Agile & Kanban In Coordination

Ryan Polk
WMS Gaming Inc.
rpolk@wms.com
Blog: www.spryyeti.com

Abstract - Iterative development and Kanban are not mutually exclusive competing methodologies; in essence they are complementary processes that when used together can facilitate higher performance in large development teams. This paper is a walkthrough of the Network Gaming team’s implementation of Iterative Agile alongside Lean Kanban here at WMS Gaming. The intent is to cover the processes created and the improvements achieved with this coordinated Agile / Lean system.

Keywords – Scrum; Kanban; Lean Software Metrics; Agile Software Development; Agile and Lean;

I. INTRODUCTION

WMS has been a leader in interactive entertainment since Harry Williams founded the company in 1943. A Stanford University-trained engineer who devised the “tilt” mechanism for pinball machines, Williams changed the nature of pinball in wartime America. The company brought this spirit of innovation to the home video market (as Midway Games) in the 1980s and to the casino gaming industry in the early 1990s.

The NG (Network Gaming) system development team at WMS Gaming Inc. is comprised of 18 Engineers of varying skill sets and backgrounds. Located in Reno Nevada, this development team is responsible for development on a relatively mature and expansive codebase using primarily a Microsoft technology stack including C# / .Net and MS Team Foundation Server (TFS). The Remote Configuration & Download (RCD) management system is now in its 5th version with a history of over 4 years in development. The RCD functionality is a supporting mechanism for casino operators to deliver games remotely to slot machines on the casino floor.

Much like most systems development teams we are constantly working to balance large scale system feature upgrades that make up about 60% of the work with small-scale changes and bugs that cover the remainder of our time. Over the last 2 years we have worked to transition from a “Laissez-faire” waterfall team to a simple and well tuned Lean / Agile team.

My role in the team is both as one of the development team managers and as Agile coach for WMS as a whole. As a manager on this team my role as been to advise on our agile process and to facilitate continuous improvement. I try to give as much credit as possible to the team by the use of the term “we” in this paper, with the understanding that all the decisions made were made as a team and all were done in the process of continuous improvement.

II. AGILE / LEAN EVOLUTION

A. How We Got Here

About 18 months ago our 4-person management team was given the opportunity to transform our group of 18 developers from a so-called “agile” team to a truly effective Agile team. What we called agile wasn’t very agile. We held 15 minute daily stand-ups but our team was coordinated using a bug tracking tool. We had no systematic process.

Fig. 1 shows the timeline for our improvements. At the beginning of our agile improvements we had three teams of 5, 6 and 7 developers respectively. Together, all three teams were working on the same product. Over the first few months we found it a problem that there was no well-defined purpose for each team. As work came in, our product owners struggled to define what work should go to what team. They tried to break things down by individual projects / features and different layers of the system, but most times these projects ended up spanning teams. In addition, projects were difficult to estimate and track as we were using fractions of a team’s Velocity in our planning exercises. Most concerning to us, the distribution of project tasks across teams forced developers to jump back and forth between teams to get work done.

Overall, the overhead of running three separate teams was found to be wasteful, and eventually we started to work as one large team. This was not ideal but it created a much more cohesive flow of work. One problem though… With this large of a team (18 people) about 3 to 5 of the team members were falling through the cracks in various ways. They were failing to complete their work and generally hiding in the large team structure. Also, the concept of a self organized team was lost when team members began to disengage from the process as they had a hard time knowing how best to organize themselves within such a large structure. To make things even worse, the overhead that we were trying to escape by combining the teams re-appeared in the amount of work it took to coordinate iterations with a team of this size.
One interesting thing we noted when working in the large team environment was that certain team members gravitated towards selecting different types of work. After several months of debate the management team decided to create 2 groups that were defined by the size and type of work they would commit to. Team 1, the Iterative team, had 9 members dedicated to large scale / large project based work with very little up front definition. Team 2, the 6 member Kanban team, was dedicated to developing smaller more defined features. Also, the Kanban team was primarily comprised of those team members who had seemed to be falling through the cracks in our previous agile process. These team members were regularly missing stand-up meetings, failing to complete work inside iterations and were generally performing below what our expectations were for our team. Our theory was that they were performing below par because they were unable to fit into the team dynamic and structure of a large team. Our hypothesis in making this change, was that if we created a smaller team with more structure, these developers would be able to self organize and perform. As part of this re-organization, I was assigned as the manager of this team. I asked the team to try out Kanban as a way to help control our environment and manage the work. With their buy-in we moved forward with this transition.

