse how work is flowing through your process. To see the actual benefits of the Cumulative Flow Diagram, you will need to update it on a daily basis thereafter.

You should first decide the unit of work that you will track over time. We'd suggest picking something like a user story, task or requirement that is small enough that you will see progress in days rather than weeks. Keeping the information as granular as possible will help you spot problems.

To capture the data that you'll need, start by creating a table, similar to the one below, that details all of the distinct states a piece of work follows through your development process. The table should enable you to record how many items of work exist in each of the states for a given day. This data should be recorded at the same point in time every day.

Date	Backlog	Analysis	Development	Test	Done

The simplest way to create a Cumulative Flow Diagram is to draw it by hand. Given that the diagram is cumulative, starting out with a large piece of paper is advisable; at minimum we'd suggest a piece of A3. The vertical axis represents the cumulative total of work items over time, the horizontal axis, the date at which the information was captured. Each of the states in your process should have its own colour so that it can be distinguished from the others in the diagram. At significant events, such as a team member being on holiday, or the introduction of a lot of new work, some teams find it useful to make a note on the diagram for future reference.

Working backwards through your process, starting with the last state, (Done in our example) draw a stacked bar that represents the amount of work items in each state. Remember that this diagram is cumulative and so should maintain a historical account of the movement of work items through your process. Over the course of use, you will build a diagram similar to the one below.

The above process can also be achieved in a spreadsheet application such as Microsoft Excel. To do so, create the table as described above, and then build a stacked area graph from it. Make sure that the series of data are ordered correctly – the last state of the work (Done in our example) should be at the bottom.

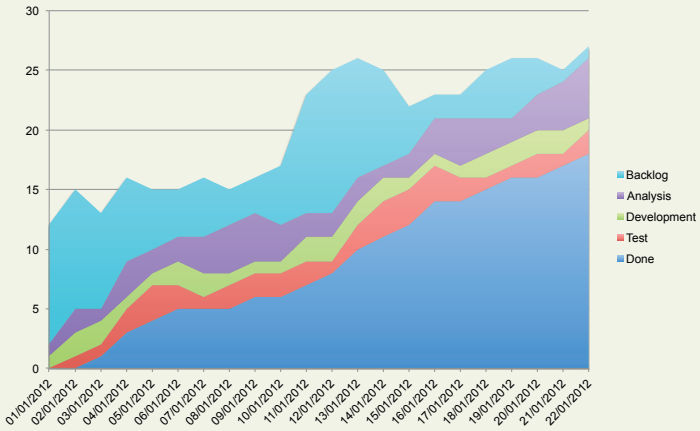


Figure 9. A stacked area graph created using Microsoft Excel

Commentary:

Over time the Cumulative Flow Diagram will build a representation of the flow of work through your process. It allows you to calculate cycle time, queue length and helps to identify bottlenecks. For help analysing a Cumulative Flow Diagram see the Technique Library entry.

As an example, however, the diagram above can be 'read' to reveal what was happening in our development process. We can see that on 11/1/12 there was a sharp increase in items entering the backlog – this was the point at which approval was granted for a new batch of work. We didn't manage to make much of an impact on the extra work until we brought in the Product Owner on 17/1/12 to help us with the analysis. We needed her help because items stuck in analysis were beginning to starve development for tasks to work on.

1.4. Make process policies explicit

Transparency is in the DNA of Kanban. For many departments, when under pressure from unrealistic business demands, there is a tendency to hide knowledge away. If you tell everyone that their request will take 50 days, they'll be pleased when you give them a result in 45 days. Department A's request will only actually take 10 days, but Department B's request is so complicated it could take 60 days. You need to claw the time back somehow...

This is the opposite of the way Kanban works. The policies need to be set in advance, whatever they may be. We only accept work items that take an average of 10 days. If it's predicted to be longer than that, then it must have its own project team assigned to it. In return, we guarantee that we will complete the item within 20 days.

Such clarity of agreement is intended to manage risk and improve customer satisfaction by delivering against expectation. As the policy is seen to work, levels of trust build significantly, something which stops a great deal of the unnecessary pressure created by departments pushing back against one another over cost and timing estimates.

Everyone must understand the reasoning behind them, from the WIP limits per process to what justifies an 'expedite' class of service. This also means, of course, that anyone can challenge the policies and then adapt them and test out the new policy.

It has been said that this practice is a key enabler of evolutionary change. It also complements the good advice described in the Teams session in terms of having a common set of working agreements. Later in the Improvement Kata section, we discuss running experiments to improve overall performance. Improvements can lay in experimenting with new explicit policies and measuring the performance.

It is important to ensure common understanding (an explicit policy, if you like) as to who has the authority to change a policy. For instance, who can change a WIP limit? Why is a WIP limit being changed – is it permanent or merely a temporary override based on an unexpected issue?

Case study – *Biomni, overlaying Kanban*

Creating software to assist large organisations structure and manage their IT services to different business user groups, Biomni is now in the sixth generation of its service catalogue offering. The development team has been working together for nearly 10 years and adopted Scrum about five years ago. In each retrospective the same issue came up again and again – a gap seemed to have opened up between the work being processed by the seven developers and the two quality analysts. In fact, quite often at the end of a sprint a piece of work wasn't really 'done', it still needed some final testing and then a few tweaks of rework.

This might not sound like a great deal, but rather than congratulating themselves on how super-agile they were compared to other companies, the team recognised that continuous improvement meant exactly what it said. Since the problem was primarily one of flow, in 2011 the team decided to overlay Kanban on top of their existing working arrangement.

The team was distributed, and so the first task was to visualise the workflow in a meaningful way. The team chose the 'kanbanery' tool to help them with this, providing a virtual taskboard. It proved an almost instant improvement on the spreadsheets they had used previously.

The team took the most important (and painful) step, to create and enforce WIP limits. There were immediate effects. Quality analysts don't like standing around with nothing to do while the developers create some code. So naturally, they got involved in discussing what kind of tests they ought to be thinking about. The developers pointed out that a lot of the tests were already covered by the unit tests that they wrote as part of coding. The testers shouldn't waste their time with those – they should focus on all those tricky mind-bending tests that developers didn't even consider!

