Testwell CMTJava

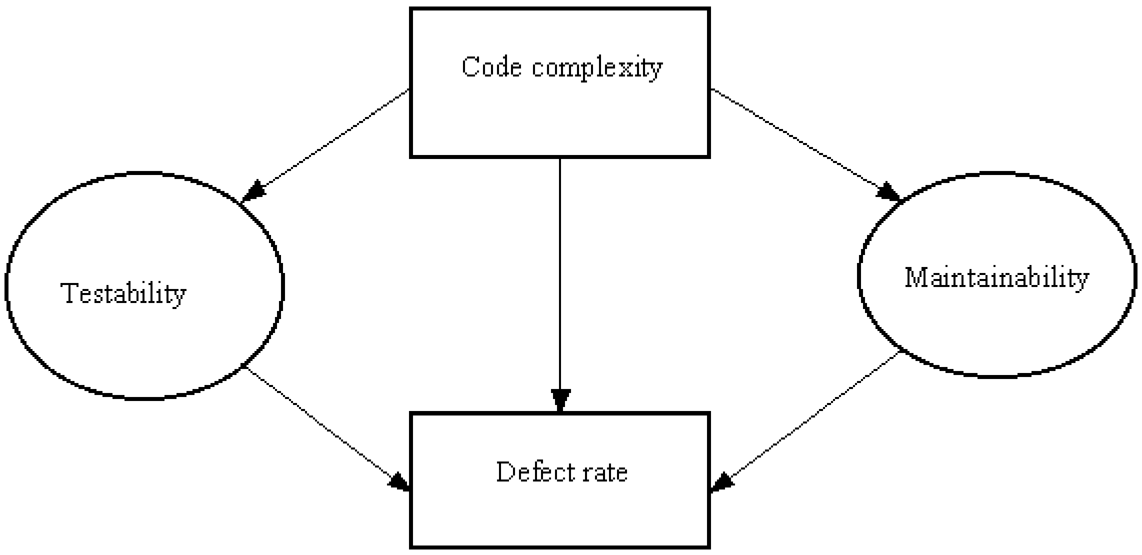
Complexity Measures Tool for Java

Information in this document corresponds to the CMTJava version 3.0.

1. INTRODUCTION

CMTJava, Complexity Measures Tool for Java, is an easy-to-use code metrics tool for the Java language. It is intended for mature software development organizations striving for productive development process resulting in high quality products.

Code complexity has effect on how difficult it is to test and maintain the program.



CMTJava helps you to locate the likely problematic complex code functions and files from big code masses. Then you can assess them separately: put special testing focus on them or perhaps redesign them. You can use CMTJava also for just measuring how much code you have: physical lines, comment lines, program lines, statements. For example measuring the whole project code base is no problem to CMTJava.

Here are some quick links to some of the CMTJava capabilities:

- Command line mode use
- GUI mode use (Windows)
- Default textual report and same report with top level class summary only
- HTML report (textual report converted to HTML and assigned to source files, started in new window)
- Long XML report and same report with operand/operator frequencies and of top level class summary only
- Excel report and same report with top level class summary only

CMTJava Facilitates

- Calculating the basic code complexity metrics
  - McCabe's cyclomatic number (can be calculated in 4 "flavors": basic, extended (=default), basic_modified, extended_modified)
  - Halstead's software science metrics
  - Lines of code metrics
  - Some other metrics like: number of semicolons, number of Java comment blocks /** ... */, number of method parameter, depth of control structure nesting
- Calculating Maintainability Index (MI)
  - MI is a single number value (or index), which expresses the relative maintainability of the code
  - MI is derived with an arithmetic formulae from the code's McCabe, Halstead and lines of code measures
  - The used formulae is widely accepted at the industry and is defined at various places in the web.
- Extremely fast processing
  - CMTJava can measure many thousands of Java source lines in a second
- Usage in the large
  - In one CMTJava session you can analyze a single file, a couple of files, your whole application, or the whole code base of your company possibly containing thousands of Java files
- Publishing the results in HTML
  - The CMTJava report, linked to the actual source code, can be converted to HTML form and put e.g. to company intranet. The representation is hierarchical and color-coded. Items whose measures are out of their acceptable limits are highlighted in red. With one mouse click it is possible to see the actual source code of the method or class at hand.
- Independent of the used Java development environment
  - CMTJava does not use, not even assume the presence of, a Java development environment. The tool only assumes that it can read the Java code from normal operating system source files.
  - Supports Java language upto Java SE 7 level
- Further processing of the measurements
  - CMTJava can write the measures into a text file, which is readily suitable input for Excel (or any other spreadsheet program), by which the measures can be processed further, say, for various company summaries or for graphical representation.
  - The measurements can be generated also in XML form.
- Ease of use
  - CMTJava is simple to use from the command line
  - On Windows platform there is a GUI layer for using the tool
  - Includes thorough on-line helps in Windows, man pages in Unix
- Can be integrated into your build scripts for collecting the measures into the database
- Available on many platforms
  - These include Windows and many Unix environments (see CMTJava availability)

