School of Innovation, Design and Engineering
DVA422 – Software Engineering 3
17 September 2018

Naida Demirovic
ndc17001@student.mdh.se

Seminar 2: Software Processes and Methods

Abstract

This report aims to discuss some of Agile principles and reflect them to Scrum and XP practices. Also it will be shown how each of these principles were evident during the Exercise 1 on the course Software Engineering 3: Software Architecture and Processes. After the reader has that knowledge he/she is able to see the differences between those practices and imagine if they can be mixed together in a specific situation.

Table of Contents

1 Introduction

The Agile software movement began to receive considerable public attention approximately a decade ago, with the release of the "Agile Manifesto." Its roots extend at least a decade earlier than that, in practices such as Extreme Programming and Scrum. The Agile software movement is emblemized by the Agile Manifesto and a set of principles that assign
high value to close-knit teams and continuous and frequent delivery of working software.
The authors of the Manifesto go on to describe the twelve principles that underlie their reasoning. [1]

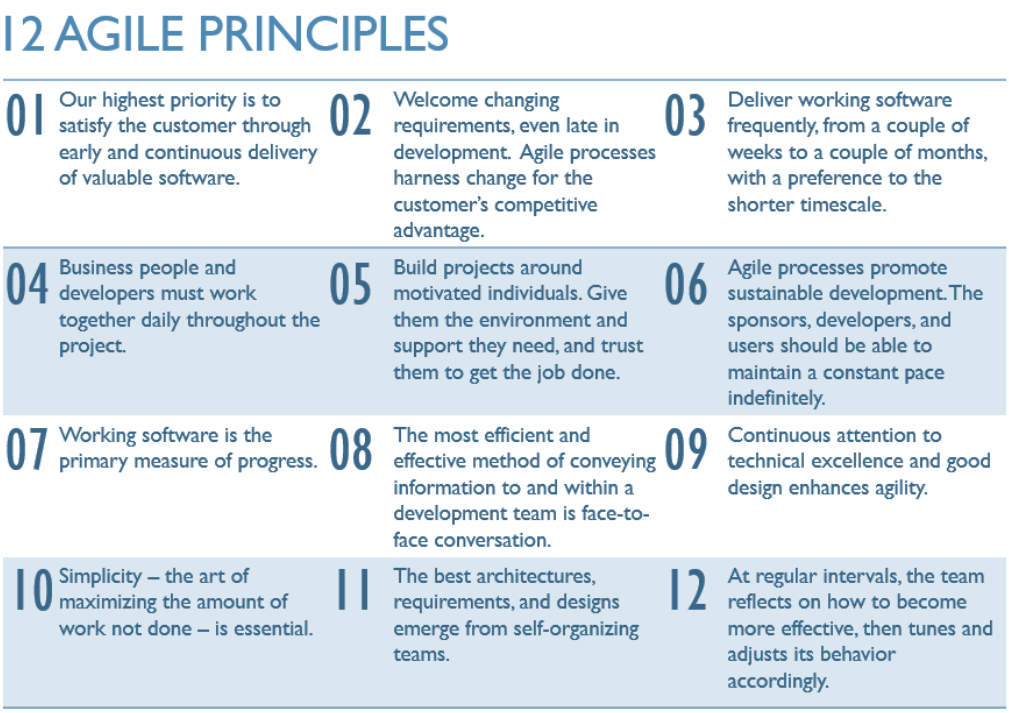

Figure 1. 12 Agile principles

2 Extreme Programming

In this section it will be discussed how each of the two selected principles are reflected in the practices of extreme programming as described in "Embracing Change with Extreme Programming".

2.1 How is 7th principle reflected in the practices of extreme programming as described in "Embracing Change with Extreme Programming"?

• Simple design

At every moment, the design runs all the tests, communicates everything the programmers want to communicate, contains no duplicate code, and has the fewest possible classes and methods. This rule can be summarized as, "Say everything once and only once.". [2] This principle is reflected to this practice in a way that all the tests run and everything is as simple as possible because the main thing is that the software is working.

•Tests

Programmers write unit tests minute by minute. These tests are collected and they must all run correctly. Customers write functional tests for the stories in an iteration. These tests should also all run, although practically speaking, sometimes a business decision must be made comparing the cost of shipping a known defect and the cost of delay. [2] This practice is reflected in a way that if the unit tests are written minute by minute and therefore there are no situations that it turns out late that something doesn't work. Everything is discovered on time.

•Refactoring

The design of the system is evolved through transformations of the existing design that keep all the tests running. [2] This reflects in a way that if all tests run, design is changed if needed.

•Collective ownership

Every programmer improves any code anywhere in the system at any time if they see the opportunity. [2]

•Just rules

By being part of an Extreme team, you sign up to follow the rules. But they're just the rules. The team can change the rules at any time as long as they agree on how they will assess the effects of the change. [2] This reflects in a way that every change is welcomed if it will make the software works properly.

2.2 How is 12th principle reflected in the practices of extreme programming as described in "Embracing Change with Extreme Programming"?

•Pair programming

All production code is written by two people at one screen/keyboard/mouse. We can say that the team is more effective in this way of programming because it ipas easier to work and think in a team. [2]

•Collective ownership

Every programmer improves any code anywhere in the system at any time if they see the opportunity. [2] This principle is reflected to this practice in a way that the programmer adjusts his/her behaviour according to the situation.

•Open workspace

The team works in a large room with small cubicles around the periphery. Pair programmers work on computers set up in the centre. [2] In that way the team has a good and opened communication during all day at work and they can adjust behaviour and figure out how to become more effective in every second.

