Xue Cheng

Dr. Deborah Scally

ECS 3390.5H3

October 28th, 2019

Comparative Analysis of Extreme Programming and Scrum

Introduction

Nowadays businesses operate in a rapidly changing environment. To get new business opportunities or markets, they have to respond to changing economic conditions and competing products quickly. And almost all business operations rely on software, so rapid software delivery is critical for business systems. Also, it's impossible to get a set of stable software requirements since requirements change frequently. However, traditional software development relies on complete requirement documents, high-level design, implementation, and integration (Awad,1). A new software development model has to be produced.

In the 1990s, engineers began to look for suitable software development processes that could meet market demands constantly within the shortest time. A variety of lightweight software development processes were produced such as rapid application development, the Scrum process, Crystal Clear, extreme programming, and so forth (Anwer, 1). In 2001, the seventeen software engineers gathered to discuss the development processes and decided to name the processes "agile", which means light and sufficient (Awad, 8). Also, they wrote the Agile Manifesto which contains the core values and principles of the Agile software development process.

The purpose of this paper is to compare two commonly used Agile software development processes, Extreme Programming and Scrum. This paper introduces the process of these software development phases, analyzes the similarities and differences, and discusses the advantages and disadvantages.

Extreme Programming

Extreme programming is a lightweight incremental software development process that was popularized by Kent Beck in 1996 (Anwer, 1). Extreme programming is widely used because it emphasizes teamwork and customer satisfaction, which helps to improve the efficiency of development and deliver high-quality software rapidly according to customer requirements (Mahajan, 699). Extreme programming involves high-degree disciplines and five core values to guide the practices.

The life cycle of Extreme programming contains six phases: exploration phase, planning phase, iteration to release phase, productionizing phase, maintenance phase and death phase (Anwer, 1). In the exploration phase, customers write the user's stories, and developers define the architecture of the system and estimate the cost and time of the implementation. The planning phase focuses on prioritizing the user stories, planning for the next iteration, and estimate the effort for the tasks. Iteration to release phase contains the basic development activities such as designing, coding implementation, unit testing, and integration. In productionizing phase, developers deliver the software in small releases to test whether it's ready for production. The next phase is the maintenance phase which aims to maintain the running of the software system. Also, new features can be released incrementally. The death phase is the last phase of extreme programming. If the customers are satisfied with all the functionality, the system will be released finally. If the customers have new requirements that can't be developed, they may consider closing the project.

