TDDC93 Exercises:
Planning and Processes

Martin Söderén
marso329@student.liu.se
900929-1098

March 14, 2015

1 Planning and Processes

1.a

Small releases:
In XP you add small packages to the project continuously then build the project and send the pre-release to the costumer for feedback. So the costumer is given alot of releases which is given more features everytime. This means that the requriments and the definitions of the features can change during the project when the costumer realizes that the feature does not meet their requirements or that they do not need it at all. Since XP is agile it has no problem with change during the project. If the same project was developed with the waterfallmodel this might be detecked when the product is shipped to the costumer late in the process. Kent Beck is saying that the cost of change during a project rises exponentially over time and that by having constant feedback from the costumer you reduces the numbers of changes late in the project and that reduces the cost.[1]

Coding Standards:
Using XP requires you to have standars how you name variables, functions and classes. You also need standars for the coding style and formats. This makes its easy for the whole team to read and refactor the code. If the code looks different depending on who was the programmer it is easy to proclaim ownership on that piece of code but that not a part of the XP way since there is a collective ownership of the code but that another practise. This is a part of the implementation part of the waterfallmodel and is probably used in every project model there is.

Simple Design:
Always look for the implementation that is the easiest and meets the requirments. This speaks for itself, if you can make a system less complex but keep all the functionality it is a better solution. This will make it easy for everyone to understand the code, refactor it and maintain it. Unfortunate the complexity of the system is in the eye of the programmer. Something that seems like an easy solution might be rather complex and difficult to understand for someone else. The team decides together what is the best and easiest solution. This is not part of the waterfallmodel but it could be used in it. It is pretty straight forward that you want the least complex solution to a given problem.

Testing:
Everytime you release a small release to the costumer it needs to pass some predetermined tests to make shure of its functionality. These tests are normally written before the code that will be tested so the code is written with these tests in mind. This is a part of the iterative process and when faults are detected the programmers can immediately fix the problem so the release can be released as quickly as possible. This is unique for XP. The waterfallmodel has a testing phase but this is all done in the end of the project and the tests are defined in that phase and does not have any similarity with the testing in the XP model.

1.b

Since the code in the project will be owned by the team collective and these newly graduated programmers each have a different skillset they can learn from each other. They will also get feedback on their code compared to if the worked completely alone on one part of a system and no one but themself would ever look at the code and

they would continue to make the same mistakes.

The XP model has one practise that is called Metaphor. This means that parts of the system can be explained with metaphores that makes you think of something similary in your life which you can compare the part of the system with and make that part more understandable for other programmers. It might be that you refer the systemparts to parts of a car. The number crunching part is called the engine and the part that is storing data is called the trunk and so on. This will make it easy for the programmers to see the whole of the system.

The XP modell has a practise called pairprogramming which means that two programmers share one computer and switches back and forth. The will help improve productivity but also make shure that the programmers learn from each other and can help one another to figure out problems when they get stuck.

1.c

Scrum is used to manage project while XP is used to develope software. Scrum is used to create a process that will be used to create software or something else. It does not tell us what will be done during the process to create the actual software. XP on the other hand tells the programmer how to work during the project to create a quality software. So XP can be used to manage the project but all the different parts of the project/the process can come from the XP model. When it comes to how to work Scrum is more freely on that point. XP tells the programmers how they should work during the process, for example pairprogramming, code-style and so on while Scrum does not bother on how you make the code. As long as it is done within the timeframe.

1.d

Of course, you can use scrum to handle the project with its sprints, scrummaster and scrummeetings. During the sprints you can use XP make all the different parts of the sprint come to life for example incoperating pair-programming, have testing and feedback from costumer as a part of the sprint. So you can use scrum to estimate how long the development will take and use XP the help during the project to get things done as quickly as possible.However in XP change is a part of the process, scrum does not use change as a part of the routine but you can combine these. One major difference is that scrum can be used on almost any kind of project, it does not have to be a software development project while XP is a pure software development tool.

> "To say that Scrum and XP (eXtreme Programming) can be fruitfully combined is not really a controversial statement"

[2]

References

[1] Beck, Kent (2007) Scrum and XP from trenches Ch.5. Addison-Wesley

[2] Kniberg, Henrik (2000) Extreme Programming Explained: Embrace Change p.81 . C4Media inc