Continuous Integration

CS-C3150 Software Engineering

A modern software development requires a large team of developers working simultaneously on different parts of a software. All the parts must be compatible and integrated together to build a fully functional system. In continuous integration, team members integrate their work frequently, usually daily or even multiple times a day (Sommerville, 2016). Before merging the work done by different developers, several automated tests are run to check any possible bugs like conflicting changes by different developers, compilation errors, failed tests etc. First, the latest code is fetched and then compiled. If it passes, then unit test cases are run to check if there are any trivial bugs in the code. If that passes, then other automated test cases are run. If all the steps are passed, then a new build is published to a common repository along with a log of changes made. (Stolberg, 2009)

This kind of development process helps in detecting trivial bugs early and makes it easy to keep track of the changes. It also helps in the speedy delivery of the product and makes the code easily maintainable.

The problem in agile development and the solution:

In agile development, where code is being changed continuously by many developers, it becomes difficult to discover the problems caused by different developers. Moreover, given a base code, we can't simply tell where the possible bugs are, without integrating the whole code and testing it. This is precisely what continuous integration does. It integrates the code several times a day and checks if everything is going smoothly or if the build has been broken by faulty code. In case of a fault detection, the code can be changed immediately when the code is still fresh in the developer's mind. Thus the code doesn't look very complex and foreign after a few days, and developers don't lose track of the changes.

Software Configuration Management and Continuous Integration:

Software configuration management is a part of the software development process which is concerned with policies, processes, and tools for managing changing software systems (Sommerville, 2016). It includes *change management, version management, system building,* and *release management*. Change management deals with keeping track of requested changes in the system by the users and deciding whether to make the change or not, depending upon cost and impact of change. Version management keeps track of multiple versions of the build while ensuring conflict-free code. System building is integrating everything to build an executable. Finally, release management deals with preparing software for external release for the users and keeping track of versions that have been released. (Sommerville, 2016)

An ineffective configuration management can make the code very complex and hard to understand after a number of significant changes have been made. Continuous integration can be termed as a kind of system building activity which is the process of assembling program

components, data, libraries, etc frequently and then compiling them to create an executable system, along with keeping track of changes and versions.
Since integration happens so often in continuous integration, the build (integration) time must be very short and tools like git are required to keep track of the changes. Some automated tests are needed to be designed to verify the validity of the new build (Stolberg, 2009).

References

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 23 March 2020] Available from: https://mycourses.aalto.fi/pluginfile.php/1177243/mod_resource/content/6/Stolberg2009.pdf