B. Defining Agile

In our organization we find it valuable to define Agile by the key practices that each team implements. Although we do not follow an exact methodology, we are heavily influenced by Scrum and XP. I believe it is valuable to list our practices and compare how we implemented Lean / Kanban in our context. The 8 key practices that we define as the WMS Agile implementation are:

- User Stories
- Acceptance tests
- Iterative Development
- Burn down charts
- Project Boards
- Daily Stand-Ups
- TDD / Automated Unit Tests
- Continuous Integration

Beyond this, all developers have been trained in the basic principles of Agile. We work constantly to make sure everyone has an understanding of how the things we do relate back and support these agile principles

C. Adding in Kanban

As the Kanban team was formed, they were introduced to the practices of Kanban as a part of their everyday routine and as a simple modification to their already familiar agile process. To explain how this would work we created a simple comparison of the agile practices that we currently followed with the new Kanban practices. As you can see from Fig. 2, the substitution of Kanban was considered a simple plug and play substitution of practices. Previously, the team had been using story boards to manage work inside of their iterations. Using the Kanban board was simply equated to a more complex project board with specific rules for each queue. After adding the Kanban board to the process, I started to teach the team the concept of limiting work in progress and coaching them on the importance of flow through the system. Weekly cycle times were eventually posted on the board as well. Since we were unable to fully embed testers into the team, the developers worked to be even more cross functional; on a rotating basis the team members would also work to do simple testing on all new code. In addition to testing this rotating team member would coordinate the delivery of these new features to our Product Assurance (PA) department. With the understanding that we don’t just deliver code, we deliver...
tested and completed product, came the biggest changes in mindset for the Kanban team.

<table>
<thead>
<tr>
<th>Typical Agile Adoption</th>
<th>Simple Lean Adoption</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. User Stories</td>
<td>1. User Stories</td>
</tr>
<tr>
<td>2. Acceptance tests</td>
<td>2. Acceptance tests</td>
</tr>
<tr>
<td>3. Iterative Development</td>
<td>3. Iterative Development</td>
</tr>
<tr>
<td>4. Burn Down Charts</td>
<td>4. Burn Down Charts</td>
</tr>
<tr>
<td>5. Story Boards</td>
<td>V.S.</td>
</tr>
<tr>
<td>7. TDD / Unit Tests</td>
<td>7. TDD / Unit Tests</td>
</tr>
<tr>
<td>8. Continuous integration</td>
<td>Continuous integration</td>
</tr>
</tbody>
</table>

Figure 2 – Our training slide for the beginning of our Kanban adoption.

III. THE NG / RCD CURRENT STATE

A. Organization & Product Ownership

Early in our transition we created a product owner team comprised of one of the team managers and two developers / systems engineers. Their task was to marshal all incoming work and help maintain a regular schedule of meetings to break out User Stories with the help of the development teams. Our product owners regularly maintain the priorities of the Kanban board with the understanding that unless there is an extreme emergency they are only allowed to reprioritize, add to, and change work in the waiting queue. All other changes are off limits.

Since we take a more XP style approach to User Stories, we involve portions of both the Iterative and Kanban teams in the story creation and estimation process instead of leaving this responsibility solely to the product owners. As part of our planning process both teams maintain a standing meeting every other Wednesday for 2 hours of story planning and estimation. Since the teams worked together to estimate and define User Stories, we are able to keep both teams in sync. A piece of work can flow to either team as it has been estimated by a sampling of members from both teams.

