Four Years of Scrum: The Good, The Bad, and The Ho-Hum

Heather M. Wilcox

heatherwilc@yahoo.com

Abstract

Three years ago, a paper was presented at the 2014 PNSQC titled, “How to Fail at Agile Without Really Trying”. It was based on the concept of achieving a successful implementation of Agile Scrum by leveraging the “lessons learned” from failure.

The company in question is now over four years into using the Scrum Methodology. Some things have changed significantly since the original launch and implementation. Some things have remained remarkably the same.

This paper examines what Scrum looks like when the newness has worn off and the process matures. It also addresses what we’ve learned about when Scrum really works and when it really doesn’t.

Biography

Heather has spent over 20 years working and learning in the software industry, choosing to focus primarily on start-up and small companies. As a result, she has had a broad range of job descriptions which include, but are not limited to: Tech Support Engineer, IS Manager, Technical Writer, QA Engineer, QA Manager, and Configuration Management Engineer. This has given Heather a wide range of experiences to draw from in her current roles as a Senior Quality Assurance engineer and Scrum Master.

1 Introduction

Three years ago, I presented a paper at the 2014 PNSQC titled, "How to Fail at Agile Without Really Trying". It was based on achieving a successful implementation of Agile Scrum by leveraging "lessons learned" from failure. At the time the paper was presented, I promised to return at some point and report on how everything evolved. Since then, enough time and change has happened that it feels like there are lessons worthy of sharing.

The company in question is now over four years into using the Scrum Methodology. Some things have changed significantly since the original launch and implementation. Many of these changes have been the result of a rededication to Scrum process and knowledge achieved through trial and error. Some things have remained remarkably the same – again as the result of lessons learned through trial and error. However, some less positive issues have remained and have not improved over time.

This paper examines what Scrum looks like when the newness has worn off and the process matures. It also addresses what we've learned about when Scrum really works and when it really doesn't.

2 Where We Were

Three years ago, at the point "How to Fail at Agile Without Really Trying" was published and presented, the company in question was still figuring out how to make its Scrum implementation work. A few critical errors were made during the initial rollout: Not training everyone on the Product Development team, not training the rest of the Company on Agile Scrum (or even giving more than a cursory explanation of what it might be about.), changing up and/or reorganizing teams every 6 months or so, and temporarily robbing people from one team and assigning them to others to increase productivity on current projects which, in turn, stunted progress on future works. Not every team had its own Business Analyst (BA) and, on some teams, the Product Owner (PO) exclusively drove feature work. There was also a generalized undercurrent of chaos which, in hindsight, was likely due to trying to figure out how to make scrum work in our organization given the rules we were initially given.

We also had a very "top down" approach to scheduling, which meant that points were nearly worthless. The expectation that certain features had to be available at certain times was the driving force behind all projects. Whether there was sufficient time to get the work done wasn't an integral part of the discussion, so putting points on stories was seen as a waste of effort and, therefore, wasn't done.

3 Where We Are Now

Since that first year, we've evolved significantly. During the second year, after the paper was published (although not necessarily as a direct result of the paper), Product Engineering rededicated itself to Scrum. We also reorganized our processes and made changes with respect to the personnel associated with each team.

3.1 Things that Changed

Each Scrum team now has a Business Analyst that works directly with the Team's assigned Product Owner. Interestingly, BAs do not serve as Product Owners. The PO position is occupied, in most cases, by a project manager or, in some cases, a people manager. We also have at least one POs who is a Senior Quality Engineer. In all cases, BAs or Marketing do not drive the pace of development although they do have influence on it. A PO works with the team to come up with a loose estimate on the number of sprints required for a particular Epic or Feature. From there, the PO confers with the BAs to create and rank stories that encapsulate the work to be done. Each team then refines the stories to suit their unique needs. During the final step, the PO works with the team to determine which stories are brought in to each sprint. So, in that respect we now schedule our projects from the bottom up, which is much more Scrum-like that our previous “top down” approach.

This has meant that our once irrelevant story points are now wholly relevant. Regular grooming sessions are used by most teams to make sure their stories are pointed out in advance. Additionally, estimated points are used to prioritize and schedule project work, so getting a good estimate for story points is now a vital part of our process.

Instead of holding several different meetings to keep work activity across all teams synched up, we’ve split our scrum teams up under business units. To spare us from spending time worrying about how to keep everyone’s work lined up, we now have a top-down driven “Seating Plan”, which prioritizes activities for each business unit for the coming year. Teams within a business unit are coordinated by their Project Managers and POs. Any coordination that needs to happen across business units is normally handled by the Product Owners.