Of course, this conversation was always meant to happen anyway, but somehow, before, it just... hadn't.

The idea became an explicit process policy: there must be a conversation between developers who worked on the story and quality analysts to establish the best plan for tackling the testing of the story. The Biomni team adopted a name for this conversation – the three amigos. A developer, a quality analyst and the Product Owner were expected to get together and discuss ideas up front – everyone got excited as the collaboration started to pay dividends in slightly unexpected areas like integration and regression testing.

Unsurprisingly, the team still had occasional flow problems and bottlenecks. A big queue built up in front of testing at one point, but because strict WIP limits meant that the developers couldn't just keep cracking on with writing more code, the whole team was forced to swarm in order to crack the flow problem. It gave the team a more structured way of dealing with problems and fostered a greater sense of collaboration.

Finally Kanban's insistence on lead time as a metric, backed up by a Cumulative Flow Diagram to show pinch points, really helped the team focus their energies quickly. It was impossible to build up an internal queue which only came to light at the end of the sprint – instead the flow was something every team member was aware of, and which helped make retrospectives less a matter of subjective opinion and more about the empirical data of their performance. Indeed, so helpful had the enforced conversations been to planning the work, that the idea was mooted to expand the three amigos to four, and include input from customer service representatives into initial development discussions.

Practicalities

Negotiate these policies up front. They will probably include some form of service level agreement (SLA), some rules on work item size, frequency of batch delivery, what permits an expedited request, how responsibility for prioritisation will work, how frequently prioritisation meetings will occur and who will be involved.

Once these are set, exceptions are rarely tolerated (please, pretty please I need two features completed this month not just my allocated one... No. I'll buy you dinner. No. I'll complain to the chairman. No. I'll fall on the floor and scream and scream till I'm sick. Hey everyone, come and watch this!).

However, the policies need to be frequently reviewed. Perhaps the weekly prioritisation meeting is too frequent and needs to be scaled back, or it's clear that too many expedite requests are arriving or even that the SLA can be improved because lead times have reduced.

Activity 4 – Explicit policies reduce rework

This activity should take 20 minutes.

The intention of the activity is to identify the sources and causes of rework within your team, and to help you reduce or potentially eliminate the problem. The activity is broken down into 3 sections.

Part 1 (5 minutes):

Use a technique called Silent Brainstorming to generate ideas about why rework is occurring within your team. Silent Brainstorming involves everyone noting down their ideas about why the rework occurs and where it comes from. Note down one idea per piece of paper/Post-its/index card. Work in silence with no interaction with those around you.

Part 2 (5 minutes):

Place the ideas that have been generated on a board or table where everyone can see them. Work as a team to group together any that are similar. Having done that, spend a little time to pick one issue that you would like to work to improve or resolve.

Part 3 (10 minutes):

Discuss as a group the smallest change that you could make to bring about an improvement to this particular issue.

An example might be:

"Issue – we pick up lots of problems in testing that we ought to have noticed earlier.

Smallest step: we should peer-review a piece of code before handing it over for testing.

Who's going to organise that?

The person who developed the code should organise for the review to take place.

What about if the code was written in a pair?

Good point. If the code was written by two people in a pair, then it doesn't need to be code reviewed."

Write the policy up and display it somewhere that the entire team can see it. Plan a review of the policy for three weeks time and use a meeting format similar to the above though instead you should answer the question "Did the introduction of the policy give us the benefit we planned? What could we do differently?"

Commentary:

The idea of policies evokes strong feelings: some people feel constrained and restricted by them; others relieved that with standards in place they can worry about something else. Each team finds its own happy medium between the chaos of no policies and rigid, rule-bound policies everywhere. In either case, the activity described above can be used as a powerful tool to establish a shared understanding of a problem. That way at least assumptions are stated explicitly. 'Documenting a policy' can sound overly formal – what it might really mean is 'we scribbled an idea on the whiteboard which we're testing this week'.

1.5. Implement feedback loops

This principle was not originally explicitly stated in Kanban. But shortening the feedback loop is inherent in the idea of improving cycle time. If you write and deploy a feature faster, then obviously you will gain feedback on it sooner. By formalising the practice, however, Kanban now makes it clear that teams must seek feedback – not just pile up completed features waiting to release them!

Feedback is what lets you know whether you are doing the right thing and doing it right. It exists on many different levels: from whether customers are paying for your product to feedback from automated tests pointing out you 'broke the build'.

We go into the many feedback loops required in depth in the Feedback session. Here it is worth mentioning the 'process feedback' loops that is specifically called out in this description of the Kanban methodology. In particular, feedback helps us realise whether we are achieving against the underlying goal. It is all too easy to get involved in asking questions about what people think about particular features rather than thinking about whether they are delivering the expected value.

To this end, David Anderson called out three meetings or events designed to embed feedback loops on how the product, team and organisation are performing.

The Daily Stand-up – most proponents of Agile will recognise this meeting. It provides feedback on how the team is doing right now. Within the Kanban Method, it is typically held in front of the board. It is product focused but also asks the team to offer feedback on how they are working together to improve the process.

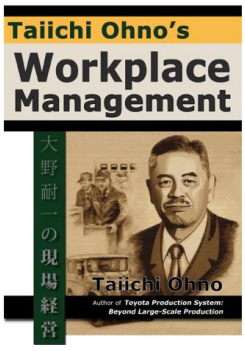
Improvement Kata – this is designed to highlight opportunities to improve within the team or department, looking at blocks, problems and system capability. Targets should be set, defined and measures or actions agreed that offer progress towards these targets.

The Operations Review Meeting – an explicit practice (more below), the Operations Review Meeting brings different parts of the organisation together to provide feedback to one another on the larger picture.

1.6. Improve collaboratively using models

Kanban inherits one further characteristic from its Japanese gestation: a belief in kaizen or continuous improvement.

Until the addition of the Improvement Kata, Kanban did not explicitly state how you improve. Because every improvement depends upon individual circumstances and context, the choice of technique or action is left entirely up to the individual team. The only mandate is to 'use models' and the scientific method.



