Oregon State University

CS 362

Spring 2016

Test Report of CS 362

Author:
Zhi Jiang

Instructor:
Alex Groce

1 Introduction

As we know, software is similar with an engineering, so it still has some problems and it also need to be maintained. Testing, as one of necessary skills for programmer, plays important roles on software development. We need to use proper testing to check this software engineering in order to ensure that it is stable and reliable. The purpose of CS 362 is to introduce some basic principles and methods of testing for students. According to this class, I learned many kinds of testing methods such as unit testing and random testing, and I still understand some rules and concepts for testing such as coverage.

2 Unit Testing

In the assignment 2, I created eight unit tests for my program. As we know, unit testingis the most basic method to test program. The implementation of this kind of testing is to divide the entire game and then choose individual units ofsource code to do testing. Usually, size of unit testing should be small because the target codes of unit testing is also small in the program. When I did unit testing for assignment, I follow these two steps:

- Understanding which part should be tested
  The assignment requires me to test effects of card, so I chose function cardEffect() and put corresponding parameter in this function.
- Setting conditions to check this card implements the effects or not
  For example, I compared number of hand cards before and after calling adventurer function. If the result does not meet real expectation, which means there is a problem in this parts. I followed these two pints to complete all of unit tests.

After I done assignment2, I think advantage of unit testing is that it can be quickly and easily implemented, because it is pretty small. In addition, it is able to provide a clear and direct feedback for you, so you can locate problems immediately and fix them. The process of writing a unit test code can prompt developers to think about the content and the logic of the code. Unit testing still has some disadvantages. The most obvious disadvantage is it cannot to make complete testing for the entire game.

3 Random Testing

In the assignment 3 and assignment 4, I started to learn random testing. Random testing is also a very useful test method for software developers. The purpose of Random testing is to generate more random inputs to test program. As we know, there are many conditions and branches in a program, so it is impossible to write unit testing for each branches. Random testing can use a lot of random inputs to cover all of branches in all possible ways. When I did these assignments, I followed these rules:

- identifying interface
  For example, if I need to test a function, I have to adjust all of parameters for this function. I have to consider what is range of valid value, or some parameters do not need random inputs.
- Building generator
  According to information about interface I gain, I should build a generator for create random inputs.

- Checking behavior
  Eventually, to write proper codes check behavior on random inputs.

The advantage of random testing is it is able to check all branches, so which it is very hard to miss bugs in program.In my assignment 3, first thing I did is to set 2000 to time of loop. Usually, the time of loop should be big enough because it can ensure generate many random inputs. Secondly, I read codes and found the range of valid inputs. For instance, choice in card steward is a special variable, so I need to determine valid inputs for each choice. As for assignment 4, it is also a random testing, but the difference between assignment 3 and assignment 4 is the latter is used to test the whole game. Actually, I adopt the same ideas with the assignment 3 to write this random testing. Entire game is more complex, so I need to consider and generate more random inputs for more function.

```
void random_kingdomcard(int *r_kingdomcard){
    int i;
    r_kingdomcard[0]=rand()%(treasure_map - 7 + 1) + 7;

    for(i=1; i<10; i++){
    do{
        r_kingdomcard[i]=rand()%(treasure_map - 7 + 1) + 7;
    }
    while(check_kingdomcard(r_kingdomcard, i));
  }
}
```

Listing 1: random_kingdocard function in testdominon.c

The function is one of random input generator in testdominion.c. The purpose of it is randomly choose ten kingdom cards for game. The example is a good evidence to prove quality of random testing for whole game, because random testing for whole must consider all of details for important variables in each function.

4 Mutantion Testing

In the final project, I chose mutation testing, so I still learn some knowledge about mutation testing. The purpose of mutation testing is to make a tiny change such as changing bound of loop in order to check feasibility of correlative codes. The version that have been changed is called mutant. When I did mutation testing, I usually follow these steps:

- Using mutant testing generation to create a number of valid mutants
  The role of mutant testing generation is to help user quickly create mutants. Usually, each mutant only contains single change. The change of mutants includes value mutation, decision mutation and statement mutation.
- Choosing a test to check these mutants and original file, and then to compare their output.
  As for this step, I write a bash shell to compile these files at the same time, and then output results of them to output files
- According to the comparison results to determine the effectiveness of the software test.
  In these results, we probably meet condition, which is original file and mutant file have the same output, so condition represents this mutant is lived and it cannot detect change. On contrary, the mutant can be killed if they have different outputs.

In mutation testing, there are two important hypotheses, one is competent programmer hypothesis and another is coupling effect. Competent programmer believes that most of errors are due to small syntactic error. The coupling effect states some simple errors can cascade other errors.

5 Coverage

Coverage is a crucial concept in software testing. It is a useful method to measure the integrity and effectiveness of the test. In the other words, essence of coverage is to check how much this testing cover for the program. The types of coverage contain basic block coverage, branch coverage, path coverage, data flow coverage, logic coverage and active clause coverage. In my assignment 2, when I test adventurer function by unit testing, the coverage is pretty low because unit test is used one time. But in my assignment 3, the coverage increased obviously after I used random test for adventurer function. So which means that random testing has high coverage than unit testing. On the other hand, cover can be also used to check effects of random testing. There are two results from my output in assignment 4, and they are coverage of randomtestadventurer and randomtestcard2.

```
Function 'adventurer_function'
Lines executed:100.00% of 16
Branches executed:100.00% of 12
Taken at least once:100.00% of 12
No calls

Function 'smithy_function'
Lines executed:100.00% of 4
Branches executed:100.00% of 2
Taken at least once:100.00% of 2
No calls
```

Listing 2: output of randomtestadventurer and randomtestcard2

All of lines and branches are executed, so it represents that my random tests are effective to test program. On contrary, if the cover of certain random testing is lower than expectation, which means that the input is not random enough. The random testing which has low coverage will ignore many branches and then cause that bugs existing in branches cannot be discovered.

6 Reliability of Classmates codes

In assignment 4, I picked Wenbo Hous dominion implementation to do differential testing, but the result I gain is TEST FAILED. Our codes cannot pass differential testing because there are some errors in my codes or Wenbos code. In this final project, I used unit testing to find a bug in his codes. In opinion, his bugs are simple because he just changes some independence statement. For example, in his card_Effect_great_hall(), the bug is drawCard function was deleted. Actually this kind of bug only make influence for itself and it cannot affect other codes in program, so I think the reliability of his code is enough. Another classmate I tested is Wang Zhao. I used unit testing and random testing to check his dominion.c. I also found bugs in his codes. His condition is similar with Wenbos codes because his bugs are also simple. For example, he just changed the time of loop, so it will not affect other code outside this function. So I believe that the reliability of his code is enough.

7 Conclusion

Overview, I really feel a lot of benefits of software testing from this class. Before I did not know importance of software testing, but now I implemented three kinds of testing methods by myself. When I did these things, I not only learn knowledge about testing but also understand a structure and logic for a program. In the other word, software testing provides me a new perspective to consider own codes. When I was doing a testing, I need to understand this system and to expect some complex condition. After I gained results, I also need to analyze coverage and reliability of

these results. It is obvious that software testing can improve tremendously my professional skills. I believe that software testing will always be with my study and work in the future.