AGH

AGH UNIVERSITY OF SCIENCE AND TECHNOLOGY

Subject of the Master's thesis:

Continuous integration tool that supports the process by an optimal test suite selection.

Author:	**Andrzej Szewczyk**
Field of Study:	Mechatronics engineering
Specializations:	Mechatronics design

Supervisor:	**dr inż. Lucjan Miękina**
Reviewer:	**dr hab. inż. Mariusz Giergiel, prof. AGH**

Motivation to take up the subject of CI

I have been working on several embedded software projects. Let me briefly describe the typical difficulties these projects were suffering from:

Project A

- Safety system for turbines, compressors and engines,
- IEC61508 **SIL-3** certified,
- 1100 test cases (95% automated),
- **Execution of all tests cases took around 3 weeks on two independent test stations,**
- **A lot of unclear ties between software modules**,
- **Bugs were often found after 2 weeks or later,**
- **If any bug found, there was a need to run all tests cases once again for the updated firmware.**

Project B

- Electronic Engine Control for a turboprop aircraft engine,
- Challenging certification process,
- 65 software engineers involved in the project on 3 continents,
- Project activities carried out 24/5,
- **Numerous software updates every single day,**
- **Build process and configuration management not well defined,**
- **Need to build 3 separate projects prior to loading the firmware – major integration issues.**
- **If a test fails, can it be stated with absolute certainty that a bug is found?**

Continuous Integration

Continuous Integration (CI) is a development practice that requires developers to integrate code into a shared repository several times a day. The starting point when implementing continuous integration is an assumption that a single command should have the capability of building the system.

There are two main objectives of CI: build and test software automatically and provide developers with immediate feedback about quality of the last code build. In order to achieve those objectives, continuous integration relies on the following principles [1]:

- Maintain a code repository,
- Automate the build,
- Make the build self-testing,
- Every commit should be built on an integration machine,
- Keep the build fast,
- Test in a clone of the production environment,
- Make it easy for anyone to get the latest executable version,
- Everyone can see the results of the latest build,
- Automate deployment.

Proposed CI tool

Fig. 1. Jenkins web interface for the build pipeline view.

1) **CI_tool_build_trigger_job** – listens for push events to the github repository.
2) **CI_tool_synchronize_local_repositories** – updates local repositories to include all commits that have been pushed to the remote repository.
3) **CI_tool_run_python_script_to_select_optimal_test_suite** – executes the python script that selects an optimal test suite based on the changes made to the code in the last commit.
4) **CI_tool_run_optimal_test_suite** – executes an optimal test suite.
5) **CI_tool_run_integration_sanity_test** – executes the sanity check.

If the **CI_tool_build_pipeline** finishes successfully, the last commit will be merged to the production branch on the github remote repository.

How does the CI tool work?

Fig. 2. Github side-by-side diff for the commit that introduces bugs to the code.

How does the CI tool work?

```
Jenkins › CI_tool_run_python_script_to_select_optimal_test_suite › #29

10) Parse file blocks (method body) for methods that have been added to separate names of the methods,
    as an output the normalized path to the file that contains the methods is also returned:


11) Concatenate the results from points 8., 9. and 10. (i.e. build single list of all methods that have been affected
    with normalized path to the file that contains the methods):

        [['src', 'main', 'java', 'tcpServer', 'ComputeEngine_Runnable.java'], 'private double _1h_Watchdog_close_to_expire()']


12) Create list of the methods affected by changes in newer commit against older commit:

        affected method: _1h_Watchdog_close_to_expire()


13) Create lists of the methods that call the methods affected by changes in newer commit against older commit:

       package: tcpServer,     method name: run()


14) Based on the lists of methods from points 12. and 13., create a list of unit tests that are
    an optimal test suite for changes in newer commit against older commit (the list of tests is returned in format readable by pom.xml):

        unit test to be executed: <include>**/tcpServer/_1h_Watchdog_close_to_expireTest.java</include>
        unit test to be executed: <include>**/tcpServer/RunTest.java</include>
        unit test to be executed: <include>**/tcpServer/Run_ClientMessage_ACKTest.java</include>
        unit test to be executed: <include>**/tcpServer/Run_ClientMessage_BootUpTest.java</include>
        unit test to be executed: <include>**/tcpServer/Run_ClientMessage_MeasurementDataTest.java</include>
        unit test to be executed: <include>**/tcpServer/Run_ClientMessage_MeasurementHistoryTest.java</include>
        unit test to be executed: <include>**/tcpServer/Run_ClientMessage_SensorInfoTest.java</include>


15) Update pom.xml file with names of unit test to be executed:

        updated pom.xml with optimal test suite is saved in: E:\Praca magisterska\CI_tool_source_code\JavaWorkspace_MT\UnitUnderTest\updated_pom.xml

E:\Praca magisterska\CI_python_engine\PythonScripts>exit 0
Triggering a new build of CI_tool_run_optimal_test_suite
Finished: SUCCESS
```

Fig. 3. Console output of the Jenkins job that selects an optimal test suite for the changes made to the code shown in figure 2.
(#29 CI_tool_run_python_script_to_select_optimal_test_suite)

AGH

How does the CI tool work?

Jenkins ▸ CI_tool_run_optimal_test_suite ▸ #36

```
        at tcpServer._1h_Watchdog_close_to_expireTest.test_run_4(_1h_Watchdog_close_to_expireTest.java:291)

test_run_5(tcpServer._1h_Watchdog_close_to_expireTest)  Time elapsed: 0.127 sec  <<< FAILURE!
java.lang.AssertionError: expected:<0.6> but was:<0.48>
        at tcpServer._1h_Watchdog_close_to_expireTest.test_run_5(_1h_Watchdog_close_to_expireTest.java:335)


Results :

Failed tests:
  Run_ClientMessage_ACKTest.test_run_2:278
  Run_ClientMessage_ACKTest.test_run_3:347 expected:<0.6> but was:<0.48>
  Run_ClientMessage_MeasurementDataTest.test_run_1:233
  Run_ClientMessage_MeasurementDataTest.test_run_3:390
  _1h_Watchdog_close_to_expireTest.test_run_1:159 expected:<0.5> but was:<0.32000000000000006>
  _1h_Watchdog_close_to_expireTest.test_run_2:203 expected:<0.6> but was:<0.48>
  _1h_Watchdog_close_to_expireTest.test_run_4:291 expected:<0.5> but was:<0.32000000000000006>
  _1h_Watchdog_close_to_expireTest.test_run_5:335 expected:<0.6> but was:<0.48>

Tests run: 34, Failures: 8, Errors: 0, Skipped: 0

[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 26.505 s
[INFO] Finished at: 2018-09-09T22:22:46+02:00
[INFO] ------------------------------------------------------------------------
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-surefire-plugin:2.19.1:test (default-test) on project UnitUnderTest: There are test failures.
[ERROR]
[ERROR] Please refer to E:\Praca magisterska\CI_tool_source_code\JavaWorkspace_MT\UnitUnderTest\target\surefire-reports for the individual test results.
[ERROR] -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MojoFailureException
Build step 'Invoke top-level Maven targets' marked build as failure
Triggering a new build of CI_tool_run_integration_sanity_test
Finished: FAILURE
```

Fig. 4. Console output of the Jenkins job that runs the previously selected optimal test suite. (#36 CI_tool_run_optimal_test_suite)

AGH

How does the CI tool work?

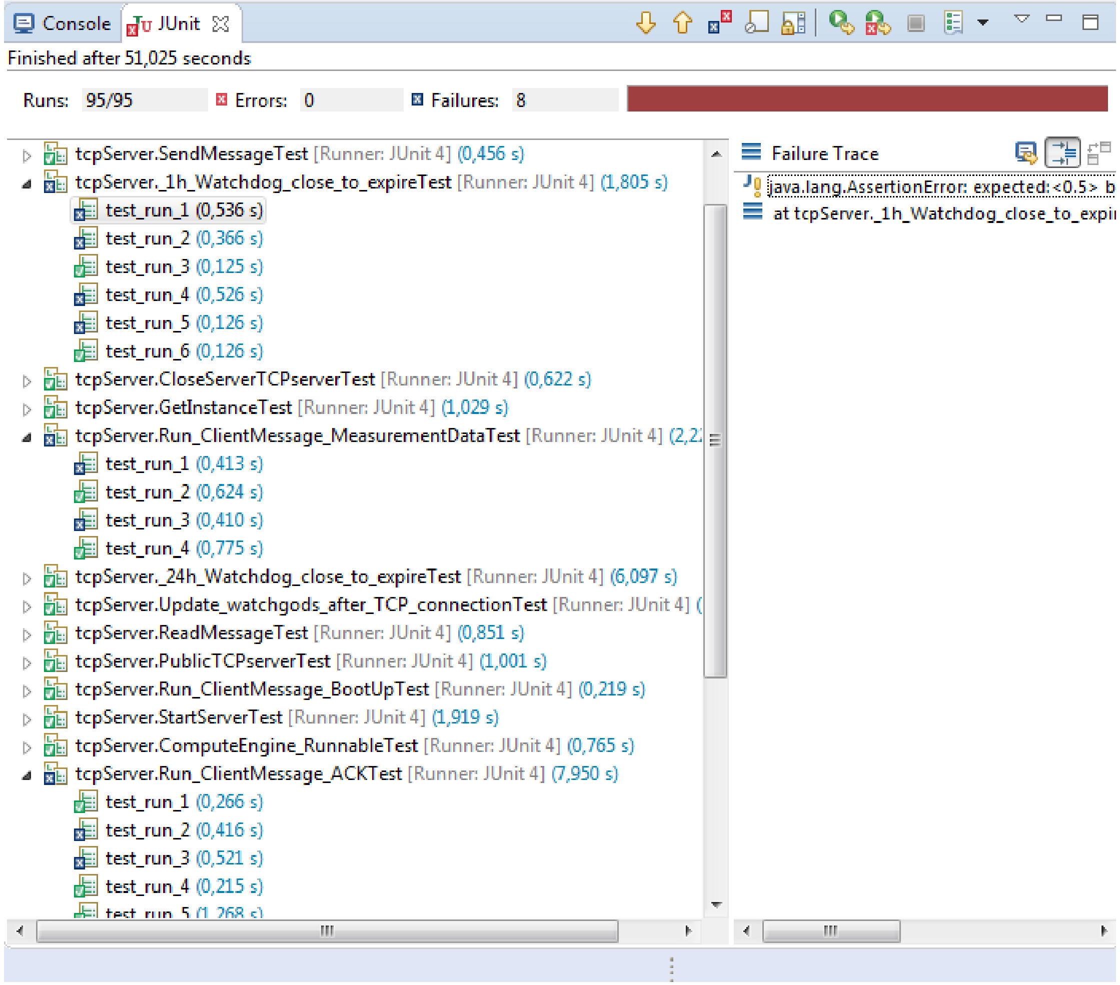

Fig. 5. Results of the execution of the entire test suite for TCP server.

How does the CI tool work?

Reverting changes from commit: CI tool validation for malfunctioning …

…software

Browse files

develop + production

xAndrew94 committed 4 days ago

1 parent 66969fd   commit 94eb99ae22d003a20d9f9144ec0fd3689cdf978c

Showing 1 changed file with 2 additions and 2 deletions.

Unified | Split

4 JavaWorkspace_MT/UnitUnderTest/src/main/java/tcpServer/ComputeEngine_Runnable.java

View

```diff
@@ -430,8 +430,8 @@ else if (close_ComputeEngine_Runnable) {
         protected double _1h_Watchdog_close_to_expire(double _1h_watchdog, double watchdog_scale_factor, int sensor_number_of_measurements ) {

                 // define duration of an additional delay that is used for watchdogs synchronization
-                double global_delay_factor = 0.20;
-                double local_delay_factor = 0.60;
+                double global_delay_factor = 0.25;
+                double local_delay_factor = 0.50;

                 //Global_1h_Watchdog.getInstance().setEnabled(false);
                 if ( sensor_number_of_measurements == (TCPserver.getMeasurements_limit() - 1) ){
```

Fig. 6. Github side-by-side diff for the commit that fixes the previously introduced bugs.

AGH

How does the CI tool work?

Jenkins › CI_tool_run_optimal_test_suite › #37

```
                Test Run 5 Purpose:
Verify outputs of the _1h_Watchdog_close_to_expire() function call for input_1h_watchdog higher than Watchdog_Thresholds.LOWEST and no_of_measurements equal to
TCPserver.getMeasurements_limit() - 1
                Test Run 5 Logic:
Server Thread Started.
[Compute engine Runnable] Multithreaded Server Service has been started
[Compute engine Runnable 1] _1h_watchdog equals : 0.6 when leaving _1h_Watchdog_close_to_expire()
[Compute engine Runnable 1] Global_1h_Watchdog equals : 1.2 when leaving _1h_Watchdog_close_to_expire()
                Test Run 5 teardown section:
Server Thread Stopped.
Serversocket closed
[TCPserver] all attributes of the static TCPserver class are reinitialized to default values

                Test Run 6 Purpose:
Verify outputs of the _1h_Watchdog_close_to_expire() function call for input_1h_watchdog lower than Watchdog_Thresholds.LOWEST and no_of_measurements equal to
TCPserver.getMeasurements_limit() - 1
                Test Run 6 Logic:
Server Thread Started.
[Compute engine Runnable] Multithreaded Server Service has been started
[Compute engine Runnable 1] _1h_watchdog equals : 0.4 when leaving _1h_Watchdog_close_to_expire()
[Compute engine Runnable 1] Global_1h_Watchdog equals : 0.8 when leaving _1h_Watchdog_close_to_expire()
                Test Run 6 teardown section:
Server Thread Stopped.
Serversocket closed
[TCPserver] all attributes of the static TCPserver class are reinitialized to default values

Tests run: 6, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 2.056 sec - in tcpServer._1h_Watchdog_close_to_expireTest

Results :

Tests run: 34, Failures: 0, Errors: 0, Skipped: 0

[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 28.406 s
[INFO] Finished at: 2018-09-09T22:33:37+02:00
[INFO] ------------------------------------------------------------------------
Triggering a new build of CI_tool_run_integration_sanity_test
Finished: SUCCESS
```

Fig. 7. Console output of the Jenkins job that runs the previously selected optimal test suite. (#37 CI_tool_run_optimal_test_suite)

How does the CI tool work?

Jenkins › CI_tool_run_integration_sanity_test › #30

```
[Compute engine Runnable 4] all Measurements Datas for sensor ID: 4 have been deleted
[Compute engine Runnable 4] set local 24h watchdog flags in the 24hWatchog_timestamp_table array to FALSE
[Compute engine Runnable 1] all Measurements Datas for sensor ID: 1 have been deleted
[Compute engine Runnable 1] set local 24h watchdog flags in the 24hWatchog_timestamp_table array to FALSE
[Compute engine Runnable 3] does not respond to ClientMessage_ACK.
[Compute engine Runnable 3] TCP connection with sensor: 3 is being closed
[Compute engine Runnable 3] all Measurements Datas for sensor ID: 3 have been deleted
[Compute engine Runnable 3] set local 24h watchdog flags in the 24hWatchog_timestamp_table array to FALSE
[ApplicationSanityCheckIT sensor: 1] Measurement History check file.getName():  measurements_2018-09-09_22-42-30.measurement_history
[ApplicationSanityCheckIT sensor: 1] Sensor Info check file.getName():          sensor_1_2018-09-09_22-42-30_gotoOPERATIONALafterRESET.sensor_info
[ApplicationSanityCheckIT sensor: 2] Measurement History check file.getName():  measurements_2018-09-09_22-42-30.measurement_history
[ApplicationSanityCheckIT sensor: 2] Sensor Info check file.getName():          sensor_2_2018-09-09_22-42-30_gotoOPERATIONALafterRESET.sensor_info
[ApplicationSanityCheckIT sensor: 3] Measurement History check file.getName():  measurements_2018-09-09_22-42-30.measurement_history
[ApplicationSanityCheckIT sensor: 3] Sensor Info check file.getName():          sensor_3_2018-09-09_22-42-30_gotoOPERATIONALafterRESET.sensor_info
[ApplicationSanityCheckIT sensor: 4] Measurement History check file.getName():  measurements_2018-09-09_22-42-30.measurement_history
[ApplicationSanityCheckIT sensor: 4] Sensor Info check file.getName():          sensor_4_2018-09-09_22-42-30_gotoOPERATIONALafterRESET.sensor_info
[ApplicationSanityCheckIT sensor: 5] Measurement History check file.getName():  measurements_2018-09-09_22-42-30.measurement_history
[ApplicationSanityCheckIT sensor: 5] Sensor Info check file.getName():          sensor_5_2018-09-09_22-42-30_gotoOPERATIONALafterRESET.sensor_info
[ApplicationSanityCheckIT] passed since number of TCPconnections reached the threshold
                Test Run 1 teardown section:

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 520.318 sec - in deliverables.ApplicationSanityCheckIT

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO]
[INFO] --- maven-failsafe-plugin:2.19.1:verify (integration-tests) @ UnitUnderTest ---
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 08:45 min
[INFO] Finished at: 2018-09-09T22:42:32+02:00
[INFO] ------------------------------------------------------------------------
Pushing HEAD to branch production at repo origin
 > E:\Program Files\Git\bin\git.exe --version # timeout=10
using GIT_ASKPASS to set credentials Github-password
 > E:\Program Files\Git\bin\git.exe push https://github.com/AndSze/CI_tool_for_an_optimal_test_suite_selection.git HEAD:production
Finished: SUCCESS
```

Fig. 8. Console output of the Jenkins job that runs the sanity check.
(#30 CI_tool_run_integration_sanity_test)

How does the CI tool work?

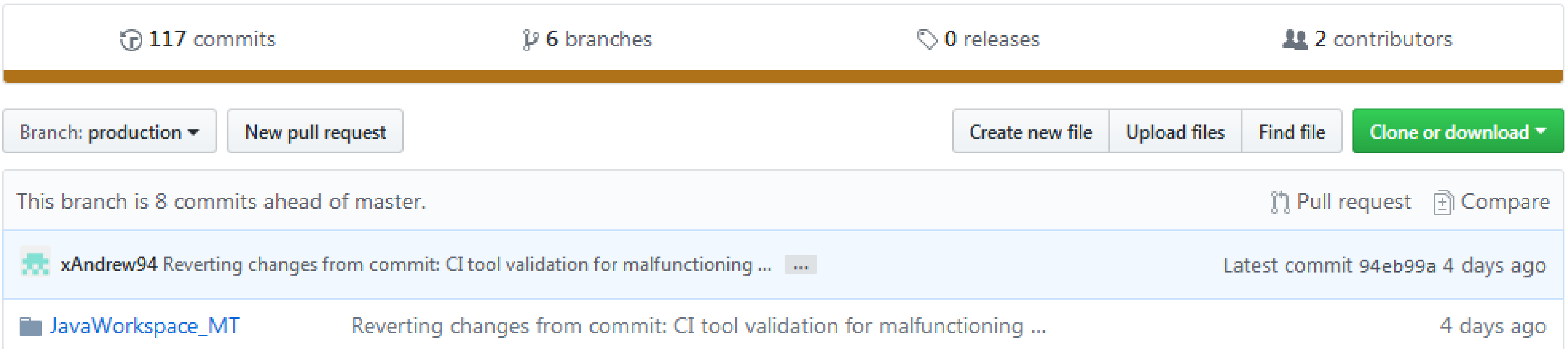

Fig. 9. Code merged to the production branch after successful execution of the Jenkins pipeline. (#43 CI_tool_build_pipeline)

Branches on the github repository [2]:

1) **master** - development branch for code of the application and all tests.
2) **develop** - temporarily development branch, used for testing the CI tool.
3) **production** - contains the code that has been successfully built and tested by the CI tool pipeline. The code on this branch is always up and running and can be delivered to a customer at any time.
4) **tests_selector** - contains python scripts used for an optimal test suite selection,
5) **JenkinsJobs** - contains the Jenkins jobs the CI tool comprises of.
6) **documents** - contains the official documents required to get Master's Degree.

The general principles of CI

In order to summarize the material covered by the scope of the thesis there should be a retrospective look at the general principles of the CI process made:

- Maintain a code repository,
- Automate the build,
- Make the build self-testing,
- Every commit should be built on an integration machine,
- Keep the build fast,
- Test in a clone of the production environment,
- Make it easy for anyone to get the latest executable version,
- Everyone can see the results of the latest build,
- Automate deployment.

The above list was mentioned once again on this slide for the purpose of illustrating that **the CI tool successfully uses each of the principles**.

The benefits of CI

The proposed CI tool besides the general advantages of adopting the CI principles [3]:

- Immediate feedback on software quality,
- Prevents integration problems, avoids last-minute chaos at release dates,
- Repeatable build process,
- Constant availability of a “current” build for testing, demo, or release purposes,
- Automated testing: code is tested in the same way for every change,
- Increases visibility which enables greater communication,
- Spends less time debugging and more time adding features,
- Helps break down the barriers between developers, testers and customer,
- Ease of tracking all of the changes, possibility to revert the code to stable version,

places additional emphasis on:

