Michael A. Walker
2003-09-01

Assignment 2 Report

Having gone through and written as good of unit tests as I can. I haven't found any code that “doesn't work”. However, I did refactor heavily some classes. The point of this refactoring was to reduce the coupling between classes by either dependency injection or just object creation outside of classes. This resulting in much improved testability of the code.

Another technique I used was making use of the Strategy Pattern to decouple components. This resulted in me being able to initially create 2 interfaces, and then after further review, to combine them into a single file accessor class. This class is the only place where direct file access is done. This allows me to substitute in a Stub class where I can manually inject the return values I want to receive and/or give.

I did have two sets of major roadblocks, first is the file to test “impl/Asgn2/src/org/cs27x/dropbox/test/DropboxProtocolTest.java” the DropboxProtocol.java's test class. I was able to test the class fine. Except for an issue where [verify(transport, times(1)).publish(cmd);] would result in an error, that wouldn't give any stack trace or other output. However, this only occurred when testing other methods that would call publish within them. When I would call publish myself, it would work properly. I'm unsure as the root cause of this issue.

The other major roadblock I had was the FileEvent class “impl/Asgn2/src/org/cs27x/filewatcher/FileEvent.java”. For some reason, when I initially tried to construct my own FileEvent object or mock objects, I was having difficulty creating a valid eventType. (despite having the appropriate imports in the class, etc.) Eventually I skipped it and came back to the problem, I restarted Eclipse after applying OS patches. And I was able to properly construct the FileEvent objects. So I'm leaning towards the problem being Eclipse, though I'm not sure.

What I would have liked to do to test this code further was to do some kind of integration testing between the two components. However, I am not aware of how you can do that with Unit Testing. Therefore I didn't attempt that at all.