**Implementing Agile Scrum: A Case Study**

Clyde D’Souza

**Abstract-** Every organisation developing a software follows a methodology and each methodology when implemented turns out different from theory. In the process, the team encounters aspects that worked for them and aspects that didn’t. The following case study is of an organisation using Agile Scrum as their development methodology, their success factors and the challenges faced by them. For them, the main success factor has been the small member team and the ability to understand requirements better. This of course, is a result of using the methodology. They claim that the biggest challenge is getting their employees confident with using various tools, processes and technologies.

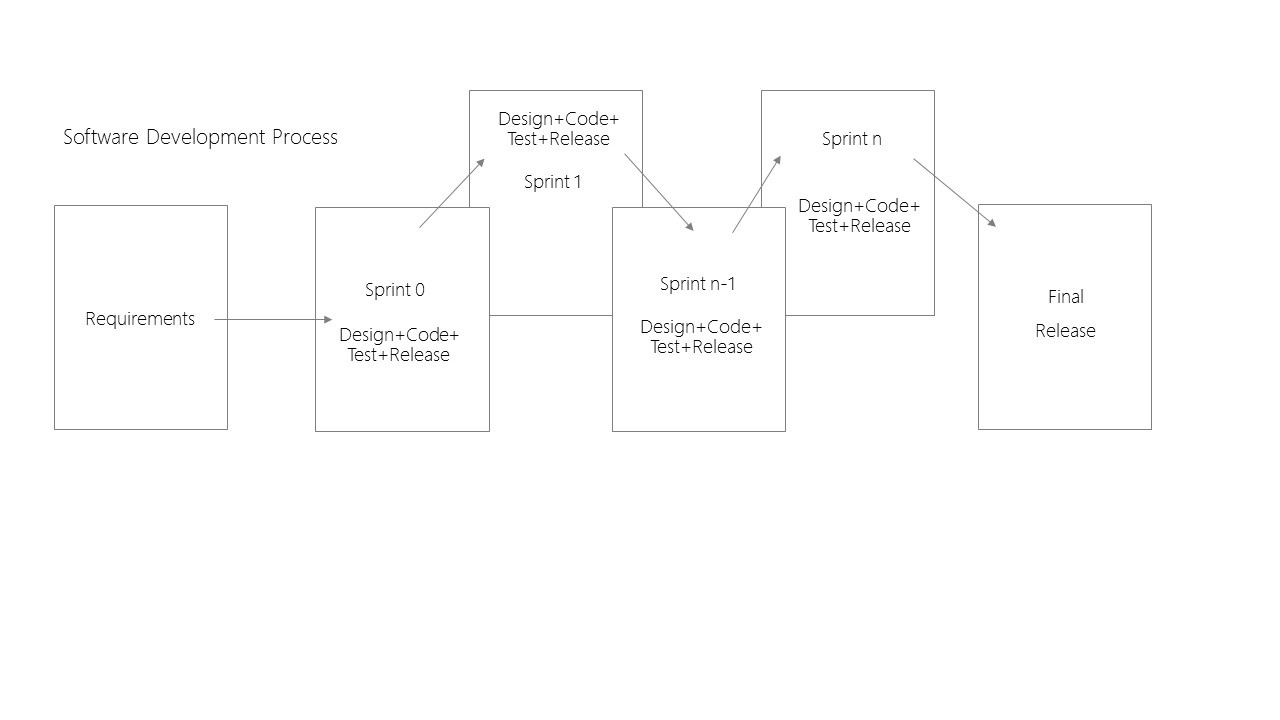
1. **Introduction**

The organisation named OnRoad is a mid-sized organisation having about 200 employees working across three locations in New Zealand. This organisation mainly focuses on applications for the transport industry like navigation software including web applications and mobile applications, transport management systems, fuel management systems, etc.

Ages ago, this organisation used to use waterfall model. But realising the drawbacks of the waterfall model, they have now, gradually, over the years switched over to agile scrum. They considered this approach mainly because requirements could be better understood, people relationship could be better developed and they could work collaboratively in a heterogeneous team.