'Use models' is a slightly grandiose term for reminding you that continuous improvement is built on objective data and empirical evidence; that is, on actual results, not gut feel or propaganda. If a team manages to produce a working product with fewer defects, worth more value to the company, at a lower cost, more swiftly, then that is the better team – not the one doing a presentation about how great they are.

Similarly, a project's performance is often measured against the business case that supported it. If the revenue does not match then difficult questions need to be asked about how we are doing our analysis and research. Kanban believes you should be explicit about the assumptions (the model) with which you begin. Collect the data about how you are doing and then make changes and measure the difference these changes make. We discuss the subject in more detail in our Feedback session.

The crucial factors that permit continuous improvement tend to be about how the team feels. A team that feels empowered will make plenty of suggestions and expect to trial and debate them. Management will be tolerant of failure when change was instituted for the right reasons, because a focus on measuring and managing flow means that success or failure is visible and can be built on or corrected with low risk.

Practicalities

The basic model that underlies Kanban is that flow is improved through the use of WIP limits. Naturally examining this assumption is an important part of Kanban. You should be using various data tools (like the CFD) to check whether and by how much lead times are improving. Each time you experiment with reducing or increasing a WIP limit or with adding or removing buffers you should measure the impact of such changes on lead time.

Another key tool to use is control of batch sizes. The assumption is that by reducing batch size you should improve flow and thus lead time. Changes should be measured and analysed.

Other tools all come with their own assumptions that you should test through the measurements at your disposal. In general, we recommend incremental change as the best way to test models step by step.

Case study – FNGP: enough is never enough

Jim Benson is one of several Kanban advocates who insists that kaizen is a mindset, rather than any set of practices. It's a point emphasised in the case of the Freudenberg-NOK General Partnerships (FNGP). This manufacturer of vibration dampers (amongst many other types of seals and gaskets) began to introduce lean practices to the factory in 1992 – specifically they searched for ways to eliminate waste through continuous improvement.

An initial kaizen event (in which employees discussed and began implementation of improvements) at a factory in Indiana achieved a 56% improvement in labour productivity and a 13% reduction in factory space needed. They revisited the activity in five additional three-day kaizen events over the next three years. With each session, productivity increased, while the space needed reduced.

The point was that FNGP never said to themselves, 'right, we've done enough', they continued a relentless quest for improvement.

Interestingly, James Womack and Daniel Jones in their book ***Lean Thinking***, comment on how the process was perceived by managers from other companies. In general the response was negative. One group questioned why FNGP didn't stop bothering with kaizen and concentrate on running at a 'steady-state' because the process was fixed. The other group felt that these improvements revealed that FNGP had set the process up wrongly to begin with. If they'd devoted more time up-front, went the thinking, they wouldn't have needed to spend three years improving!

Both views help demonstrate the point of Kanban's proponents – a belief in continuous improvement is a mindset, not a series of specific tools.

Activity 5 – Tulips from Amsterdam

Figure 10. Tulips in Amsterdam

You're due to meet your partner for dinner tonight. As the highly creative owner of an imitation flower boutique, you decide you will make a special bouquet for your partner. After thinking hard, you decide on the perfect flower – tulips!

This activity will take about an hour. You'll need 5 other people to help you play it. This is a perfect activity for running with multiple teams at an all-hands meeting or for providing the warm-up at a team meeting.

Objective:

The objective of this activity is to allow the people taking part in the simulation to experience some of the main aspects of Kanban in a safe, simulated environment.

Preparation:

To play the activity, you'll need:

- A few sheets of paper (A4/US Letter as a minimum size).
- Print out of the assets associated with the game (available on the valueflowquality.com website).
- Colouring pens.
- 3 pairs of scissors.
- Stapler.
- Timer.

This is the first time that you've made tulips in your flower boutique, so you'll need to create a new process. Discuss what the process might look like given the description of how a tulip is constructed below. You'll use this same process throughout the rest of the activity so ensure you think carefully about the creation of the flowers. 5 of the 6 people should be assigned specific roles within the process, the remaining person should act as the observer.

Observer: For each of the following rounds, one person in the group will need to act as the observer. It is his or her job to record specific pieces of information as instructed in each round.

How to build a tulip:

Use the assets for this activity that are provided in the Additional Resources section on www.valueflowquality.com. A tulip should be built by taking the following steps:

1. Cut the flower and leaves out from a page.

2. Colour the leaves and stem green.
3. Colour the petals to a colour of your choosing or as directed.
4. Staple the leaves to the stem.
5. 12 (a dozen) flowers should be assembled into a bouquet and a bow stapled to them to hold them all together (the bow is also in the assets provided).

Round 1 – Oh goodness, is that the time? In the fun of designing a new process to deliver your product you've completely ignored the time! In 10 minutes you need to leave to meet your partner. In that time you need to produce a bouquet of 12 flowers. Start the timer and using your process deliver as many tulips as you can (if you can deliver more than 12 do so). The Observer should time the round, and record the number of flowers that were produced.

At the end of the round, run a little retrospective – a meeting to review the assumptions you made in designing your process and how they worked in practice. How many tulips did you actually manage to create? Divide 10 (the amount of time you had for the round) by the number of tulips produced - this will give you the average lead time per tulip.

Good news – when you were out with your partner, other people spotted the lovely bouquet and have asked how they can buy some. You've uncovered demand for a new product! That means that you need to start producing tulips on a much grander scale. You've heard of this wonderful thing called Kanban, which is said to increase throughput.

Round 2 – Enter Kanban: visualisation. Start this round by visualising your process – take a piece of paper and draw a column for each of the activities in your process. Add a column that records the number of completed tulips. Finally add some rows to your diagram. At 1-minute intervals throughout the round, the Observer must record the inventory at each of the activity states.

Once you've completed the diagram, run the round as per Round 1, seeing how many tulips and/or bouquets you can create in 10 minutes. Remember that the Observer needs to record the amount of work in progress at each of the activity states.

At the end of the round, run another retrospective; in this one review the data that was collected by the Observer – think about how work flowed through the round. What does the information you have tell you? Calculate your average lead time again.

Round 3 – Enter Kanban: WIP limits. Start by redrawing the process diagram you used for Round 2, but split each column in two. Label the left hand one 'Ready' and the right 'In Progress'. Review the data you collected in the last round. Identify the busiest time for each activity in your process and how many items were either in progress, or waiting to be progressed. Take that number and divide it by 2, rounding down if necessary, your result is the WIP limit that you will apply to this activity.

