A Kanban primer for Scrum Teams

Steve Porter

June 21, 2017

★ 4.3 from 6 ratings

Subscribe

All you need to know to improve your Scrum using Kanban

Several of us in the Kanban and Scrum community got together recently to build a bridge between Scrum and Kanban. A more accurate way to put is that we got together to document the bridge that is already connecting these two worlds. We are writing a series of blog posts looking at this bridge from different perspectives.

This time around Yuval Yeret joins us to write this blog post. Yuval is known as "Mr. Kanban Israel" for his work helping establish a strong Kanban community with several enterprise product developments in the "Startup Nation." Today, he leads the Scaled Agile practice and North America consulting services at AgileSparks, a global Agile consulting firm. Yuval and AgileSparks have been bridging Scrum and Kanban in the trenches for several years.

Yuval Yeret

Kanban Values / Core Principles

Kanban is a method that uses a work-in-process limited pull system as the core mechanism to expose operation (or process) problems and to stimulate collaborative improvement efforts.

Kanban practitioners follow a set of core principles that guide the way they apply the Kanban practices. These principles are:

- Start with what you do now
- Agree to pursue incremental, evolutionary change
- Respect the current process, roles, responsibilities, and titles

Kanban’s principles - Easy to live by if you’re starting from an existing Scrum implementation

If you’re already using Scrum and want to add some Kanban practices, you’ll find its principles make a lot of sense. Use Kanban’s practices to guide your empirical inspect and adapt process. Kanban’s practices don’t require you to change the current Scrum roles, events, and artifacts as documented in the Scrum Guide.

The Kanban Core Practices

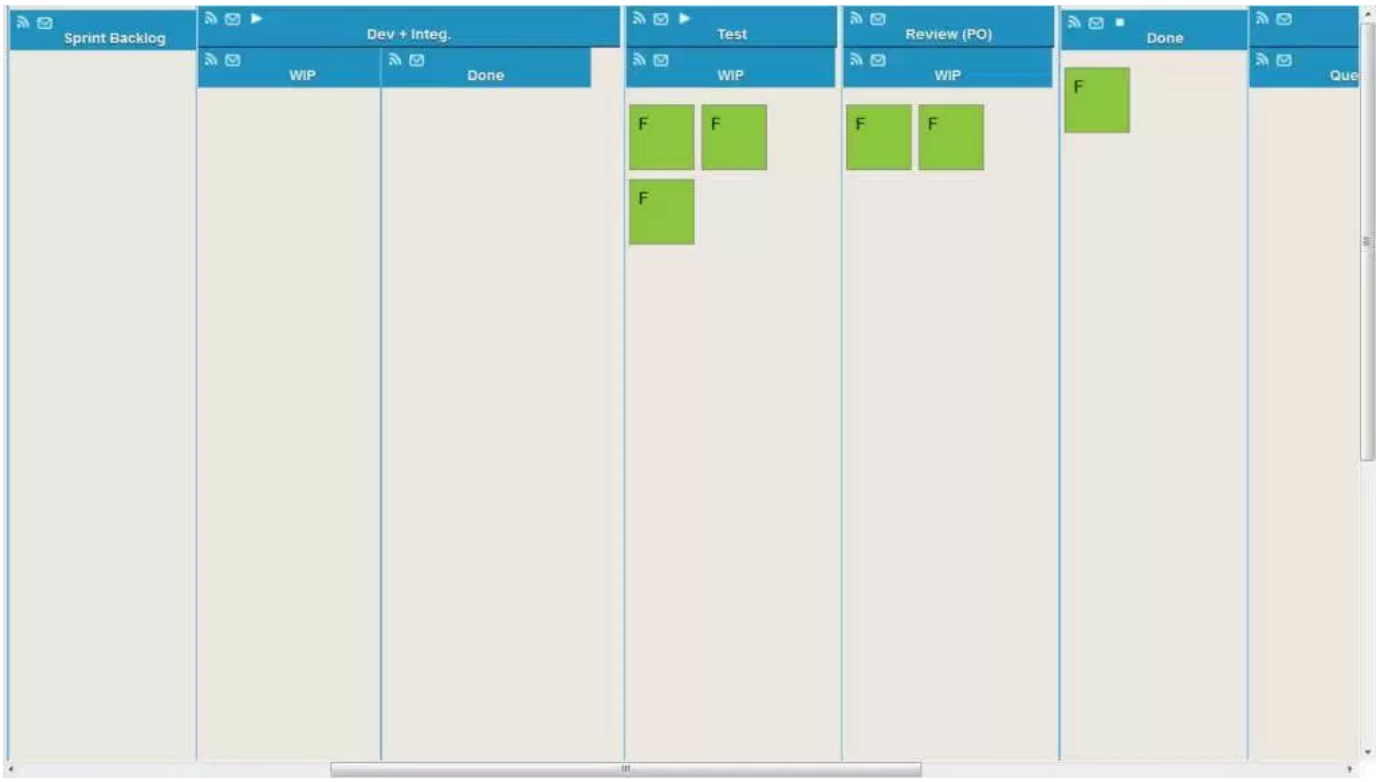
1. Visualize (the work, workflow, and risks to flow/value delivery)
2. Limit WIP (Work in Process)
3. Manage Flow
4. Make Process Explicit
5. Implement Feedback Loops
6. Improve Collaboratively, Evolve Experimentally (using models and the scientific method)

