Software evolution and configuration management

2018-03-15

Configuration management involves processes, tools, and policies to manage to change software. Four activities of configuration management are *change management*, *version management*, *system building*, and *release management*. Change management ensures that the system evolution is managed process and the changes that are made to the software are the most effective ones. Version management keeps track of different versions of configuration items in the system and ensures that changes made by other developers do not interfere with each other. Since configuration management tracks change made into the system it releases the developer from having to remember them. Configuration management also makes distributed development possible because it ensures that the teams do not interfere with each other's work (Sommerville 2010, chap. 25).

Continuous Integration

Continuous integration (CI) is software engineering practice in which the development work is integrated into mainline frequently and each integration is automatically tested. Integrating frequently reduces the risk of merge conflicts. Not integrating frequently could lead to "merge hell", in which the time to integrate the work exceeds the time it took to make the original changes (Sommerville 2010, pp. 697–699).

Continuous Integration in Agile Development

Continuous integration can be part of agile testing and should be carried out using an automated framework. Because the development in agile is done incrementally, whenever the developer merges their local changes back to the mainline the software has to be built and tested. Since small increments are preferred, testing has to be done often and should be automated using continuous integration. Continuous integration can help to discover problems caused by interactions between developers. Apart from test automation, continuous integration can also automate the building and deploy the application, the release an executable of the system and generation of documentation. Continuous integration is not always possible or easy to implement. A large software may be impractical to test often, due to long build times. Large projects can employ daily builds instead of building everytime a change is made (Sommerville 2010, pp. 697–699).

Stolberg (2009) describes their implementation of continuous integration as:

1) Change in the source code is automatically detected which triggers
2) the latest code to be fetched and compiled if the compilation succeeds then
3) the unit tests an ran if unit tests pass then
4) the automated acceptance tests are run, if automated acceptance pass then
5) the build is published by the team and the team is notified.
6) Finally a build report is created by the process.

Relationship between Configuration Management and Continuous Integration

Let's think the relationship between configuration management and continuous integration through an example of releasing a programming package.

1) Development happens in persons local workspace using version management system such as *Git* to tracks changes to the codebase.
2) New release version is tagged.
3) The changes are pushed to *upstream* to a central repository hosted at web-based services such as *GitHub* or *GitLab*.
4) The push then triggers continuous integration service such as *Travis* or *Appveyor* to perform automated tasks such as running the tests or generating documentation. The test may be run against multiple versions of the programming language in question, against different machines and operating systems to verify that they work on these as well. Since new version was tagged it triggers an automated process to release a new version of the package into the programming languages official package repository and builds the documentation of the new version.

Bibliography

Sommerville, I., 2010. *Software Engineering: 9th Edition*. 9th ed. Addison-Wesley Publishing Company.

Stolberg, S., 2009. Enabling agile testing through continuous integration. *Proceedings - 2009 Agile Conference, AGILE 2009*, 369–374.