Yichen Duan

CS 362

6/6/16

After all the tests I've done for the assignments, I realized that when you want to test a piece of code, one of the most important things is understanding the subject. I was really confused when I saw that half of the first assignment requirement is to understand the game, but when I started writing the tests I learned that fully understanding the subject is the base of the testing, not just understand the rule of dominion game, but the dominion.c, like all the variables, how they work together, the logic and all the function calls. At first my card tests kept failing, but I didn't think it was because of the source file because I checked the card code and they were logically correct, then I examined my test code, I found that the assert values were wrong, because in the initialize function it draws 5 cards for the 0 player, when assertequal handcount for player 0 the value should increase 5. So if I understood the source well then would be no confusion. When I wrote tests for other cards, the variable handpos also confused me, I didn't have any idea about what it was for and how it interacts with other variables, so it took a long time to get it. For all the 4 assignments, I only wrote tests for the easy ones like the cards only added actions or card numbers, and simple functions, that's why after running the unit tests the coverage was less than 5%, and gcov after card tests was only 27%. The tests could show more if they can cover more code, and there would be a greater chance of finding a bug. I didn't write code for hard functions because I didn't get the source code well, and it was

hard to tell the value of variables after calling those functions. If I could know the code better, i could get a higher coverage, and it would be much more fun.

Unit test is simple to implement, but it may not cover everything because of that. The functions and cards that were unit tested were not complex this time, so the gcov of those functions were not bad, but for all the code the gcov was poor. Before I took this class, unit test is the only test I've ever done, it is useful, but the difficulty is that you have to think of those crazy inputs and possibilities by yourself, and there is a great chance that you miss something.

Random testing is really an efficient way to test, because as long as you set the times it runs and give a seed it could run automatically with all the input and a variety of possibilities. Random tests could test more than unit tests, generally. The cards I random tested were not complicated also, so the coverage of the function didn't change a lot. If there were enough branches or need a variety of inputs I think random testing would be more useful. Anyway I wish I could use it more in the future, so it could really help by generating different inputs.

Tarantula is a magic way of testing, it is easy to tell which statement is suspicious and which is not that suspicious if you don't need to write a script by yourself each time. And it not only shows the very suspicious ones, it also tells you the less suspicious and very safe ones, so you can check those statements with priority and it would be hard to miss a potential bug.

I tested wangzhao and liujiaw's dominion.c with my unit tests, random tests and testdominion. For liujiaw's code, after running the card test of smithy, the message showed that the test failed, which means that there is a bug in function smithy, and after making some changes to his dominion code it proved that I was right. All of the four unit tests passed and

after the unit tests the coverage was “Lines executed:48.39% of 560” and after the random tests the coverage was “Lines executed:47.14% of 560”. When I compared his result and my result after running testdominion.c, my coverage was “Lines executed:63.23% of 563” and his was “Lines executed:63.57% of 560”, which was almost the same, but there was a difference when he bought gold while I bought silver, and then there were more difference without surprise. It’s hard to say which one is correct because both of us have cards that contain bugs, but I’m not sure which bug caused it or if it was the bug in the cards caused it.

For wangzhao’s code, the cardtest1 and cardtest2 failed, which tested card council_room and smithy, it means that there are bugs in these two functions. The unit tests passed too. After running the random tests, the coverage was “Lines executed:48.75% of 558”, and the coverage was “Lines executed:47.49% of 558” after card tests and unit tests. It showed “Lines executed:63.44% of 558” after running the testdominion. When compare his results to mine, there was the same difference – he bought gold while I bought silver, which means this difference is very likely caused by my bugs. So I compared his code to liujiaw’s code, and they have more results in common, but still have difference when a player plays a card, and the coverage was 63.57% and 63.44%, which are slightly different.

It is so hard to tell the reliability of the dominion code, because there are so many functions connected to each other and one single mistake may cause a lot of difference. And the card tests can not tell a lot either, because they only tested four cards out of twenty cards. Also the coverage was less than 70%, which means that there were still hundreds of lines that were not executed at all, there may be bugs among those lines too. It would be better if we print more details in every step or have a whole game tester instead of only run the unit tests

and a “random playdom”. But it is so hard to write one because of the complexity of the dominion code.