Curtin University — Department of Computing

Software Metrics (ISAD4002)

2016, semester 1

Assignment 1

Due: Week 7 – Friday 15 April, 4:00 pm

Weight: 20% of the unit mark.

This assignment requires you to access the compileable source code for a moderate-to-large software system of your choosing.

Once you have the code, your task is twofold:

(a) Analyse the either the performance (speed/timing) or the memory usage of any single operation of the software.
(b) Analyse the overall code quality.

The Software System

The system you use must have the following characteristics:

- It must contain at least 2000 lines of code, excluding blank and comment lines, and any HTML, CSS, XML and other non-programming code. You can measure the lines of code using the "`cloc`" command (cloc.sourceforge.net).
- It cannot be written (in whole or part) by you, or by anyone or any group associated with you.
- It cannot be ImageCafe (the software used in previous years for this purpose), or any derivation of ImageCafe.
- It must have functionality for processing an open-ended, potentially large amount of input data. (See below.)

To find such a system, consider browsing repositories of open-source software:

- projects.apache.org;
- code.google.com/hosting;
- sourceforge.net;
- github.com.

Note: you must have access to the source and compiled code, *but* you do not necessarily have to compile it yourself. (Compiling can occasionally be tricky with large programs.)

Task A: Performance/Memory Analysis

You may choose to analyse EITHER (1) the performance (CPU usage) of the software, OR (2) the memory usage of the software.

- Identify one functional requirement of the software. It must involve processing a *potentially large* amount of input data.

  There should be no specific upper limit on the amount of data allowed to be processed (except as imposed by the OS or the machine architecture). "Processing" could involve reading, writing, sending, receiving, transforming, summarising, etc.

  Note: real-time operation like "playing music" or "recording video" present a more subtle problem for performance analysis. They are not designed to run "as fast as possible", so analysing their performance could require more care.

- Use a profiler to determine how the CPU or memory usage of the operation (i.e. the time taken) is affected by the size of the data.

  Notes:

  - You will need to construct a dataset, comprising data of increasing sizes, in order to perform the analysis.
  - "Size" can be measured in many ways, depending on the nature of the data. Choose a size metric that makes sense for the data being analysed.
  - The relationship between size and CPU/memory usage, when you determine it, should be expressed as a mathematical equation.

- Find the parts of the software (methods and classes) most directly responsible for the time or memory required to complete the operation.

  Investigate why they take the most time.

- Report your analysis, making good use of quantitative evidence, in the form of numbers, graphs and equations. Explain and justify the choices you make. Comment on the feasibility of improving performance.

  All the raw timing/memory usage data on which your analysis is based should be included as a table in your report.

  Explain how your dataset was constructed. Provide enough detail to allow someone to re-create your dataset (or at least something similar enough that it would achieve the same results). You do not need to provide the dataset itself, but you may do so if practical.

Task B: Quality Analysis

- Identify the three most important problems with the code's readability and maintainability, taking into account the complete source code of the software.
  - You can use RSM, PMD, PyLint or any other such static analysis tool appropriate for the language used.
  - "Most important" is subjective, but you should be able to justify your choices.
  - Some software may not have any huge quality issues, but it should nonetheless be possible to find things that could be improved.
- For each identified problem:
  - Demonstrate that it is actually one of the three *most important* problems (and not just *a* problem);
  - Identify and measure a metric that best illustrates it;
  - Determine how it could be rectified without creating additional problems.
- Report your analysis, making good use of examples and quantitative evidence. Explain and justify the choices you make.

Note: when identifying a metric that illustrates each problem, you may invent your own metric, or you may suggest a previously-proposed metric. In either case, the metric should be justifiable. The idea is to quantify the problem – find a way of assigning a number to it. A certain amount of creativity may be called for.

Report

Include all your discussion, results and analyses in a report. Your report should:

- Have normal spacing and a sensible layout.
- Be logically structured, using headings where appropriate.
- Use good English, with proper spelling and grammar.
- Comprehensively demonstrate that you know what you are doing!

Any source code segments included in your report should be easy to read, and visually separated from any surrounding text.

```
forExample("I like to put code in a box, with a fixed-width font (and maybe
    syntax highlighting, but don't stress over it).");
```

Note: don't make your report enormous and unreadable by including the complete source code, complete profiler output, etc. These things can be submitted *alongside* the report. See the following section for details.

Submission

Submit the following electronically to the Assignment 1 area on Blackboard:

- A completed Declaration of Originality (whether photographed, scanned or electronically filled out);
- Your report (in .pdf format);
- The complete source code of your chosen system (*not* just a link to it);
- A sample of the raw profiler output;
- A sample of the output of any tool(s) used in your quality analysis.

After submitting, please verify that your submission worked by downloading your submitted files and comparing them to the originals.

See the unit outline for the policy on late submissions. Additionally, if your Declaration of Originality is missing, incomplete, or in some way erroneous, your assignment will not be marked.

Marking Guidelines

The performance/memory analysis will be worth approximately twice as many marks as the quality analysis.

Marks will be based on (among other things):

- Carefully explaining *why* you made certain decisions.
- Your attention to detail.
- Your reliance on hard evidence.
- Demonstrating depth and rigour in your analysis.

Academic Integrity

Please see Curtin's Academic Integrity website (and the unit outline) for information on plagiarism and academic misconduct.

The unit coordinator may require you to provide an oral justification of, or to answer questions about, any piece of written work submitted in this unit. Your response(s) may be referred to as evidence in an Academic Misconduct inquiry. In addition, your assignment submission may be analysed by Turnitin and/or other systems to detect plagiarism and/or collusion.

End of Assignment 1