Description for Evaluation Reproduction

(for „Getting to Know You... Towards a Capability Model for Java“ by Ben Hermann, Michael Reif, Michael Eichberg, and Mira Mezini)

June 6, 2015

1. Introduction

The purpose of this manual is to guide you through the steps necessary to run our analysis and reproduce the results in the paper's evaluation.

As we make assumptions about the folder structure in our software, it is important to check whether the JAR files of the pure analysis and the evaluation are located in the same folder as the resource folder. This is the case, when you use the replication package ZIP file provided by us. When you build from source, please make sure this is the case.

The "resource" folder contains the mapping of native calls to capabilities and the "rt.jar" file of the unofficial windows build of the OpenJDK 1.7 update 60. (There is no official build for Windows available)

2. Prerequisites

The analysis and the evaluation tooling is written in Scala and compiles down to Java Bytecode. Thus, it will run on every machine with a Java 1.8. Runtime installed. We were successfully able to run it using the Windows 8 and MacOS X 10.10.3 operating systems.

For reproducing the evaluation charts and statistics you may need R (http://www.r-project.org/) and a standard browser (we recommend using Firefox).

3. How to use the Tool (Quick Software Manual)

If prerequisites are met you can start our tool from the command line interface. Open a console and navigate to the folder where you placed the jar file and the resource folder.

If you want to run the analysis you have to provide a project that you want to test. Therefore, you have to specify it via the "-cp" parameter.

To execute the jar and analyze the "xstream" library you type in the following command:

"java -jar PEAKS_JavaCapAnalysis.jar -cp=resources/projects/xstream"

Notice that "xstream" is not a .jar but a folder. If you pass a folder to the analysis every class file and jar in this folder will be included to the analysis.

If you start the analysis, there will be a menu.

[1] Start capability analysis for libraries.

[2] Sliced capability analysis for projects.
[3] Help.

The normal usage (used in the paper) is the first option. If you only provide a project it will print capability set to the command line. You could specify other parameters if you are only interested in some capabilities or if you want get the methods which transitively use certain capabilities. Use the third option to get an overview over all available parameters.

This is the complete list of all parameters that are available:

[-cp=<Directories or JAR/class files> (If no class path is specified the current folder is used.)]
[-libcp=<Directories or JAR/class files>]
[-lm] - All found methods with capabilities gets listed.
[-CL] - Print all methods with the CLASSLOADING capability.
[-CB] - Print all methods with the CLIPBOARD capability.
[-DB] - Print all methods with the DEBUG capability.
[-FS] - Print all methods with the FS capability.
[-GU] - Print all methods with the GUI capability.
[-IN] - Print all methods with the INPUT capability.
[-OS] - Print all methods with the OS capability.
[-NT] - Print all methods with the NET capability.
[-PR] - Print all methods with the PRINT capability.
[-RF] - Print all methods with the REFLECTION capability.
[-SC] - Print all methods with the SECURITY capability.
[-SD] - Print all methods with the SOUND capability.
[-SY] - Print all methods with the SYSTEM capability.
[-UN] - Print all methods with the UNSAFE capability.

Option 2 from the menu above is a bit more advanced. The analysis will only take care of the actually used part of the libraries in an application. So far, this works only under the assumption that all the application dependencies are packaged in the same jar as the application. As this is the common case for the deployment of most applications it should fit most development processes nicely.

4. Reproducing the Evaluation

This section describes how to reproduce our evaluation results.

4.1. Requirements

Note that you will need a connection to the Internet as we used the online documentation for some libraries. If you work without an Internet access the capability set from the keyword scan of libraries with online documentation will be empty and the results will differ from ours. Please make sure that the PEAKS_Eval_JavaCapAnalysis JAR is located in the same folder as the "output" and "resources" folder.

4.2 Reproducing the Evaluation

You can start the evaluation by opening a console. Then navigate to the folder where the jar is located and execute the following command:

"java -jar PEAKS_Eval_JavaCapAnalysis.jar"

After execution this command, a menu will appear. Press "1" and Enter to trigger the evaluation process. The evaluation will start now, the output will be written to the output folder. The evaluation can take up to 6 hours (depending on your computer). Due to heavy parallelization of the OPAL Framework your computer can be quite busy while running the evaluation.