Let’s look at what these mean when applied in combination with Scrum:

1. Visualize (the work, workflow, and risks to flow/value delivery)

Many Scrum Teams already visualize work and its flow using a Scrum board. This practice can be a great way for teams to keep their plan up-to-date and transparent, to see opportunities to collaborate in order to complete the Sprint Goal, and to identify forecasted items that may not be completed by the end of the Sprint.

One of the keys to effective visualization for a Scrum Team is to make sure it sees the flow of Product Backlog items from idea identification, through refinement, analysis, design, coding, testing, and deployment; all the way to "Done." This is what Kanban teams call the value stream. It is not enough to simply visualize the Sprint Backlog. It is important to make the flow of work into and out of the Sprints visible as well because many teams have difficulty with this aspect. Visualizing the flow throughout the value stream means we should mainly visualize the flow of Product Backlog items (PBIs) rather than tasks.



2. Limit WIP (Work in Process)

Once you have a Kanban board visualizing the flow, the next step is to implement a Kanban system. This calls for limiting the Work in Process, which in a Scrum context

means you limit the amount of PBIs in progress at any time. As professional Scrum practitioners should know, the Sprint itself is a WIP Limit. A Sprint limits the number of PBIs that the Developers forecast they can complete during a fixed period. This is such a powerful aspect of the Sprint that I wish more Scrum Teams would understand and use it. What typically happens when I talk to Scrum Teams about Kanban and limiting WIP is that they either come away with a deeper understanding of why the Sprint has been helping them focus and collaborate or they recognize that they've been missing something important in how they have been implementing Scrum. For both these cases, taking it one step further and using more granular limits on the number of PBIs active at the different stages of the team's flow during the Sprint helps improve that flow, focus, and collaboration even further.

Typically, Kanban boards specify WIP limits in different stages of work that the Developers perform (e.g. design, coding, testing, deploying, etc.) Once the limit is reached in one stage, instead of starting a new PBI, a Developer would help others deal with the current PBIs already in progress. This can mean helping others in your area of expertise or helping others in their areas of expertise (e.g. coders help testers, content marketers help social marketers, etc.). This virtual Kanban system can extend beyond the Sprint boundaries to include activities across the entire value stream such as discovery activities, refinement of PBIs, and production support.

