Xue Cheng

Dr. Deborah Scally

ECS 3390.5H3

December 1st, 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 (Cohen, Lindvall and Costa 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 (Beck).

The purpose of this paper is to do a comparative analysis of Extreme Programming and Scrum and then give a feasibility recommendation. This paper introduces the processes of these two methodologies, compares them from the scope, agile values, and project processes, and gives a recommendation.

**Extreme Programming**

Extreme programming is a lightweight incremental software development process that was popularized by Kent Beck in 1996 (Anwer et al). 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 and Kaur 699). Extreme programming involves high-degree disciplines and five core values to guide the practices.

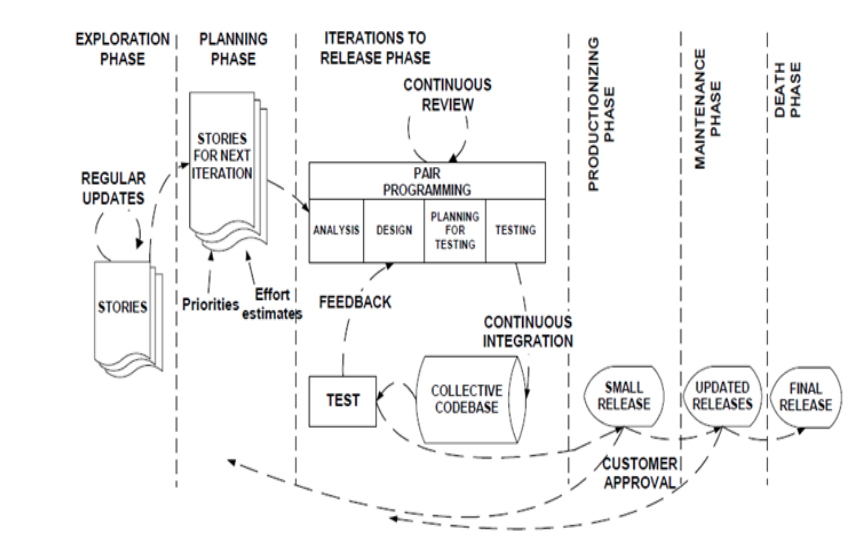
The life cycle of Extreme programming contains six phases: exploration phase, planning phase, iteration to release phase, productionizing phase, maintenance phase and death phase (Anwer et al). In the exploration phase, customers write the user's stories, and developers define the architecture of the 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 estimating 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 (Anwer et al).

Extreme programming has five core values which include communication, simplicity, feedback, courage, and respect (Beck and Gamma 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 and Gamma 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 and Kaur 703).

**Scrum**

The term “Scrum” comes from the rugby game, which means “getting an out-of-play ball back into the game” (Schwaber and Beedle 23). Scrum was created by Jeff Sutherland and Ken Schwaber in 1995 (Awad 30). It is an object-oriented development cycle that is based on iterative development and provides an incremental process for product development (Schwaber 117). 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 usually lasts one to four weeks. Customers, product owners, users, and the scrum team attend the Sprint Planning meeting to discuss the functionalities and set goals. Afterward, 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.

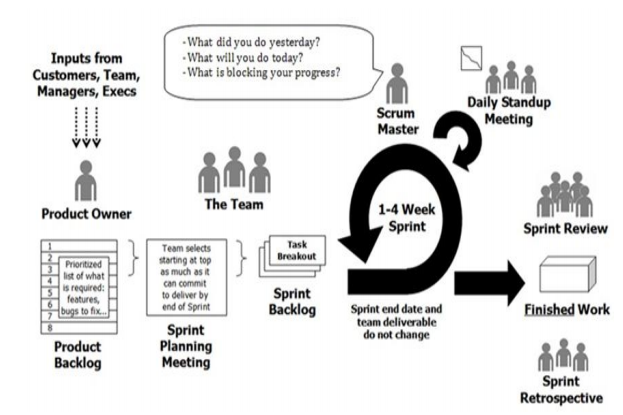


Figure 1: Scrum Framework (Anwer et al).

**Comparative Analysis**

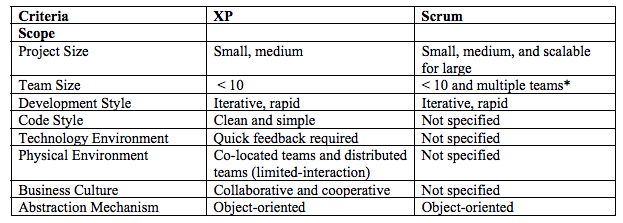
Table 1 analyzes the scope of Extreme Programming and Scrum comparatively (Qumer and Henderson-Sellers 6). This table shows that both Extreme Programming and Scrum can be used in small and medium-sized projects while Scrum can be extended for large projects. The team size of Extreme Programming and Scrum is less than ten, but Scrum can have multiple teams. Both of these two development methodologies are based on the iterative processes and use an object-oriented approach to develop software rapidly. Extreme Programming defines code style, technology, and physical environment as well as business culture, while Scrum does not have clear specifications on those aspects.