For example, if you have one person colouring petals with a limit of 2, as soon as they have 2 sets of petals waiting, the person cutting out petals cannot cut out any more. Instead they could help to colour in and then go back to cutting out.

Run the round again as per Round 2, with the Observer recording the amount of active work every minute and the number of tulips and/or bouquets completed at the end of the 10 minutes.

Typically when using a Kanban system the introduction of WIP limits helps improve the flow of work - the bottlenecks are alleviated as tasks move more smoothly between the different states. Run another retrospective at the end of the round to discuss whether you have observed a similar effect. Are there changes to your current WIP limits that you think could help you make further improvements? Our way of identifying your WIP limit is fairly arbitrary – now that you've run the experiment you should have a better idea of what's right.

Round 4 – Enter Kanban: experiment. Your tulips are now selling very well; there are customers everywhere! You want to make sure that your process is as efficient as you can make it so that orders flow smoothly to your customers. The main tool at your disposal is adjusting the WIP limits. Based on the discussions from Round 3 set new values for the WIP limits and run the activity again: 10 minutes for the round with the Observer recording data as you go.

Commentary:

In this activity we've introduced you to four of the main practices in the Kanban Method.

The main missing point is an overall (systemic) view of how work flows through the system. Indeed, there was no variation in the item being built (the bouquet), and nor were you permitted to change the process. This is very different from what you would expect in a product development scenario, but the basics are sufficient for you to imagine how these techniques might be applied.

Spend just a few minutes with the teams discussing whether any of these techniques might be helpful in your work.

2 KANBAN EVOLUTION: CLASSES OF SERVICE

We mentioned that Kanban had developed beyond its use in manufacturing. A key reason is that manufacturing deals with very low variation – it doesn't matter whether you make this or that chip first – they are interchangeable. First in, first out, makes perfect sense as an order of process. In contrast, knowledge work, such as product design or development, is made up of tasks of different sizes and, potentially, very different value. How do we know how to order them? We've discussed the theory behind prioritisation elsewhere. The point is that by offering classes of service we can optimise our flow and economic outcomes for different situations.

Put simply – do you want to pay more to get something done faster, or pay less for a slower service. Even if the customer is not paying – imagine a group of product owners debating between themselves which of their many requests takes priority – the principle of value prioritisation remains the same.

In his blog post *Kanban: What are Classes of Service and why should you care?,* Dennis Stevens explains that 'Each Class of Service will have its own policy to reflect prioritization based on the different risks and business value. Classes of service and knowledge of the policies associated with them empower the team to self-organize the flow of work to optimize business value.'

Figure 11. Postal services offer different Classes of Service

Practicalities

Classes of service must be defined by the impact of a task on the business outcome. 'I just really really want it done fast', is not a business outcome. 'The customer will walk away from the sale unless this error is fixed', is.

Establish a number of possible classes of service. Don't make it too complicated by offering twenty options. There are four classes that are often used:

Expedite – This class means that the item can jump all queues, becoming the first item to be pulled by each team. In theory, that should give the item the fastest possible lead time. Special rules might include that expedite requests enter the system outside of normal meetings and schedules and are launched out of the normal cycle of releases.

Fixed delivery date – A specific schedule date may be caused by an external deadline – end of the financial year, for example or some other regulatory requirement. This gives the item a sudden spike in cost of delay at a certain point. Such requirements might be moved through the system with a specific label that gives them priority over other, more financially valuable, items.

Standard – Most items should come under this heading. These are work items with a measure of urgency, although the tasks might be split by size with appropriate service level agreements and policies assigned to each – a few days for a small item, running to several months for a huge task. There is a cost of delay associated with each standard item, which is due to be incurred soon. In other words, when the task is completed then the benefit can be realised – there is value in delivering early.

Intangible – Items under this (interestingly named) class are typically internally focused and not for the customer. It would normally equate to non-urgent work that is still important. The work can fit the team's schedule, fitting around their more urgent demands, although still to be delivered within an agreed framework. This class is where maintenance or upgrade work often sits. It often has a low cost of delay, but must be fitted into the schedule because there will come a point when it will have a very high cost of delay that will stop any other valuable work being done. It is important to keep a fairly close eye on SLAs to make sure the item is not continually being bumped by more 'urgent' work.

While it is normally recommended to have four classes of service, we have come across more, but believe that six should be the maximum number. Each should be explicitly delineated on the board – either using swim-lanes or different colours.

As in the image to the right, each class of service still obeys the WIP limits, although there is usually an extra rule to ensure that only one expedite request can be accepted at any one time. Sometimes a customer might decide the class of service – either through what they are paying or through discussion regarding business outcomes. But often it will be up to the team to decide the class of service required and then to organise how jobs are pulled through the system because they understand the policies involved in setting them up. Ideally you want a team who know exactly which job they should be pulling next, not people who whine, 'well, no-one told us the server collapsing was more important than designing a new flash graphic for the Christmas party'.

Figure 12. **Classes of service rules visible for all to see**

In either case there still needs to be an allocation for how many items can be taken on within each class of service. We've said you can't fill the WIP limit with all 'expedite' items. Equally, it would be foolish to fill the entire system with fixed date delivery items. What happens when an expedite request comes along? Fine-tuning the mix is a matter for every individual team and company. Having some 'super

saver' items which can be easily put on hold is important – first of all, unless you provide space for them in the schedule, they won't get done until they suddenly become time critical; secondly, they represent a kind of spare capacity or slack, because these are the items that can be delayed in order to process expedite or fixed date items.

Aligning your capacity to the demand for each type of class of service is an iterative process, and one which may change from month to month. As long as you are monitoring and measuring the flow, you can see the result of your decisions and then continue to adjust accordingly.

2.1. Input cadence

As a term 'input cadence' is rather over-engineered. It simply means – how often new items will be selected for input to the system. As we know, Kanban means that items will be 'pulled' from the backlog as and when slots are available, enabling product owners to ask themselves: what are the most important items that I wish to see go into development right now? Will something be pulled every time there is a slot? This would require the whole product backlog to be continually prioritised and reprioritised. Once a week? Once a month?