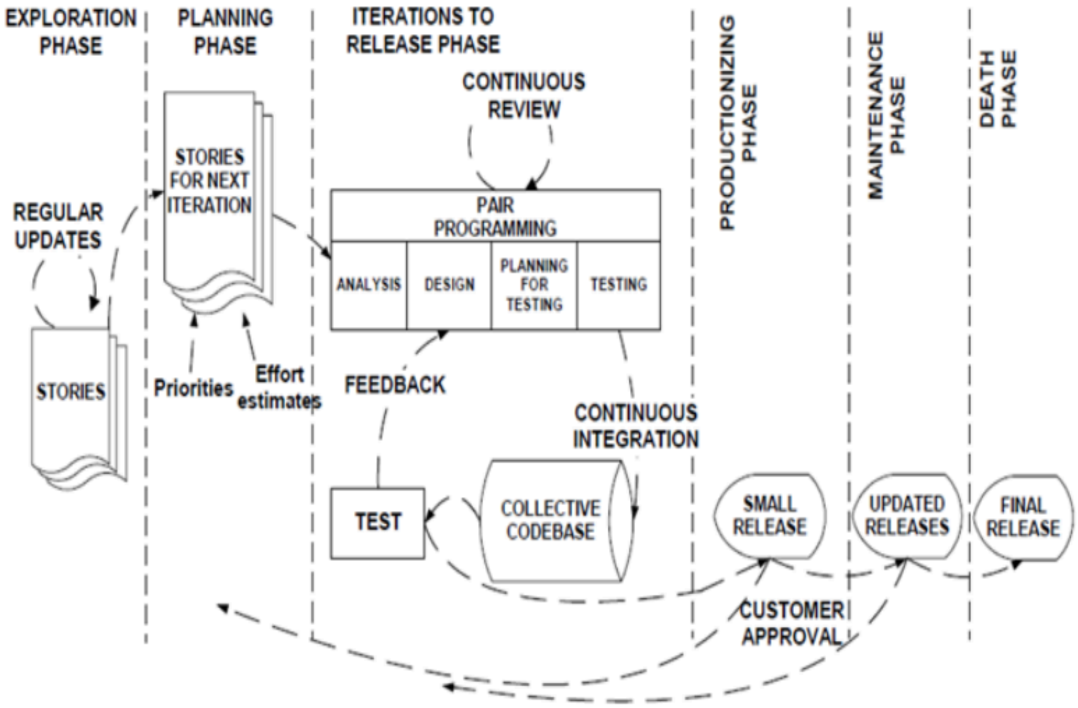


Figure 1: Life Cycle of Extreme Programming

Extreme programming has five core values including communication, simplicity, feedback, courage, and respect (Beck, 30). In the traditional software development process, code implementation relies on intact requirement documentations, but extreme programming stresses communication among team members, customers, and managers. Simplicity means to figure out the simplest solution for the problem. Beck states that it's better to do a simple thing today than to do more complicated things since the requirements change frequently (Beck, 30). Programmers can get direct feedback from the system by writing and running tests, and functional feedback from the customer reviews, which helps the software steer in the right direction. Courage is valuable when combined with communication, simplicity, and feedback since it helps programmers to speak out crazy ideas, explore multiple solutions to find the appropriate one, and persisting in figuring out solutions. In Extreme programming, team members respect the contributions of each person and help to seek for the best solution (Mahajan, 703).

Scrum

The term "Scrum" comes from the rugby game, which means "getting an out-of-play ball back into the game" (Schwaber). Scrum was created by Jeff Sutherland and Ken Schwaber in 1995 (Awad). It is based on iterative development and provides an incremental process for product development. Scrum emphasizes cooperation among team members and cross-functional teams.

Scrum activities can be divided into 3 phases called Pregame, Game, and Postgame. Figure 2 lists the framework of Scrum. In the Pregame phase, the product owner is responsible for collecting the requirement from customers and prioritizing required features to the Product Backlog. Also, this phase contains other important activities such as time estimation, risk management, architecture design, funds approval, and so on. In the Game phase, the development process is based on iterations called sprint. Sprints are usually one to four weeks in length. Customers, product owners, users, and the scrum team attend the Sprint Planning meeting to discuss the functionality and set goals. Then all the features are assigned to the specific sprints to form a Sprint Backlog. Team members and Scrum master focus on the implementation. In the Daily standup meeting, the Scrum team members report the process of the task and talk about obstacles or issues. In the Postgame phase, Integration testing is necessary for the final release. The final release indicates all the goals and features of the software are met.

