Individual Project

Re-engineering Apache Commons BCEL

Version 1.0.0 (30/4/2021)

1. Overview

BCEL is a long-standing software framework to analyse Java class files. It is similar in nature to the ASM framework we've encountered on this module; it enables decompilation and direct manipulation of class files.

It is truly a legacy system. It has existed for over 20 years. Over this time it has had to constantly evolve to accommodate changes in Java language specifications and compilers, to add features, and to keep up with the advances of its competitors (such as ASM).

This assessment is to be carried out on an individual basis. It is important that you do not discuss your solutions to this with group members.

2. Setup

Log in to GitLab, and create a "fork" of the following project:

https://stugitlab.dcs.shef.ac.uk/courses/com3523/2020-2021/individual-project/com3523-2020-21-individual-project

This contains three directories: A clone of Apache BCEL, a clone of the Reengineering Toolkit, and an almost empty directory called "submission".

BCEL and Reengineering Toolkit have been stripped of their specific git metadata, to avoid any confusion.

You should use the commons-bcel source code as the basis for your analysis and re-engineering. This will be where you'll commit your changes (see instructions below).

You should use the tools in the reengineering-toolkit directory to carry out your analysis (see instructions below). You are welcome to make your own commits that enhance this tooling. I have included some pull-requests from a couple of you, who found ways in which to improve the tool during the group projects (*many thanks to those of you who proposed them!).*

Finally, the submission directory is where you will include your written submission, and any additional data or figures that you'd like to include.

3. Instructions

For this assignment, the goal is to put all of the skills that we have covered throughout the course into practice. The main assessed part of the work will be a write-up, but figures and committed code will be required as well. The report will be split into two parts:

1. **Analysis of the system**

You should find out what you can about the system. What is the design? What are the important elements (classes or methods)? Which classes or methods are particularly important for the functioning of the system? Which ones are particularly unusual?

You must highlight three (potentially) substantive design weaknesses, and back them up with appropriate evidence. This evidence must be obtained using only the analysis techniques that we have covered in our lab work. Your discussion of the three approaches must at some point involve the application of:

- Bash-script based analysis (this can include an analysis of the version repository).
- Class diagram analysis.
- Call graph analysis (this can include fan-in / fan-out metrics - see below)
- Code clone analysis.
- The computation of appropriate metrics.
- Dynamic analysis.

Feel free to modify and enhance the techniques that we have covered in the labs if they help you to accentuate something within the system.

You do not need to apply all three approaches to all three design weaknesses that you identify.

Your write-up *must* refer to the concepts that were covered in the lectures. One of the objectives is to demonstrate that you have followed and understood the lecture material *as well as* the practical activities. For example, you are expected to refer to design principles, and relate these to the evidence that you have collected using the above techniques.

If you analyse the version repository for BCEL, you will need to clone the original repository into a separate directory, and analyse it from there (we have stripped out the Git meta-data so that it would not conflict with the Git data for the assignment as a whole). For this you should run:

```
git clone https://github.com/apache/commons-bcel.git
git checkout 50a023e
```

Approximate word-length for COM3523: 1000 words.
Approximate word-length for COM6523: 1500 words.

The word count is meant as a guide - you can fall within 100 words below and 300 above the limit.

2. Reengineering of the system

Pick one weakness in the system (that you identified in the previous section). It should be a weakness that is non-trivial in nature — not something that can be fixed by merely extracting a method, for example. It should be something more significant, such as a God Class, extensive coupling between one or more classes, a lack of cohesion, etc.

Describe the strategy that you would apply to address the weakness. Use class diagrams to illustrate your re-engineering approach (the diagrams only need to contain the classes and methods that are specifically relevant to the reengineering effort). Take care to mention any risks that might arise (e.g. where a re-engineering step might lead to a deterioration in a different aspect of the design).

Set up a test set. Do not use existing test cases. Commit your test cases to the version repository. In your write-up, describe how you generated your test set. Describe any problems you encountered.

Apply your re-engineering strategy to the code base. Do so via a series of commits to the version repository. Describe any problems you encountered.

