Lessons Learned from a Failed Attempt at Distributed Agile

Mark Rajpal^(✉)

ARC Business Solutions, Calgary, AB, Canada
mrajpal@arcbus.com

Abstract. What do you do when you have endured an Agile experience where things didn't go so well? You can abandon Agile altogether or you can take those lessons learned and apply them to future Agile projects. This paper discusses the journey travelled from that painful experience to becoming a more confident and experienced Agile practitioner. We will also look at some of the challenges that I still encounter today.

Keywords: Organization · Project · Team · Agile · Scrum · Extreme programming

1 Introduction

The 12 principles of Agile [1] indicate what is needed to make Agile effective. However, it does not specify what elements can render Agile ineffective. Agile (like anything) can only be successful in the right situation. There are components that pair well with Agile, but there are also factors that may not be well suited for Agile.

Some organizations have experienced failed attempts at Agile and claim that Agile does not work. In many cases this is a fallacy. Sometimes Agile uncovers pre-existing issues that have been around for years that were simply neglected. In other cases, the organization is structured in such a way that it is not conducive to the Agile mindset.

In this paper, we examine a failed Agile project that uncovered many practices that were not in fact Agile. These elements range from non-technical to technical. In each case we discuss the journey from the failed project to today where we apply common practices as a result of lessons learned.

The rest of the paper is organized in five separate sections. In Sect. 2, the failed project is explained in detail. Section 3 indicates the lessons learned that were applied. Section 4 discusses the remaining challenges. Finally, Sect. 5 summarizes the key points in conclusions.

2 The Failed Project

In 2008, an ambitious initiative was undertaken to implement a scheduling system for a large organization in the energy sector. The Scrum methodology was selected in an attempt to deliver high quality software in a short amount of time using month long

sprints. Consequently, teams were allocated at various locations including Alberta Canada, North Carolina USA, and various parts of Europe. Matters were complicated by the multi-vendor approach where one vendor provided the Canadian teams and the other vendor provided the USA/European teams.

The project was scheduled for just one year. At completion, the project took over 3 years and was over 5 times its initial budget. As a result, the project was considered a failure and the Agile approach was to blame. I was not convinced of this and I decided to explore the truth behind what really went wrong.

This paper will focus on some of the challenges and lessons learned from the failed project. These lessons learned aided me on ensuing projects. That is not to say these subsequent projects were free from challenges. Instead, the failed project provided a starting point, and the successive projects provided a means of moving forward while continually learning and improving.

The lessons learned can be categorized into two main categories— communication and requirements.

2.1 Team Communication Problems on the Failed Project

2.1.1 Intra-Team

I was part of a Scrum team (1 of 2 Scrum teams for Vendor A) consisting of 5 team members where everyone (except the ScrumMaster) was collocated. Team members started their day by updating the time remaining on each of their tasks before the standup meeting. The standup meeting was usually in excess of the standard 15 min as team members typically asked questions instead of answering the 3 questions. Sometimes these meetings went on for over an hour as problem solving was incorporated that unnecessarily occupied everybody's time.

At sprint end, a retrospective was held. The team discussed what did not work well. But the problems continued to exist as no corrective action was taken. Furthermore, the retrospective was cancelled once the project schedule started to slip.

2.1.2 Inter-Team

In all, there were a total of 4 Scrum teams. Vendor A was responsible for 2 Scrum teams located in Edmonton and Vendor B was responsible for 1 Scrum Team that was distributed across North Carolina and Europe. Additionally, the client provided an operations team that was also located in Edmonton but not dedicated to the project.

Communication between Vendor A and Vendor B teams was problematic from the beginning. The teams were uncomfortable talking to one another so they avoided it until they absolutely had to. Phone calls were rarely used. Instant messaging tools were not used. Instead, email was overused so there was no sense of any kind of real-time information. The problem was further complicated having the Vendor B resources in completely different time zones and geographies. It became an “us vs. them” mentality especially after failed integration attempts at the end of each sprint.

A similar relationship existed between the client operations team and the Vendor A teams. There was no sense of team unity between these collocated teams. Both sides did

the bare minimum in terms of supporting one another. Complications arose mostly due to the fact that the operations team was not dedicated to the project. As a result, they had various demands that they could accomplish themselves but did not have the time to do because they were inundated with other tasks and projects.

With so many teams and participants, pressure was introduced to refrain from talking which led to less effective communication [2].

2.1.3 ScrumMaster

As mentioned, the ScrumMaster for both Vendor A teams was not collocated with the team and had an additional role of project manager. Even though the ScrumMaster attended the standup via conference call, it was evident that he was not an active part of the process. The ScrumMaster would often ask for the progress of certain tasks that had already been completed or in some cases ignore tasks that were incomplete. Although the ScrumMaster visited the team approximately once a month, during this time he was immersed with meetings amongst senior management.

