From Scrum to Kanban - A Team's Journey

AKI NAMIOKA, Marchex

Due to the changing nature of the work my team was assigned to, we needed to make some changes to our Agile practices. Since other teams in our company had experience with kanban, we decided to make the transition from scrum to kanban. This report will discuss the factors that went into our decision, how we made the transition, some lessons learned, and suggested best practices.

1. INTRODUCTION

In 2014, my team was in the middle of a very aggressive and complex transition. We transitioned from supporting an established product that had been running in production for 7 plus years, to a team that was developing a brand new product.

As my team tried to respond to these changes, we found that our current scrum paradigm had issues. We had been successfully using scrum for more than 2 years, but now our scrum practices weren't working as well.

As the Development Manager on the team, I observed that morale became an issue as we found it harder and harder to execute successfully on our sprint plans. In general it was a stressful time for the team as we tried to adapt to the demands of a new product, as well as new technology, and a new organization structure.

In the Summer 2014, we decided that we should make the transition to kanban, to see if it would help mitigate some of our issues with sprint planning.

2. MARCHEX AND AGILE

My company, Marchex, is a mobile advertising technology company. Since 2011 my team had supported the Free411 product, a free directory assistance service that was driven by a speech recognition engine. This product had already been in production for several years, but Marchex acquired the company in 2011. In 2013 my team had finished migrating the Free411 product into a smooth running, supportable product in a new data center. At this point, Marchex's highest priority was to invest the engineering resources of my team into an area that had more growth potential, i.e. search. Search advertising is a multi-billion dollar business, thanks to the Google AdWords platform. Because Marchex already had an extensive Call Analytics product that was processing millions of advertising campaigns per month, it seemed like a natural next step to expand our Call Analytics product to support Call Analytics for Google and Bing search campaigns. Because this was a brand new product, it was essentially "green field" application development. To minimize change for the team, we tried to stay with our basic scrum structure as we transitioned from one product to the next. But as it turns out, this plan wasn't as smooth as we had hoped.

Marchex is no stranger to Agile. Our VP of Technology is a strong advocate for the Agile philosophy and the entire product organization has adopted Agile in some form or another. True to Agile, each team is empowered to select the methods and practices that work best for them. Therefore, some teams are strictly XP, some teams are using scrum + XP, and some teams look more scrumban - a combination of scrum and kanban. Many of us had had kanban training in 2013, so we were already familiar with the concepts.

In 2014, before we transitioned to kanban, my team looked like a combination of XP + scrumban. We had adopted some standard XP practices, e.g. TDD, daily standups, pair-programming, demos, and regular retrospectives. We also were using our white board to track our sprint progress, through a kanban-style board, which helped us visualize the work better.

Since Agile was deeply ingrained into the Marchex culture, we had the support to adopt a new Agile paradigm, if necessary. And by the summer of 2014, it looked like a change was going to be necessary.

Aki Namioka, 520 Pike Street, Seattle, WA; email: anamioka@marchex.com

3. MOTIVATION FOR TRANSITION TO KANBAN

3.1 Unfamiliar Domain and Technology

Our first problem was our story definitions and point accuracy. One of the decisions we made, as we started our new product development, was the decision to use Scala for application development. Previously, the team had been using C++, Ruby, and Java as their primary languages. Unlike better-established languages, Scala uses a functional paradigm that is different than the more familiar object-oriented paradigm. As we started to delve into our new search domain, and adopt the new Scala technologies we found it more and more difficult to write stories with sufficient acceptance criteria, and we were getting worse about assigning accurate points. For example, we had a set of stories around creating a new service. One of the stories was defined as creating an API for this new service. The acceptance criteria for this story was vague, i.e. create an API that is usable by a client application to fetch and push data. This story languished in the same status while the team talked to internal clients that would use the API, and then designed a solution, and then implemented. Looking at our retrospective notes from the summer of 2014, there were many comments about stories with insufficient information, and underestimating how long it would take to do X. We didn't have a "yesterday's weather" type of comparison we could use to accurately gauge story size, so we often didn't complete the stories in our sprint plan.

All of this was creating a morale problem for the team members, as sprint after sprint the stories were not getting done as planned. Seeing the same stories in the same status day after day, or week after week, gave the feeling of being bogged down. Up until now, the team did their best to plan what they thought they could achieve, and felt good when all stories were in the "Done" state at the end of the sprint.

The apparent lack of progress also was causing a problem in how we communicated with our Program Manager. Week after week he saw the same stories on our board, and they appeared to be stalled. As he was also new to the technology and domain, he didn't know what questions to ask to get better clarity on scope and schedule. The team felt they were on a constant journey of discovery, and each new story was a new area to learn about. Thus, from the team's perspective they found it difficult to explain the lack of clarity around their estimates. This lack of clarity around deliverables was also causing some concern from upper management.