You will then find the results in the “output” folder in the file “EvaluationResults.csv”. You can compare them to our results from the “Evaluation” folder.

If the evaluation is done and you may want to recreate the capability distribution chart from the paper (Figure 4) you have to execute the jar again. Choose the second menu item this time. It will trigger a transformation of the “EvaluationResults.csv” to another representation needed to execute the R script. This new file is also located in the “output” folder and is named “transformedResults.csv”. The R script (peaks.R) is located in the Evaluation folder. If you want to use it, adapt the working directory to the “output” folder before.

To create the capability matrix (Figure 3), copy the “EvaluationResults.csv” file from the “output” folder into the “Evaluation” folder and open the capmap.html file with a browser. As Google Chrome (and other browsers) suppress locally run java scripts from loading files (in this case the CSV), we suggest Mozilla Firefox or running a small webserver (e.g. with NodeJS).

5. Differences to the Paper

There are no significant differences between the claims made in the paper and the reproducible package. However, we did update of the Bytecode Analysis Framework from OPAL to a newer version and noticed that the NET capability was not detected in four projects used in the paper’s evaluation anymore (i.e. sunflow, sandmark, lucene-query, xstream). We investigated the differences and found that all on them stem from a more precise call graph construction now available in OPAL.

Another difference that could influence the evaluation results on the documentation side is the use online API documentation. (We had to use these because they were not available for download.) If a documentation is not available or has changed for some reason, the resulting keywords could be empty or could slightly differ to our evaluation results.

5. Content Overview

Resource	Description
PEAKS_JavaCapAnalysis.jar	Runnable ".jar" file where the analyses are implemented. Executable via the command-line interface.
PEAKS_Eval_JavaCapAnalysis.jar	Runnable ".jar" file which triggers the evaluation. The output is written as csv file to the output folder.
Folder "src and javadoc"	This folder contains several jar files which include the source files and the Javadoc of our analysis and the evaluation project. In addition, it contains the source files jar of our performed Javadoc keyword scan. The executable jar of the keyword scan is included in the lib directory of the Evaluation project archive.
Folder "resources/capabilities"	The content of this folder is our keyword mapping for the Javadoc keyword scan (Section 4 of the Paper) and the bootstrap mapping of the native methods to a capability set (Section 3.2 of the Paper).
Folder "resources/csvs"	This folder contains the meta data for our projects used to evaluate the analysis. Essentially it contains the library name, the library version and the location of the Javadoc.
Folder "resources/jre_7.0_60"	This folder contains the rt.jar of the unofficial OpenJDK Build for Windows (version 1.7 update 60)[1] We have used the 64-Bit version.
Folder "resources/projects"	This folder contains the jars and in some cases the Javadoc of all projects we used for our evaluation.
Folder "output"	Target folder for the evaluation results.
Folder "Evaluation Results"	The results of our evaluation.

[1]

Windows 64-Bit can be Downloaded here:
https://bitbucket.org/alexkasko/openjdk-unofficial-builds/downloads/openjdk-1.7.0-u60-unofficial-windows-amd64-installer.zip

Windows 32-Bit can be downloaded here:
https://bitbucket.org/alexkasko/openjdk-unofficial-builds/downloads/openjdk-1.7.0-u60-unofficial-windows-i586-installer.zip

6. CSV Data Schemata

File / Description	Column	Description
`resources/capabilities/NativeMethodsRT.csv` Bootstrapping of native methods of the JCL in OpenJDK 7 update 60 (Windows) (Section 3.2 in the paper)	package	Package of the method
	class	Class name of the method
	method name	Name of the native method
	capabilities	List of capabilities associated (comma separated)
	<unnamed columns>	Used to identify a method by its signature
`resources/capabilities/keywords.csv` Mapping from capabilities to keywords (Section 4 in the Paper)	Capability	Capability
	Keywords	Associated keywords to the capability
`resources/csvs/EvaluationProjects.csv` List of all evaluation projects and their documentation)	Tool / Library	Project name
	Version	Project version
	Link to Javadoc	Link to the documentation (either local or remote)
`*/EvaluationResults.csv` Result dataset of the evaluation	project	Project name
	peaks	Capability set derived from code analysis
	docs	Capability set derived from documentation analysis
`Evaluation/caps.csv` Mapping of capability names to their abbreviation for capmap.html	full	Full name of the capability
	sh	Short name of the capability