2.1.4 Product Owner

The Product Owner was only collocated (in Calgary) alongside a single team member from Vendor A. This allowed that team to gain an edge in terms of face time but the team also suffered in terms of effective communication with a distributed team member. The Product Owner had a difficult job mostly because each team competed for her time. As a result, much time was wasted as teams waited for their turn.

In the early stages, the Product Owner travelled to the various teams. While on-site, the particular team she was visiting was productive. Unfortunately, the other teams were at a standstill. As the project schedule slipped and the overall budget escalated, travel was limited.

In one sprint, each team travelled to Calgary for sprint planning. Even though everyone had face time with the Product Owner, the same problem persisted in that the Product Owner was constantly bombarded.

2.1.5 Executives

The relationship between the client and Vendor A's management was turbulent from the beginning. Their interpretation of Agile seemed to be the core of the problem. The relationship declined even further when milestones were not achieved.

Sprint reviews were held with the client's senior management asking questions about why things were done in a certain way. Then in one review, the application crashed. The management team was convinced this was a waste of their time, and sprint reviews were cancelled.

2.2 Requirements Issues on the Failed Project

2.2.1 Confusion

The project was engulfed with many different forms of requirements. This led to much confusion as to what was acceptable and what was not.

Before the project initiation, the client incurred a sizable cost to conduct requirements gathering that resulted in a vast amount of documentation. Consequently, their expectation was that no further requirements were necessary. Upon project commencement, it was obvious that the requirements were outdated and was of no use. This angered Vendor A's management because their fixed price bid was based on the Big Design Up Front (BDUF) requirements and they were concerned about the impact to their profitability. BDUF is where the architecture phase is accompanied with a vast amount of documentation [10].

2.2.2 Process

Before developers could begin coding, they were required to write a design specification. This deliverable was not contractually bound and was only stipulated upon project commencement and yet all vendors agreed to adhere to this. In some cases, developers spent an entire sprint writing a specification and delivering nothing. What I found quite peculiar is that many people agreed that this was a good use of time even though sprint deliverables were not being met.

2.2.3 Change

Change requests were frowned upon because they implied a change to the requirements. As mentioned, there was an assumption that the large cost of BDUF negated the need for any changes. As change requests arose, executives required that the vendors provide thorough documentation. Consequently, teams stopped working on their sprint goals as they context switched to accommodate the executives. In most cases, the change request was rejected and the vendors had to absorb the cost.

3 Lessons Learned Applied

3.1 Good Communication is Important

Be Serious About the Standup: On subsequent projects I experienced many different interpretations of the daily standup. For some teams, the meeting turned into a 1 h water cooler discussion that included the weather and the sporting events from the previous night. Other teams used the time to problem solve. In some cases, my team members decided that the meeting was optional and decided to attend when it suited them or when they felt like it. On a more recent project, I stressed the importance of the standup meeting by explaining “why” it was important. Team members began to realize that it was in their best interest to understand what other team members were doing. At first, they struggled with the 15 min rule. However, I received a really good tip at an Agile conference. The session speaker suggested that at the start of the meeting, the ScrumMaster

should set a 15 min timer on their smart phone. When 15 min expires, simply end the meeting. After 3 or 4 times, teams will get the hint and start adhering to the timebox. In fact, one of my teams introduced a post standup discussion. Immediately after the meeting, any team member was allowed to discuss anything they chose as long as it benefited everyone. This included anything from upcoming vacation schedules to the progress of testing.

Aim for Totally Integrated: We live in a world where distributed teams are becoming the norm. According to Sutherland [5], there are three types of Scrum implementations when teams are distributed. Isolated scrums, where teams work completely independent of one another and have very little communication. In fact, some teams abandon Scrum and fall back to waterfall. Distributed scrum of scrums, where teams are mainly isolated but are integrated by a scrum of scrums that meet regularly. Totally integrated scrums, where teams are cross-functional [6] and the project is integrated as a whole. Overall, I would describe the failed project as the isolated scrums model. Not only did the teams work independently, they had very little knowledge of what the other teams were doing. Despite my best efforts, the totally integrated model was never reached on any project. However, having those aspirations allowed for the progression from isolated scrums to distributed scrums of scrums on more than one occasion.

Collocate the ScrumMaster: On ensuing projects the distributed nature of the ScrumMaster greatly affected the team. There was little to no confidence in the ScrumMaster figurehead who was rarely seen and had very little involvement. Also, it seemed that on most projects it was assumed that the project manager should take on the role of ScrumMaster. Much more success was achieved when one of the team members (that was collocated) took on the role of ScrumMaster. On one project in particular, we decided to rotate the role amongst ourselves which made the experience much more enjoyable. Even though this approach worked much better, I'm not sure this would have been successful with a non-collocated team member facilitating the role of ScrumMaster.

