

4.7: Kanban(ish) Variant of Scrum

There is another agile method, called Kanban, that is becoming popular for software development. In this chapter we describe its main strength, and how it can be integrated into a Kanban(ish) version of Scrum.

Table of Contents

<i>Brief Description of Kanban</i>	1
<i>Integrating WIP into Scrum</i>	3
<i>Sprint is still there</i>	5
<i>Discussion</i>	5

As much as we love scrum, even we would have to admit that it's not perfect. Nothing is. In fact, a large part of this book describes workarounds for various deficiencies that scrum presents to us in certain circumstances.

One of the more commonly noted deficiencies in scrum is that it plans its work a whole Sprint at a time. This "batch" planning process is often not agile enough to cope with the actual rate of change of requirements. In fact, Chapter 4.4 on Placeholder Stories, the discussion of the mid-Sprint Re-planning in Chapter 4.8, and the discussion of renegotiating the scope of a Sprint in Chapter 4.3 are all about resolving this deficiency.

There is another agile process, called Kanban, which solves this problem and is becoming popular for software development projects. In this chapter we will describe the main strength of Kanban and how to integrate it into scrum.

Brief Description of Kanban

The "Kanban for software" movement is led by David Anderson¹, and is really gaining some traction in the agile community. The main idea of Kanban is very simple and based on the Lean "pull," "Just in Time" (JIT), and "reduce inventory" principles: eliminate planning inventory by making sure that you don't commit to doing work until you are actually ready to start the work.

¹ <http://www.agilemanagement.net>

In other words, eliminate the “future” planned-out work that causes us problems in scrum by planning Just in Time; only commit to new Stories when the Team becomes available to work on them by finishing old Stories.

This leads to the notion of Work In Progress (WIP), which is a fixed-length collection of items (or Stories) that are actually being worked on all the same time. This WIP replaces the Sprint Backlog, which is a collection of work that has been committed to being done during the Sprint, which means that some of it will not be started until later.

In scrum the Sprint length is the primary variable we play with, while in Kanban it is the WIP Length, which is the number of items that are being worked on at the same time. Typically, the WIP Length is a small number like two or three, and is usually slightly less than the total number of coders (not developers) that are on the Team.

What does this change give us? Well, here’s a short list of benefits:

- There is no prediction about what we’ll be doing later in the Sprint. The Team is actively working on all of the Stories it has committed to so far. This makes the Team more agile by reducing whatever predictive thinking may have been present in the scrum team;
- If the Stories that are being worked on are small, there is very little need to renegotiate what is being worked on when “something comes up.” This is because if the Stories are small some part of the Team will be finishing their current Story soon, and will be available to take on additional work as part of the normal WIP selection; and
- By its very nature, this process causes Team Swarming behavior, because of its focus on the “single item flow” concept of Lean. That is, the subset of the Team that just finished its Story now just “jumps on” the next Story that is elevated to the WIP. Also, the system lends itself to having one coder on each swarming TeamLet, and having one floating coder to help out.

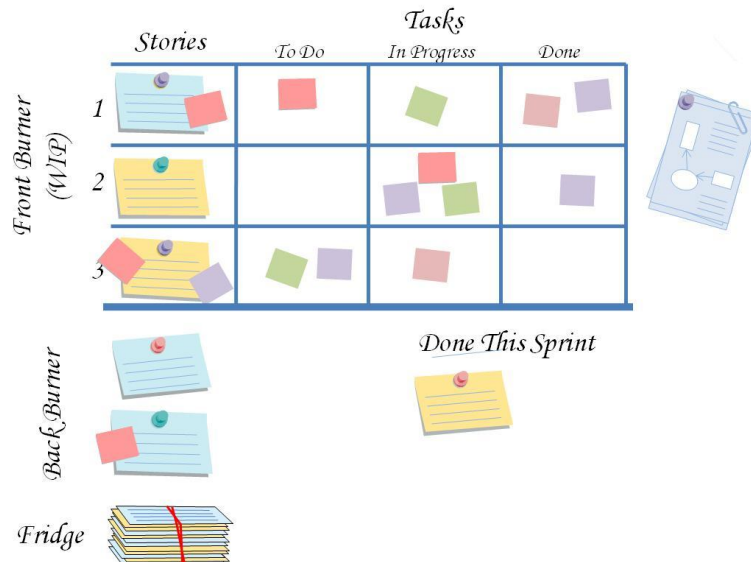
Now, Kanban can be made much more complex than I stated here. Is not our intention to adequately, or accurately, define Kanban in all its glory. It is our intention merely to capture the essence of what makes Kanban good, and figure out a way to integrate it

into scrum. If you want a very thorough discussion of integration of kanban and scrum, read Corey Ladas' book².

Integrating WIP into Scrum

Well, now we know what the basic idea is, what actually happens when we integrate the WIP concept into scrum?

First of all, we simply replace the Sprint Backlog with Work In Progress. In terms of the planning board, the Sprint Backlog is replaced with a fixed length Work In Progress, as shown in the following picture, where the WIP length is three (3).



This picture makes it seem like a relatively simple thing to do, but it isn't really. It raises a lot of questions about process and practices, so we'll just make a list of the issues/changes here (in no particular order):

1. Clearly, Sprint Planning changes. The Sprint planning meeting we would normally have goes away and is replaced by on-the-spot planning meetings that take place whenever a Story in the WIP is completed. When a Story is completed, the Product Owner and the Team reprioritize the backlog and select the next Story to be done, finalize the negotiation about the "doneness"

² Corey Ladas, ScrumBan and other Essays on Kanban Systems for Lean Software Development, Modus Cooperandi Press, 2008.