As of now, for a given project, this organisation is structured as follows. Each project is broken down into modules and each module is taken up by a team. This team is a small heterogeneous team consisting of a business analysts, three developer, a tester and a scrum master. At the most, the team size increases to seven or ten but not more than that. This increase is due to the module size that they could be working on. According to them, if the team size is over ten then something is wrong in the implementation of the methodology.

There is no concept of project manager. The scrum master is responsible for managing the team. Apparently, this is one of the reasons of having a small team. The business analyst is responsible for gathering requirements from the product owner. He schedules meetings with the product owner and discusses the product. The product owner is the person who owns the product. The developers in the team are placed as per their skills and knowledge concerning the product under development. The tester(s) is responsible of testing and ensuring the quality of the product.



Complementing the diagram, the entire software development process can be summarised as follows. Initially, the requirements are gathered which is then passed onto the team. The team then starts working on the requirements where the development is done in sprints. Each sprint is about 2 weeks in length and the number of sprints depend on the project. The output of each sprint adds value to the project and combined they produce the working output as expected by the client.

1. **Deep dive**

To deep dive into how this organisation works, let’s start with the software development method being used. As mentioned earlier, OnRoad uses Agile Scrum as their development method. Like any other implementation, practical scenario of agile scrum tends to be a little different than theory. However, the main reason for the adoption of this methodology is the flexibility that it offers. It encourages and accommodates change and the benefit that this organisation has achieved very significantly is the opportunity to fail early and make corrections at an early stage rather than wait till the last moment.

Notably, OnRoad now finds it easier to manage their teams (since it’s smaller in size per module) and manage the whole project (since small manageable teams make up a manageable project altogether).

2.1. Requirements gathering

Another aspect of adopting agile scrum is that requirement gathering becomes effective. The members of the project are certain that they are working on a requirement that is exactly what the product owner is expecting and is of high priority. The business analyst of the team arranges a meeting with the product owner in view to capture all that the product owner is expecting of their development team. The requirements are gathered in the form of user stories where each user story is an individual requirement. Thus, all the requirements (user stories) go into the product backlog which is an analogy for a buffer. It’s a place where the requirements reside until they are taken up to be developed. It could reside there for a couple of weeks or even for the entire duration of the development (meaning that the requirement was not at all incorporated into the project, perhaps, due to its low priority).

Once the requirements are in place, the product owner decides the priority as to which user stories needs to be developed first. Since, the product owner is aware of the market and ‘knows’ the product, he (or she) would be the best person to decide on the priority.

Apart from the requirements gathered above, the non-functional requirements are also gathered. These are requirements that define how the system should be. It is also called the qualities of the system. In this organisation, the non-functional requirements are captured as user stories just like the other requirements. This gives a clear understanding of what kind of system the client is expecting.

* 1. Pre-development

Now, the time has come to take up user stories and start work on it. But how will this be decided? At OnRoad, they follow a variant of the planning poker estimate method. Basically, the effort required to accomplish the user story must be estimated by the team so that they can come to a decision on how many (priority) user stories they can complete in the (here, first) sprint. The team uses t-shirt sizing estimation where the team has cards labelled ‘XS’, ‘S’, ‘M’, ‘L’ and ‘XL’ and just like the original planning poker, the team reveals their card on what they suspect the effort to be. For each user story the all need to agree to a particular t-shirt size, or, what is meant to be the estimated effort required to complete that particular user story.

* 1. Development

Once each user story has been taken up and the sprint has been started, members of the team communicate either during daily stand up meetings or via email. These meetings are held every day and last for about 10 minutes or 15 minutes at the most if it’s unstructured. Three main questions are answered by every member of the team: what was done yesterday; what will be done tomorrow and; is there any impediments that are stopping your work. The scrum master is actively involved in such daily stand up meets to ensure that nobody hits a roadblock. The scrum master must ensure that their project is running smoothly. Email is another mode of communication, preferred to share something important during the day. It also acts as an evidence that some data was sent from person X to person Y. The general advantages of email also come along in this situation like asynchronous communication, fast and cheap mode of getting in touch, etc.

During the development process, it is very important to keep a track of each person’s activity. It is essential to know who is doing what. In this organisation, team members use TimeZone. This is a tool used to keep a record of the team members’ activity. TimeZone is fed with your current status- what job are you doing at the moment and since when you are doing that? Using this software it is also possible to measure an individual’s performance. Overall, the performance of the team is also measured using burn-down charts. The burn-down charts show the teams velocity, i.e., the number of story-points remaining to be done in the given days left.