On another project we split the role of the ScrumMaster. Essentially, we had a team ScrumMaster who was collocated with the team. Additionally, we had a client facing ScrumMaster who was mainly tasked with communicating with the Product Owner. Initially I thought the "co-ScrumMaster" approach would not work because Agile evangelists do not seem to support it. However, this allowed the junior Agilists to feel more comfortable taking on the team facing ScrumMaster role.

I am not aware of success stories where the ScrumMaster was distributed from the team. That does not mean it cannot exist. In fact, it is highly probable that there are such cases but the lack of supporting information could indicate there are rare cases or that more research needs to be explored in this area.

Involve the Users Early: After one year from the start of the project, the client brought in an additional resource that functioned as a subject matter expert. The idea was to alleviate the barrage of questions on the Product Owner. However, the subject matter expert was not empowered to make decisions. Often she would have to go back to the Product Owner for clarification or confirmation. This approach introduced an unnecessary layer of communication. Furthermore, neither the Product Owner nor the subject

matter expert was an active user and Agile projects require active participation from the users [7]. The users did not have an active role in the project.

On later projects, the end users were more than happy to provide feedback. As we listened to their concerns, they felt their trepidations were being addressed. Including users early on in the process tends to give them a finer grain of control over the project [8].

Get Buy-in From the Top: It seemed that the executives (or possibly the organization) were not quite ready to make the switch the Agile. Or maybe they just assumed it would be seamless. Agile migrations are not free from issues and there will be obstacles to overcome. “If you don’t involve your executives in the move to Agile, there is a good chance that they will stop the move as soon as they learn of any issues with the migration.” [9]. Having dismissed the sprint review, the executives were even less engaged at a time where they needed to be more involved.

On successive projects, we posted displays (aka information radiators) for everybody (including executives) to see. Posting an impediments list prompted executives to inquire about the impediment and eventually aid in removing it. For example, one of my teams listed “Administrator Access” as an impediment. Essentially, the team did not have the ability to install tools on their development machines without going through a formal approval process. It turned out that one of the executives heard this from other people and also experienced this pain point. Consequently, the executive authorized all team members to have Administrator access. However, this example worked well because the executive walked past the information radiator many times a day. In many cases, executives are not located anywhere near Agile teams.

Retrospectives Are Necessary: Even though the retrospectives encouraged good discussion, they were completely ineffective especially once they were cancelled. Following the project, I researched various techniques. Almost everyone I reached out to pointed me to the book, “Agile Retrospectives: Making Good Teams Great”. There were many helpful suggestions for the novice and also the more experienced. On most projects I introduce the “Timeline” activity [3] for the Sprint 1 retrospective. It seems to be a good way to get the team (as well as a rookie facilitator) accustomed to retrospectives. The book provides many other activities for the other retrospectives so the team is not repeating the same thing again and again.

Incorporate the Necessary Tools: On the failed project there were many areas that could have been incorporated to bridge the communication gap. For instance, equipping all computers with webcams would have allowed distributed resources to have a face-to-face conversation. Also, teams could have adjusted their working hours to have some overlap with the other teams. Even a one-hour overlap would have made a significant difference. Furthermore, it seemed that the travel budget was not adequate. When travel was requested it was quickly denied due to cost. It is difficult to say whether or not these things would have made a noticeable difference. However, it is likely that these modifications would have gotten the teams closer to a totally integrated scrums model.

On other projects, I have used add-on tools that facilitated more collaboration. For example, Jira is a popular issue tracking system I used on some Agile projects. There are many supporting tools for Jira but one that stands out is Hipchat. Hipchat facilitates instant messaging, screen sharing, and video. It can also produce notifications once the

state of a Jira item has been modified. On smaller projects, my teams benefited from Kanban boards by using tools like Trello and AgileZen. While these tools can make our lives easier it should be noted, “technology can maintain relationships but it won’t build them” [4].

3.2 Managing Requirements is a Necessity

Time and Materials (T&M) Over Fixed Price: Fixed price can lead to false expectations and mistrust [8]. On this fixed price project, the client felt they could change the requirements whenever they wanted, and the end product would be delivered as expected, at the same cost, and on time. Ensuing projects that were T&M based allowed the client to make modifications but it also allowed the teams to incorporate quality, present alternatives, and build trust by delivering often. In fact, it was noted that while extreme programming (XP) practices are possible for fixed price, it is not proven because accurate estimates of scope are necessary [6].

Change is Inevitable: Requirements will change regardless of how much analysis is spent before the development work commences, so change requests are inevitable. However, many large organizations cause change requests to be as tedious as possible. In this case it simply detracts from getting the work done. Later projects involving smaller organizations viewed change requests as a waste of time. Other projects addressed changed requests by accepting the fact that change will happen. Sometimes that meant the user stories were pushed to another sprint. In other cases, the team put in extra time to accommodate the change to meet their sprint goals.