3.2 Fluctuating Backlog

Another problem was our fluctuating backlog. For teams that transition from waterfall to Agile, a 2-week sprint seems short and immediate. However, we were finding that 2-week sprints were too long in the evolving business needs of new product development. Our Business Development team was discussing our product to potential customers, and returning with feedback about our offering. In response, we had to constantly adjust our backlog priorities. Sometimes it felt like we were changing priorities on a daily basis. During this time we also went through a major product shift i.e. the product we thought we were planning and executing on, turned out to not resonate with our clients as much as some of the data analysis we were providing for them. So, we decided to refine our product offering in response to this feedback. In this environment a 2-week sprint plan was difficult to maintain.

Even after our product launch in February 2015, the fluid nature of our backlog still hasn't changed as our customers continue to ask for additional features. As a result our product roadmap is still in flux. In this environment sprint planning feels too awkward, and no longer fits our business needs. Between the rigidity of sprint boundaries, a backlog that is in constant flux, and the constant need to release feature enhancements and robustness, a new paradigm is required.

3.3 Need for Change

After several retrospective where we noted our inability to finish a sprint plan, I started looking around for ideas on how to mitigate these issues. Many members of the team had had kanban training the previous year, and we had taken the initial steps of moving our electronic scrum board onto a large white board in our standup space. This made the work more visible.

Also, as part of our transition to a new product team, my team went from being a single development team working mostly on its own, to part of a 3-team development effort. All 3 teams were required to develop and launch our new product. One of our new sister teams was already using kanban, and they were very happy with the results. They also had a reputation for being consistently productive.

I spoke to one of the developers who had moved from our team to the kanban team, and asked her how she felt about the transition from scrum to kanban. She said she liked the kanban style better for two reasons: the team just focused on the top 1 or 2 stories from the backlog, and when they were done they moved to the

next story. She said she enjoyed the simplicity of just picking the top story from the backlog and just finishing the work. She also mentioned that their process was more manageable because their stories were much smaller in scope. After talking to her, I was convinced that this was the right direction for my team.

From a logistics point of view, we had to be careful in how we transitioned. We were in the middle of new product development, and customers were being identified. We didn't have the luxury of slowing down our productivity.

3.4 Things That Didn't Need to Change

As we considered our transition away from scrum, there were still some practices that we firmly believed were still productive for the team. As is widely known, scrum is a method of structuring a project, while XP practices are mostly how to develop code. Advocates for XP often adopt scrum+XP. Similarly, we decided to continue using some XP practices, while using kanban for structure instead:

- Pairs-Programming - in 2014, the team had recently adopted pairs-programming as the normal way to develop software. We found that in adopting the new Scala language, and learning about the search paradigm, that pairs-programming was an imperative to maintain productivity on the team. We assigned pairs at our standups each day, though often the same pair would work together until a story was done.
- Test-driven Development - almost all, new development was created using test-driven development, and we felt that this practice was still the best way to develop software.
- Retrospectives - this is a standard Agile practice that should never go away. We find it useful for the same reasons other Agile teams do – it is our way of doing continuous quality improvement.
- Story points - We changed our point definition, but fundamentally we didn't see the need to change the idea of using points to estimate story size. We would discuss a story, agree on the acceptance criteria, and then assign points by group vote.

4. PROCESS OF TRANSITION

My next step was to talk to the Development Manager of the kanban team to understand his team processes and some best practices. He is also our Jira Administrator, so I also talked to him about what our new kanban board would look like. Even though his team had a continuous flow of stories, there were three things he still did on a weekly cadence: (1) planning every Monday afternoon, (2) weekly demos, and (3) weekly retrospectives. He also mentioned that smaller stories made it easier to measure flow, and keep a consistent WIP (work in progress) limit. Though this sounds a bit like a 1-week sprint, the important difference is that his team works off a constantly groomed backlog and reprioritizes as needed throughout the week.

I discussed the concept of moving to kanban with the team, and organized a Lean Coffee (leancoffee.org) style meeting to solicit feedback and address concerns from the team. The Lean Coffee style meetings explicitly solicit feedback from everyone by asking participants to submit discussion topics, and then they are prioritized by group vote. The priority of the topics is organized by the vote count.