- **Measuring system-wide impact of local changes.**

Possibility for further improvement

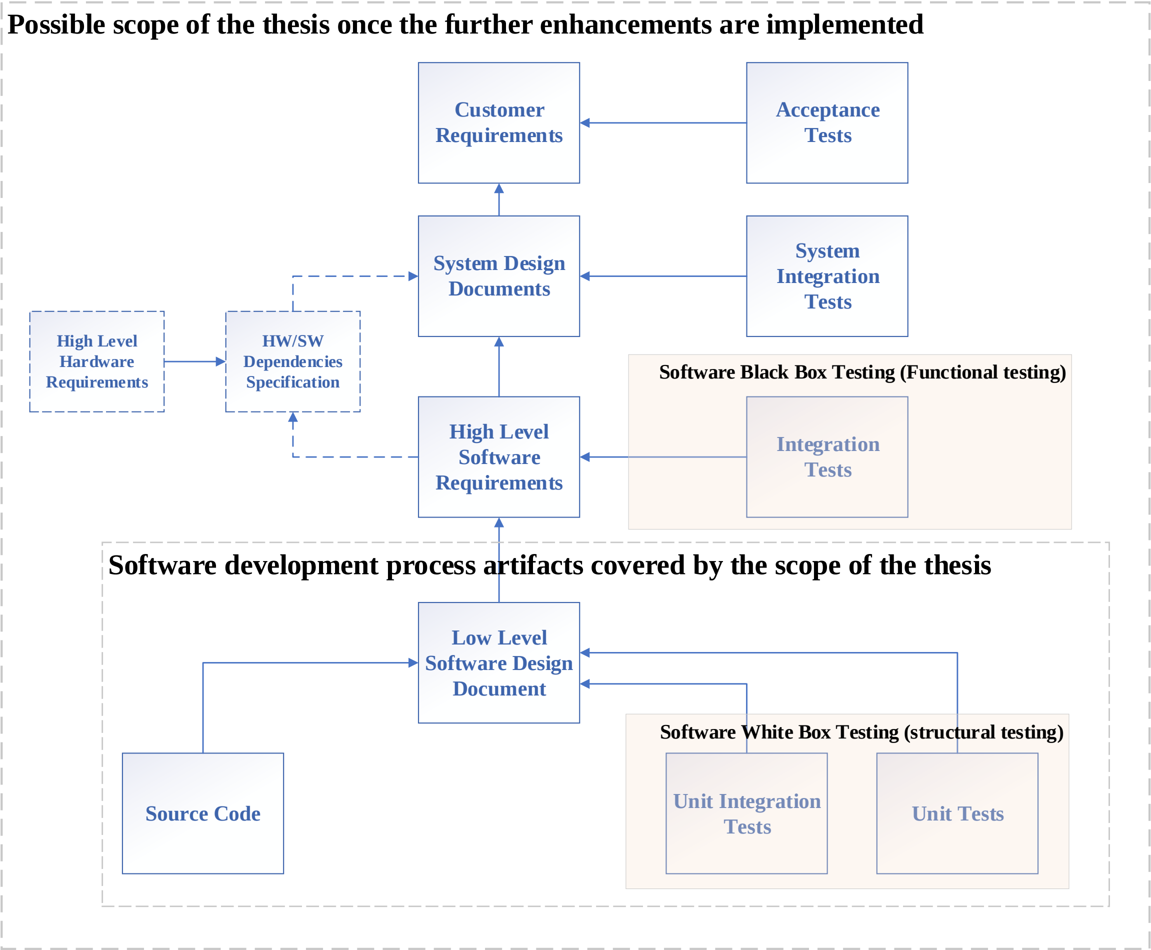

Fig. 10. The types of the software life cycle work products with an indication where the scope of the thesis applies to the software development process.

Amount of work for the CI tool

Summary of the software project artifacts the thesis is comprised of:

- Application:
  - more than **5000 lines of code in Java** (refactored multiple times to be testable),
- Tests:
  - **173 tests in JUnit** (60% unit integration tests, 40% unit tests), 95 tests for TCP server, 78 tests for TCP client (updated multiple times to be stable),
  - **sanity check** - comprehensive integration test,
- Build configuration for the project:
  - **2 Maven POM files**,
- Optimal test suite selector:
  - more than **2000 lines of code in Python** (powered by **GitPython** lib.),
- Automation server:
  - **Jenkins build pipeline** containing **5 Jenkins jobs**.

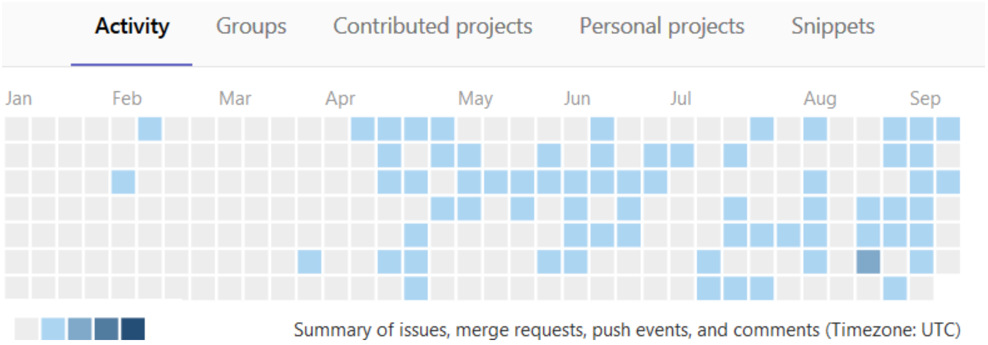

Fig. 11. Summary of the push events to the remote repository.

Bibliography

[1] - Continuous Integration: important principles and practices.
Available: https://www.thoughtworks.com/continuous-integration (visited September 14th, 2018)

[2] - Github public repository: CI_tool_for_an_optimal_test_suite_selection
Available:
https://github.com/AndSze/CI_tool_for_an_optimal_test_suite_selection

[3] - Top benefits of continuous integration.
Available: https://apiumtech.com/blog/top-benefits-of-continuous-integration-2/ (visited September 14th, 2018)