Inter-team Coordination in Large-scale Agile

Xinkai Wang†
Ira A. Fulton Schools of Engineering
Arizona State University
Tempe Arizona US
xwang585@asu.edu

Yu-Ting Tsao
Ira A. Fulton Schools of Engineering
Arizona State University Tempe
Arizona US
ytsao2@asu.edu

ABSTRACT

Agile development is a new type of software development method that has attracted widespread attention since the 1990s. It is a software development capability that can respond to rapidly changing requirements. Because of its efficient nature, more and more small teams are turning to agile development as their first choice. But agile development can be challenging when it comes to large software projects where multiple small teams work together.

This paper mainly describes the difficulties encountered in coordination in large-scale agile development. How to avoid these problems is the focus in our future work. It also discusses the importance of Design/Architecture in large-scale agile.

CCS CONCEPTS

• Software and its engineering

~Software creation and management

~Software development process management

~Software development methods

~Agile software development

KEYWORDS

Inter-team coordination, Large-scale agile, Design/Architecture

ACM Reference format:

Xinkai Wang and Yu-Ting Tsao. 2020. Inter-team coordination in large-scale agile. In *Proceedings of ACM Woodstock conference (WOODSTOCK'18). ACM, New York, NY, USA, 2 pages.* https://doi.org/10.1145/1234567890

1 Review Agile Concepts

With the rapid development of the new generation of information technology and the upgrading of consumer demand, product functional demand is increasingly diversified and uncertain. The traditional software development methods represented by the waterfall model can no longer adapt to the pace of product updates. As technology updates and user needs continue to diversify, agile development gradually replaces traditional development. As a new method of software development that can quickly respond to the functional requirements of users, agile development is not only widely used in the software industry, but also gradually spread to other industries.

The main feature of agile development is that the software development process attaches great importance to human initiative. Agile software development method mainly has the following five advantages:

• Precise requirements and accurate results. Agile development has a very short development cycle. This allows agile development to more accurately accomplish the task of each phase. At the same time, the shorter development cycle also allows the program to better adapt to changing requirements.

• Guaranteed quality. Agile methods have strict requirements for the quality of each iteration. Agile development teams have a high level of development methodology, and some test code will be finished before the formally development.

• Customer cooperation is better than contract negotiation. Good teams care more about working with customers.

• Highly return on investment. In an agile development, the most valuable features are always prioritized to development, so that the customer will always get the maximize return on investment.

• High speed is one of the most significant benefits of agile development. Agile teams focus on developing only the parts of the project that are most needed and most valuable today. Because that the software can be quickly put into development.

Agile development, of course, has its own disadvantages. Agile development focuses on human communication and ignores the importance of documentation. If the turnover of project personnel is too large, it will increase the difficulty of software development and maintenance. Nevertheless, the advantages of agile development have made it increasingly the most popular method of software development.

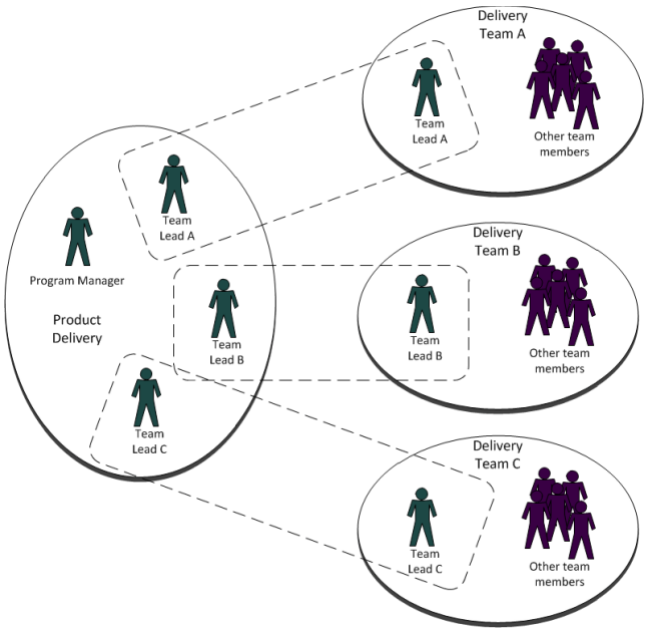

Figure 1. Organization structure of an agile product-delivery team.

2 Impediments and Solutions

When agile teams form together into a bigger agile team, they usually need to deal with the following impediments. First, it always comes to the **unfamiliarity** to each other from the other side. Knowing roles and skills from team members is considered a majored part in agile methodology. However, people usually like to stick together with someone who is already familiar with. Especially when the bigger agile team is formed in a short period of time, a barrier between the two sides is often an obstacle to cross. This would cause a higher cost when it comes to communicate during the agile process.