The biggest issue that came out of the meeting was story size. It was felt that kanban would work better if all the stories were smaller and more consistently sized. It would also help mitigate the unknown nature of working with the new search and Scala technologies. As a result of that meeting we decided on the following processes to support our new kanban model:

- Weekly 1 hour planning meetings on Mondays.
- Weekly 1 hour retrospectives on Fridays.
- If we started getting low on stories mid-week (i.e. before the next Monday planning meeting), we would do a mini-planning at the daily standup.
- New points "measuring stick". 1 point = 1/2 ideal workday for 1 pair of programmers. Previously our scale was 1 point = 1 ideal workday for 1 pair. This reduced scale helped us keep our stories smaller as a 5 point story meant it was supposed to be done in about 2.5 days, rather than 5 days.
- If, during planning, we assigned more than 8 points to a story, we would break it into 2 or more stories.
- Daily standups would focus on discussing story status and moving them across the kanban board, rather than going around the circle and giving status. This meant our standups were shorter and more

focused on the immediate tasks on hand. This change seemed like a natural extension of our move to kanban, as it underscored kanban's emphasis on making work and cycle-time more visible.

- We assigned pairs during the standups. We didn't necessarily change the pairs every day, especially if the pair was in the middle of a story, but we discussed each pairing every day. Given our team size, it only took a minute or two.
- We kept the concept of 16th minute - i.e. if anybody wanted to discuss an issue in more depth, then we would write it down on our white board and park it for the 16th Minute discussion. The 16th minute items were discussed after we were done discussing story status on the board and assigning pairs.
- WIP would be 1 for each pair of developers on the team, i.e. 6 developers would mean a WIP of 3 for "In Dev". We also assigned a WIP of 3 on the "In QA" column. We never changed this WIP, so it seems to suit the team well.

We made the transition from scrum to kanban at the sprint boundary, i.e. we finished our current sprint, and at our next planning meeting we created a new points "measuring stick" and started planning as if we were a kanban team. This meant that we only needed to scope and plan stories for the coming week, as planning was on Monday morning.

Making the change to smaller stories solved one of our existing problems - the underspecified stories. By discussing stories that had a smaller scope, we also naturally tended to tighten up the acceptance criteria. Since our stories were 8 points (4 days) or less, we discussed the goals of the stories in smaller granularity. For example, instead of having a story that was simply creating an API for a new Service, the new stories were broken down into smaller stories like initialize, publish, validate, logging, finalize. In working on our acceptance criteria for each of those smaller stories, since the scope was smaller, our acceptance criteria also become much more modest in scope and granularity. For example, creating acceptance criteria for a story on logging became very specific. In contrast, the type of acceptance criteria that was created for a story on creating an API was much more vague.

It is true that creating smaller stories also meant creating more stories. Creating a good story hierarchy was critical to make sure that the larger features were being implemented adequately. A story hierarchy is often used to organize sub-stories under a broader scoped story. The broader story is often called an "epic". Using this mechanism we can create some order around a larger number of smaller scoped stories.

We were able to easily plan about 1 week's worth of work in under an hour. This was less than our previous 3 hour bi-weekly sprint planning meetings. Somebody used to fondly call it "poke your eyes out day". In all fairness, however, our old sprint planning meetings also included a retrospective. Now, we had a separate retrospective each Friday. But breaking the planning and retrospective into two meetings felt more palatable than 1 long bi-weekly meeting.

We started our new kanban life using the same white board we were using for scrum, with no adjustment to our column names. The column names were:

| In Dev | Ready for QA | In QA | Ready for Release | Done |

Then a couple of weeks later, when we acquired a large screen, we moved to using the electronic Jira board during our standups and planning. This made it easier for the team to see the backlog. At this point, we added in the use of 4 additional columns. This was a practice we got from the other kanban team. The 4 additional columns we added to the Left of our existing 5 columns were:

| New | Bucket | Backlog | On Deck |

New = New stories
Bucket = Unordered backlog
Backlog = Prioritized backlog
Start = Stories that the team feels were ready to implement, i.e. good acceptance criteria and points assigned.

The transition was very smooth, and after 9 months the team is still happy with the kanban paradigm. The planning is more "just in time" (JIT). The stories are shorter and more manageable. Using the rule of having to break down any story that is bigger than 8 points forces us to keep the stories small in scope. For a while, we

had a picture of our new "measuring stick" posted on our Jira board monitor, and we often referred to it during the first few weeks of our transition. After the first few weeks, the picture continued to hang on our Jira board, but we didn't have to explicitly discuss it. Also, we find that writing acceptance criteria for smaller scoped stories makes it easier to write more specific acceptance criteria that are less ambiguous. In other words, our story definitions are tighter.

We never did an A/B comparison in terms of productivity before and after the transition, but the team feels more productive, and morale improved as we were able to see progress each day as we moved our stories across the board. In addition, our communication with the Program Manager improved, as he had a better idea of status and how long it would take to complete a feature.

Even after our product launch in February 2015, our business requirements are still changing on a regular basis, so this JIT style planning is still a good match for our environment.

5. LESSONS LEARNED AND TAKE-AWAYS

There were several factors that made our transition fairly smooth.

