Learning Journal Week 6

(For COMP1003J Introduction to Software Engineering)
(Lecture: Mooney Catherine)
(BDIC1003J)

Software Engineering
Second Semester, Academic Year 2017-2018
Beijing-Dublin International College

Authors

Group 9: *Cirno9*

Name	UCD Number
Xinzhu Wang	17205965
Zihao Wang	17206179
Xienan Wang	17206001
Yaochen Wang	17206183
Ziyin Wang	17206182
ZhouHeng Wang	17205972

April 11, 2018

1 What did you learn today?

What we learnt today is divided into three parts. The first part is the definition of software processes and software process models. The concepts of software process is very clear that it is the way for programmer to make a software product which is required by customers. As for software process model, it is a rough work process for software process. However, we cannot finish all programs with just one model, so there comes the part two: five different types of software process models, which are "Build and fix", "Waterfall", "Incremental", Spiral" as well as "Rapid prototype" model. we will discuss these in the question two so we do not go in details here. Finally, we learnt something about Distributed Version Control and Git. By using version control system, we can make our teamworks more efficient and convenient. We also learnt the history of version control system and we knew something about Git, which is the latest version control system. After the class, my teammates and I decided to use GitHub to work together easier and better. We not only knew many new knowledge but also aware that we can improve our group work and do much better.

2 Discuss a number of software process models and when they might be used, advantages, disadvantages, etc...

As we have learnt, in general, there are five types of software process models, which are "Build and fix" , "Waterfall' , "Incremental" , "Spiral" and "Rapid prototype" . Every of them has its own merits and demerits.

First come to build and fix model. Briefly, when a product is being programming without enough instructions and design, programmers always need to fix it several times to meet the needs of the clients, which means it fits for the small products. Its advantage is that it has limited complex and often cost a little, however, it has a number of disadvantages. For example, as the product needs so many times fixes, it' s unpredictable when it can be finished and delivered, and even though it has been completed, the quality is low sometimes. Moreover, as no document and enough specification will be produced, maintenance can be extremely difficult.

Secondly, the waterfall model, fits for the product whose requirements are clear and easy to understand. It is a great model that every stage will be verified before proceeding, and faults detected can cause rolls-back to certain stage, maintenances rolls-back process appropriately. While its merits are obvious, its demerits are also clear. If discovering faults late, they may cost a lot to fix them and there is no requirement to illustrate the client anything but documents until the end.

Next is the incremental model. It divides the whole project into several blocks, and when we finish one of them, we can show it to clients to ask for their advice, so it can reduce the cost of the changes raised from clients and it' s easy to get feedback from them anytime, which can help to produce a product that is truly useful. It has some disadvantages, for instance, the process is invisible when proceeding, and the structure of system may degrade when adding new

increments.

Then the Spiral model. It is like the mixture of the Waterfall model and Incremental model, so it includes all of their merits. While it still needs long time to complete the task, it is easy to be controlled so it fits for the product that may meet risks during programming.

Last comes to the rapid prototype model. It need the programmers to establish a brief model at first to understand the details of the requirements and it can help them to reach a consensus with clients on the requirements at the beginning, however, it means it needs clients to involve a lot and sometimes the initial prototypes cannot be used.

The words above describe the five types of lifecycle models briefly, and we come to conclusion that if we fully understood the use of them, we could finish the tasks whatever we meet perfectly.

3 Have you used any Version Control software before? If so, what was your experience? If not, would you like to try Git/GitHub?

I used to enjoy in Version Control software and still using it for daily coding and team working. Before I took the plan with Version Control software, the main problem in my programing experience is the redundant codes in program. As completing, maintenance or developing a software, I always save the codes of current edition down in different periods of working. For what more that I have multiple computers with different functions, need to often switch to the device for developing, I have to keep every machine has a copy of each editions of the software. The Secondary problem is waste of hard disk space, and keeping all files in synchronization is only inconvenient. The fatal problem is I will forget the particular differences among those editions ever why and how I partition versions as time goes by. It will make mess on working of integration and store up hidden danger for continue designing and programing. What worse is that at the time, the main thing I did was game programming. Although it was a small project, there were still 4 people involved, and the engineering files had more complex operating systems, tedious file types and corresponding relative storage paths. The cost of repeated copies, mutual synchronization and version control is further improved.