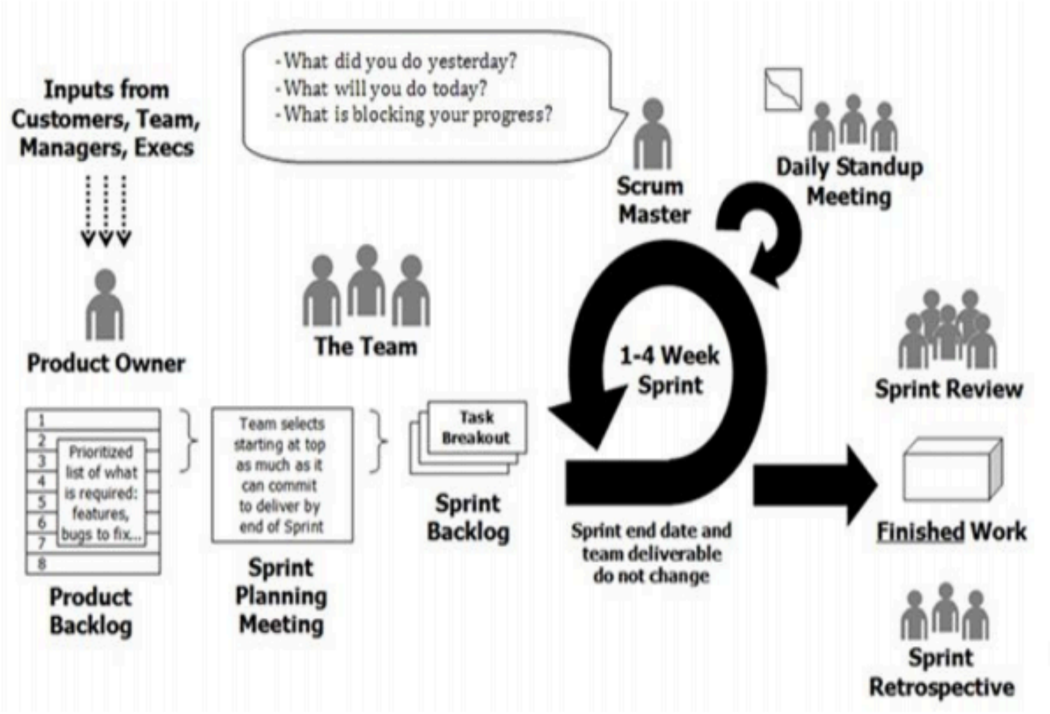
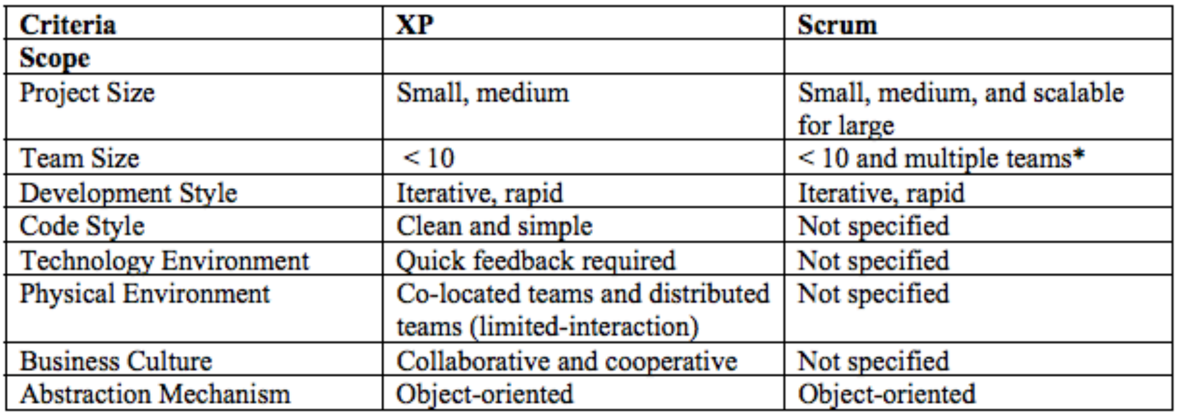

Inputs from Customers, Team, Managers, Execs
-What did you do yesterday?
-What will you do today?
-What is blocking your progress?
Scrum Master
Daily Standup Meeting
Product Owner
The Team
1-4 Week Sprint
Sprint Review
1
2
3
4
5
6
7
8
Prioritized list of what is required: features, bugs to fix...
Team selects starting at top as much as it can commit to deliver by end of Sprint
Task Breakout
Sprint Backlog
Sprint end date and team deliverable do not change
Finished Work
Product Backlog
Sprint Planning Meeting
Sprint Retrospective