We now use a tool called Target Process to manage our Epics, Features, User Stories, Test Cases and Test Plans. This has helped us to unify some of our processes across teams and has enabled more precise measurement of and metrics for test cases, test plans, test runs, and other valuable data. Although the tool isn’t ideal, it does serve its purpose reasonably well and has provided a framework upon which we could build our Scrum process.

Finally, team membership is now much more constant. People occasionally trade groups as they tire of working on the same project but, for the most part, teams stick together and have sufficient time to not only jell but to get a solid understanding of how each person works best and how to most effectively leverage their skillset.

3.2 Things That Stayed the Same

“The more things change, the more they stay the same….” This is certainly true in our case. Some of the best parts of the Scrum process have stayed with us throughout our journey. Each team is still expected to demo any new work at the end of each sprint. We are also expected to write and keep track of our stories, have planning meetings and hold retrospectives. The basic tenets of scrum are still in place for every group, even if each one handles those requirements differently. As explained above, every team has a Scrum master, a product owner, and a Business Analyst who all work together to guide development work.

Unfortunately, some of the less positive things stayed around too. We don’t have a consistent Scrum onboarding process for our new hires. Once an employee is brought in, they are invited to all the relevant meetings and then dumped into the mix to learn as they go. Some teams have a more formal indoctrination procedure than others but the one constant is that our Scrum training is inconsistent at best.

3.3 What Our Teams Look Like Now

As it turns out, our scrum teams have become very siloed both between and, in some cases, within business units. Each team has evolved its own version of Scrum process based on what was originally defined for us four years ago.

I sent out a short survey to all of the Scrum Masters. As it turns out, a few teams are using Kanban. Some adhere to a reasonably formal definition of Scrum process. Others “wing it” and follow the rules as they can but either can’t or don’t choose to follow Scrum process very closely. Most teams use two week sprints but a few utilize a one week sprint. All teams follow some sort of Scrum or Scrum-like process and no teams have reverted to a full waterfall format. However, the level at which the teams adhere to "true Scrum" clearly varies widely.

3.4 What My Team Looks Like

Since receiving the initial training in 2012, I've gotten five solid years of Scrum Mastering under my belt. As a result, I've formed some opinions about what I think works and what doesn't. The process below is the result of the opinions of myself and my teams translated into our work paradigm.

3.4.1 Standups

My team does some sort of standup every day. Since we have a few folks that work from home, we usually do a quick "Standup Status" email on Tuesdays and Fridays (the most common WFH days.) Mondays, we "walk the board", meaning that we look at our task board and get an actual status on each story. On Thursdays, we do a quick status based standup that usually takes less than five minutes. If we can't solution an issue quickly, it gets moved to a separate offline conversation. Wednesdays are a little different and will be explained separately. Ultimately, however, the goal of each of these interactions is to ensure that nobody is "in the weeds" and that there are no blockages. In other words, we stick to a pretty traditional version of the Standup meeting most of the time.

3.4.2 Planning Meetings

Although my team runs two-week sprints that end on a Tuesday, we have a one-hour planning meeting EVERY Wednesday. The first Wednesday of our sprint is a true planning meeting, where we step through our stories to verify which ones are finished and determine which ones need to be carried over from the previous sprint. Then we add enough stories to (we hope) get us through the whole two weeks. If there is any planning time left over, we work with our PO and BA to groom upcoming stories.

The second Wednesday of every sprint is more of a check-in and grooming meeting. We start by walking the board, which lets us know whether we're behind, on, or ahead of schedule. If we are ahead of schedule, we use the opportunity to add enough stories to keep everyone busy for the rest of the sprint. We tend to end up with more leftover time in the Second Wednesday meeting, so it often becomes primarily a story grooming session.

Theoretically, we understand that we should only need one planning meeting per sprint, but what we've discovered is that theory and reality don't always jive. To the good, we are often ahead of schedule and people feel like they might run out of work, so we have our PO handy to help us decide what should be pulled in next. During those times when we are behind schedule, the planning then serves two purposes – our PO gets an early warning that we may be falling behind but she also then helps us re-prioritize the remaining work so that the most important things really do get done.

An interesting side effect of this meeting schedule is that we have eliminated additional story grooming meetings. We occasionally meet with our PO to do broad estimates on upcoming Epics, but the days of marathon backlog grooming meetings are gone!

