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, companies are required to respond to changing economic conditions and competing products quickly. Almost all business operations rely on software, so rapid software delivery is critical for business systems. Also, it is 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.

This paper introduces the processes of Extreme Programming and Scrum, compares these software development methodologies from scope, agile values, and projects processed, and give a recommendation at last.

**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.

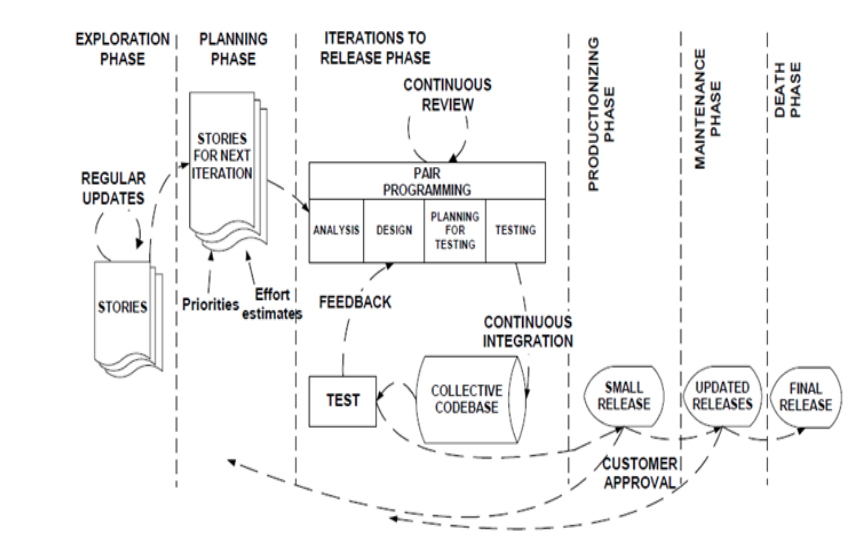
The life cycle of Extreme programming contains six phases: exploration phase, planning phase, iteration to release phase, productionizing phase, maintenance phase and death phase (Anwer, 1). In the exploration phase, customers write the user's stories, and developers define the architecture of the system 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. The iteration to release phase contains the basic development activities such as designing, coding implementation, unit testing, and integration. In the productionizing phase, developers deliver the software in small releases to test whether it’s ready for production. The second to last 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 ultimately released. If the customers have new requirements that cannot be developed, they may consider closing the project.

Figure 1: Life Cycle of Extreme Programming

Extreme programming has five core values which includes 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 is 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 along with functional feedback from customer reviews, which helps steer the software in the right direction. Courage is valuable when combined with communication, simplicity, and feedback, as it helps programmers to speak out crazy ideas, explore multiple solutions to find an appropriate one, and persist 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, and funds approval. In the Game phase, the development process is based on iterations called sprint, which 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. Afterwards, all the features are assigned to the specific sprints to form a Sprint Backlog while team members and Scrum master focus on the implementation. In the Daily Scrum 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, which indicates all the goals and features of the software are met.

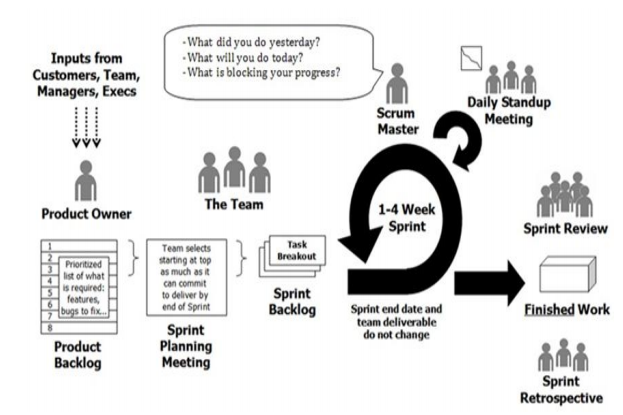


Figure 1: Scrum Framework

**Comparative Analysis**

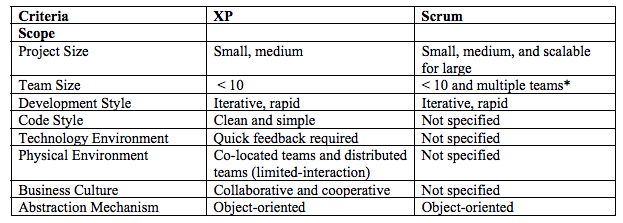
Table 1 analyzes the scope of Extreme Programming and Scrum comparatively (Qumer). This table shows that both Extreme Programming and Scrum can be used in small and medium-sized projects while Scrum can be extended for large projects. The team size of Extreme Programming and Scrum is less than ten, but 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, as well as business culture, while Scrum does not have a clear specification.

Table 1: Scope of Extreme Programming and Scrum

Table 2 compares Agile values in Extreme Programming and Scrum (Qumer). From this 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.

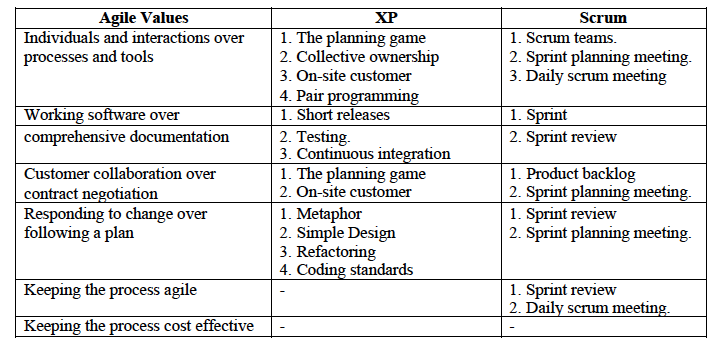
****

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 processes, but they do not include practices for support or process management processes.

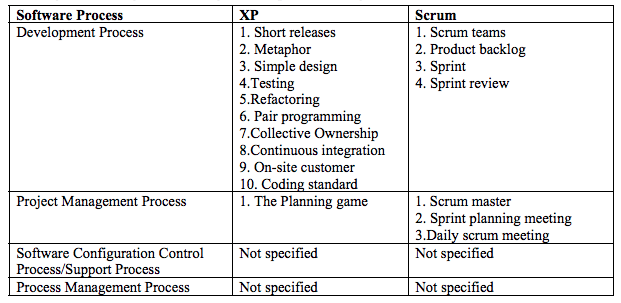
****

Table 3: Process of Extreme Programming and Scrum

**Recommendation**

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. I recommend Extreme Programming, and the reasons are listed as below. Firstly, Extreme Programming teams are much more flexible to change during their iterations. As long as the team has not started on a particular feature, a new feature can be swapped into the iterations. However, for Scrum teams, they do not allow any changes within their Sprints once they have made commitment to develop the features after the sprint planning meeting. Secondly, Extreme Programming teams develops the features based on priority that is decided by the customers, which means they develops the most basic and important features first, and then deliver other features incrementally. But Scrum teams can determine the development sequence after product owner prioritizes the features. Last but not least, I like practices of Extreme Programming such as pair programming, simple design, and refactoring.

**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.