CS31310: Design within eXtreme Programing

Edgar Ivanov
edi@aber.ac.uk
Department of Computer Science, Aberystwyth University

December 16, 2013

Extreme programming is one of the software development methodologies, with its own set of rules and approaches. Some aspects of the eXtreme Programming are criticised due to the different strategy it takes to develop software. Some of the aspects include: continuous design, incremental definition of the system requirements, there is a mandatory pair programming requirement, customer representative may be part of the project etc. In this report I will discuss the design aspect of the XP.

Design on its own is a fairly broad topic. In general it is a phase when the team makes decisions about the data structures, classes, data flow, modules, programming language, platform etc. In comparison to the other methodologies XP tries to keep its design simple and encourages to do it while coding and adding new features to the system.

XP uses continuous design instead of the planned one. Why is continuous design chosen? In order to understand this, we will first need to take a look at the planned design, when developers consider all the issues they may face in advance and plan accordingly. They think about the best solutions and known patters that can be applied while building the system. When design is complete it is then down to the coding and the system should be ready to use. It is not as easy as it may sound, though. It is impossible to predict all the issues that project team will face in the future and it is even harder to predict the changes a customer may request during the development. The continuous design was chosen to overcome the issues and become more flexible about the system requirements in the future and to satisfy the customer at the same time since it is one of major principles that agile software development stands on[1]. However there are drawbacks in this approach as well, as Martin Flower points out in his article "Is Design Dead?": if the continuous design is not used properly it may become "code and fix" nightmare, when it gets hard to maintain it and add new features to the system.

My experience in designing is limited to what I've learnt during the first two years of my studies. When completing a group project in the second year our group used the planned design which we tried to stick to. We had to change it slightly, though, during the implementation phase since we found some parts of our design could not be implemented in the code. I haven't worked in the environment where continuous design would be used yet, so it is hard for me to judge which design approach is the best one to use. I do like the idea of the continuous design in XP, where everything is flexible and changes can be made at any time. I think it provides the developers with more freedom and space for creativity. Although it cannot be considered to be a universal solution, as Flowers points out in his article[2]. There may be systems where requirements are not set specifically at the start of the project or they change every month, that is when continuous design may be of good use[3]. On the other hand when it is known that there will be no major requirement change and all specifications are known prior the start of the project BDUF may be used successfully. There are many examples of success stories using XP as development methodology in the projects[4]. Examples of XP being used successfully provided me with the evidence that design approach that XP takes is sufficient for building software systems.

Simplicity and YAGNI were two of the points in the article "Is Design Dead?" that I really liked. There is a great example that will help me to illustrate how these techniques work. Consider having a class where the addition method needs to be implemented during the current iteration. We know that there will be a requirement for the subtraction method in the future, but it should not be implemented at this stage because that would purely be a waste of time, resources and make code to be more complex than it should be at this stage. XP insist that instead we should concentrate on completing the goals of the current iteration.

I had an experience of writing a simple PHP program when I spent the whole evening adding functionality. I thought this would be useful in the future. As the result of that evening I had a class with bunch of methods, but it was not doing anything useful since I didn't have the rest of the code yet which could use these methods. I have deleted or completely redesigned most of the methods later on because I found out that original ones were not doing what I needed them to do. This could have been avoided if I had concentrated on the current goals instead of building questionable features. The problem here is that it is sometimes tempting to add a few extra lines of code because you know that you will need them.

The key points of the design phase in XP project are use of CRC cards, the refactoring and pair programming [5]. I found the idea of CRC cards use very beneficial. It is very simple to use them in comparison with the usual modelling methods. Instead of drawing classes on the whiteboard or modelling them in some piece of the software we can write the class names on the small paper cards and then manipulate them in any way we want. In case something needs to be changed in the model we don't need to spend time wiping off the old image from the board and drawing a new one; all we need to do is to move the card to the new place. As well as being simple to manipulate CRC cards also allow "more fully appreciate object technology"[6].

Martin Flowers' article "Is Design Dead?" gave me a better understanding of the design techniques used in XP and deepened my knowledge of the differences between the different design approaches.

Bibliography

[1] Agile manifesto. Principles behind the Agile Manifesto. [Online] Available at:`http://www.agilemanifesto.org/principles.html`, [Accessed 25 October 2013].

[2] Martin Fowler, 2004. Is Design Dead?. [Online] Available at:`http://martinfowler.com/articles/designDead.html`, [Accessed 29 October 2013].

[3] Extrime Programing, 1999. Don Wells, When should Extreme Programming be Used? [Online] Available at:`http://www.extremeprogramming.org/when.html`, [Accessed 25 October 2013].

[4] Cunningham & Cunningham Inc, 2006. Successful XP Projects. [Online] Available at:`http://c2.com/cgi/wiki?SuccessfulXpProjects`, [Accessed 29 October 2013].

[5] Extrime Programing, 1999. Don Wells, Design rules within XP. [Online] Available at:`http://www.extremeprogramming.org/example/crcsim.html`, [Accessed 25 October 2013].

[6] Extrime Programing, 1999. Don Wells, CRC Cards. [Online] Available at:`http://www.extremeprogramming.org/rules/crccards.html`, [Accessed 25 October 2013].