3.4.3 Demos

When we first started, many of my teams insisted (myself included) that we should demo something EVERY Sprint. What we ended up with was more than a few ten-minute demos that didn't really provide anything interesting and that took far more time to prepare than we spent showing our work. This made demos an expensive proposition.

What we settled on was waiting until we had real and valuable features to demo. Sometimes we go two or three sprints without showing anything. Sometimes we demo several sprints in a row. We don't save things up, but we also don't have a demo just for the sake of having it. What we've discovered is that, since we have fewer demos, people are more interested when we do hold an event, so our participation levels are much higher and those who attend are strongly engaged in the presentation.

3.4.4 Retrospectives

My team holds a retrospective every sprint. The rule I have for retrospectives is to use the same technique for as long as it works. Once the valuable input slows or stops working, I switch it up. I believe I've been fortunate in that my team takes the retro process very seriously and they work hard to come up with valuable output. Having a similar format every sprint saves time in that I don't have to explain a lot and my team is prepared for what is coming. However, eventually the format stops working. I then come up with a new technique, or revisit something that we haven't done in a long while.

Currently, I'm using a very simple retro format which is producing good results:

- Draw an emoticon that expresses how you think this sprint went.
  - This gives me a heads-up on how everyone is feeling about our current work and conditions
  - It gets everyone laughing and engaged immediately.
- Write down one thing that has had a significant effect on you this sprint.
  - It can be anything – an event, a conversation, a rule sent down from Upper Management, etc.
  - This forces people to really think about the sprint and all the events in it.

This retro usually takes 30-45 minutes (sometimes less than 30) and, so far, has been extremely productive in exposing issues that need attention. If this format does as well as my previous method, I should be able to get at least six to twelve months use out of it before I need to invent or locate a new retro technique.

3.4.5 Chartering

Whenever I am moved to a new team or if the team I'm on has a significant membership turnover, I take the group through a chartering exercise. Initially, the teams tend to feel like this is a waste of their time. However, I've discovered that, if a team has created certain artifacts, they can and will reference them when they feel like they've lost touch with what they're supposed to be doing or how they're supposed to be doing it. When I explain this value, most teams then become more willing to step through the process. Through some trial and error, I've learned that certain portions of the Chartering process are more valuable to my teams. I focus specifically on Vision, Mission, Mission Tests, Shared Values/Simple Rules, and working statements. For whatever reason, these seem to be the touchstones a team needs when they lose their way.

3.4.6 The Goal

As our team's process has evolved over time, our goal has consistently been to respect and preserve the ceremonies of Scrum while also being as mindful of time as possible. We know that each step in the process serves a specific and important purpose for our team. However, we also balance the time required for each ceremony against our workload and available bandwidth. So far, this has worked well for us as we have been able to successfully hold all the required scrum events but we have also managed to prune them down to the point where we have all the time we need but that time is nearly always an hour or less.

4 What We Learned

4.1 When Scrum Really Works

One of the things that has become clear to us is that, when the availability of a feature or epic needs to be predictable, but doesn't have a hard and fast delivery date AND the feature or epic is of "reasonable" size (typically less than 6 sprints), scrum works very well. This is especially true when we are doing incremental additions or fixes to an existing product. The team can follow all the rules of Scrum and complete the project "on time" and with an elevated level of quality. These are, by far, our most enjoyable sprints.

4.2 When Scrumfall Works Better

Over the last four years we have had a few occasions where the team has had to temporarily abandon true Scrum in favor of a technique we call "Scrumfall". We found that this tends to happen when we have an excessively large project. (e.g. Code refactor of an entire application.) A large project combined with a hard completion deadline usually assures the transition to "Scrumfall".

In the "Scrumfall" process, we tend to abandon true planning sessions and, instead, we bring every story we need to complete into the sprint in prioritized order. From there, we simply work through all the stories until they are done. At the "end" of the sprint, all incomplete or un-started stories are pushed to the next sprint. This eliminates the need for full planning meetings and backlog grooming. We still hold standups and the regular planning meetings, but they become more of a check in/status/strategy meeting than actual planning and backlog grooming. Along with the modified meeting structure, we also tend to jettison sprint retrospectives in favor of an end-of-project retro. This saves the team precious hours in meetings but allows us to deliver a high-quality product within the desired time frame.