Table 1: Scope of Extreme Programming and Scrum (Qumer and Henderson-Sellers 6).

Table 2 compares Agile values reflected in Extreme Programming and Scrum (Qumer and Henderson-Sellers 7). From this table, we can find that both Extreme Programming and Scrum activities support the basic 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.

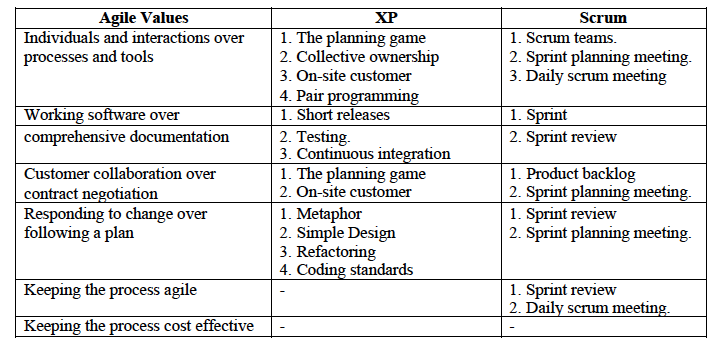
****

Table 2: Agile values in Extreme Programming and Scrum (Qumer and Henderson-Sellers 7).

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 and Henderson-Sellers 7). 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.

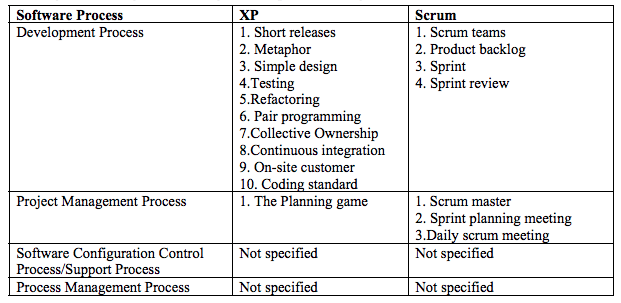
****

Table 3: Process of Extreme Programming and Scrum (Qumer and Henderson-Sellers 7).

**Recommendation**

Extreme Programming and Scrum are widely used in different technological companies. Both of them are suitable in small and medium-sized projects and use an object-oriented approach to develop software features iteratively and rapidly. I recommend Extreme Programming for several reasons.

Firstly, I like practices of Extreme Programming such as pair programming, simple design, and refactoring. Pair programming means two programmers write the code on the same computer, and they are shuffled. The person writing the code is called a driver while the other one is a navigator who thinks about high-level issues. Pair programming not only results in better code but also provides an opportunity for programmers to learn new skills as well as better coding styles. The simple design is a great practice to solve the problem. The simple design focuses on choosing the simplest way to solve the current problem and do not worry too much about the future since the future is unpredictable. For example, a programmer spends a long time to figure out a complicated solution to handle large input data but later found the input changed based on the new requirement. Refactoring out any duplicate code is also an excellent practice, which helps to reduce the code complexity and improve the code structure.

Secondly, 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 committed to developing the features after the sprint planning meeting.

Last but not least, extreme Programming teams develop the features based on the priority that is decided by the customers, which means they develop the most basic and important features first, and then deliver other features incrementally. But Scrum teams can determine the development sequence after the product owner prioritizes the features.

In conclusion, I recommend Extreme Programming because of its practices, flexibility to changes within the iterations, and its development priority.

**Works Cited**

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): n. pag. Web. 3 November 2019.

Awad, M. A. "A Comparison Between Agile and Traditional Software Development

Methodologies*." University of Western Australia,* 2005: 1-84. Web. 3 November 2019.

Beck, Kent, et al. *Manifesto for Agile Software Development.* 2001. Web. 3 November 2019.

Beck, Kent, and Erich Gamma. *Extreme Programming Explained: Embrace Change.*

Addison Wesley Professional, 2000. Books.google.com. Web. 3 November 2019.

Cohen, David, Mikael Lindvall, and Patricia Costa. " An Introduction to Agile Methods." *Advances in Computers* 62.03 (2004): 1-66. Web. 5 November 2019.

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. Web. 3 November 2019.

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*. 6-7 July. 2006. Web. 12 November 2019.

Schwaber, Ken, and Mike Beedle. *Agile Software Development with Scrum.* Upper

Saddle River: Prentice Hall, 2002. Print.

Schwaber, Ken. " Scrum Development Process. " *Business Object Design and*

*Implementation.* Springer, London, 1997: 117-134. Bookmetrix.com. Web. 1 December 2019.