Agile Multi-Team Coordination

Cohesively Building Small Components of Large Projects

Jonathan Bush  
School of Computing, Informatics, and Decision Systems Engineering  
Arizona State University  
Tempe, Arizona, United States  
 jeb@asu.edu

ABSTRACT

Conventional software development lifecycles (SDLCs) focus on designing a complete system early in the process and rely on documentation for further guidance. In large-scale projects with many developers and a large codebase, these processes provide clear division of labor and direction. However, conventional SDLCs also come with large process overhead that impedes the ability of the organization to consistently generate new business value. Additionally, these processes are prone to high bug rates and long development cycles.

Agile software development methods attempt to alleviate some of the problems with conventional methods by eschewing the principle of big up-front design. In scrum, the most popular agile method, software is developed iteratively through a series of sprints. Frequent interaction between the scrum team and the customer generates new requirements over time. This ultimately yields a higher quality product that aligns more closely with the customer’s actual needs.

Special precautions are required when applying agile methods to large-scale projects. Since these projects are too large for a single scrum team to address, the work is divided among several teams. These teams must coordinate their efforts to yield a coherent product. One common approach to this is scrum-of-scrums, where meetings are held between representatives from each team. Alternate approaches combining conventional SDLCs with agile methods have also been proposed.

Ultimately, a key consideration in any software development process is software architecture. The system architecture guides the division of tasks between teams within the organization and facilitates component integration. With agile methods, how to develop and maintain a complex system architecture must also be considered.

CCS CONCEPTS

• Software and its engineering---Software creation and management---Collaboration in software development---Programming teams

KEYWORDS

Agile, architecture, Scrum

1 Conventional Processes and Agile

In many large-scale development projects, traditional software development life cycles (SDLCs) have been utilized because of their well-understood natures. Large projects often dictate the creation of substantial documentation and may have additional traceability requirements for safety-critical systems. Although conventional methods have advantages in these areas, they also represent a substantial overhead that does not necessarily contribute business value that can be offered to the customer.

Considering this overhead, agile processes attempt to directly provide business value by relying on iterative design and customer interaction to guide the development process. This approach has been successfully applied to many small-to-medium sized projects but experiences some difficulties when applied to large projects with many development teams working on different parts of the overall systems.

One potential difficulty of applying agile methods (namely scrum) to large-scale projects lies in coordinating the distribution of work and the communication between many smaller agile teams working on the same overall product. Communication difficulties in agile methods are particularly apparent in global software development, where teams do not have access to the face-to-face communication typically required for agile software development [1]. In addition to geographic constraints on communication, the meetings involving many teams with distinct roles may also be difficult to manage and may provide little value to the teams. A typical approach is scrum-of-scrums (SoS), where meetings consisting of delegates from each scrum team are held in an attempt to coordinate the efforts between the teams. However, it was found that such “Grande SoS” meetings are limited in effectiveness due to time limits on scrum meetings and the relevance of information presented during the meetings [2].

Despite these challenges, the benefits of agile methods in terms of business value and flexibility [3] have driven agile methods to be adapted for large software projects.

2 Adapting Scrum for Large-Scale Projects

When building a complex software product, it is necessary to break down the development into parts that can be developed by separate groups and individuals before integrating the parts into the final product. In scrum, it is customary to deliver new working functionality at the end of each sprint. This ensures that the development process is continuing to meet the developing requirements of the customer. This iterative development makes scrum suitable for high-risk projects, where all requirements cannot be clearly established at the beginning. Agile methods have also demonstrated “shorter development cycles, higher customer satisfaction, [and] lower bug rates” [3]. However, as the complexity of a project increases, it becomes more difficult to manage through scrum.

One possible way to deal with the complexity issue is to utilize scrum within the context of a conventional SDLC that is intended for managing large projects. Such a system was proposed by Cho, who suggested combining a streamlined version of the Rational Unified Process (RUP) with scrum to accommodate the “strengths while suppressing the weaknesses of both methods” [3]. This hybrid system works by bounding the scrum process within the phases of RUP, as shown in Figure 1. This union provides the structure and predictability of RUP while also accommodating the benefits of an iterative agile development process. The structure of RUP and the big-up-front design in terms of system architecture guide the scrum teams to work cohesively throughout the remainder of development. Without this structure, significant additional communication between scrum teams would be required to coordinate product development between teams focused on distinct components.

