Practical Lab Series 1 – Software Metrics

In Series 1 we focus on software metrics. Software metrics are used (for example) by the Software Improvement Group (`http://www.sig.eu`) to gain an overview of the quality of software systems and to pinpoint problem areas that may cause low maintainability. Some relevant questions are:

1. Which metrics are used?
2. How are these metrics computed?
3. How well do these metrics indicate what we really want to know about these systems and how can we judge that?
4. How can we improve any of the above?

In other words, in this assignments you concern yourself with the motivation, interpretation and implementation of metrics. The SIG Maintainability Model provides an answer to question 1. You can read about it here:

- I. Heitlager, T. Kuipers, and J. Visser. "A Practical Model for Measuring Maintainability". In: *Quality of Information and Communications Technology, 2007. QUATIC 2007. 6th International Conference on the*. 2007, pp. 30–39. DOI: `10.1109/QUATIC.2007.8`.
- Additional reading is provided by Baggen *et al.* [3], Visser *et al.* [34] and online `https://www.sig.eu/resources/sig-models/`

Question 2 is partially answered by the 2007 paper referred to above and by you in your programming for this assignment. The remaining questions are answered in the report.

Collaboration

Series 1 and 2 are executed in (the same) pairs. You can work together as a pair on all aspects of this assignment. You can brainstorm with anybody else about the contents of your report, but for this assignment you are not allowed to look at code from other groups or exchange solutions in detail with other groups. Golden rule: "exchange ideas, not solutions!"

Assignment

Using Rascal, design and build a tool that calculates the SIG Maintainability Model scores for a Java project. Document your approach in a report that complements the implementation, e.g., by describing relevant design decisions, tests, results, and what you did to address threats to validity. Make sure that your report (also) answers all the questions of the above introduction.

Calculate at least the following metrics:

- Volume,
- Unit Size,
- Unit Complexity,

- Duplication.

For all metrics you calculate the actual metric values, for Unit Size and Unit Complexity you additionally calculate a risk profile, and finally each metric gets a score based on the SIG model ($--$, $-$, o, $+$, $++$).

Calculate scores for at least the following maintainability aspects based on the SIG model:

- Maintainability (overall),
- Analysability,
- Changeability,
- Testability.

You can earn bonus points by also implementing the Test Quality metric and a score for the Stability maintainability aspect.

Your tool should print textual output that contains all of the above information in a way that is efficient with space, easy to read and makes it easy to confirm the calculations.

Use the following zip file to obtain compilable versions of two Java systems (smallsql and hsqldb): zip file[11]

- **smallsql** is a small system to use for experimentation and testing. Import as-is into Eclipse and ignore build errors.
- **hsqldb** is a larger system to demonstrate scalability. Import into Eclipse. Make sure to have only `hsqldb/src` on the build path, and add the following external jars from your `eclipse/plugins/` directory:
  - `javax.servlet_$VERSION.jar` and
  - `org.apache.ant_$VERSION/lib/ant.jar`

Hints

- Create a Java project with example files to test your solution on (using the Rascal test functionality).
- Create a Java project for each of the two systems, smallsql and hsqldb. Some few lines of code will still not compile, but commenting them out would not change the metrics too much. So commenting out just a few lines is ok in this case. It saves time!

Grading

You submit a **single** zip file containing the *source code*, a PDF of your *report*, and a *document* containing the output your tool produces for the test projects. The files are checked for plagiarism automatically. You will be graded using the following model. The base grade is 7. For this grade you need an implementation that conforms to the assignment described above.

[11] `http://homepages.cwi.nl/~jurgenv/teaching/evolution1314/assignment1.zip`

Condition	Max grade modifier
The metric value (total LOC) or ranking for Volume deviate without good motivation	-1.0
The metric value (%) or ranking for Duplication deviate without good motivation	-1.0
The risk profile or ranking for Unit Size deviate without good motivation	-1.0
The risk profile or ranking for Unit Complexity deviate without good motivation	-1.0
The scores calculated for the maintainability aspects deviate without good motivation	-0.5
Your report critically reflects on the implementation of the metrics, discusses design choices and possible alternatives	-1.0 to +1.0
Your tool produces output that makes it easy to reproduce and verify the (intermediate and overall) results of your analysis	-0.5 to +0.5
You have implemented Test Quality and Stability and can argue the correctness of your implementation	+0.5
Your tool scales to larger projects regarding running times as demonstrated by an analysis of the algorithmic complexity of your algorithms and/or through an empirical analysis of running the tool on projects of various sizing (including smallsql and hsqldb) in the report	+1.0
Your code is well-structured, modular, separates concerns, has automated tests and is easy to maintain	-0.5 to +1.0
You have found another metric in the literature that is not in the SIG Maintainability Model, can argue why and how it would improve the results, and implemented the metric.	+1.0

Table 7: Grading Conditions and Scoring for Series 1

Furthermore, your solution has a sensible design and code implementation. In your report you explain and motivate how your solution reads the Java code and calculates the metrics, the rankings, reflects on (additional) metrics and threats to validity. Your implementation should be runnable when submitted. To demonstrate scalability empirically, also submit a *script* that makes it possible to reproduce your experiment(s) without effort. Table 7 shows conditions and how they modify the grade (the teachers have a reference implementation that provides outputs for comparison).

Deadline

The deadline for handing in your submission is given on canvas.