Communication in Agile Inter-team 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

As the complexity of software development increases, companies relying on agile processes are asked to apply agile at large scale. A typical approach is to run several agile teams in parallel, however, this can conflict with the basic tenets of agile which focus on interaction and flexibility. A key aspect in this transition is maintaining quality communication to ensure progress, which can be addressed with appropriate adaptations in process, technology, and architecture. In this paper, we review the use of agile for large projects, and appropriate adaptations.

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. 2019. Communication in Agile Inter-team 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

Agile approaches are popular in software development for many reasons such as adaptability towards change, early customer satisfaction, and increase in team effectiveness with time. However, the concept of an agile process does not lend itself to large teams and is not suitable for large projects [6]. In complex software, situations arise where multiple agile teams need to work together to achieve a common goal. When scaling to multiple teams, the most important consideration is enabling effective communication, both personal and architectural, which is the focus of this paper. For an example of a company developing a complex piece of software, consider Spotify AB who has over 30 teams [4]. Spotify is a streaming music client which supports media, commercial, and social features. The basic element of development organization at Spotify is the Squad, analogous to a Scrum team. Squads in related areas are grouped together into Tribes. The overall structure is illustrated in Figure 1.

As outlined in the Agile Manifesto [2], an agile process gives individuals and interactions value over processes and tools. An agile team should contain all that is required for them to do their work, thus interactions are intensely intra-team. However, diverse feature sets such as required in the Spotify application necessitate many skill sets and substantial manpower. By using a multi-team model, large projects can benefit from agility, provided that the process is adapted. Multiple teams increases the importance of communication, from interpersonal to system specification. The agile approach of interacting in-person can fail due to reasons such as the vertical nature of a company and layers of middle managers [3]. The large number of intermediaries can cause an agile process to breakdown as it takes longer for messages to reach their destination than can be afforded. In agile, where Big Design Up Front may be avoided, the move to multiple teams also motivates a need to communicate design. For successful integration of components, system elements must have well defined functional specifications and interfaces.

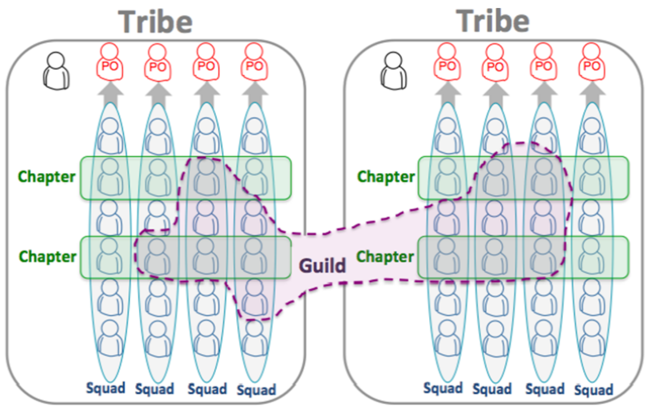

Figure 1. Agile team organization at Spotify [4].

This document is organized as follows: Introduction, Communication Concerns, Supporting Communication in Inter-team Development, and Conclusion. In Section 2, we discuss communication issues. In Section 3, we present process adaptations that can help resolve issues. Section 4 concludes with an overview of key aspects of inter-team agile development.

2 Communication Concerns

A fundamental issue in tasking teams with separate goals is that teams can become isolated from other teams pursuing similar or related goals. Consider a scenario at Spotify [4]: a tester in a particular Squad invested time to solve a problem, which was also faced by another tester on a different team. Without communication, members would perform redundant work. To address this problem Spotify defines Chapters, which are crosscutting groups which gather individuals with similar roles across different Squads. At Spotify, offices are laid out so that the Squads making up a Tribe are spatially close [4]. This leads to an environment where an informal exchange of information on the Tribe's work will occur.

Consistency becomes an issue when expanding to multiple teams. The product owner in an agile process model is the person with the project vision. In a small team, they can clearly communicate it to every team member, and members can even query the product owner. However, in large projects with multiple teams, it is practically impossible for the owner to continue what they did in a small team. This motivates a need for a formal definition of their vision as it gives teams something to consult to when encountering a design decision. Without a functional specification, teams will be unable to produce components which can be easily integrated.

3 Supporting Communication in Inter-team Development

The general problem of communication can be tackled from two perspectives. First, stakeholders at and across different levels need to have communication channels. This can be addressed by adaptations in development process and leveraging technology. Second, the quality (e.g., accuracy) of communication needs to be high. This can be addressed by thorough system specification through architecture.

3.1 Process Adaptations

There are many changes [1] that can be incorporated into a process model to decrease the chance that parallel agile teams run into problems during development or integration.

One aspect of an agile process is daily standups which are a platform for letting each team member know the work being done by other team members. This uncovers differences in understanding early and prevents last minute discovery of discrepancies. For multiple teams, this problem is compounded as a stand up is a closed activity, preventing other teams from knowing the discussions of each other. This can be resolved by having a representative from each team be present in a standup, e.g., a Scrum of Scrums, where stand up questions are given but reframed for a team [6].

Although the product owner is responsible for agile teams under his supervision, projects with multiple agile teams may have multiple product owners, and over time the vision of the owners may drift and lead in different directions. This can be limited by regular meetings of product owners where the scope and vision of the project can be synchronized.

All the initial, intermediate and final planning sessions should be organized such that 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

Selecting appropriate technology plays an important role in enabling communication and task management so that teams can work together efficiently. Since agile focuses on interactions, like face-to-face conversation, conferencing tools can lower the bar to inter-team interactions. They make communication real-time, reducing delay in communicating ideas across teams. Continuous integration tools can go a long way in identifying inter-team problems early in development as it reveals the impact of changes on the entire project.

3.3 Importance of Architecture and Design

System architecture acts as a common vision for not only the teams but also for communication between the product owners, and is the main topic for most of the inter-team communication that occurs in a multi-team environment. It needs to be as formal, and as clearly defined as possible, as it is the system design that acts as a “deadlock breaker in decisions” [7] for situations where teams disagree on a particular design path, and ensures that components will be constructed so they can be integrated into the whole. A formal architecture description also helps to mitigate communication problems by reducing the need for inter-team communication as a good design document provides enough guidance for a developer to incorporate their work into the system. Thus, a good design document sometimes acts as a proxy for a product owner when it comes to detailing the requirements of the customer. A second approach addresses this issue from an organizational standpoint 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 a long term stewardship over a sub-system.

Architecture can also be used to enable team agility. 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 is ideal for agile development since it defines components in terms of hiding design decisions, which empowers teams to design their own solution.

4 Conclusion

As we have seen, inter-team communication can be an obstacle when agile teams work together, as intermediaries in communication and lack of personal communication causes disruption in the agile process. To reduce this problem a number of adaptions can be made to the process to ensure the flow of information. However, the most important adaption is a well defined design and architecture which is available across teams and enables teams to produce individual system components which can be easily integrated.

References

[1] U.S. General Services Administration. 2018. 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] Kenneth S. Rubin. 2012. *Essential Scrum: A Practical Guide to the Most Popular Agile Process* (1st ed.). Addison-Wesley Professional.

[7] Scott W. Ambler. 2012. The Architecture Owner Role: How Architects Fit in on Agile Teams. http://agilemodeling.com/essays/architectureOwner.htm/