Justin Thrailkill
CS 362
Spring ‘15

Test Report

Over the course of this term, I would say overall I enjoyed the testing of dominion. I had not played it before, but I had heard of it. I can now say I am a bit experienced on the subject. Having testing based around a game made the assignments a lot more meaningful, and made it so I could get more engaged in what I was doing. Compared to other classes where the projects and assignments feel like you would not be doing anything like that for the rest of your life, I could see myself testing and implementing something like dominion out in the wild for a job or internship. It gave the class real world value, and that is something I feel like the computer science program here could use a bit more of.

I learned a lot about bugs, debugging, and the process of finding and fixing them. There is a lot to learn about the art of bugs, and I feel like I have only scratched the surface. But I know I can apply the skills I have learned here towards other projects. Even with a program with the main file being fewer than 1000 lines, it is still possible for numerous bugs to be found, even after a term of finding and attempting to fix, or not fix, said bugs.

Using a version control system for submitting work was an interesting and practical method, which taught us about them at the same time. Even though I came across some problems with submitting my work riding up to the deadline, they were not huge now that I look back at them. I now do see why Alex advised people who did not know how to properly use Git to use SVN instead. There were at least two occasions where a student forced a commit or merged their directory with mine, and other students’ work. This was a bit annoying, but did not tend to cause me much of an inconvenience. There was one time I did have to recommit work after a revert, which if I left unnoticed passed the deadline, would technically make my assignment late.

The first assignment for this class was fairly easy, but gave us an opportunity to get acquainted with the code that we would be working with for the rest of the term, and to understand dominion if we did not already. For me, since I did not know anything about dominion before this class, it was interesting to learn about a game from its code implementation, as opposed to purely learning from playing the actual game. The second assignment really kick started the amount of learning done through dominion. Unit tests were a semi new concept to me, having only really heard of them or maybe doing one small one for a past assignment. Writing eight of them really forced me to understand what exactly they are trying to accomplish. I can see how I can use this knowledge in pretty much any other area of programming and code review.

The third assignment was quite unique, as I have never done purely random testing before. It was interesting to have to think about the results you were getting, and to put it into a frame of mind of pure randomness, and what that could mean. Then to think about all the edge cases where a random number would generate something that would never happen in a real game, and to add exceptions to the randomness, making it ever so less random.

The fourth assignment was like taking the third assignment a step further, by making it play entire games of dominion. This was easily the hardest of the assignments to implement. There were a lot of factors to consider, from making sure cards were actually where they said they were, to making sure they actually did their intended purpose, to having different phases for each player. It also just pushed your knowledge of dominion itself. I bet everybody in this class is a pseudo-expert of this game by now.

When measuring the code coverage of the code of my classmates, I did not find a major discrepancy between the percentages. But when I compared my numbers to other people's coverage, I could see that my tests did not cover nearly as much as some peoples. I started with running my tests with haagensa's dominion.c, and I got a 51.43% coverage. Next I ran my tests on fahlmant's code, and got a 50.96% coverage. I noticed that some people were getting coverage way higher than fifty percent, around seventy percent for some people. It was obvious that my tests were not the best, but now I have a comparable number for how much worse it was.

Since I did not get a huge difference in code coverage between the two dominion.c's I tested, I would conclude that haagensa's and fahlmant's code is reliable. This does not say much towards the application of production of this code. If this code were to go public and be released, it would be discredited due to the number of bugs that are still remaining, both from my code and the original source code.

Overall, this class has been a pleasant and unique experience that has taught me a lot of valuable information in which I can apply to nearly any other field of work in computer science. Working with dominion has given me an opportunity to learn more about the other side of writing code, or post-code if you will, while still being an interesting project in of itself. This class has opened my eyes to a wider view of positions in a computer science project. There are more slots to be filled on a team besides working on creating the main project. As someone who does not particularly likes to code the main meat of a program, it gives me hope and an idea of the other types of positions that I can fill in a company and still be an integral part of the development process.