Agreement, and then simply elevate the Story to the WIP and commence work on it. This may also cause a reorganization of the people across the Stories in the WIP.

2. The Back Burner must have Stories that are “ready to go” whenever the WIP has room to move a new Story up. This means there must be continuous grooming, and probably requires the use of analysis Stories (which must be selected into the WIP just like any other Story), just as before. However, the Back Burner can be shorter than it would have been to have a full Sprint’s worth of Stories to plan with. This is a good thing, as it follows the Lean principle of “minimize waste” by having fewer well-thought-out Stories ready to go at any given time.
3. Renegotiations during a Sprint are different, in practice if not in theory. Since there is no pressure to finish “what we planned for” it is easier to stop working on a Story that is in progress in order to go work on an emergency one. Not only that, if the Product Owner can wait a day or two for the new Story to start, then it can just be moved into the WIP as a matter of course.
4. Since we’re not trying to accomplish a certain number of Stories (or Story Points) within the Sprint, there is no need for Sprint BurnDown/BurnUp graphs.
5. However, since we’ll still be Reviewing and Reporting on a Sprint-by-Sprint basis (see the next section), we need to keep track of which Stories have been finished in the current Sprint.
6. We still need to make sure we have a robust definition of “done” in our Agreements. It is tempting to relax the definition of “done” because there is no apparent time pressure within the Sprint. Don’t do this! It can lead to gold-plating Stories (or scope creep inside the Story), which will “steal” time from the future, which will make it harder for us to be successfully agile in the long run.
7. Finally, it is easier for the Team to have a rule to manage Chores, such as the “70/30 rule.” All the Team has to do is make sure that one third (or some appropriate percentage) of the Stories that are elevated to the WIP are chores, and the deal is done. Even easier, if the WIP Length is three (or whatever is appropriate), we just need to make sure that one of our “slots” in the WIP is reserved for chores.

This list may not be exhaustive, but it should give you a good idea of the changes you have to make to accommodate integrating the WIP into scrum, and having it replace the Sprint backlog (Front Burner).

Sprint is still there

It is important to remember that we are still doing scrum – there is still the Sprint. However, its purposes have slightly changed, as we no longer plan the whole Sprint at one time – we plan it continuously by managing the Work In Progress.

So, what do we use the Sprint for? Why is it still Important?

Good questions, with good answers, as follows:

1. The Sprint still defines our major Review Cycle. We still review the results of the Sprint with the Stakeholders on a consistent cycle. As usual, there is no “partial credit” in scrum, so we review only the Stories that were completed in the Sprint, including those that were in the WIP at the beginning of the Sprint and finished within the Sprint. The Stories that are still in the WIP at the end of the Sprint are not reviewed. As usual, this consistent drumbeat of Sprint Reviews is comforting to our Stakeholders, and also allows us to produce metrics and do Release Planning as we normally would.
2. We continue to do Retrospectives at the end of every Sprint in order to improve the Team’s process.
3. We continue to update the Release Plan at the Sprint boundaries, based on the Sprint Review and Stakeholder needs. Each Sprint still has Sprint Goals, and they are used as guidance when the WIP is updated.
4. We calculate our Release-based Metrics Sprint by Sprint, in order to show “how we’re doing” – see the sections on Release Planning and Measuring (sections 6 and 7).

Basically, all that has changed by adding the WIP to scrum is the method that we do the first level of Strategic Agility – determining what work we are doing in this Sprint. The Tactical Agility is the same – the Team still does its self-organization to meet the “doneness” Agreements they have committed to. The higher levels of Strategic Agility are also unchanged – the Release Planning is still at the Sprint Boundaries as before, and the Sprint Goals are still useful.

Discussion

It is very tempting to want to do this Kanban(ish) version of scrum; at first appearance it is quite appealing. However, it is actually harder to do correctly than “normal” scrum, and should only be attempted once it is clear that the benefits outweigh the risks.

It is beneficial when there is constant re-planning within a Sprint. Managing to the Work In Progress replaces the re-planning with constant planning and avoids unnecessary planning that later turns out to “be wrong.”

However, when using this variant it is tempting for many teams to relax the commitment to using the “doneness” Agreements to manage scope creep and technical debt inside Stories. This is because many people see Kanban as saying “just do the Story until it’s done” rather than “do it as quickly as you can, while meeting the “doneness” Agreement.”

For basically the same reason, it is also tempting to commit to larger and larger Stories rather than break them into small bits. This “just do it until it’s done” attitude can lead to never-ending Stories with significant scope creep inside them – look out for this!

However, if the Team can stick to the discipline of using the “doneness” Agreements, then go ahead and try it. Because there are only a few items on the WIP, Kanban leads to swarming – where the team (or subset) works together on a story until it is done.

Unfortunately, it sometimes leads to stove-piping – the subdividing of the team into permanent sub-teams that work together. This is because when one team finishes a story in the WIP, it is tempting to just assign the next story to them as it moves up into the WIP.

So, you want the Swarming to involve most of the team swarming across all of the stories. That is, when we get a new story moved up to the WIP, assign one person to work on that story full-time. So, each story in the WIP has one full-time person assigned, and the rest are swarmers.

In short, I think that this Kanban(ish) variant of scrum is a good and natural thing to do when a team matures, if the Team is encountering significant requirements churn that results in constant re-planning and re-negotiating. It is a good tool to have “in your back pocket” to use in those occasions.