Chapter V

Fault Injection

Part of the process was—funny as it sounds—was to test the testing framework. The purpose of the testing framework was to detect errors in the program where it did not meet the program requirements. So, a simple way to test the testing framework would be to insert errors into the project's source code which would cause certain tests to fail. If the report shows that the proper tests passed and failed, then we would know that we are catching the errors which we are trying to catch.

The following errors have already been written into the code and are simply commented out. Removing the comments, and they will inject the errors mentioned hereafter.

Adding the line *ret.getMemberIds().remove(5);* to the Cohort Class in the *intersect* method after the conditional at line 138 will cause any test case which has a 5 in the result to fail. Specifically, this is test case 8.

In the *union* method of the Cohort class, adding the line *ret.getMemberIds().add(5);* after the conditionals at line 119 will put a 5 in the resulting Cohort. This will cause any test case which does not have a 5 in the returned Cohort to fail. Or, more precisely, test case 3 will fail.

If the line *ret.getMemberIds().remove(4);* is added to the *subtract* method of the Cohort class after the nested conditionals at line 158 will cause any test case which has a 4 in the result to fail. Hence, test case 15 fails.

Inserting the error *ret.getMembers.remove(-3);* to the *union* method of the Cohort class after the conditionals at line 120 will remove -3 from the returned Cohort, and thus any test case which expects a -3 in the result will fail. For this reason, test case 22 will fail.

Injecting the line *ret.getMembers().add(1);* to the *subtract* method of the Cohort class after the nested conditionals at line 159 will add 1 to any resulting Cohort. Any test case which does not have a one in the expected output will therefore fail. These would be test cases 13, 15, 17, and 23.

At this point in the process, things are going very smoothly. Faults were injected into the source code, and it was very reassuring to see that they did in fact produce errors which were caught by our testing framework. This in a way solidified the feeling of accomplishment that comes after work is done. We saw that our program really was working properly, and we saw it do so in a way that a series of tests labeled "passed" somehow did not. Maybe it was seeing that they indeed could fail, and simply were not, that brought the reassurance. However, in the end, it was a great way to end the project.