The answer depends entirely upon your organisation. For teams who sit together no meeting may be necessary. For disparate teams, meetings may need to be held on video conference or by email. In general, regular, short meetings will help keep batch size small and foster collaboration.

Practicalities

It is sensible to begin with a regular weekly meeting – often known as the 'Queue Replenishment Meeting'. Diarising regular meetings and keeping them to time lowers the transaction and coordination cost of such activities.

The preceding week, an email can inform the product owners how many slots you estimate having available to fill and advising them to bring their top items to the meeting for discussion. Try to keep the meeting to an hour only – this in turn will encourage regular attendance. To keep it to this, there needs to be great clarity in the questions asked. Kanban helps with this by making your lead times more predictable.

For example, if lead time is currently 30 days, the question we would ask is simply: what will be the most crucial initiatives for the company in 30 days time?

Make sure that the business leaders assisting with prioritisation know what homework they are expected to do. If a business case is required to justify including an expedite item, for example, this needs to be prepared prior to the meeting.

2.2. Delivery cadence

The overly fancy terminology continues with 'delivery cadence', which in simple terms means how often we do a regular release of working software. For example, your team might agree to do a release every Thursday (a weekly pattern), or the last Friday of every calendar month.

An important point of Kanban is that you can decouple the input from the output. Unlike Scrum, which runs a set Sprint with input at the start and working software released at the end, in Kanban the two can run on entirely different rhythms.

As we know there are transaction costs related to releases – meetings to discuss deployment, final checks, ensuring staff are available in case anything goes wrong, etc. Even something as simple as upgrading another employee's machine still requires coordination – is there a day that the person is offsite or in a meeting? When the task in question is to upgrade the whole company's databases, the coordination of the deployment can be enormous.

Much of Agile is about automating parts of these processes to enable more frequent releases. Yet since each project is different, even with heavy automation some releases will be small, some inevitably larger. According to Kanban, accepting this and choosing a release schedule accordingly makes more sense.

It is possible to calculate a cost for a given release (taking into account such possible factors as training, extra staffing hours, developers who need to attend several meetings, etc.) and to set this against the value that we expect to receive from the release. The value we expect to receive is an estimate – in some cases, when you are simply improving a function slightly, for example, the value may be an intangible brand quality association rather than a cash order.

Kanban recommends using a discussion of the cost/benefit to determine the release schedule.

The key thing to understand here is that you can still couple the cadences if you want or need to. In a rather amusing blog post, Cory Foy explains how you can re-create Scrum using the Kanban method using the 'magic' of explicit policies.

Remember what Henrik said, '...Kanban teams start doing Kanban because they didn't like Scrum and then later discover that Scrum was actually pretty good...', they 'gradually discover (or sometimes rediscover) the value of many of the Scrum practices.'

The point is that it is *your* choice based on *your* context and things will change over time.

Practicalities

Begin with a relatively conservative release schedule, but also begin to invest in cutting the transaction costs associated with a new release. From investing in tools that assist automation to achieving a higher quality of code that requires fewer fixes, it should be possible to bring costs down. With this in hand, the team can begin to commit to more frequent releases. This may require a formal Release Plan – deciding what will be included in each release and what steps are necessary to deploy (a technical checklist might help).

2.3. Technical excellence

We mentioned that high quality code requires fewer fixes. Kanban has an explicit goal of delivering high quality code, both as part of continuous improvement, and because poor quality code which has to be fixed is an obvious form of waste.

```
Public ReadOnly Property BatteryPercent()
    ' This code will retrieve the BatteryLifePercent property and convert it to a percent.
    Get
        If SystemInformation.PowerStatus.BatteryLifePercent.ToString = "1" Then
            Return "100%"
        ElseIf SystemInformation.PowerStatus.BatteryLifePercent.ToString = "0.99" Then
            Return "99%"
        ElseIf SystemInformation.PowerStatus.BatteryLifePercent.ToString = "0.98" Then
            Return "98%"
        ElseIf SystemInformation.PowerStatus.BatteryLifePercent.ToString = "0.97" Then
            Return "97%"
        ElseIf SystemInformation.PowerStatus.BatteryLifePercent.ToString = "0.96" Then
            Return "96%"
        ElseIf SystemInformation.PowerStatus.BatteryLifePercent.ToString = "0.95" Then
            Return "95%"
        ElseIf SystemInformation.PowerStatus.BatteryLifePercent.ToString = "0.94" Then
            Return "94%"
        ElseIf SystemInformation.PowerStatus.BatteryLifePercent.ToString = "0.93" Then
            Return "93%"
        ElseIf SystemInformation.PowerStatus.BatteryLifePercent.ToString = "0.92" Then
            Return "92%"
        ElseIf SystemInformation.PowerStatus.BatteryLifePercent.ToString = "0.91" Then
            Return "91%"
        ElseIf SystemInformation.PowerStatus.BatteryLifePercent.ToString = "0.9" Then
            Return "90%"
        '...
        'snip
        '...
```

Figure 13. This is not the technical excellence we'd be looking for

We described how a small amount of WIP, shorter lead times and smaller batches had a positive impact on quality in the other sessions. Here, we just want to remind you that you retain responsibility for the product you are designing and building. If the software you write today is due to be released tomorrow and you will be fixing any problems that result, then you are highly motivated to get it right first time. Low WIP limits mean that you are unlikely to get distracted or confused by the inevitable switching cost of working on several different projects. Together, the three practices lead to improvements in overall product quality, which in turn frees up more time.

Kanban takes a certain level of technical ability as standard. You have to be able to produce working software in order to improve the process using Kanban. If you are failing to produce a working product, then Kanban is not a series of technical improvements (in the way that XP is, for example) that will assist you.

Many would argue, however, that Kanban will expose quality problems with code more quickly. Because defects sit on the board as cards that must be moved through the process before new work can be pulled in, everyone's attention will focus on quality – making it more likely that the core problem will be addressed.

