Agile Inter-team Software Development

Sarthak Tiwari
School of Computing, Informatics, and Decision Systems Engineering
Arizona State University
Tempe, Arizona, USA
sarthak.tiwari@asu.edu

Ruben Acuña
School of Computing, Informatics, and Decision Systems Engineering
Arizona State University
Tempe, Arizona, USA
ruben.acuna@asu.edu

Abstract

With the boom in the use of agile process model, imperfect implementations of agile are frequently being seen, out of which the problem of inter-team communication in agile teams is one of the most common issue. An agile team by its definition is a group of people who are self-sufficient to bring their responsibilities to closure, the interaction between different teams is thus considered minimal in most projects implementing agile. This causes problems when the integration of end-products of different teams is carried out. In this paper we have taken a detailed look at this problem and how we can mitigate this.

CCS Concepts • **Software and its engineering → Agile software development**; *Programming teams*; • **Social and professional topics** → *Project management techniques.*

Keywords Agile, Software Engineering, Software Development, Process Models, Inter-team, Management

ACM Reference Format:
Sarthak Tiwari and Ruben Acuña. 2018. Agile Inter-team Software Development. In *SER574 '19: Advanced Software Design, January 20, 2019, Mesa, AZ*. ACM, New York, NY, USA, 3 pages. https://doi.org/12.1337/1122456.1122456

1 Introduction

Recently, agile approaches have become popular in software development. As outlined in the Agile Manifesto [2], an agile process gives individuals and interactions value over processes and tools. As software grows in complexity, situations arise where multiple agile teams need to work together to achieve a common goal. In this paper we survey several the issues that can arise when agile teams must work together. We also discuss approaches, which if implemented correctly, provide a more robust and functional process for inter-team agile development. Although an independent team may use an agile process with ease, additional considerations are required for complex systems where running multiple agile teams in parallel becomes necessary. The agile approach of interacting in-person fails due to a few reasons such as the vertical structure of the company organization and the existence of middle managers [3].

The transition to multiple teams increases the importance of inter-team communication, from simple interpersonal communication, to system specification. In an agile process, where Big Design Up-front may typically be avoided, the move to multiple teams motivates a stronger need to communicate design decisions. In order for sucessful integration of techonological components, system elements must have well defined functional specifications, and interfaces. This large number of intermediaries in inter-team communication causes the process to fall apart as it takes longer than usual time for messages to reach their destination then what agile can afford. This problem is difficult to fix as this is so embedded in the working agile culture of the current organizations that changing them will take a long time, thus an immediate patch that works is required.

This document is separated into four sections: Introduction, Sportify: A Canonical Example, Proposed Approach and Conclusion. In Section 2, we discuss the a real world example of large scale agile project and show a problem encountered in that. In Section 3, we present some adaptations that can be done in agile process model to resolve the problem discussed.

2 Spotify: A Canonical Example

For a typical example of a company using many agile teams, consider Spotify who has over 30 teams [4]. The basic element of the development organization at Spotify is the Squad, analogous to a Scrum team. Each Squad is the autonomous developer, with a localized working area, of a particular subsystem in the product. Squads in related areas are grouped together into Tribes. The overall structure is illustrated in Figure 1. Although larger than Squads (7-10 members), Tribes are limited in size (< 100) to ensure smooth communication.

2.1 Communication

A fundamental issue with tasking teams with different goals is that teams can become isolated with respect to the context of their work. At Spotify [4], offices are laid out so that the Squads making up a Tribe are spatially close. This leads to an environment where an informal exchange of information on the Tribes work will occur. Consider another scenario: a tester on a particular agile team invests time to solve a particular problem, which is also faced by another tester on a different team. Without communication, these team members would perform redundant work. To address this problem Spotify defines Chapters [4], which are crosscutting

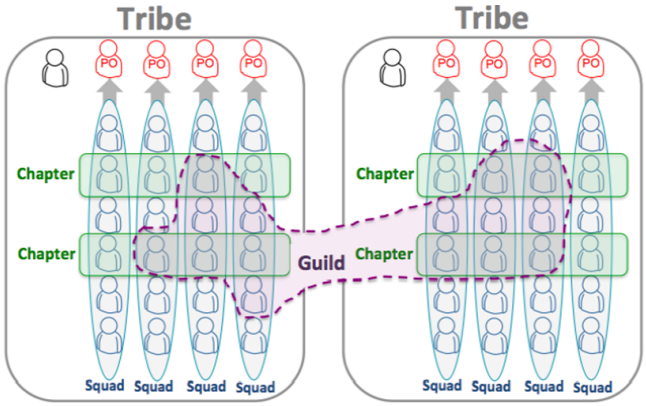

Figure 1. Agile team organization at Spotify [4].

divisions which gathers individuals with similar roles across different Squads.

