Lab assignment

1 Introduction

This is the lab assignment for the Software Quality Management course. The aim of the assignment is to develop a maintainability model using a number of software metrics. The model will be used to analyse the maintainability of two software systems, which have both been written in Java but differ in size. Next you have to apply visualisation techniques aimed at gaining more insight into a software system. The visualisations must be validated using a number of aesthetic and usability criteria selected by yourself.

1.1 LEARNING GOALS

After completing this assignment, you will be expected to be able to

- extract facts and draw conclusions from these facts in order to analyse the quality and structure of an existing system.
- weigh up the advantages and disadvantages of software metrics when determining the quality of a product.
- apply software metrics to an existing system.
- visualise data regarding software artefacts and assess these visualisations using Tufte's principles of graphical integrity and Shneiderman's interaction principles.

1.2 STUDY GUIDE

This assignment will be carried out in pairs. Individuals can only carry out the lab assignment by themselves with the permission of the examiner. Before you begin, familiarise yourself with Rascal by completing the introductory exercises and thoroughly review the article about the maintainability model by Heitlager, Kuipers, and Visser, the articles about visualisation techniques by Heer, Bostock and Ogievetsky, as well as the article about the CodeCrawler software visualisation tool by Lanza and Ducasse.

1.3 SOFTWARE

We will be using the domain-specific programming language Rascal for this assignment. Visit the Rascal page for information on installing Rascal in the Eclipse environment: `http://www.rascal-mpl.org/`.

1.4 VERSION CONTROL

In order to develop good software, version control systems are indispensible. Therefore, in this assignment we will use GIT to manage the developed Rascal code.

1.5 SUBMISSION

Submit your work via the digital learning environment. Clearly state your name, student ID number and the version number of the assignment.
The interim submission for part 1 consists of the computed metrics (in text format), as detailed below.
The main submission will consist of the Rascal code and a report (in PDF format). Please ensure you book the appointment with the examiner to explain your assignment orally in good time. It is possible to make your appointment before the work is submitted. Just be aware that the code and your report must be in the possession of the examiner at least 24 hours prior to the appointment.

2 Computing software metrics

Organisations including the Software Improvement Group (SIG, `http://www.sig.eu/`) use software metrics to quickly obtain an overview of the quality of a software system and identify areas that might be difficult to maintain. A number of relevant questions with regard to the use of metrics are:

1. What metrics will you use?
2. How are these metrics calculated?
3. How closely do these metrics represent the information you really want to obtain about the system and how do you establish this?
4. How can you improve on the above aspects?

The maintainability model developed by the SIG answers the first question. You can find more information about the model in this article:

> I. Heitlager, T. Kuipers, and J. Visser. A practical model for measuring maintainability. In Proceedings of the 6th International Conference on Quality of Information and Communications Technology, QUATIC '07, pages 30–39, Washington, DC, USA, 2007. IEEE Computer Society.

The second question ('How are these metrics calculated?') forms the subject of this first part of the lab assignment. The other questions could form interesting starting points for further research.

2.1 ASSIGNMENT

Write a Rascal program that implements the SIG maintainability model. It must be possible to use this program to analyse systems written in Java. The unit test coverage metric is optional. Implementation of the four other metrics, however, is mandatory:

- volume
- size of each unit
- complexity of each unit
- duplication

The metrics must be applied to the following Java systems:

- SmallSQL, a small-scale system. In order to pass the assignment, you must determine the metrics for this system as a minimum. The source code can be downloaded from SourceForge[1]:
  `https://sourceforge.net/projects/smallsql/files/`
- HyperSQL Database Engine (HSQLDB), a more extensive system. You are not required to calculate the metrics for this system, but doing so will result in a higher mark. Again, the source code can be downloaded from SourceForge:
  `https://sourceforge.net/projects/hsqldb/files/`

2.2 HANDING IN

You must hand in the result of this part of the assignment, the calculated metrics, via yOUlearn. Please use the following format:

```
ExampleSQL
----
lines of code: 123456
number of units: 12345
unit size:
  * simple: 60%
  * moderate: 10%
  * high: 10%
  * very high: 20%
unit complexity:
```

[1] Use the 'Files' tab to download the code, selecting the latest version

```
  * simple: 70%
  * moderate: 5%
  * high: 15%
  * very high: 10%
duplication: 42%

volume score: ++
unit size score: +
unit complexity score: +/-
duplication score: --

analysability score: +
changability score: -
testability score: ++

overall maintainability score: ++
```

Make one text file for SmallSQL and one for HSQLDB. You can include more information than what is shown in this example.

