CS31310 Worksheet 3

Daniel Atkinson (daa9)

12 December 2012

1 Design in XP

The design stage in XP is very diferent from most methods of project design due to it being very short and sweet.
Define the intended end result, break it down into pieces, design a piece, then implement it. Repeat this for all pieces to build up the project. If something doesnt work, go back to it and fix imedietly. This may not be possible due to the fact that the programmers may not be involved in the design process at all. Being passed a design and instructed to build it. If in issue with the design then comes up and the designer is no longer around or is on another project then a workaround is likely to be written. This type of practice can lead to coide that is very hard ot maintain or adapt for either a change in design or additional functionality.
In my opinion it is always a difficult task to incororate new design aspects into a project when the goal posts are constantly being moved, no matter how much planning has gone into making the project flexible. Trying to code these changed to a deadline can lead to rushed code and working around the design to get it done, when coming back to either fix or expand this code it can be messy.

Using XP practices rather than a fully planned upfront approach enable you to develop more simple designs. If you have simple designs it reduces the complexity and thus the cost of change. When something is more complex, it is harder to understand and can cause mistakes to be made. Also only write what is needed to this itteration even if you can implement something to be using in a later itteration at little or no current cost. If you do not have the full requirements then you may make a mistake in implementing somthing early, and even if there is no cost currently it could cos to correct it later.

Having as many member of the team involved in the desgn process could also be a good idea. The more people contributing to a solution can increase

the number of good ideas added to the pot. This may accelerate the process to finding a workable solution.
Personaly I find this interesting as having been in a team trying to come up with a solution to a problem. When more people get involved there seemed to be a few good ideas and a large amount of bad or confusing ideas. This can be good as the good ideas are made public to the rest of the team, but when a higher ranking membe rof the team such as a manager gets involed, many of the good ideas are thrown away in favour of their own ideas which in most cases that I have observed, did not fit the problem.
Idealy more people tackling a problem should produce more results but that depends on who the participants are. Another observation made during these situations was that most of the good ideas came from the higher level programmers, but the majority of these people were introverted and didn't defend their ideas against the more vocal and agressive members of the team.

Pair programming seems to have two main opinions aimed towards it. The first being from a manergerial point of view that two people working on a single problem is a waste of resources, for the same amount of time spent they could both be working on individual problems.
The other view being that of many hands make light work. The two programmers can aid each other with their own knowledge base and spot mistakes or suggest improvements quickly.
My personal view on this is that pair programming can be a very good thing, and in fact get jobs done much faster than if the two programmers were working seperatly. Of course it can backfire by the two people jsut chatting or getting side tracked, but a focused group can work very effectively in pairs. My experience in this area is very possitive, tasks got completed much faster and to a higher quality than when performed individualy.

My own view on XP approach to design is that I agree it should be smaller evolutionary designs with constant refactoring to keep everything simple and tidy as to avoid later complications and an attempt to plan forwards a bit to try and ensure that each itteration of the design does not spiral out of controll. I believe pair programming can also help in this as one person cannot verge off track without the other noticing this and attempting to correct this diversion.