17. November 2019

Continuous Integration

CS-C3150 Software Engineering

All the software systems are quite volatile given the ever-changing and fluctuating nature of the competitive market and customer needs. New bugs surface every day which has to be fixed. New requirements emerge and these need to be implemented in later versions. As new versions of hardware and interactive components are released, it becomes necessary to make your system compatible with them. Also, you have to keep yourself ahead of the competitors which require coming up and implementing new features to your software every day. As software changes, it creates a new version of itself. Most systems can, therefore, be imagined as a set of different versions of different software. Thus it is critical to keep a track of the version which constitutes a system and hence it is important to have efficient configuration management(CM). Having an ineffective CM can often lead you to lose track and end up modifying and delivering the wrong version to the customer. There are four activities involved in a CM which are *system building, version management, release management,* and *change management*. (Sommerville, 2016)

Software building is a process of integrating different components, external libraries, data files, etc to create a fully functional system. It involves a large amount of information, codes, etc because of which it becomes a complex process that is prone to errors. It thus becomes important to build the software frequently to discover the bugs. One way of implementing is through continuous integration. (Sommerville, 2016)

In an agile method, where the code is changed almost every day and is changed by many developers simultaneously, it becomes important to discover the problems caused by different developers and to repair them as soon as possible. This problem in an agile method is solved by using the concept of "Continuous Integration". Continuous integration can be thought of as a process in which the building and testing of a system are carried out frequently to discover problems. (Sommerville, 2016) It involves rebuilding the mainline as soon as some changes are made in the source code. According to Martin Fowler continuous integration is a practice where team members integrate their work frequently. Every member integrates at least daily which leads to multiple integrations in a single day. Each integration is checked by an automated build that detects errors as quickly as possible. Continuous integration is supported by various continuous integration tools which are servers running

automated testing suites. Continuous integration tools thus act as configuration management by giving an insight into the steps required to set up and configure a particular software system.(Stolberg, 2009) This reduces the integration period and hence speeds up the delivery. Steps in continuous integration involve getting the latest source code from the version control system into the developer's workspace and compiling and retesting it to check the system passes all the tests. If it fails, then whoever checked-in the last baseline system is informed to repair the bugs. After the build passes all the tests, changes are made to the system and the new system is again built and tested in the developer's private workspace. If the build fails, the developer continues editing the system or else the system is checked into the build server system. The system is then built and tested on the build sever system and if it passes all the tests then it is published into the mainline. (Sommerville, 2016)

Thus continuous integration enables testing the system in parallel with its development. Bugs are discovered early and they are easy to track down due to shorter integration periods. This saves both cost and money over the whole development cycle. It also avoids last-minute chaos near to the release dates when everyone tries to check in their versions which may be slightly incompatible. It also has a benefit that there is constant availability of a build that is ready to be shipped. Since code is frequently checked-in and used by various developers, it pushes them to create more modular and less complex code. This constant refactoring reduces the effort and increases the efficiency of the developers. Not only this, but continuous integration reduces the cost of testing. By incorporating quality testing within the development cycle, value is immediately added to be the product. (Stolberg, 2009) So as it turns out from the above arguments that setting up a continuous integration system to implement testing in an agile environment can be a big boon for the project.

Reference

SOMMERVILLE, I., 2016, *Software Engineering.* 10th ed. Boston: Pearson

STOLBERG, S., 2009, 'Enabling Agile Testing through Continuous Integration'. *Agile Conference 2009*, pp.369-374. [viewed 17th November 2019] Available from: https://mycourses.aalto.fi/pluginfile.php/1088074/mod_resource/content/6/Stolberg2009.pdf