Requirements Must Be Understandable and Decomposed: For the most part, the new requirements were very complex and the teams were pressured to complete them in a single sprint, which rarely happened. Since the team did not have the necessary availability of the Product Owner, there were many times where they had to guess which seldom ended well. Subsequent projects incorporated user stories as their primary means of acquiring requirements. This worked much better for everyone. The requirements became much clearer when the business analysts were able to write user stories as crosscutting layers. For example, instead of taking on a pure database user story, we developed a screen (with some functionality) that crossed the user interface, business logic, and database layers. Developers favored this approach because it allowed them to complete the user stories within the sprint.

4 Remaining Challenges

This report has explored the challenges associated with communication and requirements using Agile approaches. The results provide information that can be useful in overcoming these challenges. However, there are some challenges that require further investigation.

Lack of Support: Many Agile implementations (especially from a grassroots movement) struggle to gain support from the organization. In fact, there are those that are never going to accept Agile and they may even demean you for trying to promote it.

Empowerment: It is often the case that people are given a title of empowerment without the power to go with it. In Agile, there are cases where ScrumMasters are not empowered to remove roadblocks or Product Owners are not empowered to make decisions without the approval of their superior.

User Involvement: Sometimes users do not want to take part in the Agile process as they feel it adds to their workload. As a result, the task of writing user stories may fall upon the requirements engineer. In other instances, management may decide that the end users do not need to be involved until the modifications are deployed to production.

Scrum Ceremonies: Some (if not all) of the ceremonies can become redundant. Teams often get bored of repeating the ceremonies in the same manner. This may cause the effectiveness to come into question.

5 Conclusion

The paper presents challenges that prevent Agile teams from performing at a high level. Ultimately, organizations need to decide whether or not Agile is a good fit. If it is a good fit then they need to be prepared to make changes (in management style, working environment, team's skills, and close relationships with users [8]) and support it across all relevant levels. When individuals or teams are placed in a situation to fail, they often do just that. Agile is no different.

Of the 12 principles, the one that most resonates with me is “working software is the principal measure of progress” [1]. There have been many times when people have denounced my Agile efforts. What they cannot argue with, is success.

Open Access. This chapter is distributed under the terms of the Creative Commons Attribution-NonCommercial 4.0 International License (<http://creativecommons.org/licenses/by-nc/4.0/>), which permits any noncommercial use, duplication, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, a link is provided to the Creative Commons license and any changes made are indicated.

The images or other third party material in this chapter are included in the work's Creative Commons license, unless indicated otherwise in the credit line; if such material is not included in the work's Creative Commons license and the respective action is not permitted by statutory regulation, users will need to obtain permission from the license holder to duplicate, adapt or reproduce the material.

References

1. Beck, K., et al.: Twelve Principles of Agile Software (2001). <http://agilemanifesto.org/>
2. Hogan, B.: Lessons learned from an extremely distributed project. In: Proceedings of AGILE 2006, pp. 321–326. doi:[10.1109/AGILE.2006.37](https://doi.org/10.1109/AGILE.2006.37)

3. Derby, E., Larsen, D.: Agile Retrospectives: Making Good Teams Great. The Pragmatic Programmers. LLC, USA (2006)
4. Heffernan, M.: Willful Blindness. Anchor Canada, USA (2012)
5. Sutherland, J., Viktorov, A., Blount, J., Puntikov, N.: Distributed scrum: Agile project management with outsourced development teams. In: Proceedings of the 40th Annual Hawaii International Conference on System Sciences, HICSS 2007, p. 274a. doi:[10.1109/HICSS.2007.180](https://doi.org/10.1109/HICSS.2007.180)
6. Hossain, E.: Coordinating mechanisms for Agile global software development. In: IEEE International Conference on Global Software Engineering, 2008, ICGSE 2008, pp. 257–263, 17–20 August 2008. doi:[10.1109/ICGSE.2008.24](https://doi.org/10.1109/ICGSE.2008.24)
7. Wells, D.: Extreme programming: a gentle introduction (2009). <http://www.extremeprogramming.org/>
8. Chen, J.Q., Dien P., Wang B., Vogel, D.R.: Light-weight development method: a case study. In: ICSSSM 2007 International Conference on Service Systems and Service Management, 9–11 June 2007, pp. 1–6. doi:[10.1109/ICSSSM.2007.4280199](https://doi.org/10.1109/ICSSSM.2007.4280199)
9. Smith, G., Sidky, A.: Becoming Agile in an Imperfect World. Manning Publications Co., Greenwich (2009)
10. Hadar, I., Sherman, S.: Agile vs. plan-driven perceptions of software architecture. In: 2012 5th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), 2 June 2012, pp. 50–55. doi:[10.1109/CHASE.2012.6223022](https://doi.org/10.1109/CHASE.2012.6223022)