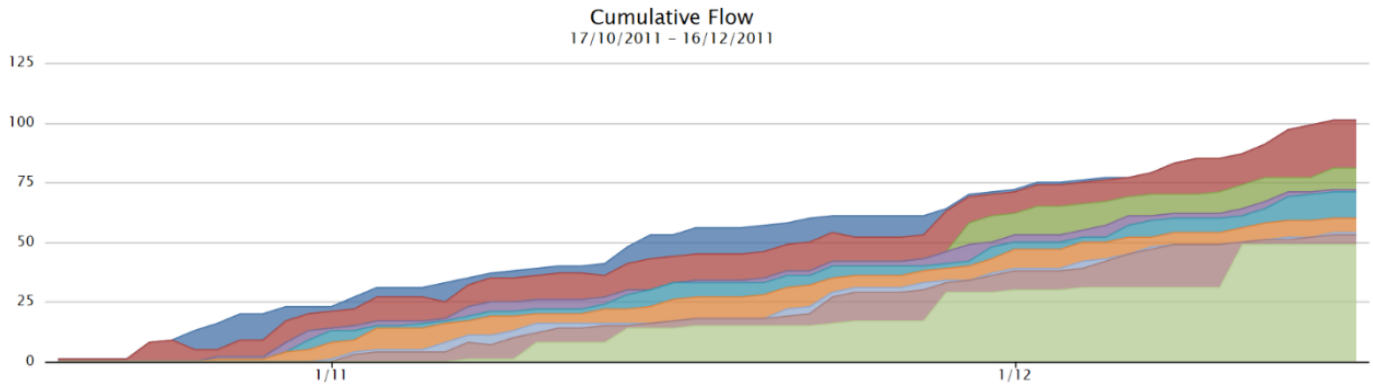
WIP limits should be low enough that they introduce pain-- meaning that they push you beyond what you're comfortable with and used to. You want to channel this pain towards ways of improving the team's process and policies so that in the future, the flow will be better and the WIP limit will be less painful. Improving engineering practices, using techniques such as acceptance test-driven development (ATDD), reducing batch/PBI size, cross-training, or any other technique are all ways to make the WIP Limit more comfortable. Over time, as team capabilities improve, the WIP limit can be tightened to catalyze even more improvement in capabilities. Limiting WIP is an effective way to drive deep collaboration at the team level.

3. Manage Flow

With the virtual Kanban system in play using WIP limits, it is now time to run the system and manage the flow. The team will look at the cycle times of PBIs for information on where there is room for improvement. Cycle time is a measure of the amount of time that has passed as an item on the Kanban board moves between two stages in a value stream. Teams monitor aging/staleness on an ongoing basis (e.g. during the Daily Scrum) to identify stalled/struggling PBIs and find ways to help them along. Here, the team's focus is on the healthy flow of work. It may start using cumulative flow diagrams, which add information about the queues within the system to

a burnup chart. The Scrum Team may also identify opportunities to decrease the amount of time it takes to get work to production. For development work inside a Sprint, Scrum directs teams to have that work be "Done" by the end of the Sprint, but if teams want to increase their flow, they should look for ways to release the work as soon as it's ready. There's nothing in the Scrum Guide that says you have to wait until the

end of the Sprint. (See https://www.scrum.org/resources/blog/sprint-review-not-phase-gate for a similar perspective)



4. Make Process Explicit

By making your process explicit, you enable empowerment. Kanban encourages Scrum Teams to bring additional transparency to their process, not just their work. Based on this transparency and clarity team members can be empowered to self-organize (See David Marquet's leadership approach based on his book, Turn the Ship Around!).

Making the process explicit also helps Scrum Teams improve their existing processes. With the "invisible" process now made visible, healthy discussions about questions such as, "why do we do things this way?" or "how will changing this policy affect our flow/results?" become easier and happen more naturally.

Scrum's definition of "Done" is an excellent example of an explicit policy. Another example is the Scrum framework itself. The Scrum Guide is an explicit policy that a Scrum Team implements. Kanban teams add explicit policies such as the WIP limit for each lane or each person, escalation/ class of service policies, How to visualize and deal with blockers, How to prioritize work.

5. Implement Feedback Loops

Scrum Teams use Sprint Planning, the Daily Scrum, the Sprint Review, and the Sprint Retrospective as examples of feedback loops for the process and the product.

To take it one step further, consider running a Sprint Retrospective in a quantitative data-driven way. By using reports such as cycle time control charts and cumulative flow diagrams, Scrum Teams can gain deeper insights into the flow of work, thus driving improvement experiments that don't surface as a result of a classic Sprint Retrospective.

Scrum Teams should naturally embrace this empirical approach to process improvement.

6. Improve Collaboratively, Evolve Experimentally (using models and the scientific method)

Professional Scrum Teams should have already embraced Kanban's "Improve Collaboratively" practice. The core purpose of the Sprint Retrospective is to give all members of the Scrum Team the ability to inspect and adapt how the team works. As part of the Sprint Retrospective, Scrum Teams should consider adding the use of models and the scientific method to guide their evolution empirically. Retrospectives are the place to design experiments based on ideas of how something might improve the team's capabilities.