Figure 1: Scrum Framework

Comparative Analysis

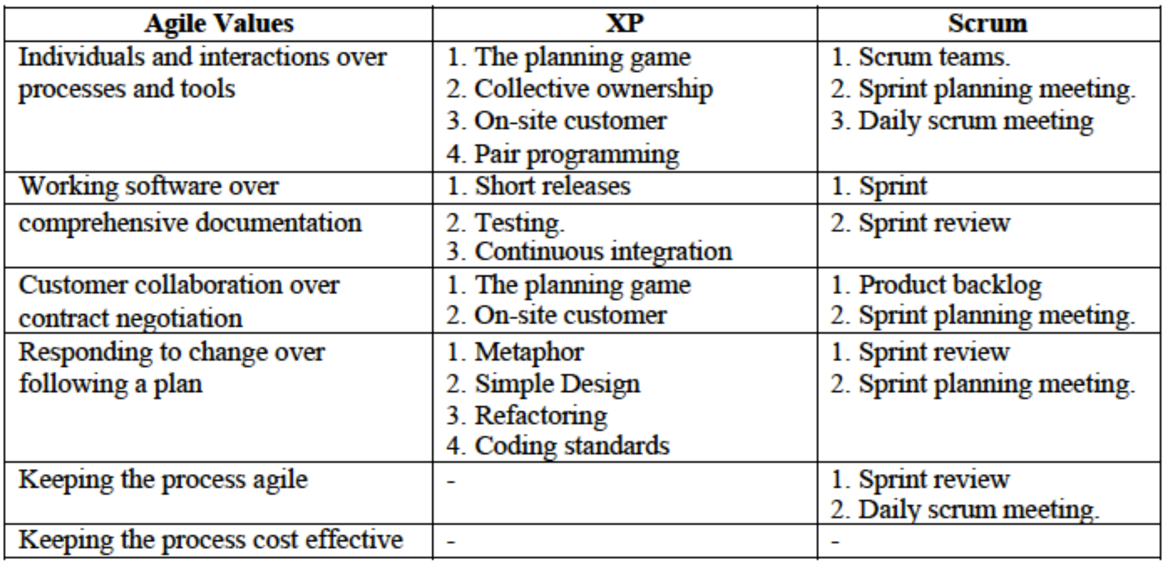
Table 1 analyzes the scope of Extreme Programming and Scrum comparatively (Qumer). The table shows that both Extreme Programming and Scrum can be used in small and medium-sized projects. Also, Scrum can be used in large projects. The team size of Extreme Programming and Scrum is less than ten, but Scrum can have multiple teams. Both of these two development methodologies use the iterative process to develop software rapidly. Extreme Programming defines code style, technology and physical environment, and business culture while Scrum does not have a clear specification.



Criteria	XP	Scrum
Scope		
Project Size	Small, medium	Small, medium, and scalable for large
Team Size	< 10	< 10 and multiple teams*
Development Style	Iterative, rapid	Iterative, rapid
Code Style	Clean and simple	Not specified
Technology Environment	Quick feedback required	Not specified
Physical Environment	Co-located teams and distributed teams (limited-interaction)	Not specified
Business Culture	Collaborative and cooperative	Not specified
Abstraction Mechanism	Object-oriented	Object-oriented

Table 1: Scope of Extreme Programming and Scrum

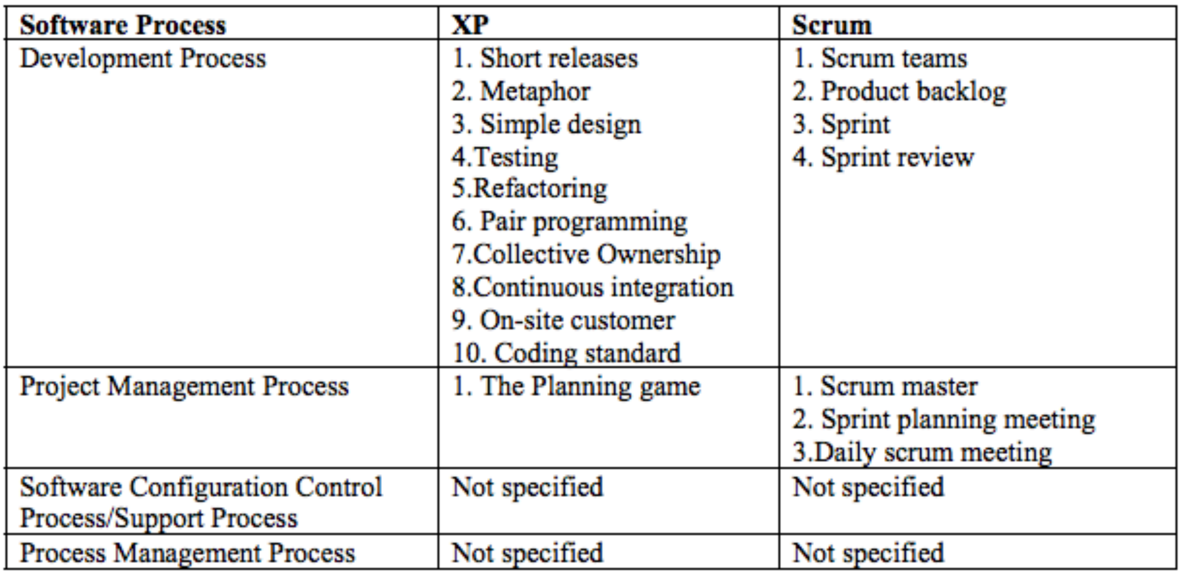
Table 2 compares Agile values in Extreme Programming and Scrum (Qumer). From the table, we can find that both Extreme Programming and Scrum activities reflect the Agile values. However, Extreme Programming does not have the practices that reflect the value of keeping the process agile. Also, neither Extreme Programming nor Scrum indicates the value of keeping the process cost-effective.