HINT

In order to calculate multiple metrics, you must first load all the Java methods from a project. The M3 library in Rascal offers an easy way to achieve this. The code snippet below shows how to print all the methods in a project:

```
module PrintMethods

import lang::java::jdt::m3::Core;
import IO;

public void printMethods(loc project) {
  M3 model = createM3FromEclipseProject(project);
  for (loc l <- methods(model)) {
    str s = readFile(l);
    println("=== <l> ===\n<s>");
  }
}
```

The method `methods` returns all the method declarations in a M3 model. This yields a collection of locations. Such a location can then be passed as a parameter to `readFile`, which will read only part of a file. Alternatively, a location can be converted into an abstract syntax tree (refer to the `lang::java::jdt::m3::AST` module). Test how the method works by calling:

```
printMethods(|project://JabberPoint|);
```

The introductory exercises contain more examples of using the M3 model.

3 Visualising metrics

The metrics that you have calculated in the previous part are not easy to interpret just by looking at the numbers. In this second part of the lab assignment we will visualise the metrics as effectively as possible, in order to obtain new insights into the structure and quality of the system. Typical questions that deserve an answer in this regard include:

- What does the global structure (architecture) of the system look like?
- What interdependencies exist between the components of the system?
- What are the calculated metrics for these components?
- How do the metrics impact on the architecture?

For this assignment, you are free to choose the information to be included in the visualisation. It is up to you to assess what can reasonably be expected within the available time. If you are unsure, contact the examiner.

3.1 ASSIGNMENT

Complete the assignment by carrying out the four steps listed below:

- Choose at least one architectural view on which to base your visualisation. This can be at a high level (a coarse-grained view) or at a more detailed level (fine-grained view). Outline the reasons for your choice.
- Use the chosen architectural view to visualise (a selection of) the software metrics from the first assignment. Use appropriate visual attributes (including colour, shadow and positioning) and any interactive features available in Rascal (such as popups, mouse events and sliders).
- Assess the developed visualisation according to design principles from the scientific literature. An example would be the work of Edward Tufte or Ben Shneiderman. You could also have a look at the article by Petre and De Quincey, in which they list a number of features of high-quality software visualisations. Remember to reference your sources!

4 Report

As part of this assignment, you must write a short report in which you outline your design choices and results. This report will be used as the basis for the oral follow-up discussion. The report must address the following topics as a minimum:

a **Design**
- *Assumptions.* What assumptions did you make with regard to the implementation of the metrics, for instance due to ambiguities in the specification? How will the results be impacted by these assumptions? What principles did you apply during the implementation?
- *Design of visualisation.* Briefly describe the design of the visualisation, and your motivation behind the choices you made there.

b **Results**
- *Metrics.* The resulting metrics for the software systems. This could be the output of the program. Please be aware that the Rascal program you submit will not be executed. This means you will need to report on and submit the key outcomes.
- *Interpretation.* What do the results say about the maintainability of the system(s) you reviewed? Base your analysis on the scores from the SIG maintainability model and on the developed visualisation. Can you identify any risk areas? Does the visualisation provide new insights into the maintainability?

c **Evaluation and reflection**
- *Validity.* How accurate are the results you found and how have you validated the accuracy?
- *Evaluation of visualisation.* Critically assess the developed visualisation, mention strength and weaknesses.
- *Cooperation.* Briefly describe how you worked together as a team and how you divided up the work. How have you used version control within your team?

Take a number of screenshots of the visualisation and include them along with the report.
The length of your report should be equivalent to 3 text-filled pages, excluding program output, figures and tables.

5 Assessment

Assessment of this assignment will be based on submission of the results of the maintainability model, the presentation of the visualisation and the Rascal code, as well as an explanation during a brief interactive session.
The assessment will focus on the following aspects:

- To what extent has the SIG model been implemented? Incomplete implementations of the model may still be sufficient to obtain a pass mark, but this will be reflected in the mark. Calculation of the unit test coverage is an optional element.
- Do the results match those of SIG? You must be able to explain any discrepancies.
- How well are the metrics from the first part of the assignment visualised?
- Have well-informed criteria been used for the design of the visualisation, and is evidence given that establishes that these criteria are fulfilled?
- How straightforward is the implementation? Additional points will be awarded to programs that are elegant, simple and easy to understand and efficient.
- How was version control used for development of the implementation?
- Is the visualisation exceptionally creative or are additional metrics explored and shown in interesting ways?

Please also see the assignment rubric for the exact assessment criteria.