Development, in this organisation, is done using languages such as java, c#, node.js, etc. The language depends on the platform being developed on and the technology being used. And depending on which language is used, the corresponding Integrated Development Environment IDE is used. For instance, if OnRoad plans to develop a transport app for windows platforms, then they will use Visual Studio as the IDE and language to develop the app could be either C# or any other language that the IDE supports for such applications like visual basic or JavaScript. An IDE is an all-in-one software application that supports programmers to develop the application. It is backed with compilers, debuggers, build automation support, version control support, a friendly visual user interface to code, emulators to simulate the functioning of the app being made, automatic error checks and much more. Such IDE’s make it a lot easier for developers to work.

* 1. Version control

Another important aspect of development and especially, developing in sprints is version control. If no proper management and definitions of versions exists, everything can get messed up. As said earlier, some IDE’s do support version control. But in this particular organisation, the team prefers to use GitHub. GitHub is an online git repository hosting service which enables version control and a centralised area to place the source code when working in a team. GitHub makes it easier for collaboration, testing the right version, wikis for every project and the ability to share code with the community.

* 1. Post-development

Quality is one aspect that this organisation focuses mainly on. With every sprint they ensure that their project gets better. Within each sprint, as the developer’s code and produce deliverables, testers ensure that the product is bug free. Of course, testing is not left after the entire process of development is done, instead, it is done parallel to the development. Testers prepare test cases for each user story based on the acceptance criteria that each user story has. For a user story to successfully move to the ‘done’ column of the story board, it must satisfy the acceptance criteria that is mentioned on it.

For testing, depending on the product (or feature) being tested, either Selenium or Cucumber is used. Selenium is an automated testing software for web applications. The main highlight in Selenium is that it doesn’t require any scripting language to be learnt by testers and that it supports web application written in different languages. The software is open source which also makes it widely used in the industry. Cucumber, on the other hand, is an automated testing software used for acceptance testing. The test cases are written in a behavioural driven development style which is basically simple English. Cucumber is a powerful testing software used to test various types of software. Cucumber also supports different language platforms.

While the above mentioned methods was about functional and acceptance testing, developers would also like to test their code. This is done by performing a code review and/or using a code coverage tool. In this particular organisation, for testing code coverage, Cobertura is used. Cobertura is used to test the percentage of your software code being covered by test cases. Different percentages of code coverage may be acceptable by different organisation, depending on any baselines set. For developers at OnRoad, the code coverage has to be 80% or more.

Jenkins is another interesting support that this organisation uses for continuous integration. Since the development of software happens in teams, always, it is essential to have code continuously integrated and being notified when any resource (or code files) change in central repositories like GitHub or TortoiseSVN. Jenkins server does the job of continuous integration, watches for code changes which can be set up to be done automatically, perform build and also rollbacks with various tools that Jenkins server supports.

The non-functional requirement’s that were previously captured also need to be tested. This is done through longevity testing which looks at the software from micro and macro perspectives. It encompasses methods like stress testing, which measures the performance of the system with heavy load on it. The main challenge with stress testing, also called load testing, is to ensure that the system doesn’t crash and performs efficiently given a certain number of users. For this particular load testing, the tool used to test is Gattling. Gattling measures the load on the system and generates reports even in HTML format. Other types of non-functional testing performed are usability testing, conformance testing and recovery testing to name a few.

Usability testing is more towards usage centred design. The aim is to make the software more usable by applying various usability principles to it and testing if the software meets the requirements. Conformance testing is verifying that the product meets any legal baselines, if any. The software must conform to various standards and policies that govern the industry or that particular software. Recovery testing is to test the systems’ performance after a crash or failure. Depending on the requirements, recovery testing will measure the time taken for the system to recover, the systems’ ability to backup data before the crash to perhaps enable a system restore.

1. **Challenges**

“During the process of software development, as a whole, there are many challenges that we face and most of them are probably unexpected”, says a developer from OnRoad. “And each new challenge gives us the opportunity to learn and explore more”, he adds.

