Solutions for Effective Cross-team Collaborations

Bingrui Feng  
 CIDSE  
 Arizona State University  
Mesa, AZ, USA  
 bfeng16@asu.edu

Huijing Liang  
 CIDSE  
 Arizona State University  
Mesa, AZ, USA  
 hliang36@asu.edu

ABSTRACT

Nowadays more and more companies start adapting using agile methodology to manage their software development process. Agile methodology provides tons of benefits such as fast, early delivery and flexibility. For the small size of a company, it usually only requires a small size of the team to work on one single project. However, in larger companies, cross-team collaborations are more common. Lack of efficient collaboration across the team slowly becomes a growing pain. So how can we make different agile teams work together efficiently? Based on our research we think solutions should at least include: clear objectives and separation, no departmental silos, collaborative architecture and design guidelines [1]

CCS CONCEPTS

* Software and its engineering [8]
* Software creation and management [8]
* Software development process management [8]
* Software development methods [8]
* Agile software development [8]

KEYWORDS

Agile, Design, collaborations, Cross-team, Architecture, source control, real-time interactions, cloud

**1    Clear objectives and separation**

Typically, agile teams are interdisciplinary and consist of developers as well as QA test engineers and some people who write documents [1]. Every team works separately with their own sprint. But all the work should be done in a uniform way (coding standard). The functions of every agile team need to be considered functionally relevant in an agile development environment [2]. Clear team goals are key to the success of the project. Therefore, at the beginning of a project, we must make a clear assignment of work. And we need to ensure the timeliness and effectiveness of communication between the leaders of each team so that when one team's work is adjusted, we can change the work tasks of the other teams timely.

Each team member should know the goals of the team and separation of work clearly. The clear separation of work among multiple teams also creates independence for every team to reduce the reduplicative work and enables the team to identify and plan dependencies across teams. The best method is developing and maintaining a team product backlog for every team to make sure every team is working on solutions towards the final goal [1].

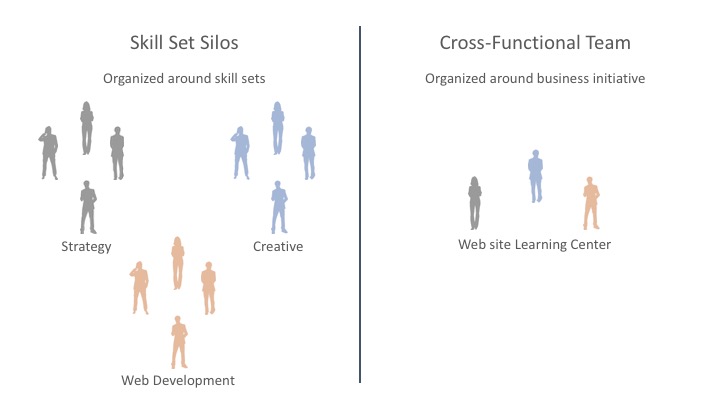
Meanwhile, cross-team daily stand-ups are necessary. each team should choose one representative to attend daily meeting, it is convenient to share team updates and remind every team to adjust their work plan [1]. So that the responsibility of every team will be much clearer, which is crucial to the delivery of the whole picture. 

Figure 1 referenced from [9]

**2 No departmental silos**

In software development industries of al size, it is very common for multiple departments being involved in the same project and working together. However, under these situations, departmental silos sometimes become obstacles of achieving collaboration. It is the duty of the executive leaders and management to prepare and equip their teams with the proper mindset and vision to break down these destructive organizational barriers [3].

Proper Communication and willingness to share necessary information are the key to solve departmental silos. Each department usually have their own perspective regarding special terms. In a big project, lack of communication or information sharing can be very harmful to the project, many times may even not be done purposely. Another reason that could result in departmental silos might be the design flaws of the project. During the designing process, it should enhance the interaction and necessary information shared between departments, and all of those should be also be documented on file, knowing who oversees what, and who to seek for information is very important. If at the beginning of the design process, all of those are clear then the communication will be more efficient and better for the final delivered product

**3    Collaborative architecture and design guidelines**

According to the 11th principle in the Agile Manifesto, “The best architectures, requirements, and designs emerge from self-organizing teams.” [7] Therefore, the organization should provide guidelines for better collaboration between teams, and some rules should be formulated, such as daily stand-up meetings between teams, Multiple Product Owners Check-in, Cross-Team Retrospectives and so on. The goals of every team will be more clear and repetitive work will be reduced by these methods.

