Combining Kanban and Scrum - lessons from a team of sysadmins

Katarzyna Terlecka
Wrocław, Poland
kate.terlecka@gmail.com
Twitter: @katerlecke

Introduction

In 2010 I have had an opportunity to work with a team of fantastically intelligent system administrators. Their team dynamics and project was completely different from software projects I have worked with before. We have tried Scrum which did not work at all due to too unstable environment. We also failed using Kanban because of the lack of boundaries, which led to coming back to guerilla-style work. Over time we have learnt that it is advised to draw as much as you can from Scrum into Kanban to make it easy and sustainable for long. The team came up with a Scrum and Kanban hybrid which is suitable for teams with high level of maintenance and ad-hoc work.

Context

In 2010 I was asked to introduce Scrum to a team of system administrators. Given my previous experience in purely software teams, it appeared to be very interesting. The team consisted of five guys: Matt, Bill, Leo, Pete and Tim (names changed). They were not exactly a team at that time, they were storming1 pretty heavily. One of the biggest challenges was the exclusivity of skills. Two Windows guys (Pete and Leo) were taking care of one sub-network and three Linux admins with their own network and source control systems. One group knew nothing about others and it wasn't only because of the eternal Linux-Windows war. They have had responsibilities and access to different parts of the network and those due to security policies could not be shared.

When it comes to skills level, Tim was an expert, but Matt and Bill were just astonishing. For windows guys Pete was experienced and Leo was just starting, but very curious and promising. All of them were extremely intelligent and those personalities and politics were sparking conflicts constantly.

First Try - Scrum

First step was the adoption of Scrum, since we were asked to do just that. I understood the reasoning behind the request – it was working nicely in other teams and the management wanted results here as well. I warned them that first few sprints would not be successful. We struggled for five weekly sprints. I have not seen anything like this before, but we failed miserably at each event, starting with planning, which was a never-ending fight, then failing to conduct daily scrums (sometimes at all). Putting all five of those guys together next to the board was like herding cats - I managed to do this just once and I had to use candy. It appeared as a disaster. All of the teams I have taken care of before grasped at least something off Scrum and they were on the road uphill. But not those guys.

So, a complex and very unstable environment, lots of testosterone mixed with extreme intelligence and me. This was looking fun.


Those five guys had around 1700 users distributed across 3 locations and 1000 square meters of a telecommunications laboratory space to take care of. They've had an engineer to look after electrical stuff, but that was all the help they could get. They were administering close to 100 various servers, spread across 4 server rooms in two locations in the city. Because of that the amount of ad-hoc work they've had usually exceeded 50 % of their time. But we didn't know it just yet. Oh, and ad-hoc did not mean a 30 minute task – that was serious business. Maintenance was usually 2-3 minutes, up to 5. Setting up accounts, verifying new user's credentials, pushing an installation to the new computer etc.

So, a complex and very unstable environment, lots of testosterone mixed with extreme intelligence and me. This was looking fun.


978-0-7695-4804-3/12 $26.00 © 2012 IEEE
DOI 10.1109/Agile.2012.20
became estimable and we saw its 50-60%) we started looking for another solution.

**Second Try - Kanban**

A friend of mine familiarized me with the idea of Kanban, which was used to manage flow. Yes, flow was something we could use.

We have already had a good understanding of the pull-approach, in fact it was implemented, so we have organized ourselves a head start. We set up an initial board, started working and after five days everything crumbled again. The team was lost – even though Scrum did not work before, they needed that impaired but still beneficial planning, measurements, the care of the Product Owner, someone to guard the process and take care of them, estimations and progress projections and before all, the feeling and view of progress. Without it they came back to worse than ever guerilla-style approach. After some time and lots of effort they probably would be able to come up with their own systems of work, but the expense of time and effort would be too great and we did not have that time to spare. One day we brainstormed how the new board should look like and we decided to marry Scrum and Kanban.

**The Scrum And Kanban Hybrid**

**Artifacts**

There was a **Definition of Done** for few types of tasks, that had to be followed.

Just like in Scrum – there was a **Product Backlog** - an ordered list of things to do. Format is up to the **Product Owner**.

There was also a physical **Board** with up to three queues

![Figure 1 - The Board created by the team](image)

**Queues (green lines on top)**

There were two queues – the Linux and the Windows one. People were pulling elements from queues according to their order – the ones in the bottom-right corner were most important. Whoever was able to take a task, took it. Mainly Linux admins were taking Linux tasks, same with Windows, but it wasn’t a rule. Who could complete a task from front of each line, pulled it. Both queues were limited to 10 elements, rest was kept in a Product Backlog.

**Competency groups (yellow lines)**

Each group had a limit of 4 tasks to be taken in total (having three people) and in addition there was a QR task in each group. QR meant being a firefighter and taking the smallest stuff. It was critical – based on what the QR was doing, new tasks were added to the line, helping automate this role as much as possible. There is also an on hold zone, for tasks that got stuck due to external interference.

**Done & rejected (red lines)**

Done was cleaned up whenever it filled up. Rejected were those tasks, that suddenly were not needed, or after little investigation did not make sense

**Quotas**

The team reached the conclusion (later empirically checked by other teams) that the recommended limit of cards is between the number of team members plus one, to number of team members times two minus two, so for a 7 people team it would be in between 8 and 12.
Roles

There was a **Product Owner**, who was taking care of the Product Backlog, and producing tasks for the board. It was his job to make sure there is always enough to do on the board, but not more than 10 in each queue. This role was combined with a role of the business analyst, since the PO was also responsible for collecting User Stories.

