

CSSE2003

Software Engineering Studio

Tutorial Week 7

Semester 2, 2009

School of Information Technology and Electrical Engineering
The University of Queensland

Goals

1. Understand some Java software analysis tools and know how to run them:
 - a. PMD (including its copy/paste detector)
 - b. FindBugs
 - c. JDepend
2. Demonstrate the ability to build and run your assignment program.
3. Collect (analysis) information on your assignment program.
4. Discuss choices arising from this information for assignments 2 and 3.

Rationale

You have chosen your target program and you are familiar with the essential tools necessary to work on it. As part of the upcoming assignment 2, you need to decide which parts of your program your team is going to study in detail. Currently, you do not have a lot of information about the quality of your program. Therefore we present a short exercise using free software analysis tools in this tutorial. Once you know how to apply these tools, you can use them to measure aspects of your assignment program. This can help you pick parts for detailed study. To ensure that your assignment is on track, your tutor will review your current state of development. Ideally by the end of this tutorial, but certainly before your next tutorial, you should have defined the parts of your program that your team is going to study in detail.

Part 1 - Use some Java software analysis tools

This part of the tutorial introduces the free software analysis tools: PMD, FindBugs and JDepend. Download the sample software and then each team member (teams of four can have two people working with the same tool) should perform one of the following tasks:

1. Analyse the software using PMD. Investigate three style bugs.
2. Analyse the software using FindBugs. Investigate two unsafe constructs.
3. Restructure the classes in the software into different packages. Analyse with JDepend.

Download the sample software

1. Download the sample software: www.itee.uq.edu.au/~csse2003/Tutorials/analysisSource.zip
2. Create a new Eclipse Java project and import the archive into the project.
3. Build the project using Eclipse.

4. The archive also contains an Ant build file called javadoc.xml file. This file contains a build target 'ydoc-impl', that builds implementation documentation. Adjust this file if necessary, and build the documentation. Refresh the project. You should now find a folder with the documentation.
5. Inspect the documentation by opening 'index.html'.
6. When all team members have reached this point, call the tutor for a team checkpoint. Show the documentation and explain what you find noteworthy about what you are seeing.

Option 1: Analyse the software using PMD

The PMD analyser is pre-installed in your Eclipse system.

1. Switch Eclipse to the PMD perspective.
2. On the icon of your project in the package explorer, unfold the context menu by right-clicking.
3. Select PMD → Check code with PMD. This will start a code check.
4. Inspect the contents of the 'Violations Overview' View.
5. Familiarise yourself with the connection between 'Violations Overview', 'Violations Outline', and the Java Source Editor.
6. Select three style bugs of different kinds and attempt to fix them.
7. Call the tutor for an individual checkpoint.
 - a. Describe the functionality of the PMD perspective.
 - b. Describe the selected bugs by showing the description and explain the impact if these bugs are not fixed.
8. Invoke the context menu again and select PMD → Generate Report. Refresh the project and inspect the report.
9. Invoke the context menu again and select PMD → Clear PMD violations. The reports should disappear.

Option 2: Analyse the software using FindBugs

The FindBugs analyser is pre-installed in your Eclipse system.

1. Switch Eclipse to the Java perspective.
2. On the icon of your project in the package explorer, unfold the context menu by right-clicking.
3. Select → Find Bugs. This will start a code check.
4. Wait until it reports '100 warnings'. Accept the default and switch to the perspective.
5. Inspect the contents of the 'Bug Explorer' View.
6. Familiarise yourself with the connection between the views 'Bug Explorer', 'Properties', 'Problems' and the Java Source Editor.
7. Select two unsafe practice bugs of different kinds and attempt to fix them.
8. Call the tutor for an individual checkpoint.
 - a. Describe the functionality of the FindBugs perspective.
 - b. Describe the selected bugs by showing the description and explain the impact if these bugs are not fixed.
9. Invoke the context menu again and select FindBugs → Clear Bug Markers. The reports should disappear.

Option 3: Restructure the software into different packages.

The JDepend analyser is pre-installed in your Eclipse system.

1. Switch Eclipse to the Java perspective.
2. Introduce two new packages: 'logic' and 'gui'.
3. Use the documentation to identify classes that depend on the program's user interface.
4. Move all those classes to the gui package.
5. Move all other classes to the logic package.
6. Clean the project.
7. Select the gui package.
8. On the icon of your project in the package explorer, unfold the context menu by right-clicking.
9. Select 'Run JDepend Analysis'.
10. Familiarize yourself with the connection between the views 'Packages', 'Dependencies', 'Problems'.
11. Call the tutor for an individual checkpoint.
 - a. Describe the functionality of the JDepend perspective.
 - b. Describe the dependencies using the data provided by the views.

Part 2 - Demonstrate building and running your program

By now, you should have familiarised yourself with the program you are working on and you should have attained the ability to build and run your target. To check that teams are on track, we will review your team's ability to do so. When all team members are ready, call your tutor for a team checkpoint.

Part 3 - Analysing your selected program

As a team, you should now be ready to decide which parts of your program you intend to study in detail. The goal of this tutorial step is to encourage you to assess your program with the analysis tools used in Part 1.

1. Collect (analysis) information on your program.
 - a. Run one or more analysis tools on your program.
 - b. If desired, inspect your program using Eclipse's built-in analysis features:
 - i. Type Hierarchy
 - ii. Call Hierarchy
 - iii. References
 - iv. Declarations
2. As a team, discuss choices arising from the information gathered.
3. If desired, consult your tutor.