3 OTHER PRACTICES IN KANBAN

Kanban adopts existing processes and merges them with ideas from other methodologies. These can range from a daily stand-up meeting in front of the card wall to the prioritisation Queue Replenishment Meetings. The only events specific to Kanban are the Improvement Kata and the Operations Review Meeting.

3.1. Improvement Kata

The word 'Kata' in Japanese means a set of choreographed movements – like the formal patterns of a dance or martial art. The term was adopted by Toyota to signify the teachable patterns that allow people to identify what is happening in the current situation, set the correct target and then create improvements that will move them towards the target. The idea is that although every set of circumstances or problems is unique, and thus every solution must also be unique, the analytical and creative skills are the same. By making these skills a 'Kata', a well-practiced pattern, the aim is to make this kind of behaviour and thinking a 'muscle-memory'. Instead of reverting to poor habits under pressure, we should fall naturally into a positive improvement cycle.

At first, 'Improvement Kata' was not included as a specific practice. It has since been included because it needed to be more explicit in order to be part of every person's normal role – asking how the department's currently performing, including how well it was serving its customers, and then considering how it could do better and what targeted steps should be taken for change.

In his work *Toyota Kata*, Mike Rother explores one of the key differences that he believes makes Toyota more able to continuously improve than other companies: the management approach. The job of the leaders is to engage with everyone in a way of keeping things in alignment and synchronised with one another. This means leaders need to provide:

- a vision as an overall direction giver
- a focused discussion about current system capability
- a forum for setting and defining target conditions
- honest and open discussions on how to improve capability.

To paraphrase Rother, the primary task of managers and leaders is not about making improvements per se but about increasing the improvement capability of everyone.

The role of the manager becomes the teacher, coach, coordinator and trainer.

Practicalities

The Swedish coach and writer, Håkan Forss, articulated the idea more fully:

1. Understand the direction – ensure everyone understands and shares the vision of where the team or organisation should be heading in terms of value, flow and quality.
2. Grasp the current condition – observe and collect real data and facts that reveal information about both process and outcome. This is what any discussion must be based on and, obviously, it presupposes that the team has been recording key metrics.
3. Establish the next target condition – decide on the next achievable step towards the business strategy or goal. This target may be a hypothesis – something that the actions will test. For example, the target may be to reduce bugs by 10% or to reduce lead time by four days. Forss advocates what he calls the Goldilocks rule for target setting, 'not too hard, not too easy, but just right'.
4. Take steps towards the target, following the Plan Do Check Adjust cycle – when taking actions it is quite likely that the team will encounter unexpected difficulties or results. These may even disprove the original hypothesis. For example, if the target involved reducing batch size, the team may learn that this increases transaction cost to unacceptable levels. That will stimulate a new round of target setting as the team strives to lower the transaction costs in order to continue progressing towards the overall goal.

The process can be summarised as a list of questions which act as a loop until we reach the desired target condition, at which point we begin again.

Figure 14. Question to reach the desired target condition

Forss recommends running at least one experiment every week and always having one active so that the team is continually focused on the idea of improvement as part of a 'kaizen' mindset.

A formal meeting can happen as a traditional one-to-one or as a larger team and manager discussion at least once a month; for some teams the process could involve a special board for improvements or a way to capture insights and the data relating to any experiment on an improvement. For smaller or less formal teams, it could simply become part of a slightly wider team meeting once a week or a fortnight.

3.2. Operations Review Meeting

This meeting has a broader focus than the Improvement Kata. The idea might be summarised as seeking to provide the kind of department-wide overview that the card wall gives for a project. It assists with broad coordination and even collaboration between teams, and offers an organisation level feedback loop that shares learning across multiple areas. Once again, its defining features are transparency and continuous improvement based on empirical data.

> *"I believe that Ops Review is the lynchpin, or keystone of a Lean transition and Kanban method implementation. It is an objective, data-driven retrospective on the organisation's performance."*
> ***Kanban***, David Anderson

Practicalities

Make the meeting as broad as possible – teams heavily affected by your department's success or failure are natural invitees.

Present thorough data analysis of lead times, defects, throughput, value from releases, etc. The point is to highlight issues and garner input as well as to celebrate successes when the data is all telling a positive story. This is an objective performance review, not a subjective 'what worked well, what didn't' discussion.

Make the meeting frequent. We've often seen quarterly reviews, and it is sometimes suggested monthly, but some companies may feel that the transaction cost of coordinating really large teams is too high. As with most of these decisions there is a trade-off between learning collectively versus creating local optimisations. The goal should be a feedback loop that encourages continuous improvement, it must be regular and not ad-hoc.

Figure 15. A team discussing the data analysis

3.3. Deep Kanban

When adopting a new way of working it is sometimes necessary to measure how the organisation is progressing against a chosen set of practices. This is a good way of guarding against only making the simple changes that fit easily into the current mindset and not realising the benefits from challenging the deeper beliefs. This implies progress may only being made on one or two dimensions only. For example, if a company were visualising work with some sticky notes on a task board, but were not yet limiting WIP, then they would have only a shallow implementation of Kanban. Although shallower implementations of Kanban might be useful and the team will see benefits – including transparency, collaboration and focus – they will not experience the full benefits.

The very idea of continuous improvement suggests the team needs to develop further, and the deeper the Kanban implementation, the greater the benefits. Those with deeper Kanban will see lead times reduce, predictability improve, throughput increase and a better relationship with business stakeholders. Full, deep Kanban should see all these benefits as well as a larger scale culture shift within the organisation. The idea of continuous improvement will be fully embedded, meaning that individuals or teams are empowered to make changes. Eventually the idea should begin spreading around the organisation as departments self-organise to improve.

To see how deep the Kanban implementation has gone we suggest you use something like a radar diagram plotting progress along each practice (shown as an axis). This helps focus attention on target areas in the Improvement Kata to show the individual team or the whole organisation's progress in Kanban. It can be measured either as relative progress (how far the team feel they are progressing towards their goal) or absolute progress (pre-setting specific targets or a scoring system for each principle). The absolute model is often more useful in a larger organisation where a central body may wish to set the targets or benchmarks across all teams.

