Project: Continuous Integration Environments, 5 ECTS.

Deadline: 2019-03-22 **Grade:** Pass/fail

Background:

The movement towards agile and more continuous ways-of-working have promoted speed and frequent delivery to the point that manual testing is no longer a feasible option for most types of testing. This is mostly justified by the benefits of more frequent feedback that allows developers to quicker find defects and mitigate defect synergy effects that add cost and effort for quality assurance.

To support continuous testing and feedback, automated tests are crucial. Preferably on all levels of system abstraction: unit-, integration-, system- and acceptance-tests. In addition, these tests should be run often, executed automatically without human interaction or input in a so-called continuous integration environment.

First step:

Send an e-mail to Emil (Emil.Alegroth@BTH.se) with the people that will be in your group (Names plus social security numbers). **Note that only 1 person in the group should send an e-mail!!!** This person will be designated the "leader" of the group, meaning that (s)he is the contact person for your group but no other responsibilities are required of this person.

Project description:

In teams of four (4) students, set up a continuous integration (CI) environment that can build the software project and execute tests automatically. The environment shall respond to developer code pushes and from push, build the entire system and run, at least, unit and integration test cases but preferably system tests. The environment must include:

1. *An integrated development environment (e.g. Eclipse or intellij IDEA)*
2. *A version control software (e.g. SVN, SourceTree or EGit plugin for Eclipse)*
3. *A version control server (e.g. Bitbucket or Github)*
4. *A continuous integration (CI) tool (e.g. Jenkins)*
5. *An automated build tool that connects to the CI tool (e.g. Maven or Ant)*
6. *An automated test framework for AT LEAST unit and integration tests (e.g. XUnit)*
7. *(**optional**) A system test tool (e.g. EyeAutomate or Selenium)*

You are free to choose any constellation of tools that you like but it shall be able to run automatically. You can either create a simple project yourselves to demonstrate that the environment works or use the pre-prepared Maven Java application. The application you use can be simple but you must demonstrate that EVERYONE in your group knows how the system works and that EVERYONE contributed to the set-up of the environment. Three tasks shall be performed with the application as explained in the following sub sections:

Task 1: CI-environment setup.

Setup the CI environment with components from the above list that can automatically build the project and run its unit and integration tests. Once the system runs, make a screen recording (Open source and freeware tools can be found online) of the environment when it is executing. The recording should start from you changing some code and pushing it until you get a log-file (or other demonstrator) that shows that the build was successful and that the test cases you've added passed.

You must also supply a model of your CI system, its components, and how they interact (e.g. how they are triggered and in what order). **OBS: You don't need to setup a remote build server, such as Jenkins, if you lack the resources to do so (A Local server is acceptable).** If you use a local server, make sure to record somehow that each group member uses the environment.

Task 2: Test-driven development

In the prepared Maven Java project, a set of test cases have been added that do not have any associated functional code. Run the tests and record that they all fail because the code is missing **(OBS! If you build your own application (in Java or other language) make sure to add the test cases first, record them failing when executing the environment and then add the code and show the tests passing).** Implement the code for the tests and run the tests, showing that the code builds successfully and all tests pass. The code needs to be implemented by all team members that should individually commit/push some code. Hence, after the project there should be AT LEAST one push from each team member. A screenshot from the repository you use (e.g. bitBucket, GitHub or Svn) shall be supplied in the submission to verifying this!

Task 3: Continuous development and testing

Extend the java project with new code and tests. The new tests should include at least unit and integration styled tests that follow best test practices but preferably (optional) also some type higher-level system tests performed with Selenium or EyeAutomate. **Note! ALL tests, regardless of type, must execute AUTOMATICALLY from the CI environment!** The recordings should clearly show this happening and give the test results.

Deliverables:

In addition to above stated screen-recordings, screenshots and models of the environment, each group must supply a report that describes the CI environment, motivate the technical choices of tools, report the challenges that you came across during development of the environment and report solutions to said challenges (e.g. if a tool didn't work or if there were problems integrating some tools together). The report can be maximum 5 pages. Each group must also provide a link to the code repository where the code is stored. **OBS!** Make sure to include some time logs of how much time each member spent in the project and be honest with time spent. Each member of the group should also write a short paragraph of their own experiences with setting up and using the environment, e.g. was it difficult, will it be useful in the future, etc.

Additionally:

You need to consider the quality of your solution such that we can EASILY extend the test suite with new tests. Hence, no hard-coded values should be used, the test suite should have a modular architecture, the application under test should be easy to extende, etc. **Obs! Make sure that all team-members (Name and Social security number) are mentioned when you submit your report, recordings, screenshots, etc., on CANVAS. The submission shall be in a .zip or .rar file.**

Support materials:

- The environment and how to set it up is discussed in Online lecture 2. There are also recordings from the course 2018 that provide more technical detail.
- The Maven Java Project is uploaded with the Project assignment on Canvas
- Additional materials can be found by googling the tools (Obs! Setting up the environment will require you to create accounts on the repository website at least)
- If you get into issues, report them on the CANVAS discussions or e-mail Emil (Emil.Alegroth@BTH.se) or Vi (huynh.khanh.vi.tran@BTH.se).

Good luck!