University of Helsinki

Faculty of Science

Department of Computer Science

TestMyCode Eclipse

Testing documentation

Juhani Heliö

Ville-Pekka Hämäläinen

Nikke Kostiainen

Erkka Kääriä

Leo Leppänen

Joel Nummelin

June 24, 2014

1 Overview

The TestMyCode Eclipse code base can be seen as consisting of three different parts. Firstly, we have the completely new Eclipse plugin code that was specifically written for this project. Secondly, there's the TestMyCode Core's new code that was also written during the lifetime of this project. Thirdly, the project is also affected by - and contains parts of - a rather large legacy code base from the TestMyCode Netbeans project[1]. Due to differences between these three parts, different testing strategies were chosen for each of them.

2 TestMyCode Core

All new code in the Core has been extensively unit tested with JUnit[2] and Mockito[3]. Cobertura[4] and PIT[5] were used in evaluating these tests to make sure the tests covered all the relevant lines, code paths and mutations. For this 'new' code, the coverage is at or near 100%.

The Core also contains legacy code from the TestMyCode Netbeans project, especially in the form of the HTTP stack that was copied over and changed only by renaming a few classes and methods. This legacy code base has fewer tests due to the hard-to-test style of the code. It was - however - felt that extensive testing of this part of the code base was less critical, mainly due to this code having been in production for years.

The team acknowledges the problem this represents from the point of view of future refactoring and development efforts, but had to prioritize other

[1]https://github.com/testmycode/tmc-netbeans
[2]http://junit.org/
[3]https://code.google.com/p/mockito/
[4]http://cobertura.github.io/cobertura/
[5]http://pitest.org/

aspects of the product for not having infinite time. In a sense, the legacy code was not tested because it was not refactored to be testable and the code could not be safely refactored in the allocated time frame before tests existed.

Other parts of the project are also heavily influenced by the legacy code base. The prime example of this kind of code is the Spyware component that was first brought over directly, but later underwent some refactoring and testing efforts.

Also influenced by the legacy code base were most of the background tasks. Most of these are well tested, with the exceptions of the Ant and Maven test runners. These tasks are dependent on a IDE specific implementation that tells them how to build projects and run Maven tasks respectively. This made testing these classes in the frame of the Core impossible without tying the Core to a specific IDE for development. Tying any future development efforts to a single IDE - and a less used that Dept. of CS at that - was deemed unwise.

3 TestMyCode Eclipse plugin

The Eclipse plugin component has no automated tests at this time. This is mainly due to Eclipse using the SWT graphics library that has very little support in the form of automated UI testing frameworks. After no good and robust automated testing solution was found, the team decided to handle all UI testing manually. The team felt that it was better to test manually that to have extremely brittle UI tests that would only work a fraction of the time. This dependency on manual testing is was made possible by the relative Simplicity of the Eclipse plugin's code and the rather small amount of features present in the plugin.

Any future efforts will probably want to research whether better automatic testing frameworks for SWT UI testing exist and possibly migrate over to using them.