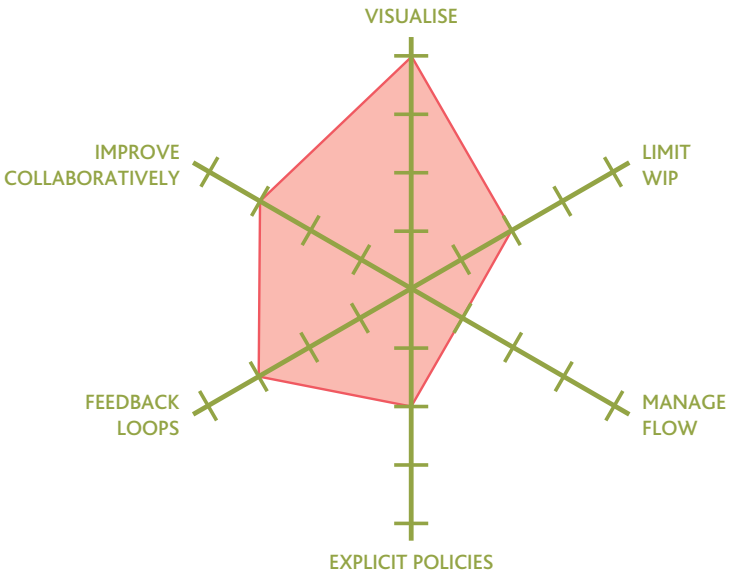


Figure 16. Radar chart to show the progress against the practices

Activity 6 – Evolutionary change

As with other lean improvement approaches you should start with your existing process. It is 'evolutionary not revolutionary'. This means that change should be easier to initiate – who wouldn't sign up for that? The best way for you to understand this is to apply practices to a product development endeavour of your own.

Before you begin, you will need to select a team that is prepared to take part in the experiment. There are many factors to be considered when choosing, most of which depend on your specific context. The best approach is to discuss it with others – start by identifying projects that are in flight, or about to start, look at the type of work they will be undertaking, consider the risks involved and finally talk to the teams – are they prepared to try a different way of working?

Okay, let's get started. First you need to visualise your workflow. Look back at section 1.1 of this session and follow the guidance there. You should end up with a kanban board mapping your current workflow.

Next you'll need to decide how to show individual pieces of the work progressing through the process. Revisit Activity 2 for guidance here, but at a minimum you should usually have a card that represents each piece of work – Post-its and index cards are most common.

Now's the time to focus on, perhaps, the most important aspect: WIP limits. The general suggestion is that there should be between 1 and 3 pieces of work per person at a specific activity state in your process. If you have 4 developers in your team, there should never be more than 12 items of work either actively being worked on, or waiting to be worked on. If you practice pair programming, you could drop that to 6 work items.

With these changes in place, run your process for a few weeks. You should be measuring the total lead time (the time from when an item of work enters the board until it has gone through all the activity states). This can be done by date-stamping each card twice – when it is pulled in and when it leaves the board – and then capturing both pieces of information.

After working with the Kanban method for a while, you should begin looking for opportunities to reduce lead time. As we suggested above, fine-tune WIP limits and batch size. Don't forget to record your assumptions and then prove or disprove them against the results you measure.

At the end of your experiment, review the data that you have gathered with the team that participated. What have you learnt by running the experiment? Facilitate an open discussion with your colleagues.

How deep have you gone?

About a month after you ended the experiment above, do this activity in discussion with your team as part of an Improvement Kata.

Draw out the 6 practices as axes, and measure how far your organisation is progressing. Of course, this is a subjective measure, so it helps if you can make the progress real using data.

For example, in 'limiting WIP' you could ask yourselves whether every team and every process has a WIP limit; or for managing flow you might consider what tools are used, what you would like to use and mark out each 'target' as steps.

3.4. Summary of Kanban benefits

- Optimise and improving existing processes rather than just swapping out an old way of working and completely replacing with something new.
- Limiting WIP and defining what type of work is recognised as 'engineering ready' allows us to optimise flow:
  - This leads to shorter, more predictable development lead times.
  - It also encourages a focus on quality, producing code with fewer defects meaning less rework or waste.
- Provides a sense of empowerment and satisfaction in their work to employees.
- Builds a customer's trust in the process and the team.
- Simplifying the process of prioritisation through classes of service saves time and ensures a focus on business outcomes.
- A measure of slack should be built into the system to enable improvement.
- The process and policies are transparent and subject to continual change based on evidence.
- Enables the emergence of a 'high-maturity' organisation.

4 CONCLUSION

As we stated at the beginning, Kanban explicitly sets itself up for evolutionary change, making small incremental adjustments to the existing system. It is simple to begin with a 'shallow' Kanban implementation where teams just visualise work. From here, the practices can be applied one-by-one, however gradually the team wishes to move.

As long as everyone understands the practices and principles in this session book, there is no need for special training or coaching. Every change is reversible – if you find that you have set the WIP limit too low and you are starving the pipeline, it is a simple matter to readjust it upwards.

That makes Kanban low-risk and light-weight. Most teams will accept it more happily because changes feel manageable and because the initial benefits from visualising work are felt almost immediately. As soon as the board is up, team members tend to gather round it, moving cards around, pointing out what help they need with a task and thinking about the impact one individual's work might have on another.

But while visualisation is powerful, it does not by itself enforce change – it simply helps focus attention on bottlenecks or other problems. Any big problems with variability, estimation and prioritisation are likely to remain. Kanban does not tell you what to do when business stakeholders cannot agree on a priority or if work cannot be broken down into equal-size tasks because the variability has value.

Thus Kanban works best when you combine it with other tools – batch size reduction, limiting variability, incremental delivery and disciplined feedback loops, for example, to drive specific quality improvements.

Sometimes teams will need to intelligently adapt the methods to suit their unique context. On large projects with big batches, lead time may not be a sufficiently granular measurement to track your progress. At other times several boards might be required – a project level board showing user story progress, and a team board showing task progress.

Kanban is an excellent idea, but it may not suit every team. Think hard about whether your company or team is at the right level of maturity to work with Kanban, or whether you would do better to combine it with other faster delivery processes.