Plan what the team aims to test and how it will validate that it is heading in the right direction. In the next Retrospective, review the results and decide whether another round of experimentation is necessary (pivoting) or whether the team can deploy the change as a standard operating policy.

Use improvement frameworks such as A3 / Toyota Kata for this process. Explore models such as Reinertsen's principles of product development flow to understand the interaction of batches, cadence, heartbeat, teams, variability, WIP, and cycle times in ways you hadn't before.

Where This Gets Us

In this post, I outlined Kanban's guiding principles and practices as applied to a Scrum context. I only hinted at the specific improvements that Scrum Teams are likely to see as they incorporate Kanban practices. I did this on purpose. These sorts of improvements are NOT prescribed by Kanban. Perhaps you will see these improvements emerge if you implement Kanban on top of Scrum or you may see something different. Kanban is just a method. It is not a methodology.

In future posts in this Scrum and Kanban series, we will provide examples and stories

In future posts in this Scrum and Kanban series, we will provide examples and stories about where applying Kanban principles can take Scrum teams.

What did you think about this post?

4.3 from 6 ratings

Share with your network

Blog Comments

26 Comments Login

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS

Name

Sort by Best 13

Dennis • 5 years ago

I'm working in an agile environment in the last 9 years. I've worked both in Scrum and Kanban. In the last few years I've come to a conclusion that it doesn't really matter which one you choose as long as you follow the core principles.
Here is my post about this matter:
“Scrum? Kanban? It doesn’t really matter” https://medium.com/dennis-n...

1 • Reply • Share ›

Steve Porter Dennis • 5 years ago

Nice post. It was a good read and had some nice principles attached to it.

• Reply • Share ›

Dennis ➚ Steve Porter • 5 years ago

Thank you

^ | v • Reply • Share ›

Rita • 9 months ago

Very nice article and introduction to Kanban methodology indeed :) Personally I use kanbantool.com at my work place to apply all the rules in practice :) Many times it helped me to survive especially with too many projects at hands :D

^ | v • Reply • Share ›

Pedro Veas • a year ago

Hi Steve, I'm starting to preparing for Kanban certification, but I'm confuse about the batch size meaning in Scrum with Kanban context, in particular the reducing batch size, It's mean reduce the size of an PBI (splitting it for example), reduce the PBIs pulled to the Sprint Backlog or both?

^ | v • Reply • Share ›

Steve Porter ➚ Pedro Veas • a year ago

The short answer is, yes. When I think of Scrum, there are a few different batches to consider.

The PBI can be looked at as a batch.
The Sprint as a whole could be looked at as a batch.
If you don't release at the end of every Sprint, then your releases are batches.

If each of these cases, you should ask yourself is your batch the right size.

^ | v • Reply • Share ›

Pedro Veas ➚ Steve Porter • a year ago

It's clear for me now, thanks Steve!

^ | v • Reply • Share ›

Govindraj Tungenwar • 2 years ago

One of the best article on Kanban in Scrum

^ | v • Reply • Share ›

yuvalyeret ➚ Govindraj Tungenwar • 2 years ago

Glad you found it valuable Govindraj! Feel free to mention @Scrum.org and myself when you share it with others :-)

^ | v • Reply • Share ›

Steve Porter ➚ Govindraj Tungenwar • 2 years ago

Thanks, I'm so glad you found it valuable.

^ | v • Reply • Share ›

Romil Gandhi • 4 years ago

After reading Scrum with Kanban guide, in the section "Definition of Workflow", I am confused

between the terms "stages" and "state" or they are same as I had interpreted?

• Reply • Share ›

Steve Porter ➚ Romil Gandhi • 4 years ago

It should be states. They are interchangeable. The next revision will replace stages with states.

• Reply • Share ›

Romil Gandhi ➚ Steve Porter • 4 years ago

Thanks.

• Reply • Share ›

Steven Spruce • 4 years ago

Why 'Work in Process' rather than 'Work in Progress'?