Very true. No matter how well defined, certain the software development method be, implementation is prone to surprises. It’s probably hard, even for the developer of this organisation, to give a straight winner. However, according to him and his past experience, he suggests that managing people is, perhaps, the biggest challenge. The entire software development process involves assorted tools, various notations, diagrams, understanding of the method itself, legal process, guidelines and what not. And every member has to be confident with all of these. Basically, this organisation has faced challenges where some members of the team are not fully aware of the process and tools they are using and still worse- why are they using?

\*\*The organisation has also noted that sometimes, team members are not even confident whether the development methodology will work or not. They accept and continue working using the methodology because either it’s been widely accepted and it is expected to work or they don’t consider that the methodology adopted to develop software will affect their performance anyway. Thus, they are only half-heartedly into the project and are not convinced. This attitude has shown a negative impact for the organisation where in these individuals are not enthusiastic of applying agile scrum concepts, neither are they interested in discussing and having meetings.

It’s not about just learning to click this button after that button without being convinced, it’s about knowing why you are doing this. What is actually happening in the background? How is the development process being affected by each step? In organisations like this, team members are not expected to perform a task because they are just told to do so. They must have a reasoning for it and have a deeper understanding.

The additional challenge is that most team members may not have deeper understanding as they are expected to have. It is vital that they should not only be curious about the project they are doing but also be curious about the methodology with which they are doing.

So how do we get our team members involved deeply into the methodology being followed? How do we ensure that our organisations’ team members are not just working but are performing- to their fullest?

1. **Recommendation**

It’s interesting to observe how individuals that lack the confidence and enthusiasm of following agile scrum like their counterparts impact of the organisation as a whole. Agile scrum is only successful if every member is successful.

To resolve the challenges faced by OnRoad, two main aspects need to be considered here. Training team members (and customers too!) and motivation. Both these aspects are people focused and will lead to a highly productive team. The first agile manifesto reads as, “Individuals and interactions over processes and tools” (Hazzan & Dubinsky, 2014; “Manifesto for Agile Software Development,” n.d.). Certainly, more focus must be given to ensure that the individuals working in an agile environment are confident about it and are thus able to interact better.

4.1. Training

Every team member must be given adequate training required for them to understand the agile process being implemented, the tools being used and technicalities that are involved in the project. It must be ensured that each team member has an in-depth understanding of what’s happening around. Javdani Gandomani, Zulzalil, Abdul Ghani, Md. Sultan, & Meimandi Parizi (2015) have suggested that inadequate and dysfunctional training can be a real problematic situation with consequences like lack of confidence, lack of effective collaboration and unrealistic expectations. In a nutshell, lack of deep understanding of agile values will not help in the proper implementation of agile.

According to Moreira (2013), the best way to train the members of the team is to bring a ‘champion’ like an agile coach, and with his help ensure that every member of the team is effectively learning and implementing agile. The agile coach also observes, during the training, the enthusiasm of the employees and he can determine which of them are willing to become ‘workhorses’, at least, if not ‘agile champions’. During the training, it must also be imparted that every member of the team gets to learn a new skill because it is the nature of agile that allows developers to assist with performing test cases and testers doing code reviews. Dybå & Dingsøyr (2008), also suggest that it is a good idea to train the customers in scrum process so they understand what is expected of them.

* 1. Motivation

Training can help, but not without the motivation. It is essential that every member of the team is motivated to learn and experiences something new. Medinilla (2012) claims that every human has a desire, a need to learn. It is a desire to get better than what we are and such intrinsic motivation must come from within. Steven Reiss (2012), has laid the 16 forces theory where one of the many desires is curiosity- the need to learn. Curiosity is required in every individual to learn deeper and to know the real reason of why this step has been performed.

Medinilla (2012) suggests that teams could be motivated by collaborating together, having meetings done with their personalised style, maybe, they could have meetings outdoor. Self-identity is crucial in motivating and developing the interest in employees. Giving them more virtual space and office space, having them to personalise things that they value the most will all benefit the company eventually.

1. **Conclusion**

This case study, thus, describes the software development methodologies used along with the supporting tools and practices followed. The case study also highlighted the importance and purpose of following the methodologies, tools and practices and also highlighted the key success factors that this organisation can brag about. Later, this case study also included the challenges faced by the organisation with recommendations made by the author to overcome them.