At first the Kanban team would break each User Story down into multiple tasks and use those as a basis for the work items on the Kanban board. After about a month’s time the team decided that tasks were too granular and creating them added too much overhead. Quickly, they moved to just using the User Stories as their standard Kanban work item. Since the Iterative team had chosen to go solely to stories about 6 months beforehand, this gave us a standard unit of work across teams allowing us to flow work to either team where necessary.

B. Iterative Development

Our Iterative development team runs in a quite typical agile fashion. They are responsible for all large-scale projects that affect the system including any architectural roadmap work that may commit us to structural changes to the software as a whole. The team is casually limited to 2 or 3 WIP (Work-in-Progress) [3] projects at any one time. Their iterations are 2 weeks in duration and managed using a typical project board. They hold an iteration planning meeting on the last Friday of the iteration and hold daily stand-ups at 10 a.m. The Iteration team’s project board is located in the same area as their daily stand-up and is adjacent to the Kanban team’s board (See Fig. 3).

Figure 3 – The Iterative team’s Storyboard alongside the Kanban board.

C. Kanban Development

The Kanban development team is responsible for all small feature requests along with bugs. This team uses a Kanban board [2][4] that manages development process only, starting from the currently selected top 5 priority items and spanning all the way to a deployed queue. The team maintains a 15 minute daily stand-up immediately following Iterative teams stand-up meeting. They maintain a cycle time and lead time metric integrated with our story point system. The queues the team maintained in this version of the board are: (Most of these queues can be seen in the board makeup in Fig. 4.)

Kanban Queues
1. Goals – Changed to “Waiting Queue”          WIP 5
2. Stories                                      WIP 5
3. Tasks – Removed Later                       WIP 3
4. Development                                  WIP 4
5. Development Complete – Buffer                WIP 4
6. Verify & Accept                              WIP 4
7. Done                                        WIP 4
8. Merged / Deployed                            WIP 4
   a) Merged into - NG Release A
   b) Merged into - NG Release B
   c) Merged into - NG Release N…

265
D. WIP (Work-In-Progress) Limits

When using Story Points with the Kanban system we found it useful to maintain WIP limits on our queues by constraining the number of stories allowed in each queue as well as imposing a limit to the maximum amount of points in certain queues. The two primary queues we were concerned with were the Development and Verify / Accept queues. With a rather inexperienced team these queues were where we wanted to focus our attention on WIP limits. What we found was that in many cases when a larger work item was put into the development queue it became a “lump” in the flow of the board and stopped us from moving other work forward. Shortly after implementing Story Point WIP Limits we also set a rule that only stories of a certain size (8 Points) and below would be allowed on the board. With these two controls we found we were able to keep work flowing smoothly without the overhead of trying to break every story down to equal sized chunks.

IV. TEAM COORDINATION

A. Our Current State

Our product owner team is responsible for nearly all coordination of the story planning and priority processes for all incoming defects and enhancements. They participate in the iteration planning meetings for the Iterative team; they also manage the top 5 priority items on the Kanban team’s board. The Iterative team maintains a typical daily 15 minute stand-up that is attended by the product owners. In the same location and immediately after that 15 minute meeting the Kanban team has their own 15 minute stand-up. If there is an urgent change to the teams’ priorities the product owners attend both meetings to get the change into the pipeline. With this system of simple meetings our product owners are capable of staying up-to-date with what the teams are doing with a small commitment of 30 minutes per day.

The product owner meeting schedule coordinates approximately 99% of all requests that our team is engaged in. The remaining 1% includes emergency bug fixes or last minute feature changes to either team. These 1% interruptions are handled by either substituting something into the Iterative team’s story board or by adding new work into the Kanban queues, bumping lower priority items back to the waiting queue to account for the emergency item. Either way these requests are few and far between and are handled quickly causing minimal disruption to either team.

B. Workflow

To help explain our work distribution process, we like to use a rock crusher / sorter analogy. As you can see from Fig. 5 below each portion of the process fits well into the analogy.

- The product owner team sorts and breaks up the work into manageable pieces of various sizes. The highest priority items go to the front of the line for both teams.
- The Kanban board high priority list acts as the sorting mechanism for all items below a certain size and complexity.
- In the iteration planning meeting the Iterative team chooses the highest priority items needed to satisfy the current roadmap priorities.