For some objectives, completely eliminating the design weakness may require more time and effort than is feasible for this assignment. Your submission will be assessed in terms of (a) the techniques that you have successfully demonstrated, (b) their appropriateness. So you can choose an ambitious reengineering task, but do not have to follow it through to completion, just as long as you demonstrate the key steps involved.

For example, for splitting up a god class, you could create a wrapper, and then show how you'd split up the class in one or two ways without fully eliminating the god class as a design problem.

For all commits that are related to the re-engineering of the source code (and setting up the test set) , add "REENG: " to the beginning of the commit message.

Approximate word-length for COM3523: 500 words.
Approximate word-length for COM6523: 700 words.

The word count is meant as a guide - you can fall within 100 words below and 300 above the limit.

Assessment

The assessment of your submission will be split into three parts: Analysis of the system (40%), and Reengineering the system (40%). The remaining 20% will be based upon an all-round assessment of the submission, analysing the coherence between the different sections, the presentation, innovation, insightfulness, the value and quality of any changes to the source code, etc.

Section	0-39%	40%-54%	55%-70%	70%-100%	Weighting
Analysis of the system	Incorrect use of techniques, poor presentation of outputs, no insights into the system.	Some basic commands correctly applied, poor presentation of results, few insights into the system. Lack of reference to the concepts and materials covered within the lectures.	Extensive application of techniques, well explained in documentation, with extensive presentation of the results, demonstrating insights. Highlighting key points highlighted within the lectures.	Extensive application of the techniques, along with some novel enhancements. All justified and well explained in the documentation, demonstrating insights. Extensively refers to concepts discussed in the lectures.	40%

Section	0-39%	40%-54%	55%-70%	70%-100%	Weighting
Reengineering the System	There is no sensible description of a reengineering strategy, and there are no commits to the source code that amount to a substantive reengineering effort.	There is a sensible description of a reengineering strategy, which is linked to specific aspects of the source code. There are some efforts to fulfil this description in the source code.	There is a sensible description of a reengineering strategy, linked to specific aspects of the source code. Most of the key steps in this strategy have been sensibly applied and are evidenced by commits to the source code.	There is a sensible description of a reengineering strategy, linked to specific aspects of the source code. All of the key steps in this strategy have been sensibly applied and are evidenced by commits to the source code. The write-up also includes evidence to show that the resulting system has been improved by the reengineering efforts.	40%

Submission

Your submission material should be placed inside the directory called “submission”, in the root directory of your Git repository [the other two directories in the root directory should be “commons-bcel” and “reengineering-toolkit”].

Your submission will be your GitLab repository (specifically, the final commit you make to it before the deadline). Your report, containing all of your written work, should be placed in the `submission’ directory as a PDF (this is strict - please do not submit other formats, such as Word documents). This should include figures, tables, and charts where appropriate.

Please ensure that the front page of your report contains your university user name (not your registration number, but ID that you use to log into lab computers, e.g. mine is ‘ac1nw’).

Please place any data files or analysis outputs (e.g. CSV files, images, etc.) that you have produced into your `submission’ folder. If the files are particularly large (>3MB, such as trace files), these should be compressed first, but may also be left out.

Please do not compress your PDF file. If you do need to use compression, please use the ZIP format.

There will not be a submission-point on Blackboard, and submissions via email will not be accepted.

The deadline for the submission is 18:00 GMT, on the 14th of May 2021.

Support

To ensure fairness, we will only respond to queries on the Blackboard forum dedicated to this assignment. As usual, please read other queries before you post your own, to ensure that your query has not already been answered. To assist your colleagues when they are checking the forum, please give your query a descriptive title.

You must under no circumstances post your own solutions (or parts thereof) to the forum - so please be mindful when you are posing questions.

Unfair Means

It is important to bear in mind the departmental rules on unfair means. Activities such as plagiarism or collusion will be treated as a serious academic offence. This could lead to the award of a grade of zero for this assignment. Since this accounts for 80% of the module mark, this would automatically result in a failure of the module, with severe implications for the possibility of obtaining a degree.

We will be closely scrutinising submissions to detect such practices, because it is important that this assessment is a genuine reflection of your own understanding of the module.

To avoid any potential accidental wrongdoings, it is especially important that you do not discuss your solutions with other students. If you have questions about this assignment, please use the discussion forum.