Collaborative Work Across Cross Functional Agile Teams

Trevor Forrey*
Aman Kaushik*
tforrey@asu.edu
akaush13@asu.edu
Arizona State University
Mesa, Arizona

Abstract

'The Agile way' of working has taken over as many organization's number one way to build products. Although Agile practices have seen a great success, it can still be difficult to get multiple cross-functional Agile teams to work together. Thankfully, there are many strategies to successfully navigate through this network of interlinked Agile teams. Communication, negotiation, challenging assumptions, having a willingness to change, learning from others, team structuring, tooling, and having clear designs are all important for the success of many Agile teams.

CCS Concepts • **Agile Process**; • **Team Success**;

Keywords agile process, team collaboration, design, architecture, SER574, communication

ACM Reference Format:
Trevor Forrey and Aman Kaushik. 2019. Collaborative Work Across Cross Functional Agile Teams. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, **??** pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Agile has quickly become the go-to way to develop software products. It allows teams to focus on what is important, like "Individuals and Interactions over Processing and Tools..." [**?**]. With roles clearly defined, regular communication and feedback, members from different departments - developers, testers, business analysts and product managers can seemlessly work on a single product with the same intent, incentive and vision. The feedback cycle that the Agile process creates also allows customers to be involved in the development of the product, which in turn helps the resulting deliverable. Most importantly, Agile practices allow for changes in requirements without a huge cost to development. Despite the success and great features of Agile, it can be difficult to productively implement it across teams. There are many strategies that allow multiple Agile teams to still be successful together, such as creating a collaborative environment, maintaining communication channels, and choosing the right tools. Having a clear architecture and design of a system can also point teams towards the same goals. This paper dives into the proper techniques for managing large, multi-team Agile projects.

*Both authors contributed equally to this research.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 Team Structure

The actual layout of an office can have an effect on the production of a software project. In traditional businesses, teams are split up based on their role - testers are separated from developers, who are separated from requirements writers. This role based separation can create silos within a company. These silos block team members from naturally collaborating with each other to fix a problem. While working on a multi-team Agile project, it is important to eliminate silos, as it allows for more organic collaboration [**?**]. Even splitting up teams based on their individual Agile team will cause silos based on the product features. For success in multiple Agile teams, there must be a balance between keeping team members together and creating a collaborative environment across teams.

3 Communication

Communication is one of the most important aspects when working on a multi-team Agile project. Similar to how communication can make or break individual teams, it can have even greater effects on multi-team projects. When you can't strike a perfect balance in the team layout, alternative forms of communication can help. Even physical siloes between developers can be overcome by having good communication channels. These events can come in the form of: cross-team stand-ups, multiple project owners communicating, and cross-team planning [**?**]. By making communication

convenient, teams can easily work together to move the project forward.

4 Clear Goals

Communication alone will not ensure a successful project, it is important to have a set of clear goals for each team. These goals should be publicly known, so each team knows exactly what every other team is working on. With each team having a defined and unique part of the project, they'll be able to work independently, and "...plan for cross-team dependencies" [**?**]. Similar to convenient communication, having clear goals helps teams organically collaborate. By knowing what each team is working on, teams with similar goals can make sure they develop something that's concise and can work when combined.

5 Tools

The right tooling can free a multi-team organization to focus on what's most important, development. Common collaboration tools like Jenkins, GitHub Reviews, and Automated Testing Frameworks, are just a few ways to make developing correct software convenient. These tools make sure teams don't step on each other's feet by wrongfully adapting another team's features. Build history from Jenkins can provides visuals to every team on the health of a sub project. Jenkins also allows for integration tests to be automatically performed, letting developers know if their recent additions have broken a part of the project. Reviewing code online, automated testing, and many other development tools aid in a less painful way to develop software. The tooling chosen between teams is important to decide when working on multi-team Agile projects.

6 Agile Architecture

6.1 Idea

The key idea behind an Agile architecture and design system/practice is that the existing system should remain operational for all users and at the same time should evolve to cater to new requirements. The overhead and delays associated with "re-architecturing" a system should be completely eliminated. From an end users' perspective the system should always be in a continuous state of flow [**?**]. The system should be designed for scalability, maintainability, testability and deployability. For example, a micro-services architecture allows for easiser testability and alllows for more decentralization but at the same time it decreases the deployability of the system (as compared to a monolithic architecture).

6.2 Need for Agile Architecture

When working on a large system with multiple stakeholders, each stakeholder will have a different investment into the solution and would measure value differently. Agile architecture plays a significant role in this situation. As its the architech who is responsible to meet each stakeholders' needs, create maximum value for them, maximize utilization of the project resources and for the continuous growth of the system [**?**]. It is necessary that the Agile architecture be a hybrid of both "up-front design" to create a reliabile system and "forward-thinking" to meet future needs. This may not necessarily apply to all kind of systems, as a small system can work well only with "upfront design" as well. Large scale systems on the other hand need both, the bootstrapping nature of "forward-thinking" and the stability and reliability of "upfront-design" to be successful. When working in a network of teams it becomes important to recognise that there will always be a trade off between the two and the same would need to be communicated clearly to all stakeholders to align goals and intent and to maximize collaboration[7]. Keeping in mind that the price of "re-architecturing" a large system may outweigh the value obtained from it, Agile architecture should balance the two.

6.3 Agile Architecture in a Network of Agile Teams

For successfully creating and maintaining an Agile architecture 'style' system, it is important that individual contributors be allowed to provide intputs to the architecture, i.e, its important to create a feedback loop for architecture and design as well and it should not be left upto one key high ranked individual. Communication is also a key aspect that not only promotes active participation from all stakeholders but also helps in identifying and streamlining the different views, intents and alignments. Finally, 'change' is at the core of the Agile methodology. A good Agile architecture should not only plan for change but should plan in a way that will best benefit the customer, the system and the company.

7 Conclusion

The key to making Agile work across a network of teams if to infuse 'the Agile way' into the system from the ground up. From the initital architecture to development and testing, every phase should have open communication and a feedback loop, so that every stakeholder can actively participate. Active communication and feedback also allow for better alignment of of values, goals and intent across the teams and hence result in more productive collaboration.

References

[1] A. Crocker, R. Cross, H. K. Gardner, "How to Make Sure Agile Teams Can Work Together", Harvard Business Review, 2019. [Online]. Available: https://hbr.org/2018/05/how-to-make-sure-agile-teams-can-work-together.

[2] "Collaboration Across Agile Teams", GSA Tech, 2019. [Online]. Available: https://tech.gsa.gov/guides/Collaboration_Across_Agile_Teams

[3] "Manifesto for Agile Software Development", 2019. [Online]. Available: http://agilemanifesto.org/.

[4] https://hbr.org/2018/05/how-to-make-sure-agile-teams-can-work-together

[5] https://www.scaledagileframework.com/agile-architecture/
[6] http://www.agilearchitect.org/agile/principles.htm
[7] https://www.ben-morris.com/relax-theres-no-conflict-between-architecture-and-agile/