Paul Minner

CS 362

Test Report

Testing dominion has been an interesting and unique learning experience for me. This was the first time I debugged somebody else's code than my own. I had helped other people debug their code before, but just being handed completely alien code full of bugs was a completely new experience for me, and a valuable skill to work on. Learning more advanced testing methods, such as mutant testing, delta debugging, and random testing, have all been valuable skills to learn as well.

The first assignment of the class was to pick five cards out of dominion and put them into their own functions. We were also supposed to introduce bugs into two of these cards. I ended up forgetting about where I put the bugs, and even now I am unsure if I fixed the bugs which I introduced. This was a good way to make everybody's dominion a little different from one another.

The second assignment was to create 8 total unit tests. 4 of them were for general functions, named unittest1-4, and the others were for cards named cardtest1-4. My first tests weren't very effective because I still didn't completely understand the dominion code. I only failed one of my tests, which wasn't very good because there were obviously more bugs in those functions than that. My total code coverage for all these functions were 36% combined. If I were to write these tests again, I would check more variables that should have been changed.

The third assignment was to create 3 random testers for three different cards, one being adventurer. This was more complicated to make than the last unit tests, but at this point I was

starting to understand the dominion code better, so I did better on these tests. My adventurer random test failed, so I knew there were bugs in adventurer, and my other random tests passed with the seed 42. One random test was for the council room card, and the other was for the remodel card. The council room tester never failed, but the remodel tester would fail on some seeds, such as 41. My code coverage after running 10 random tests on the council room tester was 22% and the remodel tester was 28%. The remodel tester worked better than the council room tester because it had better code coverage, and it actually found bugs.

The fourth assignment put everything from the previous assignments together to create an all purpose random tester. At this point, I was very familiar with the dominion code, so I didn't find creating the tester too difficult, although this was also the most complicated tester that I have written for this class. Everything about the game had to be randomized, including the kingdom cards, the amount of players, what cards were being played by the players, and the choices chosen when you play a card. I created my random tester by first randomly selecting kingdom cards, randomly selecting the number of players, initializing the game, the entering a loop. The loop would continue until the game was won. Within the loop, each player would play a random card, then buy a random card. When playing a random card, I also randomly selected the choices for playCard. This means more code will be covered. After every turn, I would print out a variety of information about what numbers changed. From my mutant testing, however, I realized that I should have printed out much more information, because while my code coverage was good, roughly 60% for a single run of my tester, my tester failed to kill the majority of mutants, because I didn't check if all variables changed. This is easy to fix however, and I think my tester is a good base that just needs more work.

The final project allowed me to test other student's code with my own testers. Two

students I tested was hollidac and liujaw. Hollidac's code, I noticed, was better written than my own. He passed all of my tests, except remodel card. When I tested his remodel card with my random remodel card tester, I noticed the wrong card was usually removed from the player's hand. Overall, though, I would rate his code higher quality than my own. His code would almost always pass my complete game random tester, but occasionally it would cause an infinite loop, or a crash. His code coverage for my tests were all very similar to my own code coverage. Overall, his dominion was in a working state, but it was still obvious there were many bugs that still need to be fixed to have dominion play correctly. Liujiaw's code is much the same story. He passed most of my tests, but I did find a bug in his smithy card, where he adds only two cards instead of three. This bug isn't high severity or high priority, but the game is still being played wrong. When I ran his code with my complete random tester, it also completed most of the time, but did enter an infinite loop or crash a few times. The code coverage from my tests were very similar to his code coverage as well. Overall, like hollidac's code, dominion is in a working state, but still has many bugs preventing the game from working correctly.

This class has been a great learning experience for me. The new ideas have been helpful, such as mutant testing, delta debugging, model checkers, and the like, but the most useful take away from this class was the mindset I developed. I have a better idea now of how to debug a program in general, even if I'm not using mutant testing or delta debugging. Knowing how to go about fixing a bug is much more important than understanding fancy versions of debugging which work very well in some circumstances, but not so well in others. Also, working on a project which I knew nothing about was a great learning experience, and similar to what many people will have to do when they get a job. Being able to debug somebody else's code will make you a great developer.