Agile Values	XP	Scrum
Individuals and interactions over processes and tools	1. The planning game 2. Collective ownership 3. On-site customer 4. Pair programming	1. Scrum teams. 2. Sprint planning meeting. 3. Daily scrum meeting
Working software over	1. Short releases	1. Sprint
comprehensive documentation	2. Testing. 3. Continuous integration	2. Sprint review
Customer collaboration over contract negotiation	1. The planning game 2. On-site customer	1. Product backlog 2. Sprint planning meeting.
Responding to change over following a plan	1. Metaphor 2. Simple Design 3. Refactoring 4. Coding standards	1. Sprint review 2. Sprint planning meeting.
Keeping the process agile	-	1. Sprint review 2. Daily scrum meeting.
Keeping the process cost effective	-	-

Table 2: Agile values in Extreme Programming and Scrum

Table 3 compares the software process of Extreme Programming and Scrum from the development process, project management process, support process and process management process (Qumer). Both Extreme Programming and Scrum have activities for development and project management process, but they don't include practices for support or process management process.



Software Process	XP	Scrum
Development Process	1. Short releases 2. Metaphor 3. Simple design 4.Testing 5.Refactoring 6. Pair programming 7.Collective Ownership 8.Continuous integration 9. On-site customer 10. Coding standard	1. Scrum teams 2. Product backlog 3. Sprint 4. Sprint review
Project Management Process	1. The Planning game	1. Scrum master 2. Sprint planning meeting 3.Daily scrum meeting
Software Configuration Control Process/Support Process	Not specified	Not specified
Process Management Process	Not specified	Not specified

Table 3: Process of Extreme Programming and Scrum

Conclusion

Extreme Programming and Scrum are two widely used Agile development methodologies. Both of them are suitable in small and medium-sized projects and use an iterative process to develop software features rapidly. We compare Extreme Programming and Scrum from their scope, core values, and process. When considering which methodology is most suitable for the project, it depends on the specific project.

Works Cited

Awad, M. A. "A comparison between agile and traditional software development methodologies." University of Western Australia (2005): 30.

Schwaber, Ken, and Mike Beedle. Agile software development with Scrum. Vol. 1. Upper Saddle River: Prentice Hall, 2002.

Anwer, Faiza, et al. "Comparative analysis of two popular agile process models: extreme programming and scrum." International Journal of Computer Science and Telecommunications 8.2 (2017): 1-7.

Mahajan, Er Rohini, and Er Pawanpreet Kaur. "Extreme Programming: Newly Acclaimed Agile System Development Process." International Journal of Information Technology 3.2 (2010): 699-705.

Beck, Kent, and Erich Gamma. Extreme programming explained: embrace change. Addison wesley professional, 2000.

Qumer, Asif, and Brian Henderson-Sellers. "Comparative evaluation of XP and Scrum using the 4D Analytical Tool (4-DAT)." Proceedings of the European and Mediterranean conference on information systems. 2006.

Cohen, David, Mikael Lindvall, and Patricia Costa. "An introduction to agile methods." Advances in computers 62.03 (2004): 1-66.

Schwaber, Ken. "Scrum development process. Business object design and implementation." (1997): 117-134.