In some respects, "Scrumfall" could be thought of as the team coasting on previous good habits. Our backlog is almost always in shape enough that we can go without grooming and regular ceremonies for several sprints in a row and then, at the end of the project, resume our normal Scrum processes without really skipping a beat. To take this whole concept a step further, it could also be considered a very Scrum-like adaptive strategy put in place to deal with a unique set of circumstances. I've often found myself reminding my teams that, even though we're trying something new, we're only committing to it for two weeks. So, if it doesn't work, we're not stuck forever and nobody dies. (This was especially true during the earliest part of our Scrum adoption.) The ability to switch seamlessly from Scrum to Scrumfall and back again, is a clear demonstration that, at least my current team has truly internalized that concept.

Finally, I should emphasize that "Scrumfall" events are rare. We work hard to not take on projects of a magnitude that forces so much change in our process. This is partially because that sort of project is inconsistent with what we know Scrum to be. Also, that type and mode of work is draining. To do a project like that very often would quickly burn out a development team.

4.3 Points Matter

As discussed previously, when we first implemented Scrum, we went through a stage where we didn't attempt to put points on stories. We were given deadlines for the completion of work, so it didn't make sense to spend time pointing stories because we already knew when we needed to be done. However, over time we have, happily, evolved into a more "bottom up" planning process. We now do an initial low-confidence estimation on epics that is used for long term project planning. Once the work has been "seated" (or put on the schedule), we work with our Business Analyst and PO to divide each epic up into small, feature-based stories.

Now that we're more Epic/Feature driven, we can confidently plan and agree to a set of features that we are able to deliver on a specific monthly release train. In most cases, we have been very successful in meeting our commitments. A large part of this success is due to diligent and consistent pointing for every story along with a good understanding of the point completion capabilities of the team under both normal and dire circumstances. Additionally, we have become extremely diligent about kicking back stories that we feel do not have all the information we need and splitting stories that are too large. We have found that any story larger than five points is, in most cases, too big and needs to be split. Occasionally, we get an odd large story (five to eight points) that must be tackled whole, but this is rare.

Lastly, given our “obsession” with small stories, we’ve let go of the Fibonacci numbers for pointing. For some reason, we have stories that we feel must be four points. They are clearly bigger than three, but not as large as five. So, we now have four-point stories periodically. We also don’t usually use .5 (half) point stories. We tend to round up to one just to ensure that there’s sufficient time to do a thorough job of testing. As a result, we have a lot of one, two, three, and five-point stories, but we also have more than a few four-point stories.

The result of all of this is that, in the last ten or so sprints (and often before that), we’ve only missed our “finish within five points of the estimated sprint work” goal once and, most of the time, we’re within one or two points of our original estimate. Sometimes we even complete more work than we originally estimated.

5 Things We have Questions about or are Still Questioning

Since we’ve adopted this process, we’ve had a few questions come up that we either haven’t answered completely or the answer isn’t completely satisfactory.

5.1 Is Continuous Improvement Continuously Necessary?

We are currently mandated to provide at least one “item for improvement” out of our retrospective for every sprint. The idea is that, by requiring an action item, the teams will be driven to constantly improve themselves. I and my team have been torn on this subject. We find that our retrospectives now tend to gravitate towards brainstorming mostly on our needed “item for improvement.” The fear is that this is happening at the expense of other discussions that, although they might be equally or more important, won’t provide the required resolution.

To the good, the team has been able to consistently come up with a thing each sprint that we could improve or fix so there is definite value in the exercise, but again, at what cost?

5.2 Is it Okay for Every Team to be Different?

Within the Company, there are currently fourteen scrum teams. Five of those are “Reporting” teams, which my group is a part of. Even within those five teams, the Scrum process varies widely from “pretty formal” to completely informal. Across the entire fourteen teams, the variation is even wider, with some groups using Kanban methodologies, one-week sprints, permanent Scrumfall…. Essentially, each team has adapted the original scrum process to suit its own needs. However, is that wide variation acceptable? Other than the learning curve associated with transferring between groups, is there any other harm that’s being done? Are there any parts of the process that we should force teams to adhere to? These topics come up for discussion periodically. So far, we’ve not come to any specific resolutions. Perhaps that lack of decisiveness is its own answer. Maybe we haven’t answered those questions because there isn’t a problem that’s forcing us to come to a resolution?

