Assignments

Lab Series 0

Introduction

In this lab you learn the basic facts about Rascal and quickly learn to program in it. The idea is to interact with Rascal using the RascalTutor environment and Eclipse. See http://www.rascal-mpl.org for download and installation instructions.

Once you have started the Rascal/Eclipse environment, you can activate the RascalTutor by selecting the menu Rascal > Start Tutor. The non-interactive tutor can also be found at http://tutor.rascal-mpl.org.

- Comments on the documentation can be given at the bottom of each page
- Discussion and questions can be posted at the online forum via http://www.rascal-mpl.org
- Bugs can be submitted via https://github.com/cwi-swat/rascal/issues

Collaboration

Please do these exercises individually. Any communication between students is allowed, but bear in mind that you should personally be capable to code in Rascal after this. Your Rascal knowledge will also be validated in a personal online test.

Assignment

Fully explore the Rascal course using the tutor and teach yourself Rascal. You will be assisted in the laboratory to install the system and type your first expressions and statements. Please ask the teachers any question about Rascal or the exercises you might have. It will be hard work!

Grading

This series is not graded. If you pass most of the exercises in the tutor, you will know enough Rascal to do series 1 and 2. However, **2 November** you will do a personal online test. You have to pass more than 50% of the questions of this test, to be able to pass for the entire course.

Deadline

You should finish Series 0 in the second week of the course.

Lab Series 1

Introduction

In series 1 we focus on software metrics. Software metrics are used (for example) by the Software Improvement Group (http://www.sig.eu) to quickly gain an overview of the quality of software systems and to pinpoint problem areas that may cause low maintainability.

Some relevant questions are:

1. Which metrics are used?
2. How are these metrics computed?
3. How well do these metrics indicate what we really want to know about these systems and how can we judge that?
4. How can we improve any of the above?

In other words, you have to worry about motivation and interpretation of metrics, as well as correct implementation.

The SIG Maintainability Model provides an answer to question 1. You can read about it here: Ilja Heitlager, Tobias Kuipers, and Joost Visser, A Practical Model for Measuring Maintainability, In proceedings of the 6th International Conference on the Quality of Information and Communications Technology (QUATIC 2007), pages 30-39, IEEE Computer Society Press, 2007, http://dx.doi.org/10.1109/QUATIC.2007.8

The second question above (“How are these metrics computed?”) is your assignment for series 1. The third and fourth questions will be addressed during the grading session.

Collaboration

Please make groups of two students. You can work together as a pair on all aspects of this assignment.

You can brainstorm with anybody else about the contents of your essay, but for this assignment you are not allowed to look at code from other groups or exchange solutions in detail with other groups.

Assignment

Using Rascal, design and build a tool that calculates the SIG maintainability model scores for a Java project.

Calculate at least the following metrics:

- Volume,
- Unit Size,
- Unit Complexity,
- Duplication.

For all metrics you calculate the actual metric values, for Unit Size and Unit Complexity you additionally calculate a risk profile, and finally each metric gets a score based on the SIG model (--, -, o, +, ++).

Calculate scores for at least the following maintainability aspects based on the SIG model:

- Maintainability (overall),
- Analysability,
- Changeability,
- Testability.

You can earn bonus points by also implementing the Test Quality metric and a score for the Stability maintainability aspect.

Use this zip file to obtain compilable versions of two Java systems (smallsql and hsqldb): zip file

- **smallsql** is a small system to use for experimentation and testing. Import as-is into Eclipse and ignore build errors.
- **hsqldb** is a larger system to demonstrate scalability. Import into Eclipse, make sure to have only hsqldb/src on the build path, and add the following external jars from your eclipse/plugins/ directory: javax.servlet_$VERSION.jar and org.apache.ant_$VERSION/lib/ant.jar

Hints

Create an Eclipse Java project with example files to test you solution on (using the Rascal test functionality).

Create an Eclipse Java project for each of the two systems, smallsql and hsqldb too. Some few lines of code will still not compile, but commenting them out would not change the metrics too much. So commenting out just a few lines is ok in this case. It saves time!

Grading

The assignment is judged by demonstrating your results and your code to us in a small interactive session. At the end of this session you will immediately get a grade between 3 and 10.

You also have to drop a zip file with all code and relevant document in the assignments. The files are checked for plagiarism automatically. If you worked in a team of two, drop your assignment twice: one for each student!

You will be graded using the following model:

The base grade is 7. For this grade you need an implementation that conforms to the assignment described above. The implementation consists of sensible design and code. You can explain and motivate how it actually reads in the Java code and calculates the metrics based on that. Your implementation can be run during the grading session on at least the smallsql project. *For grading, import the smallsql project into Eclipse as-is and ignore the 100 or so build errors.*

The following conditions modify the grade (the teachers have a reference implementation that provides the correct outputs):

Condition	Base grade modification
The metric value (total LOC) and/or score for Volume deviate without good motivation	-0.5 to -1.0
The metric value (%) and/or score for Duplication deviate without good motivation	-0.5 to -1.0
The risk profile and/or score for Unit Size deviate without good motiviation	-0.5 to -1.0
The risk profile and/or score for Unit Complexity deviate without good motivation	-0.5 to -1.0
The scores calculated for the maintainability aspects deviate without good motivation	-0.5
Your tool produces output that allows easy verification of the correctness of the result (metric values, risk profiles, scores, etc. are neatly listed next to each other)	+0.5
You also implemented Test Quality and Stability and can argue their correctness	+0.5
Your tool produces correct output for hsqldb within the time span of the grading session (~	

30 minutes), possibly with the clone detection turned off	+0.5
You can demonstrate that your own code is of high maintainability and has proper automated tests	+0.5
You have found another metric in the literature that is not in the SIG Maintainability Model, and you can argument why and how would it improve the results	+0.5 to +1.0

Deadline

The deadline for series 1 is course week 4, i.e., 17 November at the latest.

Link to assignment submission is below: