**A3: Measure Test Coverage**

**Jakob Roberts - V00484900** **Joe Czepil - V00774878**

# Part 1

**How does Emma work?**

EMMA does an instrumentation of the class files and the instrumentation code (additional byte code) is used to generate the coverage reports. The html report generator is capable of matching coverage results to line numbers.

**What kind of code coverage metrics are supported by Emma?**

EMMA supports class, method, line, and basic block coverage, aggregated at "all", package, source file, class, and method levels. Line and block metrics are offered in "normal" and "weighted" varieties.

**What kind of reports can Emma generate?**

EMMA reports fractional line coverage to help you visualize untested branches in code. EMMA reports class coverage so that you could spot classes that do not seem to be "touched" by your test suite: they could be either dead code or in need of more test attention.

**What settings can be adjusted for Emma’s report generation?**

Since EMMA bases some coverage types on basic blocks, it gives you a choice to treat all blocks as "equal" (every block contributes the same weight) or not (every block contributes a weight proportional to its size, i.e. the number of bytecode instructions it contains). This decision is made entirely at report generation time, via the report units setting. The default is weighted mode. EMMA can consolidate coverage data for multiple projects, multiple source trees, or multiple test runs. EMMA can instrument individual .class files or entire .jars .

**Description of how Emma was invoked to generate the report for task 2.**

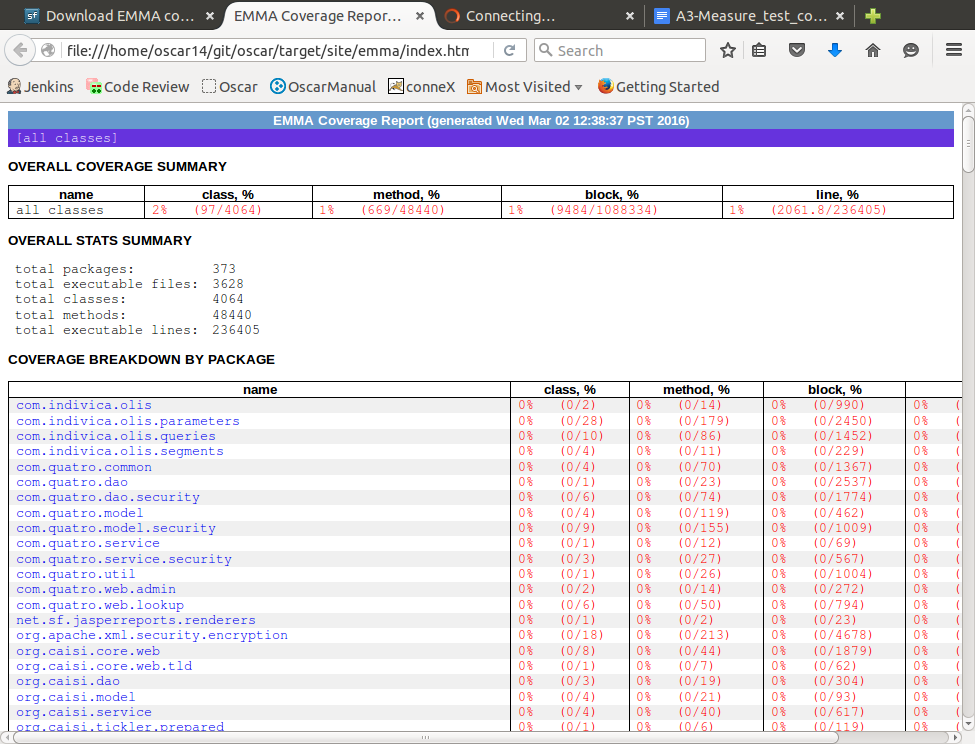
We downloaded the emma.jar file from sourceforge and put it into the \*/jre/lib/ext for the java version that oscar is using. Then all we had to run was ‘mvn emma:emma’ and it downloaded the plugin and ran it for us.

**Description of how Emma was integrated into the Cl build process for Maven.**

In order to integrate Emma into the CI build process we had to add certain plugin parameters in the pom.xml file. These parameters tell Maven what build of emma to use and how to build the reports. Once these plugins are in the pom.xml file Maven knows to generate the reports without specifically telling Maven to use emma in the command.

# Part 2

Initially when running the emma.jar file we ran into a problem that stopped the report process. After fixing this we started the report process again and it ran from 12:45 pm until 3:30 pm without any interruptions. However, do to other commitments we had to stop the process. We attempted the report process again but this time we removed the emma.jar file from the folder tree, shut off the tomcat7 service, and ran the the command mvn emma:emma then found the emma.jar file on it’s own and the html file was generated. This file is attached in the submission and a screenshot can be found below.



# 

# 

# Part 3

The following code was added to the pom.xml inside of the <reporting> and <plugins> tags.

|  |
| --- |
| <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-project-info-reports-plugin</artifactId>  <version>2.7</version>  <configuration>  <dependencyLocationsEnabled>false</dependencyLocationsEnabled>  </configuration> </plugin>  // integrate maven emma plugin to project site  <plugin>  <groupId>org.codehaus.mojo</groupId>  <artifactId>emma-maven-plugin</artifactId>  <version>1.0-alpha-3</version>  <inherited>true</inherited> </plugin> |

The following code was removed from the inside the <reporting> and <plugins> tags of the pom.xml due to errors:

|  |
| --- |
| <plugin>  <groupId>org.codehaus.mojo</groupId>  <artifactId>findbugs-maven-plugin</artifactId>  <version>3.0.3-SNAPSHOT</version>  </plugin> |

After these sections of the pom.xml file were updated the command mvn site was run in order to re-generate the html report.

# Part 4

The link to the gerrit changeset can be found below.

https://gerrit.seng.uvic.ca:8088/#/c/88/