CMTJava intended use is:

- Use as a testing tool

    - Identifies code sections which are suspiciously complex and thus more error prone, and should be tested more thoroughly. The limited and expensive human testing resources can be utilized more optimally
    - McCabe's cyclomatic number v(G) gives an estimate on how many test cases at least are needed to test all paths of a method.
    - Halstead's estimated number of bugs B gives a goal on how many bugs at least the testers should try to find from a source file.
- Use as a quality assurance tool
    - CMTJava Complexity Measures Report as supplementary material in code reviews
    - With consistent use of CMTJava, and when the alarms are assessed and appropriate actions are taken, there are far better chances to come out with more maintainable software
- Enforcing company standards with regard to accepted code complexity
    - Configurable alarm limits
    - Measures that are outside the limits are highlighted in the report
- Assessing foreign code
    - CMTJava gives you a good overall view of the coding style and maintainability expectations of a set of Java files that you are supposed to take over or your subcontractor is delivering to you.
- Use as a measuring tool
    - Simply measure how much code there is (code/comment/blank lines, files, packages, classes, interfaces, semicolons, Java comments, etc.)
- Integrated to build scripts and collecting the measures to a code metrics database

2. USE OF CMTJava

2.1. Command line mode

The "core component" of CMTJava is *cmtjava*. Its on-line help for command-line options is:

```
Usage:
    cmtjava [-h] [-H] [-c cnffiles] [-C cnfparam=value]... [-v] [-l[f]]
            [-s] [-w] [-x] [-nxh] [-f filenames] [-o outfile] [srcfiles...]
```

And it can be used for example as follows:

```
cmtjava -o report.txt *.java             (measure these Java files)
notepad report.txt                       (view the report)

dir /b /s *.java > fnames.lst            (names of .java files)
dir /b /s ..\yproj\*.java >> fnames.lst  (add some more...)
cmtjava -o report2.txt -f fnames.lst     (measure them)

cmtjava -l -o report.xml -f fnames.lst   (take "long" report, in XML)
myxmlutility -i report.xml               (process onwards...)

cmtjava -x -o xreport.txt -f fnames.lst  (take Excel form report)
start excel xreport.txt                  (start Excel on it)
```

The default form textual report can be converted to HTML form with the *cmtjava2html* tool. Its on-line help for command line options is:

```
Usage:
  cmtjava2html [-i inputfile] [-o outputdir] [-s sourcedir]...
               [-l splitsize] [-no-html-sources] [-no-javascript]
                 [-nsb] [-p prefix]
  cmtjava2html -h
```

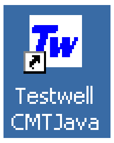
```
Command-line options:

  -i inputfile      Input Complexity Measures Report file. Default 'stdin'.
  -o outputdir      Output HTML directory. Default 'CMTJAVAHTML'.
  -s sourcedir      Source files are searched also from this directory.
  -l splitsize      Specify number of lines reported per page. Default 1000.
  -no-html-sources  Do not convert sources to HTML.
  -no-javascript    Do not generate javascript on HTML pages.
  -p prefix         File name prefix in generated HTML files. Default 
'index'.
  -nsb              Do not start HTML browser automatically (only Windows).
  -h                Display command line help.
Start browsing from <outputdir>/<prefix>.html (default 
CMTJAVAHTML/index.html)
```

And it can be used for example as follows:

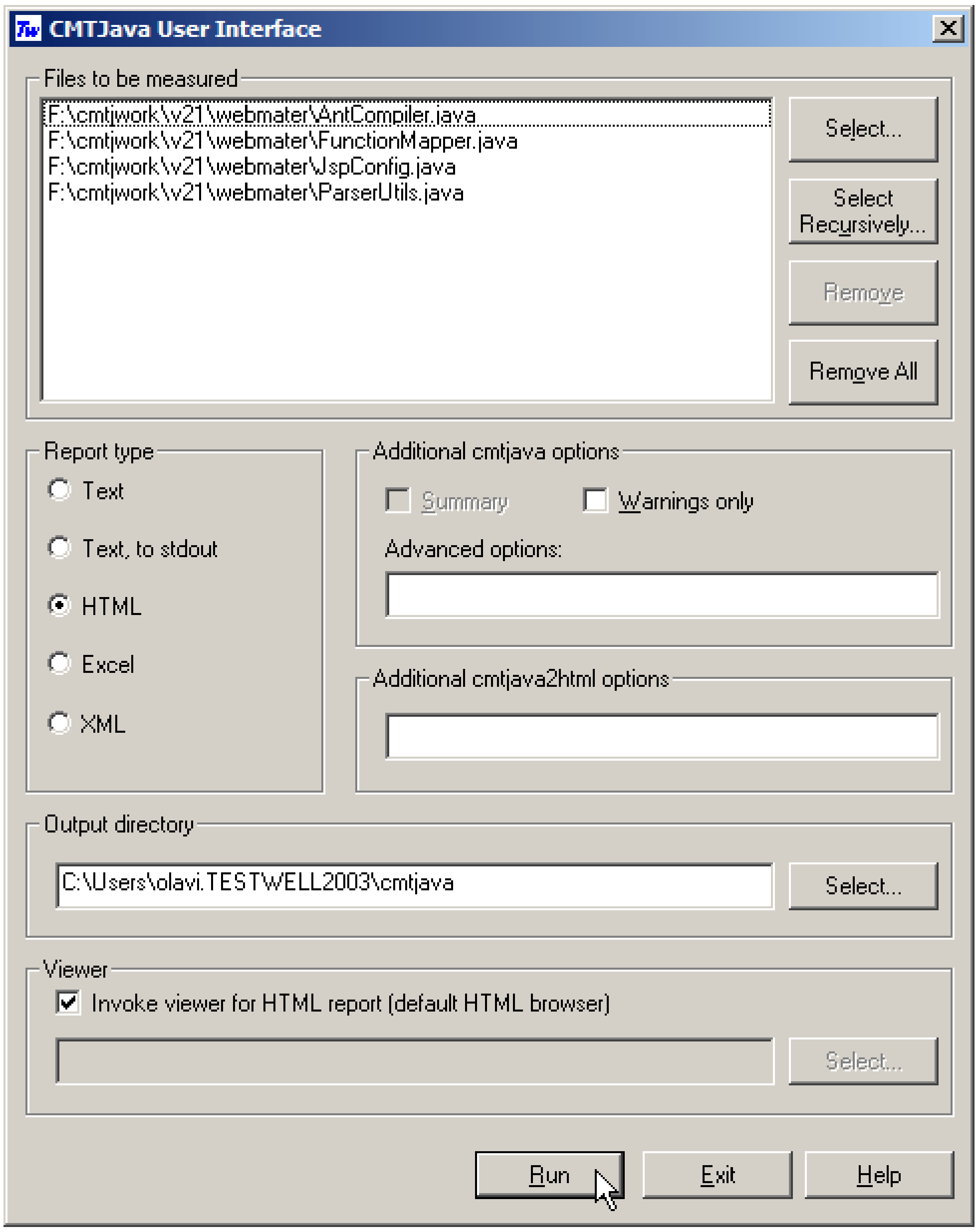
```
cmtjava2html -i report.txt -nsb   (convert the report to HTML)
start CMTJAVAHTML\index.html      (start default browser on the HTML)
```

2.2. CMTJava GUI

This is supported on Windows. The GUI can be started in many ways. For example, creating a shortcut on the cmtjavaui.exe on the desktop and starting the tool by clicking it.



It starts the following program dialog box:


CMTJava User Interface

Files to be measured
F:\cmtjwork\v21\webmater\AntCompiler.java
F:\cmtjwork\v21\webmater\FunctionMapper.java
F:\cmtjwork\v21\webmater\JspConfig.java
F:\cmtjwork\v21\webmater\ParserUtils.java

Report type
Text [✓ Text, to stdout]...

Report type: Text / Text, to stdout / [✓ HTML] / Excel / XML

Additional cmtjava options
Summary [ ] Warnings only [ ]
Advanced options:

Additional cmtjava2html options

Output directory
C:\Users\olavi.TESTWELL2003\cmtjava

Viewer
[✓] Invoke viewer for HTML report (default HTML browser)

In this dialog box you can select the files to be measured and fully control how the basic cmtjava run is done. You can specify a viewer (e.g. notepad or Excel), which will be started on the generated report. You can also ask that the CMTJava report is further processed with cmtjava2html utility and your default browser can be started on the HTML form report.

