ASSIGNMENT 1 : REFLECTIONS

EMAIL: GREGOIRE.COUSIN@UCDCONNECT.IE
GREGOIRE COUSIN STUDENT NUMBER: 18204188

Throughout our time in class, we dove deep into multiple approaches and concepts of building software with proper structure. This was to understand the fundamentals of Agile, Scrum, Traditional software methodologies like code and fix and the Waterfall Process to name a few. We looked at the best methods to properly detail the specifications and implementation of a project.

I found the Waterfall Process Interesting since it was a widely used method in history. It meant going through the entire application : from analysis, design, coding, testing, and maintenance. This was done when working in teams and the process could take multiple weeks and more.

When working in a team, it is essential to check for errors and manage them throughout your project. This is why we have to work backward from the Waterfall Process. As programmers, we found better technics including Iterative and Incremental methodologies. These methodologies consisted of working in iterations throughout your project or finishing the whole application in one go before maintenance, testing or deployment. Each process has their individual pro and con depending on the type of project at hand.

If you needed to build features and have direct access to your customers rather quickly, the Incremental method is a better approach. However, if you didn't have all of the details in front of you and needed to code the program/idea over time, the Iterative method will suit the project better.

In class, we also learned a lot about Agile and how to incorporate it within a teams project. We then compared it with the waterfall process. We found that the Agile method had better outcomes in interactions between the team themselves and their customers, building software that worked well, and a more flexible way to deal with potential unexpected changes and errors.

In Agile, we learned about extreme programming, user-stories, different meeting methods such as the planning game and stand-ups. The Agile process also helps the ability to structure a good testing environment. Software testing is vital to the process of building a useful application, and many testing technics are made to work hand in hand with Agile methods. We learned different testing methods such as Unit-Cases, Refactoring, Code Coverage, Mock Objects and what these terminologies represent.

We completed an assignment based on a video that showed the interaction of senior developers and the person responsible for the project. (This would typically be the project manager.) The project had an issue and new errors needed to be fixed. However, the testing environment was wiped from the project since the team believed they had finished all test cases successfully.

Everyone in the team realized this was not the case. The exercise was excellent to go through since it consisted of learning what could potentially go wrong when not following proper protocol. Losing test files while working on a project might be viewed as very wrong and a bad practice to fallow before being entirely sure that the software is out of your hands. Even then, it is good to keep track of your work with a history management tool such as git for any potential further work, future releases or important deployment. The video also showed us to be extremely meticulous and detail oriented with the way we approach building software, taking account all of Agile's processes, project planning, the importance of each phase and keep in mind what their values are.

I believe that structure and communication between your client and your team to be essential in the process of building good software. This means that a proper approach tailored to your program and team is necessary. In our modern ways of working in development, Agile stand out to be the best and most advantageous of team-based software building processes.