Object Oriented Software Engineering
Workshop 1
Software Metrics

Objectives

- To generate control flow graphs and understand how they are used to determine the complexity of program code.
- Understand how software metrics can be used to measure and improve the quality of software.

You will achieve these objectives by completing a set of tasks.

Setup 1

Download vcs.zip from OOSE Moodle page and unzip it to a folder of your choice. Create a Java eclipse project and name it `VehicleControlSystem`. Copy `oose.vcs`, `vehicle.types` and `img` folder from unzipped content into the `src` and project directories in Eclipse respectively. Your eclipse directory structure should be similar to Figure 1. To run the project, right click `Controller.java` and run as Java application. A class

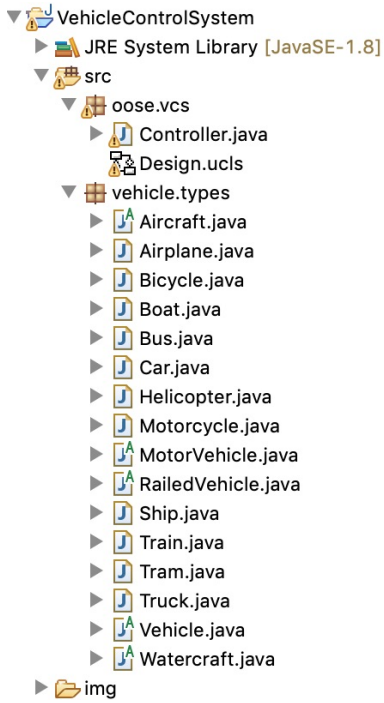

Figure 1: Project directory structure for Vehicle Control System on Eclipse

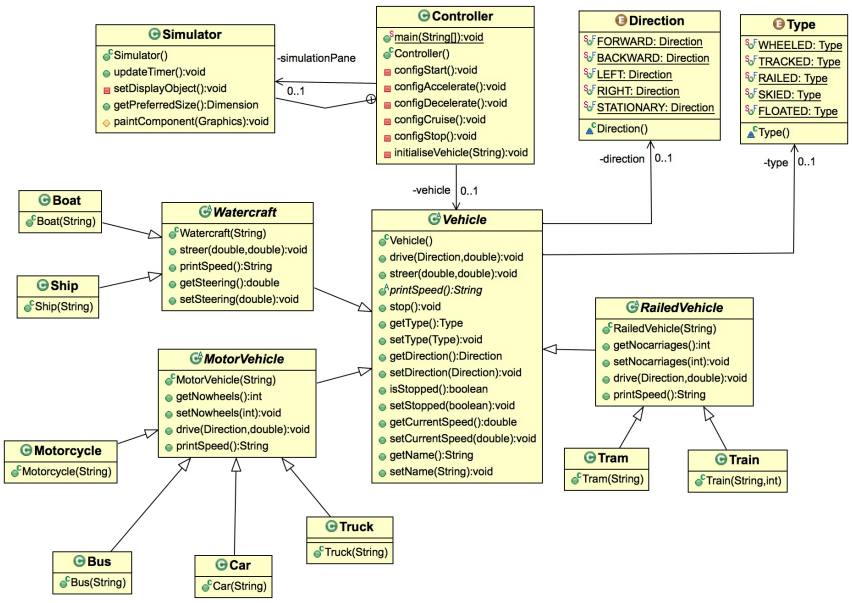

Figure 2: Design Diagram for a Vehicle Control System

diagram of the system design is as shown in Figure 2. You can also access this diagram by clicking `Design.ucls` in `oose.vcs` package. But you would have installed ObjectAid UML Explorer plugin by following instructions from http://www.objectaid.com/class-diagram.

Setup 2

Download `CKMetrics.jar` from OOSE Moodle page to a folder of your choice. `CKMetrics` is a tool for determining Chidamber and Kemerer (CK) metrics suite for java class files and Jar APIs. `CKMetrics` operates by using Apache Byte Code Engineering Library (BCEL) to analyse class files that are contained in a jar file. Assume you have `A.jar` located in your local drive ../`JarLocation`. To determine the quality of `A` using `CKMetrics`:

1. Select `Load File` button from the toolbar. You can either browse and select the folder containing `A` or select `A`. For the former, `CKMetrics` will load all the jar files contained in the folder.

2. Then select `Compute` button from the toolbar. `CKMetrics` will compute the mean CK metrics value for each selected jar file.

3. Optionally, you can include jdk class files in the analysis, or only analyse publicly accessible classes, fields and methods by checking `includeJdk` and `onlyPublic` checkboxes respectively.