There was a **Kate**, who was taking care of the process, making sure that the board looks neat, takes care of the team development etc.

Events

Every week there was a **Grooming** session – one very technical, one customer-oriented.

Every two weeks there was a **Retrospective**, unless called for earlier.

There was a **Review**, called by the Product Owner whenever a logical part of functionality was completed, maximum every two weeks.

There were **Daily Plannings** – a combination of **Sprint Planning** and **Daily Scrum** – a two-tier half hour meetings by the board, where the team accepted new elements in queues, planned their next 24 hours and briefly reviewed completed work.

Lessons From The Way

I value Scrum and I asked the team not to use this term when we will not be using Scrum. I grew an allergy for “Scrum-Bans”, “Wagiles”, “WaterScrums” and “ScrumPluses”. And it wasn’t a ScrumBut either – we were not planning on eventually using pure Scrum. So, the team started calling it Kate-Ban as a joke and eventually it stuck.

Product Owner

First big a-ha moment for the team was when they noticed that they don’t really have a Product Owner. So Matt, decided to back up from politics he was tangled in and pick up the PO role. It was initially decided that he will collect bigger requests from all users and will be trying to pull a user story out of them each time. The team decided to direct all requests to him, so that they can work undisturbed. This took some time for users to get used to this way of working, but progress was visible almost immediately.

Planning

Second invention came after all the planning pitfalls were discussed. The team was neither able to plan a sprint nor go with a continuous planning, because it simply did not happen – the board started to get empty. That’s how we decided to adopt something else and that’s how we mixed a micro planning session into the daily meeting. In fact we turned days into micro-sprints.

Firefighting

Ad-hoc work was still a problem, so they decided to set up a firefighter, who will take care of a designated mailbox with all of the tiny requests. If there was a bigger thing, the firefighter forwarded the message to the PO. This decreased the ad-hoc load to no more than 30%.

Continuous Improvement

Another challenge were internal improvements - they got lost in the flood of daily work. So the team introduced so-called reverse-groomings. All improvements were stored at the bottom of the Product Backlog, and once in a while they were reviewed and pulled up, because this certain improvement can speed up our work in upcoming days.

The Feeling Of Progress

One of the biggest problems in Kanban was that the team did not feel the progress. In the hybrid, from time to time we assembled to review finished elements on the board and Matt was able to make sure his Product Backlog was up to date. This gave the team the necessary feeling of progress, looking at the amount of finished tasks.

Multiple Threads

Because of the exclusivity of skills, mixed tasks were a problem. One day someone thought of dividing it into multiple buckets – in this case a Windows one and a Linux one. We built it into the board and it just worked. Plain and simple. No conflicts, clear progress, easy task division. We were all really happy with this improvement. After this one there were no major changes to the board.

The Lack Of Deadlines

The biggest pitfall was the lack of deadlines. In Scrum, Sprints were providing the heartbeat, the moment when the work was inspected, certain feedback – there was nothing like this in this setup and the boiling frog effect started occurring. So after some time, for
most critical tasks and the ones that were on the board for too long, the team started adding deadlines. Having no regular feedback loop also enabled some people to slack off, start doing something else in the meantime and it went through undetected for quite some time. In this case, this team had an excellent Product Owner, but it would be easy to change this from pull approach to push in an instant, so he had to be very careful.

**Estimates We Did Not Need Anymore**

The biggest surprise were estimates. Since it was too hard to estimate in this environment, tasks were not estimated. Product Owner was projecting from the amount of tasks alone, not from their size. It was working surprisingly well with such a various environment. The velocity started being the amount of tasks.

**The Problem with Phil**

In the process of introducing the new framework we stumbled upon a big problem. His name was Phil. Probably if Phil was not in the team we would have been able to perfect the framework faster. At first encounter he appeared as a very elegant, nice and intelligent man. If you observed his interactions with people, you might see that he is nice, but everyone is a little rude to him. Or that's how it appears. In fact Phil refused to collaborate with anyone and refused to share any knowledge. His switches were configured in some peculiar way that only he understood, his scripts had no comments, and whenever he could refuse service and guard himself with an existing policy, he did that with visible pleasure. "No" and "impossible" were his favorite words. The rest of the team were problem-solvers, he was the obstacle. He particularly did not like working with Matt, because he was experienced, knowledgeable of company policies and he exposed many of his "impossibles" as "I don't want to do its". Phil was also always overloaded with work because he refused to teach anything to Leo, who was supposed to help him in his daily work. Leo ended up installing Windows off a cd since he did not have access to anything else.

Phil also refused to update the Sprint Backlog and later on the team board, so the team had no idea if he really is working on something or nothing.

The team solved the problem in a nice way. They started dragging Leo into the Linux side and teaching him new things, so that he did not get bored. They surrounded Phil with a bubble of some sort. They administered tasks to him, not letting him take anything without their knowledge. Phil seemed happy with this since he had almost no interaction with people and it looked like this is what he likes.

After few months he quit, because he could not work with Matt who became his manager. But other teams in this organization adapted bubbling as a way of dealing with anti-social people.

**Epilogue**

Although this team does not exist anymore, there are multiple teams using this approach. This team dispersed due to some external events, it was not due to work changes. Right after this framework started being used, another team I was working with adopted it. It was a software team who was entering a sunsetting phase, so they stopped developing their product, they only maintained it. They set up a board with one queue and are working like this until today with only minor changes to the board layout.