Metrics Collector Software

Test Approach Document

<Version 1.0>

Murali Krishna
Vikram Patil
Vikas Nandanam

Table of Contents

1. Scope and Overview:

Metrics collector is an application which gathers process metrics from various sources such as running processes on the machine, elapsed time, memory used by the process etc. This application will also collect information about the CPU utilization by the various processes. Application will have a GUI to represent the collected information to the user. Software will be provided with a relational database which will store the metrics data and generate customized reports for the user.

1.1 Purpose:

The purpose of this document is to provide description about the test approaches that are going to be used and the estimated results for these tests. This document is used by the developers, project manager and testers to understand the robustness of the project.

2. Roles and responsibilities:

The project is modularized into different modules. Modules have been divided among the team members. Each team member does all the testing for every module they have been assigned for. The testing developers are performing is a white box testing. Later on every team member performs function, regression and integration testing.

3. Test approaches:

3.1 Unit testing:

Developers will unit test their own module or code through JUnit testing tool. Here a small piece of code is tested either manually or automated tests will be performed but we are using JUnit testing for every module of code.

3.2 GUI testing:

User interface testing is used to identify the presence of defects in software under the Graphical User Interface. We will do manual testing for GUI. Developers will also prepare dummy data to test UI and check whether UI handles data in different formats.

3.3 Performance Testing:

Performance testing is done to check the responsiveness of the application. With performance testing, throughput and scalability of the application can be figured out. The test is conducted at the last to check the performance, if test fails test the respective module of the code is fixed to enhance the performance of application. Performance testing will also be done for reading and writing of metrics data.

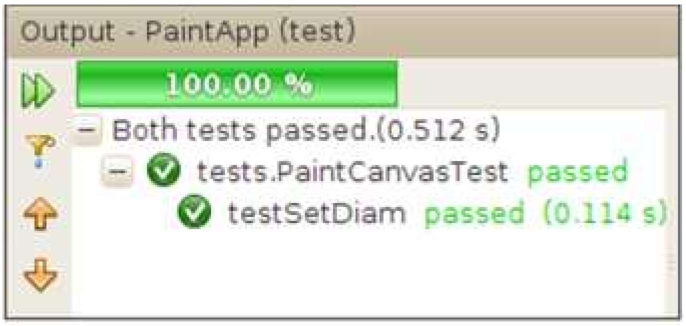
3.4 Regression Testing:

Regression testing makes sure that previously developed software works properly even after the change in the newly developed code. This testing is performed when there is any change in code. The estimated result is that any change in the code does not affect the functionality of the application. We are following an iterative waterfall model as SDLC.As soon as we find a bug in testing phase we iterate back to coding phase and fix the bug we perform regression testing in testing phase.

4. Testing Tool:

4.1 Test Framework –JUnit:

JUnit has become a standard for testing in Java programming language and is supported by almost all IDE's e.g Netbeans, Eclipse etc. So, you work in any standardized IDE environment, you would find the support of JUnit in it. If you are using an IDE for developing Java software, you can create and manage test cases with the help of JUnit wizards which would further enhance your productivity. You can execute a test case or a group of test cases by clicking on some options in the IDE and check the reports instantly using different color combination.
For example the test results are shown in a below figure

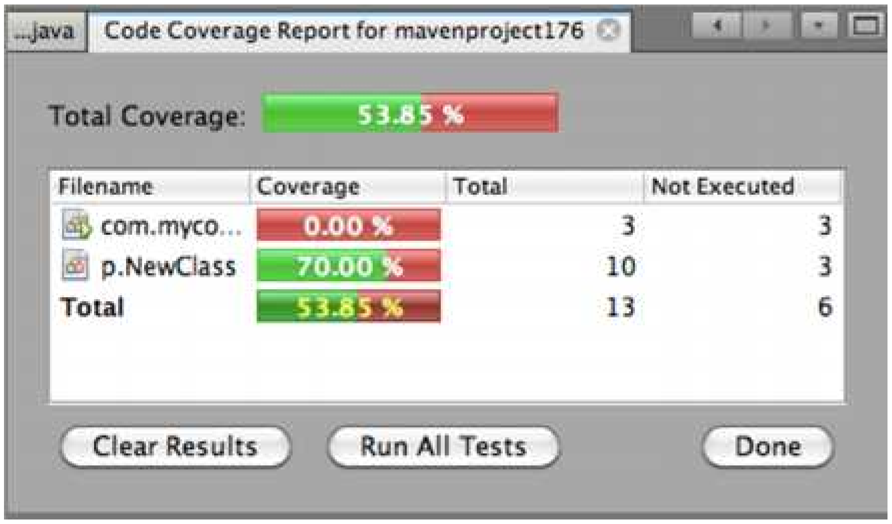


```
Output - PaintApp (test)
100.00 %
− Both tests passed.(0.512 s)
  − tests.PaintCanvasTest passed
      testSetDiam passed (0.114 s)
```

Fig: Showing the JUnit testing

4.2 Code Coverage Tool – TikiOne JaCoCo

The JaCoCoverage Plugin is a Netbeans plugin that enhances the existing NetBeans functionality with new code coverage features.
The plugin works as a transparent additional service that colors all java files according to the unit tests coverage information. With code coverage enabled user continues to work with his/her project in the usual way but can easily view the test coverage of the project classes. The code coverage plugin will update the code coverage data and refresh editors markup every time a unit test (or any selected Ant target) is executed for the project. This tool helps in knowing the percentage of code that has been tested.



Total Coverage: 53.85 %

Filename	Coverage	Total	Not Executed
com.myco...	0.00 %	3	3
p.NewClass	70.00 %	10	3
Total	53.85 %	13	6

Fig: Code coverage report in JaCoCo