Figure 3 is a screenshot showing `CKMetrics`'s output. You can also select a .class file following above steps to analyse java classes.

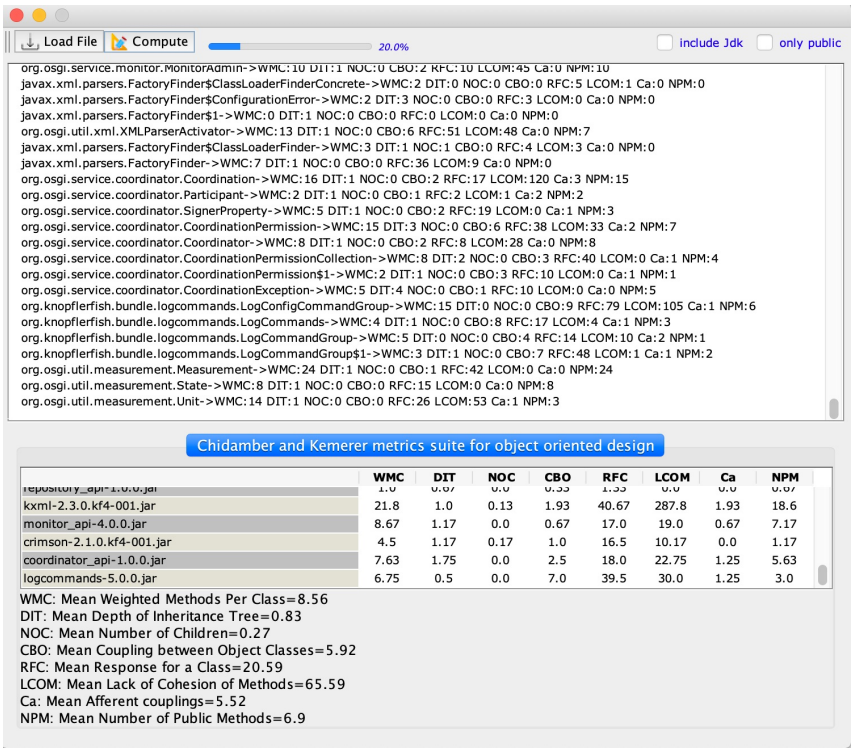

Figure 3: Analysis output from `CKMetrics`

Finally, to build the jar file required to check the quality of `Vehicle Control System` based on above steps:

1. Right click on the project in Eclipse.
2. Select export.
3. Export as runnable jar file.

Tasks

1. Draw the control flow graph for the method `initialiseVehicle` in `Controller.java`. **(1 Mark)**
2. Assume the objective is to increase encapsulation, modularity and reduce the complexity of `VehicleControlSystem`. Identify 3 metrics that can be used to measure these quality factors in `VehicleControlSystem`. **(1 Mark)**
3. Identify the most important class in `VehicleControlSystem` that should be refactored to enhance the quality factors in Task 2. Justify your choice in a maximum of 200 words. **(1 Mark)**

4. Propose and implement four refactorings for the class identified in Task 3 to enhance quality factors mentioned in Task 2. In maximum of 200 words, discuss the extent suggested refactorings enhances quality factors mentioned in Task 2. **(1 Mark)**

5. Draw a new class diagram for `VehicleControlSystem` after implementing suggested refactorings in Task 4. In maximum of 200 words, discuss any structural difference in the design of `VehicleControlSystem` after implementation. **(1 Mark)**

Deliverables

1. Turn in a pdf document electronically via Moodle for Workshop 1. You should clearly state at the top of the document your name and registration number. The document should also contain your **solution to tasks 1- 5** with relevant **screen-shots of CKMetrics** and **class diagrams**.

2. A source folder for `Vehicle Control System` containing the implementation of your refactoring tasks.

3. A runnable jar file of the `Vehicle Control System` after implementing your proposed refactoring tasks.

Assessment

Submissions is due by **16.30** on **5th February 2019**. You should submit your solution on Moodle in the appropriate upload slot for the laboratory.

Tutors and demonstrators will be in your allocated Lab to offer assistance. **Endeavour to attend your labs to discuss issues rather than sending emails**. Its likely that you will receive no response by email or response may be slow.

As per the Code of Assessment policy regarding late submissions, submissions will be accepted for up to 5 working days beyond due date. Any late submissions will be marked as if submitted on time, but reduced by 1 mark for each additional day. Submissions received more than 5 working days after the due date will receive an H (band value of 0).

Dr Inah Omoronyia, S152, School of Computing Science, University of Glasgow