**Appendix A**

1. What important software development methods and practices do you think I should understand to be part of a successful development team?
2. *How requirements are elicited (discovered)*
3. *How shared understanding of requirements (elaboration, clarification) is done What is done for release planning and scheduling*
4. *How the order of features to work on (priority) is agreed on How the expected effort to develop features is estimated for planning (eg planning poker)*
5. *How iterative and incremental development is done?*
6. *How requirements are documented/represented (e.g. user stories)*

1. What roles do you think are needed in a successful software development team?
   1. *How the team is organised – what roles and responsibilities*
2. I have been told that it is important to treat testing as important as coding through using methods like automated regression tests, automated builds, and a test-first approach. What do you think?
3. *How testing is done and what levels of testing– unit, regression, integration, acceptance, performance, load. What testing is automated? Is exploratory testing done? Is test coverage measured?*
4. *Are any quality metrics (measures) tracked? Is a test first approach used? How the build is managed– any automation? Frequency of build?*
5. We learn about practices like stand-up meetings, sprint planning and sprint reviews to keep in touch with other team members and the client in some projects. What would you recommend to keep in touch and get feedback during development?
   1. *How the team keeps in touch with each other*
   2. *How the team keeps in touch with the client (product owner)*
6. We have been told that you review the team process in retrospective meetings after every sprint and it’s ok to experiment with the process and make changes. This is part of continuous learning for the team. What do you think works well to keep the team learning?
   1. *How the progress of the development is monitored*
   2. *Do the team reflecting and continuously learn*
   3. *What the team’s reaction is to changes in features? Does the team experiment with process and practices?*
   4. *How changes to requirements are handled*
7. There are lots of tools to help with software development methods, like continuous integration tools. What do you think the important tools are to support development team over the development lifecycle?
8. *How is Continuous integration achieved?*
9. *How are non-functional or quality requirements managed?*
10. *What programming languages are used?*
11. *Any other important tools?*
12. What do you think the main success factors are, with respect to methods and tools in developing software?
    1. *What would your main focus be on- people, tools or methodology?*
13. In your opinion, what are the main challenges related to software development methods and tools I should learn about.
14. What are the chances of deviating from a particular software development method that is being followed to a method that is/was preferred by default?
15. How easy or difficult is it for everyone in the development team to follow a common methodology?
16. While developing software using the preferred methodology, what is most important to you- speed, accuracy, customer satisfaction or employee satisfaction?

**Appendix B**

The critical success factors in team based software development depends on the proper implementation of the methodology as well as many other factors. Since Agile Scrum is being used, it makes it very easy for the team to understand the requirements, know what to develop and what is important at the moment and thus, ensure that the client is happy eventually.

The challenge, unfortunately, is people. Of course, this doesn’t mean that people aren’t an asset to the company. They are very much important. The reason is that managing people is a challenge. Every member of the team must understand the methodology, the process and the product in and out. Ensuring they all understand, think and implement the same thing that is expected. Also, there are various people factors that indirectly affect the project development like minor disputes, sick leaves, etc.

**Appendix C**

1. Set A
   1. **How requirements are elicited (discovered)**

Design specification document

* 1. **How shared understanding of requirements (elaboration, clarification) is done**

Having regular meetings every sprint to focus on one feature at a time

* 1. **What is done for release planning and scheduling**

Release plan is a set of priorities that need to be done. Usually, a release plan is of no more than 90 days.

* 1. **How the order of features to work on (priority) is agreed on? How the expected effort to develop features is estimated for planning (eg planning poker)**

Since the product owner knows the best about the effect of the product in the market, he knows best which features are the most important and thus decides the priority. For each feature selected, the effort is estimated using t-shirt sizing methodology which is a variant of planning poker. The estimates here are extra-small XS, small S, medium M, large L and extra-large XL.

* 1. **How the progress of the development is monitored**

Using a job scheduling and tracking system called TimeZone.

* 1. **How the team is organised – what roles and responsibilities**

A singular team like ours may consist of a business analysts, three developers and a tester.

