4.6 Is it possible to combine process models? If so, provide an example

Yes, taking our current project into consideration we are all using a waterfall-based model for the deliverables. However; inside our particular group in order to achieve these goals we on a smaller level are using an agile-based model consisting of sprints and activity modeling.

4.8 What are the advantages and disadvantages of developing software in which quality is “good enough”? That is, what happens when we emphasize development speed over product quality?

When we focus on development speed of the product quality and create software that is just “good enough” we do a great disservice to the clients and other programmers. For example if we created in a short period of time a piece of software that does banking transactions and we just got it to work before we rolled it out, then we haven’t given it enough time to work out bugs and flaws in our product. Then in a short amount of time the client will notice these deficiencies, and because of our poor quality of work, will find someone else to correctly finish what we have started. Then it falls to the other programmers to be able to decipher what we have done, make the necessary changes, and complete the process. Essentially doing half the work we didn’t finish.

5.1 Reread the “Manifesto for Agile Software Development” at the beginning of this chapter. Can you think of a situation in which one or more of the four “values” could get a software team into trouble?

I’d like to take a look at Responding to change over following a plan. This sounds like a team is more interested in being reactive instead of proactive. This kind of philosophy is always a recipe for disaster, the old way of thinking that says “just wing it” must have been over their door when the developers wrote this in their manifesto. Ideally a plan is put out after much consideration, thought, and risk assessments. To say that a software team is just putting something out there and then working on it as it goes is piss poor planning.

5.7 Why do requirements change so much? After all, don’t people know what they want?

Yes and no, clients have a general sense of what they expect, but as for the actual implementation they generally have no clue. Requirements change after processes have been tested and shown, but then not given the desired outcome. The phrase a million ways to skin a cat come to mind quite often. As software developers we see certain things as give and take, give speed for performance etc., but the end client doesn’t know what they don’t know. And technology is constantly changing. So what is relevant today could very well be outdated tomorrow.

6.1. Based on your personal observation of people who are excellent software developers, name three personality traits that appear to be common among them.

Introversion, perseverance, helpful