•Just rules

The same as described before for 7th principle, by being part of an Extreme team, you sign up to follow the rules. But they're just the rules. The team can change the rules at any time as long as they agree on how they will assess the effects of the change. [2] In a way with this practice the team is allowed to always change the rules in line with the situation.

3 Scrum

This section describes how each of the two selected principles are reflected in the practices of extreme programming as described in "The Scrum Software Development Process for Small Teams".

3.1 How is 7th principle reflected in the practices of Scrum as described in "The Scrum Software Development Process for Small Teams"?

The role of Software Technology Group is to act of various fields covering introduction and research of new technologies and approaches. Their NBO projects were more successful than others and some comments from successful teams will be discussed in this subsection.

For example, they did the first piece of project and then reestimated—learn as you go. Reading from the present situation gave them guidelines for working software.

Another example is that the requirements document was high level and open to interpretation, but they could always meet with the systems engineer when they needed help. If the meeting with software engineer would help to continue with making the software work then it shouldn't be a problem to make that happen.

After initial planning, a series of short development phases, or sprints, the product was delivered incrementally. A sprint typically lasts one to four weeks. A closure phase usually completes product development.

The goal is to complete tasks by the sprint's delivery date. If the tasks are completed in appropriate way the result is working software.

As sprints finish, estimates become better as planners see what each team has produced in previous sprints. The organization can make one very important decision at the end of a sprint: whether to continue product development. [3]

3.2 How is 12th principle reflected in the practices of Scrum as described in "The Scrum Software Development Process for Small Teams"?

Software Technology Group held a short, daily meetings. Only those who had a need attended. If there are 20 daily meetings it doesn't mean that the team will be more effective. In this way NBO projects were more successful because they scheduled their meeting as they feel they need in that moment.

Large teams were divided into smaller subteams. When there are less people it is easier to make a deal about how to become more effective and adjust behaviour.

Before each sprint, the team updated the backlog and reprioritized the tasks—each team signed up for a number of tasks and then executed a sprint. Individual team members agreed to complete tasks they believe were feasible during each sprint.

Scrum meetings involved all team members, including those who telecommute. The meetings serve a team-building purpose and bring in even remote contributors, making them feel a part of the group and making their work visible to the rest of the team.

The development and marketing groups must work together to provide the features with the highest value for the product's first release.

The Scrum master ensures that everyone makes progress, records the decisions made at the meeting and tracks action items, and keeps the Scrum meetings short and focused.

At the meeting at the end of the sprint, anything can be changed. Work can be added, eliminated, or reprioritized. [3]

4 Exercise 1

4.1 How is 7th principle evident in the way we worked during Exercise 1?

During the Exercise 1 our highest priority was working software. In the end of the first phase of Exercise 1 me and my team had a problem with one task. It didn't work properly. Our logic and design were well done but it wasn't doing its function in appropriate way. We spent last hour on that task and worked under pressure, so that we have a software ready to be shown to the customer, but in the end we got what we wanted. To sum up, for my team the primary measure of progress was working software. We were working hard to gain all 12 principles but that one had the highest priority. We were thinking that we will not satisfy customer's needs with anything else if the software doesn't work appropriate. When you have working software to show the customer you can easily explain all other parts of the whole process. But if your software doesn't do what it needs to do then all the other parts don't make sense to the customer.

4.2 How is 12th principle evident in the way we worked during Exercise 1?

During the Exercise 1 we had stand up meetings where we were presenting what we have done previously, if we have some issues and thoughts about how to solve them or ask colleagues for help, and what we are planning to do in the future before the next stand up meeting. During those reports and discussions we were also changing the way how to become more effective and behave according to the present situation. It means that if we decided how to become more effective in one way on the previous meeting it doesn't mean that is the best way how to work. On the second meeting some new issue appeared and we were changing strategy and behaviour. Thanks to that principle we were always up-to-date with our software and issues that it had at specific time.

5 Counterexamples

Two examples of circumstances under which it would not (or at least might not) be a good idea to follow the 7th principle:

1. If the project is large it would not be a good idea to have a working software as the highest priority. In large projects it should be one of the highest priorities. But unit tests should not be done minute by minute because it takes a lot of time and there are many other tasks that need to be done.
2. If there is a project that has the design as customer's highest priority then it is not a good idea to put much effort on software's functionality.

Two examples of circumstances under which it would not (or at least might not) be a good idea to follow the 12th principle:

1. If company's departments are far away from each other then the team meetings should not be held many times per day. It should be as less as possible but well organized, so that there is more effective work during the day.
2. In some situations, the customer is not available and willing to participate in the discussions of how the project can be improved or what can be changed so that the team can be more effective. In those situations, the project leader should give the team guidelines for future work and this principle is not a good idea because it would be waste of time.

6 Conclusion

It is clear that Agile, Extreme Programming and Scrum methods play important role in software architecture. It is not easy to choose one of them. Mostly it is needed to use some parts of each one of them. But what can be learned is that those methods are useful and important for every project in a company so that all team members have a clear idea about their tasks. In that way, everything is well organized and controlled.

7 References

[1] Len Bass, Paul Clements, Rick Kazman, *Software Architecture in Practice*, Third Edition, ISBN 978-0-321-81573-6, Addison-Wesley, 2012.

[2] Kent Beck, "Embracing Change with Extreme Programming," In *Computer*, Volume 23, Issue 10, October 1999.

[3] Linda Rising, Norman S. Janoff, "The Scrum Software Development Process for Small Teams," In IEEE Software, Volume 17, Issue 4, July/August 2000.