At the same time, these methods also help ensure the timeliness of communication. The agile teams are also based on discipline all the time. Although the agile team advocates self-organization and self-management of the team, it must also follow guidelines. The team guidelines in the agile team are stricter than the disciplines of other R & D teams. For example: Whether in the initial stage of product demand or the final gray test, this requires the team to make frequent reviews and necessary adjustments; frequently deliver workable products; timely planning of iteration time; attention to the rhythm of research and development; Information visualization; continuous integration and automated testing.

Agile architecture is also very important in the development process. It is active and evolvable. It evolves with the software development process. In fact, agile architecture meets the goal of continuous value delivery within the "shortest sustainable delivery time" [6].

Agile development has a certain impact on software architecture design, giving people the impression of "light architecture design" in agile development. Firstly, I think agile development is a software process method and tool. Agile development itself does not represent the architectural design. This is like the difference between architectural architecture design and construction project management, which are two aspects of architecture. The same software industry is similar. Software architecture design describes the thing itself, while agile development describes the process of creating this thing. Firstly, traditional architecture design includes architecture and design. The design can include detailed designs, such as detailed UML diagrams (detailed class diagrams, sequence diagrams, etc.), detailed API designs and interface descriptions, and storage layer database table fields Design and much more. In today's fast-changing society, business requirements and technologies are also changing rapidly. It takes 30% (or more) of the time to design the architecture in the early stages of the software process. Either the software developed does not meet market requirements, or Demand changes, resulting in large change costs. For example, we understand an e-commerce product. The current functions were planned and designed two years ago, and if new requirements occur, they will be adopted in the next version, resulting in the entire product being out of the market and customer needs. The second point is that architecture design includes two aspects, one is architecture, and the other is design. The detailed design in the design requires a lot of time, including detailed process, API, data structure design. However, the Code coding phase of the software development phase also contains a lot of detailed design content, so there is a Repeat Yourself situation between the two. In other words, agile development now advocates Code is design, and previously it was Design is code. But the problem is that software developers maintain a set of designs and a set of codes, which is overwhelming and inefficient. So, now Code is Design is prevailing and agile is prevailing. For these two reasons, agile development is not suitable for this kind of architectural design content. Agile architecture divides traditional architecture design into Architecture + Design:(1) Retaining the architecture part of the agile architecture. (2) Transfer design to Code coding stage, refactoring stage, Unit Test.

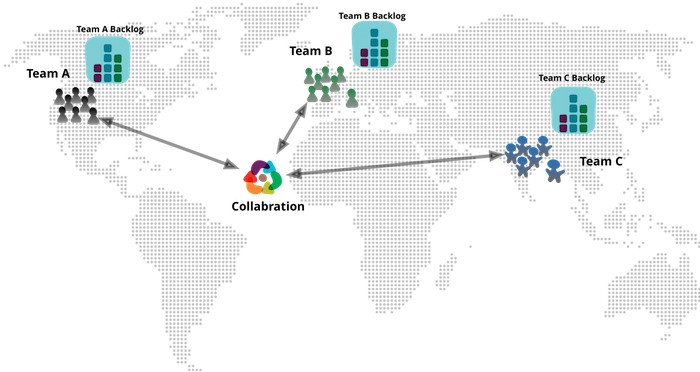


Figure 2 referenced from [5]

**4    Technologies for Collaboration**

We can use high technology to manage the Agile teams. Sometimes the teams are big so it’s difficult to manage. It is also hard to track every team’s work without proper management. So some technical tools have become necessary. We can use online task tracking and project management tools. In some cases, actual to-do items with sticky notes are good, but not good for dispersed teams. Use online tools for smooth, dynamic project and task management, easy prioritization, visibility and transparency, and better collaboration [4]. For example:

Visual boards, Scrum boards help teams plan together, make progress visualization and concentrate on their objectives [1]. Software applications such as JIRA, Trello and VersionOne provides features that facilitate discussion and collaboration between teams through joint planning, progress tracking, and frequent communication [1]. In the previous semester, we have used taiga board to manage sprint and backlog, it is a very efficient way and keeps everyone updated with other people’s progress

Video streaming software such as Skype and Zoom enables real-time interaction and communication. You can also share screen to visualize things that you have been talking about, it will be very helpful for teams that are not even located in the same office, as long as they joined the same conference room online, they are able to share their knowledge and opinions right away

We can also use cloud drive to exchange documents, like using Google Docs for easy sharing of collaboratively, which is usually used for creating documents and charts for documentation purposes. It allows editing history, commenting, creating and including charts in documents at the same time. It gets updated in real-time, so all participants can see exactly what changes are happening and review for any further enhancements. [5]

Moreover, there are some source control tools and Integration Tools, which helps teams that are not co-located or even in the same office.[1] With source control tool, each team can have its own branch and version of the source code, they can easily get the latest stable code and update their own branches without affecting other teams. When a release is around the corner, after meeting with each other, they can create 1 specific branch that contains everybody’s changes for the new release and then after unit tests, regression tests and integration tests even automation tests, and bug fixing usually, it can finally go to production.

ACKNOWLEDGMENTS

During drafting this paper, Both Huijing and Bingrui have been doing a lot of research and constantly writing in order to finish the paper. We both also help each other find any grammar mistakes or sometimes rephrase each other’s sentences. We communicate frequently to keep each other updated and we use a google doc to collaborate real-time work together. Overall this paper is a great teamwork outcome.

REFERENCES

[1] Anon. GSA Tech Guides. Retrieved January 20, 2020 from <https://tech.gsa.gov/guides/Collaboration_Across_Agile_Teams/>

[2] C.M. Tartaglia and P. Ramnath, 2006. Using open spaces to resolve cross team issue [software development] DOI: <https://ieeexplore.ieee.org/document/1609818/authors#authors>

[3] Brent Gleeson. 2017. 5 Ways to Destroy the Pesky Silos in Your Organization. (June 2017). Retrieved January 20, 2020 from [https://www.inc.com/brent-gleeson/5-ways-to-destroy-the-pesky-silos-in-your-organization.htmlConference](https://www.inc.com/brent-gleeson/5-ways-to-destroy-the-pesky-silos-in-your-organization.htmlConference%20)

[4] Tomas Gutierrez. 2015. How to Manage Distributed Teams Using Agile Techniques. (October 2015). Retrieved January 23, 2020 from <https://www.scalablepath.com/blog/manage-distributed-teams/>

[5] Sunit Parekh. 2015. ​Collaboration Techniques for Large Distributed Agile Projects. (February 2015). Retrieved January 23, 2020 from <https://www.thoughtworks.com/insights/blog/collaboration-techniques-large-distributed-agile-projects>

[6] Anon. 2019. Agile Architecture in SAFe. (September 2019). Retrieved January 23, 2020 from <http://www.scaledagileframework.com/agile-architecture/>

[7] Kent Beck et al. 2001 (2001). Retrieved January 23, 2020 from <https://agilemanifesto.org/principles.html>

[8] Anon. 2012. ACM Computing Classification System. (2012). Retrieved January 23, 2020 from <https://dl.acm.org/ccs>

[9] Jim Ewel. 2018. Cross-Functional Teams - Part Two. (March 2018). Retrieved January 23, 2020 from <https://agilemarketing.net/crossfunctional-teams-part/>

Conference Short Name:WOODSTOCK’18

Conference Location:El Paso, Texas USA

ISBN:978-1-4503-0000-0/18/06

Year:2018

Date:June

Copyright Year:2018

Copyright Statement:rightsretained

DOI:10.1145/1234567890

RRH: F. Surname et al.

Price:$15.00