A Testing Method for Embedded System Design by Aspect-Oriented

Mingrui Yang (0910070)

School of Information Science,
Japan Advanced Institute of Science and Technology

February 8, 2011

Keywords: Testing, Test case, Aspect-Oriented.

1 Background and purpose of the research

The requirement on software quality is higher and higher. The testing of software is very important to ensure high software quality. So the importance of the testing is higher and higher. We had designed and implement a control system of cooker in the PBL course of High Reliability Embedded System Course in JAIST. The quality is very important for Embedded System, so it's necessary to test the system by three levels, namely Unit Testing, Integration Testing and System Testing. We have to design test-case and then do testing at each level. Although the testing has different objectives at each level, the testing is that each level of Testing is in view of the same structure of Software system. So the same test-case maybe designed. It will effects the efficiency of the testing even the software development, so that it require a systemic testing method. I present a systemic testing method base on Aspect-Oriented Software Development With Use-Case in this paper. This method can be used to systematically design the test-case for unit testing, integration testing and system testing. I use a cooker control system to experiment this testing method.

2 Test-case is repeatedly designed

We must to test all of the classes at unit testing. Then we have to test the interface of each two classes in each component of system at integration testing. So that all the class have been tested twice at unit testing and integration testing. Finally, we have to test all the component of system one by one to check the input and output of them. But, all the three level of software testing is executed to the same structure of software system. So it is possibility that the same part of software system has been tested. Although the three level of testing has different objectives, test-case maybe repeatedly designed. Of course you can do the testing in that way and it can be used to ensure high software quality, but there are a lot of testing must be tested, so it is not good to raise the efficiency of software development. So I present the bottom line of testing at each level of software testing to ensure both of ensure the high quality of software and raise the efficiency of software development. First, we must design test-cases of unit testing first and then design test-cases of integration testing and finally design test-cases of system testing. Second, we have to make the dependency tree for each component of system. Finally, execute testing to the top classes and the bottom classes in the dependency tree at unit testing. Then execute testing to the other classes in the dependency tree by Sandwich Testing Method at integration testing. The top classes are used as a test driver and the bottom classes are used as a test stub. Execute testing to each component of system to check their input and output finally at system testing.

3 Select the component of testing

The testing method is that to execute testing to all of the classes at unit testing and execute testing to the interface of two classes in the component of system at integration testing and execute testing to the input and output of each component of system. But we have to know which part is the component of system. So I present a testing method like this. Execute testing to each use-case of system at system testing to check their input and output. Then make the dependency tree of each use-case slice and

execute testing to the classes in each use-case slice to check their interfaces at integration testing. Finally execute testing to each class in each use-case slices at unit testing. So I present the testing method based on Aspect-Oriented Software Development with Use-Case.

4 A systematic testing method

In conclusion, I present a systematic testing method like this. First, we must design test-cases of unit testing first and then design test-cases of integration testing and finally design test-cases of system testing. Second, we have to make the dependency tree for each use-case slice. Finally, execute testing to the top classes and the bottom classes in the dependency tree at unit testing. Then execute testing to the other classes in the dependency tree by Sandwich Testing Method at integration testing. The top classes are used as a test driver and the bottom classes are used as a test stub. Execute testing to each use-case to check their input and output finally at system testing.