Overall, flow is a priority for both teams. The only difference between the teams is the size and complexity of the work and the cadence of delivery.

C. Agile / Lean Metrics

Velocity / Pseudo-Velocity - Of course our Iterative team monitors their average velocity which they use to plan their iterations. We also compute a pseudo-velocity created from time slices of the Kanban team’s board. This metric is essential for our release planning process. To calculate this we compile an average velocity by placing work into iterations based on the completion date of the Kanban item. For example if an item on the Kanban board was completed on Nov. 10th and the Iterative team had iteration #8 going from Nov. 1st to Nov. 12th that item would be used to compile the pseudo-velocity for the Kanban team for iteration #8. After placing the items into their corresponding iterations, we take a rolling average of the last 5 iterations to use as an average pseudo-velocity metric for the Kanban
team. We know that this is not an entirely accurate estimate as we can't say that all items completed in a 2 week window were started in that same window. Since this metric is not used for small scale planning we accept this inaccuracy.

Cycle Time / Pseudo Cycle Time - For our Kanban team we use cycle time as a standard measure of throughput for the team. We coordinate this with the story points for each item to calculate a cycle time metric for each story size (i.e. we have a cycle time for each 1 point story, each 2 point story and so on). From this we can give our product owners an exact “lead time” [5] for any new stories at the bottom of the priority list. The product owners use this metric when scheduling high priority features and bugs to give our customers a reliable demo date for the functionality.

We also create a pseudo cycle time metric for our Iterative team. Since we started with the hypothesis that our Kanban team could be just as productive as our Iterative team we needed a way to compare the two teams and test our hypothesis. We can do this because both teams are engaged in the estimation process and so they maintain the same estimate structure across the organization with very little variance. This metric is not published as we are not looking to foster competition; we use this solely to track progress and as the teams diverge in process over time we understand that this metric will need to be abandoned.

D. Agile Release Trains

Before moving to the mixed iterative Kanban system we implemented an Agile Release Train [1] structure consisting of quarterly Potentially Shippable Increments (PSI) consisting of 5 full iterations per PSI. This structure has remained unchanged for the most part, but we added an additional layer to the release (See Fig. 6). Because of the integration of the Kanban team we do not use the concept of a “hardening sprint” [1] in our Release Train implementation. Instead, we have found that the Iterative team can have better focus on closing and hardening features as part of a normal iteration if they are buffered by the Kanban team against random interruptions. The first tier of the release plan includes all large scale projects coordinated and integrated into the quarterly plan. The second tier of the plan includes a swim lane using the Kanban teams Pseudo-Velocity metric to help plan all small scale features that are known at the time of the quarterly planning meeting. This swim lane is calculated using a velocity / 2 metric (determined from previous experience with our PA team) that leaves more than 50% open room for the Kanban team to absorb any incoming work and bugs that are intended as the team’s number one priority. This allows us to provide a very solid commitment to our customers for all large scale features in each release. With the Kanban team absorbing all small scale items, their allotment of work at the beginning of a PSI initially seems to be under committed. From our experience, we have found the 50% velocity commitment works well to keep the team fully engaged throughout the release development phase while being able to take on bug fixes and high priority items that arrive in the backlog.

V. Conclusions & Future

A. Measured Results

The results of our Agile / Lean transformation have been quite successful. Over the last 9 months we have seen a steady improvement in cycle time and pseudo-velocity of the Kanban team bringing them in line with the performance of the Iterative team. As Fig. 7 shows, after the first 2 months the Kanban team continued to make steady progress and improvement in average cycle time.

![Figure 7 – The result measured in Cycle Time – Kanban team and Pseudo Cycle Time – Iterative team show considerable improvement of the Kanban team over a short period of months.](image)

Interestingly enough it was at that two month point where we started to publish cycle time metrics on our board allowing the team to track their own performance. To reiterate, we didn’t publish the pseudo cycle time metric for the Iterative team as we were looking to avoid competition. The Kanban team has made steady progress and has come within .10 of the pseudo cycle time of the Iterative team for the last 3 months of the data collection period. This improvement has been dramatic in numbers but the amount of motivation and energy it has provided for the team has been immeasurable.