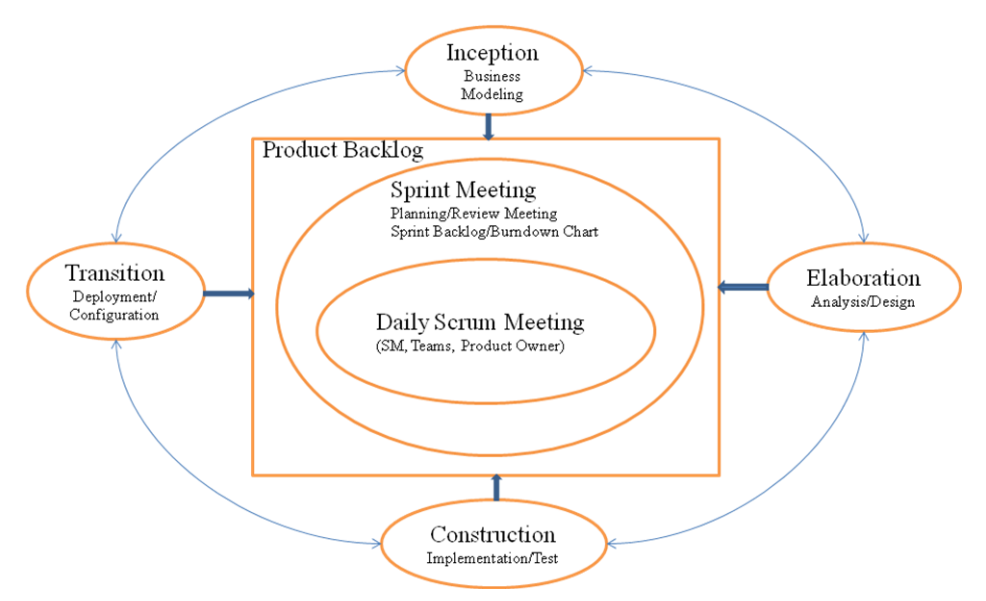


Figure 1: Hybrid RUP/Scrum model suggested by Cho [3].

Although the hybrid approach proposed by Cho takes steps to reduce process overhead within the adopted RUP disciplines [3], the process still results in constraints on the scrum process. A second possible way to apply scrum to large-scale projects is through the scrum-of-scrums (SoS) system. SoS involves scrum meetings between teams instead of individuals, typically held at least twice each week [2]. Paasivaara, et al. investigated the application of SoS within large-scale projects and identified several difficulties and possible solutions. One “challenge of the SoS meeting is not make [sic] it into a status reporting meeting for management, but to keep it as a synchronization meeting between teams” [4]. However, 15-minute meeting time may lead teams to gloss over problems, assuming that they are not relevant to other teams [2]. The case studies noted that a multi-layered approach with meetings focused by a “content/architecture-based model” were more effective than one project-wide meeting [2]. This approach increases the relevance of discussions between team representatives, yielding more useful results.

Paasivaara, et al. also suggested modifying the basic scrum meeting questions to make them more applicable for SoS meetings (Table 1). Note that these questions focus only on information relevant to other teams, as other information is shared within team scrum meetings.

Table 1: Modified SoS meeting questions [2].

|  |
| --- |
| What did your team do since the previous meeting that is relevant to some other team? |
| What will your team do by the next meeting that is relevant to other teams? |
| What obstacles does your team have that affect other teams or require help from them? |

3 The Significance of Project Architecture

Software architecture and inter-team coordination are the main factors for success in large-scale agile development. Software architecture defines the structure of the project and leads the division of tasks among agile teams. Inter-team coordination ensures that effective collaboration both within a team and within the organization as a whole [5]. In conventional SDLCs, the software architecture is designed early in the process, influencing subsequent development from preset requirements. However, in agile methods the architecture develops organically throughout the iterative development process.

The support for “emergent architecture” in agile processes is intended to focus development on the business value provided to the customer [6]. Rather than upfront architecture design which constrains the process to predetermined features, agile methods allow for adaptation to changing requirements. With large projects it becomes more difficult to manage emergent architecture, since the individual teams must maintain a consistent vision. Eckstein recommends forming a “Technical Service Team,” responsible for maintaining the architecture throughout the development process [6]. The technical service team provides an internal service to the feature teams while alleviating the need for coordinating system architecture.

Overall, “architecture provides a way to partition work around large chunks of software development, guiding the organization into teams” [7]. This critical role necessitates that architectural decisions be made wisely, as they have a profound impact on the organization. Organizing teams around architectural features rather than areas of expertise enables self-organization and consistent business value delivery [6], which are core principles of agile.

4 Conclusion

Applying agile methods to large-scale projects can be a difficult task, given the challenges of team coordination and architecture. However, the benefits of reduced process overhead and improved software quality have led many organizations to adopt agile methods successfully.

# References

|  |  |
| --- | --- |
| [1] | M. Paasivaara, S. Durasiewicz and C. Lassenius, "Distributed Agile Development: Using Scrum in a Large Project," in *2008 IEEE International Conference on Global Software Engineering*, Bangalore, India, 2008. |
| [2] | M. Paasivaara, C. Lassenius and H. Ville, "Inter-team coordination in large-scale globally distributed scrum: do scrum-of-scrums really work?," in *ACM-IEEE International symposium on Empirical software engineering and measurement*, 2012. |
| [3] | J. Cho, "A hybrid software development method for large-scale projects: rational unified process with scrum," *Issues in Information systems,* vol. 10, no. 2, pp. 340-348, 2009. |
| [4] | C. Larman and B. Vodde, Practices for Scaling Lean & Agile Development: Large, Multisite, and offshore Product Development with Large-Scale Scrum, Boston: Addison-Wesley, 2010. |
| [5] | T. Dingsøyr and N. Moe, "Towards principles of large-scale agile development," in *International Conference on Agile Software Development*, Rome, 2014. |
| [6] | J. Eckstein, "Architecture in Large Scale Agile Development," in *International Conference on Agile Software Development*, Rome, 2014. |
| [7] | R. Nord, I. Ozkaya and P. Kruchten, "Agile in Distress: Architecture to the Rescue," in *International Conference on Agile Software Development*, Rome, 2014. |

Conference Name:ACM Woodstock conference

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