3 Proposed Approach

The general problem of communication can be tackled from two perspectives. First, stakeholders at different levels need to have ways to pass information. This can be addressed by adaptions in development process and leveraging technology. Second, the quality of communication needs to be accurate. This can be addressed by sound system specification through architecture and design.

3.1 Process Improvements

There are many small changes [1] that we can incorporate in our process model to make sure that parallel developing agile teams do not run into problems when they reach the integration phase. These changes are to be made part of the entire process and are not to be implemented only in the end.

One of the most critical aspect of agile process model is to have daily standups which are the platform serving the purpose of letting each team member know the work being done in the other parts of the team and correlate it with the work being done by them. This results in escalation of differences between the development early in the process and prevents end moment discovery of mismatches in interfaces and such. In the case of multiple agile teams this problem is compounded as usually a daily stand up is a closed activity of the team itself, thus preventing other teams from knowing the results or discussions of each other. This can be resolved by having a representative of each concerned team being present in the daily standup thus letting each team know the status of other teams.

Though in usual implementations the product owner is responsible for agile teams under his supervision, in large projects with multiple agile teams where a number of product owners are present sometimes over time the vision of the owners may get too distinct from each other thus pushing the development track in different directions. This can be limited by having regular meetings of product owners where the scope and vision of the project could be synced again. This can be a bi-weekly or monthly meeting depending on the size of the project.

All the initial, intermediate and final planning sessions should be made such that all the teams which are or could be impacted by that part of the project are part of the meeting. This can assist in early agreement on high-level requirements and standardization of inter-team interfaces.

3.2 Technology Improvements

A good technology stack can be a powerful tool in maintaining a widely distributed team. Good communication and management tools can help the teams in keeping track of things that they need to do so that other teams can work as intended. The following are some of the areas where the technology can assist the teams in making a more efficient agile process environment.

As agile focuses on personal interaction with highest importance given to face-to-face conversation, conferencing tools such as video conferencing and WebEx etc. can help the team interactions become more fluid and clearer. They also make the communication real-time thus removing the lag in process due to time spent in communicating ideas across teams.

Continuous integration tools can go a long way in finding out inter-team problems early in the process as every time any team makes a change, its impact on the entire project can be seen.

3.3 Importance of Architecture/Design

As we know that the product owner in an agile process model is the person with the vision of what the project will look like and if it is a small team the product owner can clearly pass this to each and every member of the team and even each member can query the product owner directly when in doubt. But in large scale projects where multiple agile teams are working together in supervision of a few product owners it is practically impossible for the owner to keep doing what they did in a small team, that's where a formal definition of their vision comes in handy as it enables the teams to look up to something when encountering a design decision. The design or architecture in this case acts as a common vision for not only the teams but also for the communication between all the product owners. The design acts as a "deadlock breaker in decisions" [6] as when the teams can't come to a common consensus the design shows the path to take. Spotify addresses this issue with a more organizational approach by defining a "system owner" role [4]. This role is more casual than an architect role, and focuses on defining a "go-to" person who can maintain long term stewardship over a sub-system.

In addition to specification of system elements, architecture can also be used to empower teams. As discussed by Parnas [5], there are two general approaches to decomposing systems: 1) compartmentalizing a computational process and, 2) focusing on information hiding. The latter approach ideal for agile development since this approaches defines system components in terms of design dicisions, which empowers invidual teams.

4 Conclusion

As we have seen, inter-team communication can be an obstacle in agile teams working on the same thing as intermediaries in communication and lack of common personal causes things to go haywire if left unchecked. To reduce this problem a number of small additions can be done to the process as specified but the most important part of the change is a good architecture and design which is available to all the teams and thus provides a common framework to all the teams involved.

References

[1] [n. d.]. Collaboration Across Agile Teams. https://tech.gsa.gov/guides/Collaboration_Across_Agile_Teams/

[2] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. 2001. Manifesto for Agile Software Development. http://www.agilemanifesto.org/

[3] Nicolas Frankel. 2016. Making Sure Inter-Team Communication Doesn't Work. https://dzone.com/articles/making-sure-inter-team-communication-doesnt-work/ [Online; posted 13-October-2016].

[4] Henrik Kniberg and Anders Ivarsson. 2012. Scaling Agile Spotify with Tribes, Squads, Chapters & Guilds. https://blog.crisp.se/wp-content/uploads/2012/11/SpotifyScaling.pdf

[5] D. L. Parnas. 1972. On the Criteria to Be Used in Decomposing Systems into Modules. *Commun. ACM* 15, 12 (Dec. 1972), 1053–1058. https://doi.org/10.1145/361598.361623

[6] Scott W. Ambler. [n. d.]. The Architecture Owner Role: How Architects Fit in on Agile Teams. http://agilemodeling.com/essays/architectureOwner.htm/