**Jakob Roberts - V00484900 - SENG 371 - Lab 7**

**1A: Describe what this analysis did and how it is configured including what source code it is running on**

Found the lines of code on all of the subclasses for a module/framework.

Configured using the CQB file, used a visual structure to evaluate the software.

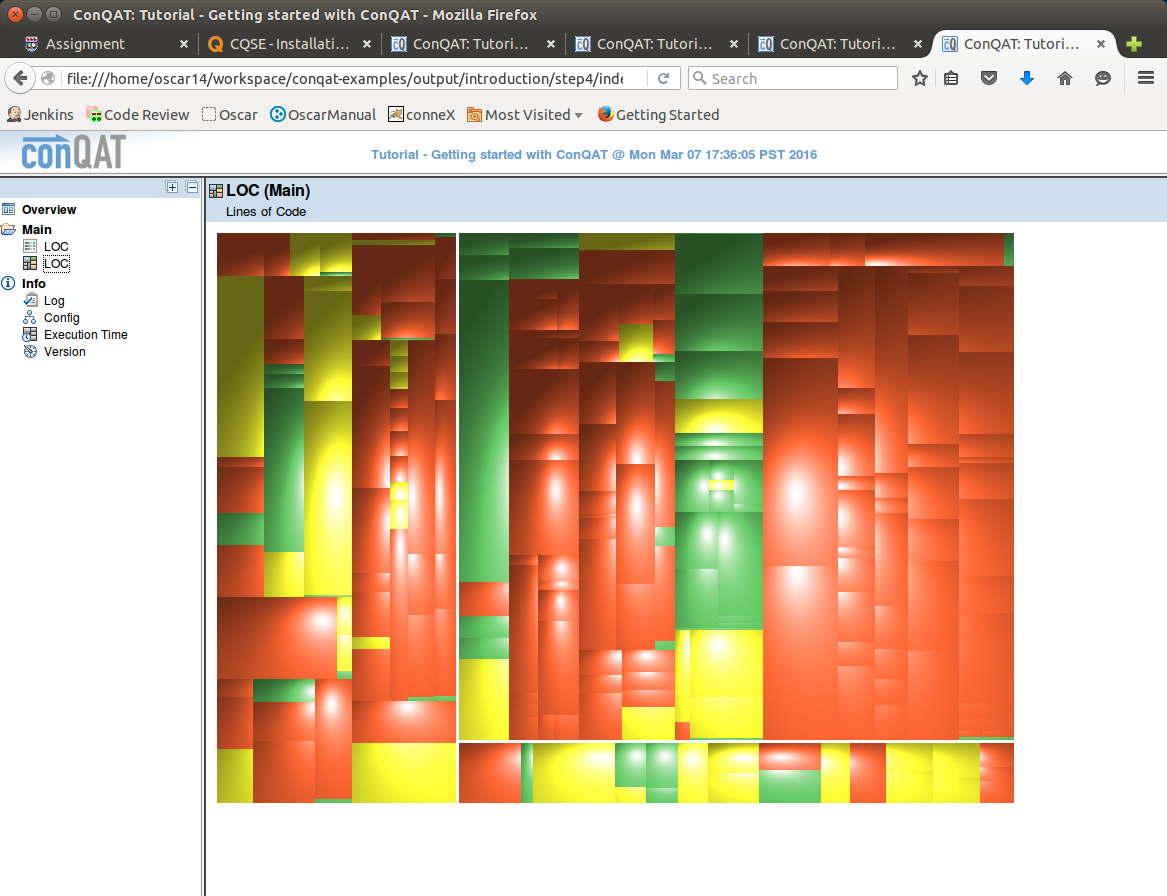
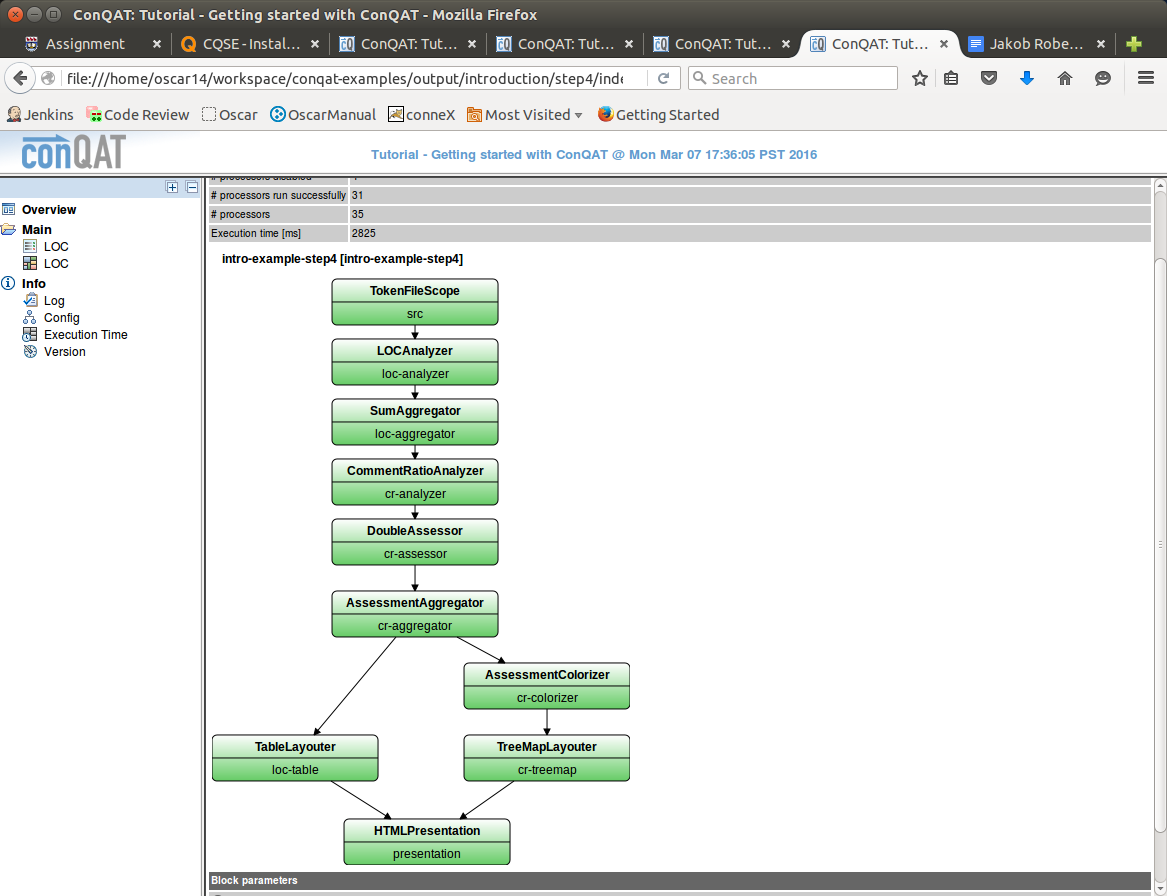
**1B: How does this analysis differ from the step 1 analysis**

It used the Sum Aggregator to get totals on frameworks, modules, and overall total for the software suite.

**1C: What does the step 3 analysis assess**

In addition to everything from the prior steps, it also analyzed the Comment Ratio of the lines of code to the lines of comments about that code.

**1D: take a screenshot of the treemap and put in in the PDF deliverable**



**2A: Investigate and report on one of the identified code clones.**

JUnit/junit/framework/ComparisonCompactor.java has a clone with JUnit/org/junit/ComparisonFailure.java

**2B: Investigate and report on a second of the identified code clones.**

JUnit/org/junit/tests/ParameterizedTestMethodTest.java has a clone with JUnit/org/junit/tests/TestMethodTest.java

**3A: Search for and report on tooling that could be used as part of the build process for preventative code clone management in OSCAR.**

Could use a plugin for MAVEN <https://github.com/basepom/duplicate-finder-maven-plugin>

**3B: Infer the type of code clone detection used by conQAT**

ConQAT implements [algorithms](https://en.wikipedia.org/wiki/Algorithms) for detecting [redundancy](https://en.wikipedia.org/wiki/Code_duplication) and architecture analysis in processors/blocks using processors and preprocessors for multiple different languages.