5.3 How To Temporarily Join Two Teams?

Within our group of five related Scrum teams, we’ve had a couple of projects where teams were temporarily combined. For one reason or another, this process has never been as smooth as it probably should have been. Scrum masters accidentally step on each other’s toes. QA and Dev leaders suffer the same issue. On the QA side, we did figure out a solution to the squashed toes problem. Even though there are five teams in our business unit, we ship all our software together. To prevent one person from getting stuck with all the work for every release, we now have a rotation within the QA team. A “Shepherd” is designated for every deployment. This person is responsible for attending all meetings and completing any artifacts that are required. We’ve found this process eases at least some of the pains associated with combined releases. The Dev team has a similar arrangement. However, even though these agreements are in place, we still have sticky issues associated with combining teams and, sometimes, things still fall through the cracks because each group does things just a little differently.

5.4 How bad is it really for a Manager to come to a Retrospective?

I have a running debate with a manager who claims that, if they are doing their job properly, it should be fine for managers to attend retrospectives. He also claims that the retrospective isn't an appropriate venue to address management issues. My response is that sometimes, the Manager isn't doing their job properly and the retro may be the only opportunity for people to voice their concerns, regardless of whether it is wholly appropriate. I also have stated my belief that, even if someone has a good relationship with their manager, the presence of a high-ranking person may inhibit or prevent some conversations. So far, he hasn't budged and neither have I. I always extend a courtesy invite to managers for our retrospectives but I also encourage them not to attend.

6 Conclusions

6.1 The Good

Scrum is working. Despite variations in usage, all the teams are getting work done in a timely matter and producing the required documentation. Whatever process each group is using, they are consistent and predictable. Teams are putting points on their work and they are providing items for continuous improvement. We don't go dark for months –features and fixes are released regularly. This has had a positive effect on our Sales and Marketing staff as they know that many features and most bug fixes are never more than a few weeks away.

6.2 The Bad

Not everything is perfect – not even close. As previously discussed, there is a lot of variability in how each of the groups runs itself. Sometimes this is a non-issue, but switching from one scrum team to another can be confusing. Additionally, internal processes vary widely. Some teams document every test case, some groups don't document test cases at all. Test automation is very inconsistent, both in technology and in the amount of code coverage. Some teams rely heavily on their Business Analysts, some rely heavily on their scrum masters, some are more harmonious and rely on both roles equally and appropriately. Lastly, as I mentioned earlier, we don't have a good program for indoctrinating new hires into our Scrum process.

6.3 The Ho Hum

The boring (and best) part of all of this is that work is getting done and, by in large, it's getting done Scrum-style. More importantly, when a team constructs a good process and sticks to it, the entire thing becomes part of the background. When we first started down this path, Scrum was the beginning and end of a lot of conversations. Development happened but we were very clearly USING SCRUM TO MAKE SOFTWARE. Four years in, we have a solid foundation in Scrum, so now WE DEVELOP SOFTWARE and Scrum is simply the device we use to do it. Scrum has been relegated to the background, where it belongs.

Few of the processes we use are innovative. The development tools we leverage are as "technology forward" as we can make them, but our Scrum methods (even though they vary widely) are still very much "by the book". We've tuned our processes to the best of our abilities and now we abide by them religiously. There is comfort in the consistency and challenge in improvement.

Interestingly, we're bringing in a Scrum coach to audit our processes. It will be exciting to hear their verdict on our adaptation of Scrum. My hope is that the coach will answer some of the questions posed earlier in this paper.

6.4 Winning Is Possible

The end goal of a software company is to successfully build and ship software and to make money in the process. This industry wants to be (and often must be) all about innovation. So, as soon as we master a technology or a process, instinct tells us that it's time to move on and invent or find the "Next Great Thing" – always skating the bleeding edge. However, sometimes the tried and true process is the best thing to use. The Waterfall development methodology is still appropriate in certain situations. Like Waterfall, Scrum may not be a bleeding edge technique any more, but it is reasonably easy to learn, extremely adaptable to many situations, and certainly enables the fast construction of quality software. That, by any measure, is a win.

7 Thanks

As always, I am thankful for the patience of my employer, who allows me to continue to write and publish papers as part of my normal job duties. I am especially gratefully to my manager who, not only allows me to get away with writing during company time, she includes it in my annual goals so that I must do it! Finally, I wish to thank my husband, Justin, who still doesn't really get all this technology doo-dah, but puts up with me talking about it all the time anyway.

References

Wilcox, Heather M. 2014. “How to fail at Agile Without Really Trying”, Proceedings of the 2014 Pacific Northwest Software Quality Conference, http://www.uploads.pnsqc.org/2014/Papers/t-014_Wilcox_paper.pdf