![Figure 6 – A view of the product Release Train taking into account the Kanban team’s additional completed work.](image)

0.00 0.20 0.40 0.60 0.80 1.00 1.20 1.40 1.60

Jun-10 Jul-10 Aug-10 Sep-10 Oct-10 Nov-10 Dec-10 Jan-11 Feb-11

Avg. Cycle Time Per Point

Team Cycle Times

Iterative Team

Kanban team

Release 1

Release 2

Release 3

Start Data

End Date

Iteration 1

Iteration 2

Iteration 3

Iteration 4

Iteration 5

Start Date

End Date

Iteration 1

Iteration 2

Iteration 3

Iteration 4

Iteration 5

Start Date

End Date

Iteration 1

Iteration 2

Iteration 3

Iteration 4

Iteration 5

Feature A

Feature B

Feature C

Feature D

Feature E

Feature F

Feature G

Feature H


CR* = Change Request
Kanban Team Results

After giving the Kanban team a simple framework to work under and giving the team members a bit more guidance and control over their environment, we have a well-functioning team that has made huge strides in their productivity. This new team is confident about the challenges they face and highly motivated to improve not only in process and practice but in their status as a team.

B. Overall Team Results

In our organization we have a system in place that helps us smooth out many of the distractions that can cause so many problems with Iterative teams. Since we are not only cross functional in our teams, but cross functional between teams, we are now able to pick and choose how we handle urgent or just disruptive new work. We have created a process framework that fits our team as well as the nature of our work and provides us with a level of flexibility that is tough to match with iterative development teams alone. By combined team estimation and marrying the rituals of Iterative Agile with the practices of a Kanban system we have successfully created a blended process. This process has given us greater flexibility as a team, and allowed us to maintain a sustainable iterative pace while adding to our velocity with an ever-improving Kanban team.

C. The Future

The future of the Kanban team is something of a question at this point. Because we were successful at improving our performance, we are now considering opportunities for adding responsibilities to the team members. The Kanban team members have proven that they are good performers in an organization that provides a bit of structure to their environment. After one of our feedback sessions where the team asked to be able to work on larger and more poignant projects, we have started to assign them larger system features that are well defined. The team has been handling these features quite well. The improved performance of the Kanban team has caught the eye of management too. This attention has led to an evaluation of a possible move to a new project providing the team with new work from the ever expanding WMS portfolio. In the end, the team of “poor performers” proved that the environment and lack of granularity in the process was the root of our problems.

D. Iterative & Kanban - A Model

In the Agile / Lean discussion circles we hear about Kanban, Scrum-ban and all kinds of other -Ban ideas every day. Some suggest mixing and blending processes. We have found that running both processes in synchronization, and not blending them, has been highly valuable for our organization. In my experience, I have found it more challenging than it should be to truly run a long term systems maintenance and new feature team with Iterative Agile alone. The random interruptions and fire fighting of supporting a product in the field always seems to cause disruptions to Iterative teams. By adding a synchronized Kanban team alongside an Iterative team we have been able to even out our iterations and create a productive and healthy work environment where we are able to meet our customers’ needs. For larger development teams that work collaboratively on a single product I would highly recommend the use of this tiered team approach to Agile / Lean development. Our successes in this process have come from a readily implementable structure and simple process improvements that can be easily adopted by teams in similar environments.

ACKNOWLEDGEMENTS

I would like to thank Rebecca Wirfs-Brock for her insight and input in the iterative process of creating this paper. As part of our enterprise Agile rollout and constant advisors on the process I would like to thank James Halprin and James Grenning for helping define Agile at WMS and helping shape my strategies both as a manager and as a coach. I would also like to thank the NG Development team members and specifically the management team: Chris Blackburn, Chris Frattinger, Sean Brooks and Vineet Vashishta their efforts and collaboration are what made our experiment in Agile so successful.

REFERENCES