* 1. **How the team keeps in touch with each other**

The team keeps in touch with the help of daily stand-up meeting and scrum meetings / retrospective meetings. Teams can also keep in touch via email if not in the same location.

* 1. **How the team keeps in touch with the client (product owner)**

The business analyst usually sets up a meeting with the product owner and is usually the mediator.

* 1. **What the team’s reaction is to changes in features**

The development team definitely tests and implemented on development environment. When they are sure it’s what they want they will implement the new features to production.

* 1. **Do the team experiment with process and practices**

Not really

* 1. **Do the team reflecting and continuously learn**

Yes, of course. Each project is a new experience.

1. Set B
   1. **How iterative and incremental development is done?**

We follow agile scrum. So every sprint is an incremental development if you compare with waterfall. Each sprint produces value- a working deliverable feature.

* 1. **How requirements are documented/represented**

From the design specification document, requirements are represented as user stories which are stuck onto the storyboard.

* 1. **How changes to requirements are handled**

Every change is accepted. It is posted as a user story back onto the storyboard for the next sprint. It will get prioritised by the product owner and decided for the next sprint.

* 1. **How testing is done and what levels of testing– unit, regression, integration, acceptance, performance, load. What testing is automated? Is exploratory testing done? Is test coverage measured?**

Selenium is used for automatic functional test. Automation testing is done using Jenkins. Exploratory testing is also done. Code coverage should be more than 80% measured using Cobertura.

* 1. **Are any quality metrics (measures) tracked? Is a test first approach used? How the build is managed– any automation? Frequency of build?**

Test first approach isn’t used. Quality is measured on the results of the test automation tools and process mentioned above. Build frequency is daily.

* 1. **How is Continuous integration achieved?**

Jenkins supports continuous integration**.**

* 1. **How are non-functional or quality requirements managed?**

Non-functional requirements are measured using stress testing, longetivity testing and using a tool called Gattling.

* 1. **What programming languages are used?**

Depends on the platform and product we are working on. It could be php, java, c#, node.js, angular js, etc

* 1. **Any other important tools?**

Rally is a good tool used to capture user stories and requirements. Cucumber for testing. Github, TortoiseSVN. Various IDE to build software like Visual Studio, Xamarin, etc.

1. **What are the chances of deviating from a particular software development method that is being followed to a method that is/was preferred by default?**

Agile scrum accommodates changes (which do happen). Since we follow agile scrum, we expect changes to happen and try our best to handle it.

1. **How easy or difficult is it for everyone in the development team to follow a common methodology?**

Not very easy. This is one of the challenges we face. Not everyone is convinced about what we do, what methods we follow, etc.

1. **While developing software using the preferred methodology, what is most important to you- speed, accuracy, customer satisfaction or employee satisfaction?**

I guess accuracy and time factor is most important. We wouldn’t expect delays in the development and delivery of the software.

References

Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and Software Technology*, *50*(9–10), 833–859. http://doi.org/10.1016/j.infsof.2008.01.006

Hazzan, O., & Dubinsky, Y. (2014). The Agile Manifesto. In *Agile Anywhere* (pp. 9–14). Springer International Publishing. Retrieved from http://dx.doi.org/10.1007/978-3-319-10157-6\_3

Javdani Gandomani, T., Zulzalil, H., Abdul Ghani, A. A., Md. Sultan, A. B., & Meimandi Parizi, R. (2015). The impact of inadequate and dysfunctional training on Agile transformation process: A Grounded Theory study. *Information and Software Technology*, *57*, 295–309. http://doi.org/10.1016/j.infsof.2014.05.011

Manifesto for Agile Software Development. (n.d.). Retrieved August 15, 2015, from http://www.agilemanifesto.org/

Medinilla, Á. (2012). Motivating the Agile Workforce. In *Agile Management* (pp. 69–97). Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-28909-5\_4

Moreira, M. (2013). Evaluating Executive Support and Team Willingness. In *Being Agile* (pp. 93–103). Apress. Retrieved from http://dx.doi.org/10.1007/978-1-4302-5840-7\_10

Reiss, S. (2012). Intrinsic and Extrinsic Motivation. *Teaching of Psychology*, *39*(2), 152–156. http://doi.org/10.1177/0098628312437704