5.1 Experienced Team Leadership

The person whose role had the biggest changes after our transition from scrum to kanban was the Program Manager (acting as Scrum Master). Throughout the week he had to stay at least one step ahead of the team in grooming the backlog. Given our fluctuating business needs, this was something he had to do on a daily basis. If the "On Deck" column started looking low, he had to make sure that he had the acceptance criteria well enough defined to add more stories to "On Deck" in case the stories in flight were finished before our next Standup. This meant that instead of grooming our backlog on a weekly or bi-weekly basis, as we did during our 2-week Sprint schedule, he had to constantly keep an eye on the backlog. He also had more stories to manage.

Another challenge as a Program Manager, is managing the transition itself. As a team, we had agreed to new practices to support the new kanban paradigm, but he had to help us adhere to the new practices that we adopted. This meant supporting the team as he guided the team to adopt the new practices, meeting definitions, and story writing.

Since I had been practicing Agile for more than 10 years, the Program Manager and I worked together to groom the backlog and coached the team to the new kanban world. We were also working with a team that was very supportive of our efforts to introduce change. Which leads to the next point.

5.2 Mature Agile Organization

Another factor that made our transition smoother, was the depth of Agile experience the organization already had. Most of the team was comprised of experienced Agile practitioners. The team was already conducting retrospectives where we discuss how well, or not well our current processes are doing, and making incremental process changes as required. To consider a fundamental paradigm shift did not feel as scary for us, as it might have felt for an organization that wasn't already practicing Agile. We already had the tools to adjust if our first attempt at kanban didn't work. Having weekly retrospectives allowed us to explicitly discuss how things were going more frequently than our previous retrospectives that came at the end of each sprint.

5.3 Kanban Training

The year before our transition to kanban, the entire development organization had gone through a kanban training that was offered by Modus Cooperandi. This training introduced us to the concepts of lean and kanban, such as work in progress (WIP), cycle time, etc. When our team started discussing the adoption of kanban we already had a common vocabulary. So, for example, when a team member said, what is our WIP for each column, we already knew what a WIP was in theory.

5.4 Keeping a Weekly Cadence

We found that having a weekly cadence helped us structure our new kanban life. We had weekly retrospectives, so we could make minor adjustments to our process if issues came up. We also found that keeping a weekly planning cadence helped us feel more comfortable with the process. Book-ending the week with planning on Mondays and retrospectives on Fridays, contributed to the feeling of a good regular tempo. So, even though we eliminated sprints, maintaining a weekly schedule seemed to make sense. However, one weekly cadence disappeared - that was our weekly release cadence that we maintained when we had scrum sprints. Instead, we started releasing features as they were completed, which seems more in keeping with kanban.

5.5 Experienced Kanban Team In Close Proximity

We were very fortunate, that we were sitting next to another team that was successfully using kanban. For example, they not only gave us the column name for our Jira board, but they also showed us how to handle "blocked" and "expedite" stories by putting the stories in their own swim lanes. We also emulated their weekly cadence and smaller scoped stories.

6. EPILOG

We are still happily doing kanban, nine months after our transition. In fact it is hard to imagine what it would be like to go back to two-week sprints. As I was writing this report I asked on one of the developers if he was still happy with the transition and he said, "Oh yes!" In talking to other developers on the team, they like the smaller scoped stories, and the visibility the kanban board gives them -- both in terms of work flow and also the forward looking visibility of the backlog.

Looking at some retrospective notes from the time we did the transition (late August - early September 2014) there were comments like, "Kanban going well - stories getting smaller and smaller" and, "Just-in-time planning!"

There is something very satisfying with seeing work items move across a kanban board. I recently created another kanban board to track the progress of a series of documentation we are creating for our new product. The Tech Writer was thrilled to see her work appear on a board. The simple representation of her work in progress gave her a sense of accomplishment in a way that was different than looking at a directory of files.

Our business needs still change frequently, but the team feels productive and likes steadily releasing new features. All of our meetings feel more efficient and productive, as we maintain the spirit of JIT planning.

When I shared our kanban experience with our VP of Technology, he suggested that kanban was an advanced form of Agile, and I agree with him. Though, Agile has many different definitions, keeping a daily focus on the highest priority stories that supports business needs, makes us feel like we were embracing Agile at a new level.

7. ACKNOWLEDGEMENTS

I would like to thank my co-workers at Marchex, who helped make the transition a success: Andrew Garbutt (Sr S/W Developer), Anna Zeitlin (S/W Developer), Bo Li (Program Manager), Craig Ulbrecht (S/W Development Manager), John Paul Wallway (S/W Developer), Jos Van Schagen (Principal S/W Developer), Kent Henneuse (Sr S/W Developer), and Madeline Heffernan (S/W Intern). I also want to thank Rebecca Wirfs-Brock for all the work she did to shepherd this report, from a session at Agile Open Northwest 2015, to this Practitioners' Report.