Second, the **tools** for both teams need to be standardized. The word "tools" I mean here isn't just for development tools. It also includes agile management tools, issue tracking tools, communication tools, code conventions, and etc. Before forming large team, each group always had their own standard tools and were acquainted with them. However, a uniform technology must rule this bigger agile team. Also, a period of learning time should be considered within the software development process.

Last but not least, a **reallocation** for similar roles should be reviewed before the bigger team starts. The software projects may look very different and need to reallocate the members into small or big size for each functional branch teams. For example, a backend-oriented software project would require a bigger number of backend engineers. And by forming a bigger team doesn't mean the value of each member goes down. It should, precisely, form a more flexible and robust team.

2.1 Organize

Agile development team is a new form of self-organization and cross-functional team cooperation, which advocates the form of self-organization and self-management of the team. Therefore, the quality of teamwork is an important factor related to the performance of agile development teams. However, it has not been fully studied which contents in the daily work of the team will affect the team performance.

Through the study of some actual projects, I found that most of the good large agile development teams have the following characteristics:

1. Unified phase goal

It is means to determine what functions need to be done and what goals need to be achieved in one cycle. During the development process, the team moves toward a unified goal, without interrupting progress by inserting new features. Once the phase goals are set, the team begins work. Once the goal is complete, the team is ready to move on to the next functional iteration.

2. Decoupling

The main purpose of decoupling is to maintain independence. It is divided into three aspects:

a. Technical decoupling

This means the underlying core architecture, applications, and UI are clearly layered, which makes it easier to develop functional extensions.

b. Business decoupling

That is, the independence of the business function module. large functions need to be divided into independent feature modules to facilitate the division of development tasks.

c. Team decoupling

Divide large teams into separate teams (5-8 people are preferred). Each team is responsible for independent feature module development. Each team includes product requirements analysts, project managers, developers, and UI designers. Although the size of each team is small, its staffing is comprehensive.

2.2 Solution to Unfamiliarity

People factors account for agile methodology a large proportion and it is always the one with the most uncertainty. But still, try to take the following ideas to tackle the ice wall.

1. Before the project start, let two teams know each other with brief introduction and skills review.
2. Get used to cooperate with the other team through unofficial events, such as team building activities, knowledge share time, and etc.

Large-scale team always bring the unfamiliarity issue at the beginning. But the main point is how short can two teams eliminate such atmosphere.

2.3 Unify Tools

Nowadays technology teams are collaborating using several different tools. Even inside the same department may vary from one to another. The two agile teams should send their leader to decide what kind of tools matches the best for the following projects, not just based on the one that they were familiar with.

When it comes to a dilemma, a good strategy to analyze it is using metrics to comparison. The leader should take the whole software development into consideration. A suitable communication tool can make the team knows information faster; A user friendly scrum board tool can clarify the large-scale team's process; A standard code convention shared by two teams can make the code readable and cleaner. A real large-scale agile team is not just a format in name, moreover, it should really share commons among team member in such details.

2.4 A Way to Reallocation

It is not a good idea to **randomly** form two teams to one agile team just because the project needs more people. It depends on the project topic, the schedule, the skill sets of each team, and etc.

The leader from each team can go to the other team and have a short conversation with them. And after evaluating each person, the leader can match the suitable team member with similar skill sets together. It seems a small and trivial step to take before the cooperation start. However, this kind of approach could usually generate an optimistic result.

2.5 Agile Architecture

In large-scale agile development, architecture is critical. This is because it is difficult for large agile development teams to communicate as frequently and in detail as small teams. Therefore, a good software architecture can form a unified development standard within a large agile development team. This ensures that different modules developed by different teams are better integrated.

Agile development is an iterative and progressive development method, which runs through all aspects of agile software development. Following this philosophy, agile architecture should also be an iterative process. Each iteration models only the current requirements. Gradually improve the architecture through iteration. But each iteration must not compromise the capabilities of the existing architecture or the functionality already implemented.

There are three main points to follow when designing an agile architecture:

• The Architecture backlog should be aligned with the sprint backlog. That is, modeling only the current sprint requirements.

• The architecture should be consistent with the domain model. Avoid domain misunderstandings that can lead to architectural failure.

• The architecture is designed to ensure conceptual integrity. This means that each team should work together in a unified and clear concept.

ACKNOWLEDGMENTS

Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here.

REFERENCES

[1] Working Together in Agile Teams: https://www.benlinders.com/2019/working-together-in-agile-teams/
[2] How to Make Sure Agile Teams Can Work Together: https://hbr.org/2018/05/how-to-make-sure-agile-teams-can-work-together
[3] Why is Sprint Zero a Critical Activity? Dick Carlson, Mr. Soukup
[4] Agile Teams: https://www.scaledagileframework.com/agile-teams/
[5] Software Architecture Design and Practice of Agile Development
[6] Large Agile Teams: https://disciplinedagiledelivery.com/agility-at-scale/large-agile-teams/