CoverageEMMA

Coverage Report

Build	Class scope	Class %	Method%	Block%	Detail
trunk5 r566713	all public API	68% (1648/2421)	61% (16737 /27307)	59% (338675/572291)	here
trunk5 r566713	public API except JMX,ORB,XML	86% (1621/1893)	71% (16637/23570)	66% (337549/511018)	here
branch6 r566010	all public API	68% (1678/2480)	62% (17256 /28027)	59% (346435/583330)	here
branch6 r566010	public API except JMX,ORB,XML	86% (1648/1925)	71% (17151 /24137)	66% (345272/520446)	here

Report Generation

- **For normal application** EMMA is an opensource Java code coverage tool under CPL v1.0. For normal Java application, we can easily generate coverage report following EMMA's doc. Given that we have a Java-based application "JApp" with its test suite "JAppTest", using Sun's JDK5, we can get JApp's coverage like this:

a. instrumentation
EMMA will modify all (or part of) .class files of JApp (not include JAppTest), and generate metadata file (coverage.em in current folder by default). By the metadata file, EMMA can know which classes will be considered, what methods and how many blocks they have. By the modified (instrumented) class files, EMMA can collect coverage data when running test.
b. test running
run JAppTest on the instrumented classes of JApp, EMMA will automatically record how many blocks and what methods in what classes are covered. The coverage data file (coverage.ec) will be stored in the main folder of test suite.
c. report generation
generate report by the metadata from a) and record data from b).

- **For Harmony**
  For Harmony, we cannot follow the 3 steps above, now we must use a customized EMMA to generate coverage report for Harmony. The reason and detail about the customized EMMA can be seen in the section below.
  The process is similar as the normal case. The only difference is that we must pass a text file of "kernel class" as a JVM argument to EMMA in step a) and pass a text file of "kernel class signature" (generated by EMMA in step a.) as a JVM argument to EMMA in step b).
- **Kernel classes list**
  Since it is hard to predict which class will be referenced by EMMA, one way to solve the problem is simulate the process of collecting coverage data and let VM tell us the part of classes are loaded.

  test.jar contains a Test.test class which has already been instrumented by customized EMMA as a mock kernel class.

  test.kernel.siganture.list A mock kernel signature list which denotes that Test.test is a kernel class.

  java -verbose:class -Dkernel.signature.list=test.kernel.siganture.list -Xbootclasspath/p:test.jar:emma.jar test.Test & > kernel.classes.list

  Edit the output in file kernel.classes.list to remove unnecessary tags.

  kernel.classes.list Besides here is the standard kernel.classes.list generated on IBM VME.

- **Example** Firstly, we must use this customized emma-customize.jar tool. For Harmony build, we can do like this:
  a. instrumentation
  java -Dkernel.list=kernel.list.5 emma instr -ip $jars -d trunk5_instr_classes -ix @java5.all.api.list -ix @java5.exclude.api.list
  1. The kernel.list.5 is the list file of kernel classes.
  2. $jars is the set of Harmony jars, like "<libpath>/luni.jar:<libpath>/nio.jar:....."
  3. The instrumented class files will be stored in the folder "trunk5_instr_classes"
  4. The java5.all.api.list is the list file of all the public API classes.
  5. The java5.exclude.api.list is the list file of classes in luni-kernel.jar and security-kernel.jar, those classes can not be collected, we must exclude them. Those classes are NOT the kernel classes mentioned before.
  6. After this step, a file named "kernel.signature.list" will be generated in current folder. And metadata file "coverage.em" will also generated as normal case.

     b. test running
     ant -Dhy.test.vmargs="-Xms256m -Xmx1024m -Dkernel.signature.list=${basedir}/kernel.signature.list -Xbootclasspath /p:${basedir}/trunk5_instr_classes:${emma.dir}/emma.jar"
  7. -Xms256m -Xmx1024m is optional
  8. "kernel.signature.list" is the signature list generated in step a).
  9. "trunk5_instr_classes" is the instrumented classes of Harmony generated in step a).
  10. "emma.jar" is the customized EMMA.
  11. After this step, EMMA'll create a coverage report file named "coverage.ec" in each module's folder. There'll be 28 "coverage.ec"s in the 28 modules which have test to run.
      c. report generation
      java emma report -r html -in <base>/coverage.em -in <modules>/luni/coverage.ec -in <modules>/nio/coverage.ec .....
  12. This step is the same as the step c). in normal case.
  13. We can use ant task of EMMA to simplify the command.
  14. We can use more argument of EMMA to sort,color,arrange... the report.

15. To filter the report (for example, to ignore the XML and ORB module in the final result), we can create a new "coverage.em" file which filter those modules out and re-generate report with the new .em file and old .ec files, no need to re-run the test.

EMMA Customization

- Why customization
  When JApp is Harmony build, and JAppTest is Harmony test suite, the problem occurs. We may find that the test can not start successfully in step b. Because emma itself is a Java application, it cannot initialize normally on the instrumented (modified) classes (the class files of Harmony instrumented in step a).
  Some kernel classes in Harmony are critical for emma's initialization, they were loaded firstly when emma was initializating and they can not be modified(instrumented).
- Solution/customization
  To resolve this problem, we divided instrumentation into 2 actions: the kernel classes firstly were recorded in a text file ("kernel.list" in example) but not instrumented, and other classes were instrumented normally; then, in step b, emma firstly read (from the "kernel.signature.list" in example) and loaded the un-modified kernel classes and initialized successfully, and instrumented those kernel classes for coverage record use.
  The list of kernel classes ("kernel.list" in example) can be cut from the "all classes list" by trying to start emma.
- Source code
  The modifed source code is modify.tar.gz and the affected original source code is original.tar.gz.
  In current customized EMMA, we only modify 3 source files and add 2 new source files. So when we unzip the tar.gz files above, we'll find 3 files in original files and 5 files in modified files. The original files are only used as a comparison.
  To make a customized EMMA tool, we can build the whole project (from CVS) in eclipse and replace those 3 original source with the modified ones and added the two new source files (pay attention about their packages). After build, we can replace the original .class files in original emma.jar with our modified .class files (we use the release version of EMMA, v2.0.5312), then we'll get the customized EMMA.
- Future improvement
  Currently, the kernel class list is not very stable, if we change test suite or change the API classes set, we may modify the list. And we do not have an automatically way to get the list yet, we must try and try to find them. If we can find a way to automatically find them, it'll be better.

Customize for BTI adaptor

This emma.jar is only modified for BTI emma instrument adaptor.We tell this emma to find kernel.signature.list from ${java.home} rather than from parameter of -Dkernel.signature.list=${location}.

Any comment about the modified source code is highly welcome.