3. CMTJava Reports

The CMTJava reports are demonstrated here with 4 Java source files that have been picked "as is" from the Apache Tomcat open-source project. You can study the source files by starting the HTML report (started in a new window), going to the DETAILED view and clicking the links at the file names. The files have been selected so that they would demonstrate how CMTJava models the Java source code. Notably,

- Input to one *cmtjava* run is one or more Java source files [certain summary information is calculated for each file and for all files together (considered "system") level]

- One Java source file contains one or more top-level classes or interfaces [detailed LOC, McCabe, Halstead and MI measures are calculated on such entities]
- A top-level class or interface contains methods, inner classes and interfaces [detailed LOC, McCabe, Halstead and MI measures are calculated on such entities]
- A second-level class or interface contains yet-inner methods, classes and interfaces [detailed LOC, McCabe, Halstead and MI measures are calculated on such entities]

3.1. Short report

In normal use, when one or only a few files are measured, the short report is simple to use. It can be conveniently viewed with a text editor or printed on paper.

CMTJava calculates quite many different measures of the source code, but the measures shown in a short report are likely the most useful ones: v(G) McCabe's cyclomatic number/control flow complexity; the lines of code measures LOCphy, LOCpro, commenting ratio (shown as alarm sign if outside the limits); the Halstead measures V volume/logical "size" and B estimated number of bugs; MI Maintainability Index.

See an example of the default short report, which has been generated effectively as follows:

```
cmtjava -o report.txt *.java
```

The short report can be restricted to contain only the measures of the top-level classes and interfaces. See an example of the default short summary level report, which has been generated effectively as follows:

```
cmtjava -s -o report-s.txt *.java
```

These reports can be further restricted to contain only the lines that contain the alarm sign "-" on some of the measured items.

3.2. HTML report

The HTML report is an html'ized version of the short report. Additionally, if the Java source files are available, links to them are generated facilitating viewing them also at the browser.

When the HTML report is started a summary view is shown. It contains the measures of the two top level classes and interfaces. Clicking on the class/interface name you can get to a detailed view, which contains also the measures of the methods. In the detailed view, clicking on the class/interface/method name the actual source code is shown.

See an example of the HTML report (started in a new window) , which has been generated effectively as follows:

```
cmtjava -o report.txt *.java
cmtjava2html -i report.txt
```

Here the *cmtjava2html* run was done so that the option -no-html-sources was not used. It means that html'ized copies of the source files are generated inside the HTML report (which is a collection of .html files in the generated CMTJAVAHTML directory). If that option had been used, only links to the source files would have been generated at the location where they reside.

Further, the option -no-javascript was not used. It means that Javascripts are generated into the HTML report. They are used to display the source code section of individual methods and classes. If your browser has Javascripts disabled, you'd better generate the HTML report with that option and so allow smooth display of the HTML report, but with the price that individual method or class source code cannot be separately displayed.

3.3. Long report (in XML)

The long report is obtained with cmtjava option -l (or -lf). It contains all the measures that CMTJava calculates. The report form is XML. This report form is meant to be used in CMTJava integrations to tool chains. The report can be taken with operand and operator frequencies (option -lf) or without them (option -l only). The report can be restricted to contain top level class summary information only (option -s used).

See example of long report (default form). See example of same report but containing top level classes only and operand and operator frequencies included). Effectively these two reports have been obtained as follows:

```
cmtjava -l -o report-l.xml *.java
cmtjava -lf -s report-lf-s.xml *.java
```

3.4. Excel report

The Excel report is a csv file, which can be input to Excel for further processing. The Excel report contains all the measures that CMTJava calculates of each measured item, a kind of "raw data", where you can study just the measures that you are interested in. From the Excel form you can also derive such graphical/summary representations that you want.

See an example of the Excel report , which has been generated effectively as follows (the methods):

```
cmtjava -x -o report-x.txt *.java
```

See another example of the Excel report , which has generated effectively as follows (the top-level classes):

```
cmtjava -x -s -o report-x-s.txt *.java
```

4. OPERATING ENVIRONMENTS

See CMTJava availability for a detailed list of machine / operating system environments where CMTJava is currently available.

A Java development environment is not needed for using CMTJava. However, CMTJava assumes that the source files to be measured are "sufficiently correct" Java.