• Reply • Share ›

yuvalyeret ➚ Steven Spruce • 4 years ago

Good question Steve. It's somewhere in between interchangeable and preference on my end. I like "Work in Process" a bit more since it encompasses all work that is in your process, not just stuff that is progressing. Actually one of the things we emphasize in Kanban and specifically in Kanban for Scrum Teams is the focus on work that is aging in the process, not necessarily progressing...

Hope this makes sense :-)

1 • Reply • Share ›

Steven Spruce ➚ yuvalyeret • 4 years ago

It does....thanks both.

• Reply • Share ›

Steve Porter ➚ Steven Spruce • 4 years ago

I'm pretty sure that's just Yuval preference. I've heard him give talks and he usually uses Work in Process. I don't think he see's a difference, but I'll make sure that he sees this question.

• Reply • Share ›

aditya pandey • 4 years ago

Very well articulated ! could you please highlight some of the approaches (information radiators) to convince client to adopt KANBAN in scrum as well.

• Reply • Share ›

Steve Porter ➚ aditya pandey • 4 years ago

Hi Aditya,

If you think that your teams would benefit from Kanban, the best place to start is to get

them to visualize their flow of work. Where the work comes from, at what point do you consider the work "started", what are the steps in the knowledge discovery process and when do the teams consider the work to be "finished".

Then decide the amount of work that you want to have active (work that has "started" and not "finished") at any one time. This is your Work in Progress (WiP) limit.

Once you've completed that, start tracking the date that work "started" and the date that work "finishes". You can additional track the date work begins a new step in the knowledge discovery process, but that's not needed to get started.

Do that for awhile you'll have a good history of your work's cycle time.

Then review your team's workflow, and collectively suggest improvements and measure it's impact on team. Is your WiP limit too high? Do things take too long to get finished? Can we improve that?

This should be enough to get you started.

 | • Reply • Share ›

Manuel Navarro Pérez • 4 years ago

Nice post!!
Only a question:
Is it the development team that must decide whether to use kanban scrum? Or the scrum team?

Thank you!

 | • Reply • Share ›

Steve Porter ➚ Manuel Navarro Pérez • 4 years ago • edited

Thanks!

There's some information on that in the Kanban Guide for Scrum Teams.

In the end, it's very dependant the Scrum Team's definition of “Workflow".

 | • Reply • Share ›

John Coleman • 5 years ago • edited

As a practitioner, coach and trainer for both Scrum and Kanban, I have run simulations using a variant of Mike Burrows' Featureban (cost of delay, features of varying size/value/cost of delay/dependency complexity). In that variant, I simulate Scrum and Kanban, and I use Toyota Kata style retrospectives, referring to the stats on lead time, time in process, WiP, delivery rate while referring to explicit policies on replenishment/planning, pull, WiP, collaboration. I have the honour of training people with the most resistance to agility and really toxic behaviour, antithetical to agility. Even the most toxic behaviour doesn't break the benefits of optimizing WiP (even if benefits get curtailed - I had an extreme example recently); more value gets delivered faster with far more predictable lead times. In those simulations, Kanban style replenishment/pull almost aways outperforms Scrum style planning (blog post at valueglide.com coming soon). I like Don Reinersten's take on Scrum being good for reducing variability through its stop/start (review of increment/retro, planning new work) approach. Kanban's approach to risk management is

sophisticated but simple. As a Development Team, consider Kanban on top of Scrum also as a coping mechanism (to keep predictability and value delivery rate) for "drive-by" scope drops in situations where Scrum is implemented badly (changes let into sprint for all sorts of reasons, which I am totally against for Scrum implementations by the way, and if I was the Scrum Master, I would fix that as manager of the Scrum framework). With regard to Kanban, I would refer to the 2016 Essential Kanban Condensed Guide which has a fresher take on the principles and practices .

 • Reply • Share ›

Steve Porter ➚ John Coleman • 5 years ago

Thanks John, it sounds like you've had a lot of great experiences with both Scrum and Kanban.

Daniel and I are hoping to get people to take a different view of 'Scrum style planning'. Stay turned for an upcoming post where were tackle that.

I've never been opposed to "drive-by" scope changes if the work can be pulled by the Development Team and the Sprint goal stays intact. In a complex environment, the opportunity to deliver value may not happen on a fixed cadence.