Assignment 1: Software metrics

INTRODUCTION

This is the first lab assignment for the Software Evolution 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.

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

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, thoroughly review the article about the maintainability model and familiarise yourself with Rascal by completing a number of the exercises. The learning environment includes an FAQ containing questions and answers about Rascal.

Software

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

Submission

Submit your work via the digital learning environment. Clearly state your name, student ID number and the version number of the assignment. The submission will consist of the Rascal code and a report (in PDF or Word 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 assignment.

Assignment 1: Software metrics

You will use the first lab assignment for the Software Evolution course to study 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 lab assignment. The other questions could form interesting starting points for further research.

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
– complexity of each unit
– duplication
– size of each unit

The metrics must be applied to the following Java systems:

– SmallSQL, a small-scale system. In order to pass this assignment, you must determine the metrics for this system as a minimum. The source code can be downloaded from SourceForge[1]:
  http://sourceforge.net/projects/smallsql/

– 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:
  http://sourceforge.net/projects/hsqldb/

Report

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

– *Cooperation*. Briefly describe how you worked together as a team and how you divided up the work.
– *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?
– *Results*. 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.
– *Validity*. How accurate are the results you found and how have you validated the accuracy?
– *Interpretation*. What do the results say about the maintainability of the system you reviewed? What are the scores at system level for the four sub-characteristics of maintainability? Can you identify any risk areas?

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

The length of your report should be equivalent to two text-filled pages, excluding program output, figures and tables.

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 ToonMethoden

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 printMethods 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: toonMethoden(|project://JabberPoint|);.

Assessment
Assessment of this assignment will be based on submission of the results of the maintainability model 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 straightforward is the implementation? Additional points will be awarded to programs that are elegant, simple and easy to understand.
– Efficiency will not be considered.

Please also see the assignment requirements and criteria that can be accessed under ‘Study tasks’ in the learning environment, as well as this assignment’s rubric.