Learning outcomes

Now that you've completed this session, you will:

Understand the practices of Kanban

- Visualise workflow:
  - A visual system to ensure the team know where each task is in the process at any given point
  - Each task should have an estimate, priority, person assigned to it
  - Pay particular attention to bottlenecks and 'blocked' tasks
  - Adapt and evolve the visual system required by your team
- Limit WIP:
  - Encourage flow, reduce queues and thus enable faster development
  - Introduce small buffers between activity states with WIP limits
- Measure and manage flow:
  - Track lead time of each item through the process
  - Use Cumulative Flow Diagrams to show progress
- Make process policies explicit:
  - Set up service level agreements to improve customer satisfaction by delivering against expectation
  - Capture and write up agreed changes to improve the process
- Implement feedback loops:
  - Understand the importance of feedback loops in building quality
  - Recognise the process feedback loops that lead to continual improvement
- Use models to recognise improvement opportunities:
  - Make explicit assumptions
  - Test and measure improvement using real data

The use of classes of service

- Optimise our flow and the economic outcome for the customer
- Expedite requests jump queues for shortest cycle time
- Limit numbers of each 'class' permitted at each time
- Fine-tune the mix to ensure low urgency but necessary items are scheduled

The separation of input and delivery cadence

- How to pull in new items in correct priority
- When/how frequently to release finished items
- The two processes do not have to be aligned, releases can be timed to fit our business model

Focusing on technical excellence

- Deliver high quality code as part of continuous improvement
- Low quality code requires rework and is a form of waste
- Through improved flow more time can be devoted to producing the best quality code

Improvement Katas

- Institute process feedback loops to ensure a kaizen mindset
- Encourage small 'experiments' towards a target

Operations Review Meetings

- Apply Kanban rigour of empirical improvement to achieve organisation-wide improvement and coordination
- Hold regularly depending on transaction cost

Measure Kanban depth and progress on goals

- Ensure a focus is placed on the more challenging aspects of culture and mindset
- Increase understanding of goals and help the team understand the direction of improvement

How to overlay Kanban on existing processes

- Presupposes an existing process
- Change occurs through evolution, not revolution

BIBLIOGRAPHY

Anderson, D., 2010. *Kanban: Successful Evolutionary Change For Your Technology Business*. Blue Hole Press.

Anderson, D., 2013. *Deep Kanban. Worth the Investment?* Keynote speech for London Lean Kanban day. [online] Available at: <http://www.youtube.com/watch?v=JgMOhitbD7M>. [Accessed 5 October 2013].

Coppola, A., 2013. *Intro to Kanban in Under 5 Minutes*. [online] Available at: <http://www.youtube.com/watch?v=R8dYLbJiTUE>. [Accessed 5 February 2014].

Cox III, J. F., Schleier, J., 2010. *Theory of Constraints Handbook*. McGraw-Hill Professional.

Forss, H., 2013. *Habits of Kanban process improvements*. [online] Available at <http://hakanforss.wordpress.com/tag/kanban-kata/>. [Accessed 5 October 2013].

Foy, C., 2011. *Recreating Scrum using Kanban and Explicit Policies*. [online] Available at: <http://blog.coryfoy.com/2011/07/recreating-scrum-using-kanban-and-explicit-policies/>. [Accessed 10 February 2014].

Hefley, C., 2011. *Make Process Policies Explicit*. [online] Available at: <http://leankit.com/blog/2011/06/make-process-policies-explicit/>. [Accessed 10 February 2014].

Howard, S., 2012. *Kanban: Understanding Kanban Method*.

Joyce, D., 2013. *Systems Thinking, Lean and Agile*. [online] Available at: <http://leanandkanban.wordpress.com/>. [Accessed 27 January 2014].

Kniberg, H., 2011. *Lean from the Trenches: Managing Large-Scale Projects with Kanban*. Pragmatic Bookshelf.

Ladas, C., 2009. *Scrumban – Essays On Kanban Systems For Lean Software Development*. Modus Cooperandi Press.

Middleton, P., Joyce, D., 2010. *Lean Software Management: BBC Worldwide Case Study*. [online] Available at: <http://leanandkanban.files.wordpress.com/2011/04/lean-software-management-bbc-worldwide-case-study-feb-2011.pdf>. [Accessed 1 August 2012].

Mundy, S., 2013. *Kanban can deliver and make you mindful*. [online] Available at: <http://www.cio.co.uk/insight/change-management/kanban-can-deliver-make-you-mindful/>. [Accessed 2 February 2014].

Peterson, D., 2009. *What is Kanban?* [online] Available at: <http://www.kanbanblog.com/explained/>. [Accessed 7 February 2014].

Ries, E., 2011. *The Lean Startup: How Constant Innovation Creates Radically Successful Businesses*. Portfolio Penguin.

Rother, M., 2009. *Toyota Kata*. McGraw-Hill.

Shalloway, A., Beaver, G., Trott, J., 2009. *Lean-Agile Software Development: Achieving Enterprise Agility*. Addison-Wesley.

Stevens, D., 2010. *Kanban: What are classes of service and why should you care?* [online] Available at: <http://www.dennisstevens.com/2010/06/14/kanban-what-are-classes-of-service-and-why-should-you-care/>. [Accessed 12 February 2014].

van den Heuvel, M., 2011. *Explicit Policies Make Life Simpler*. [online] Available at: <http://scrumfamily.wordpress.com/2011/10/10/explicit-policies-make-life-simpler/>. [Accessed 9 February 2014].

Womack, J., Jones, D., 2003. *Lean Thinking: Banish Waste and Create Wealth in Your Corporation*. Free Press.

Wester, J., 2013. *What is Kanban?* [online] Available at: <http://www.everydaykanban.com/what-is-kanban/>. [Accessed 27 January 2014].

Wikipedia. *Kanban (development)*. Available at: < http://en.wikipedia.org/wiki/Kanban_(development)>. [Accessed 29 January 2014].

Emergn Ltd.
40 Bank St, Level 18
Canary Wharf, London E14 5NR
UK/EMEA: +44 (0)808 189 2043

Emergn Ltd.
20 Harcourt Street
Dublin D02 H364, Ireland
IRE: +353 (0)1 554 7874

Emergn Inc.
One International Place
Suite 1400, Boston, MA 02110
North America Toll Free: +1 888 470 0576

bcs The Chartered Institute for IT