Erin Sullens
CS 362

Test Report

I found working with the dominion game was an interesting experience. I have had some experience in the past working with making unit tests in my Software Engineering 1 class, but that was working on code that I had written. I definitely didn't have any experience testing a large amount of code that I had not written, so that was a new experience for me.

For assignment 2 I created 4 unit tests for the following cards: smithy, great hall, Adventurer, and treasure map. When I ran the unit tests for these cards, only one came back as a failed test. There was a bug in smithy because in assignment 1 I changed it so that only 2 cards get drawn instead of 3 because it said to implement one card incorrectly for that assignment. So my unit test had no trouble finding that bug. When I ran gcov on these unit tests I got the following results: card test 1 had 93.33% coverage, card test 2 had 91.3% coverage, card test 3 had 93.3% coverage, and card test 4 had 95.24% coverage.

Also for assignment 2 I created 4 unit tests for the following functions or functionalities of the game: I checked that in initialize game you start out with the correct number of copper cards; the second was checking getCost, and making sure it returned the correct value for silver; the third was to check discardCard, which checked if when you discarded a card that your hand size was one fewer; and lastly I checked isGameOver by decreasing the supplyCount of province to zero, and then checking that it ends the game. All of these unit tests passed with varying amounts of code coverage. Unit test 1 had 92.31% coverage, unit test 2 had 90% coverage, unit test 3 had 93.75% coverage, and unit test 4 had 92.86% coverage.

In assignment 3, I created random testers for 3 cards: adventurer, smithy, and great_hall. In each test I created a random instance of a game with random numbers for deckCount, discardCount, hand count, and number of players. Then I would pass that game into a function that created an identical game. Then it would run cardEffect on the original, and make the changes to the identical game that the cardEffect would make. If the two instances of the game were equal, each test would print out Test Passed and if they failed, it would print out Test Failed. All of my random testers passed.

In assignment4, I created a random tester for the whole game of dominion. It had three functions: the main function, hasActionCardInHand, and isCardUnique. isCardUnique takes a gameState as a parameter and then checks to see if the current player has any action cards in hand. IsCardUnique is a function used when the game is being created and it helps to insure that the random set of cards that the game is being played with are all unique. In the main function of the game, first a random set of cards is selected. A random number of players is calculated (2-4 players), and the game is initialized with the random seed that is supplied through the command line. The game then loops though each of the players' turns. During each turn, the player plays as many random action cards as they can until they are out of actions or until they have no more

action cards in hand. Then the player enters the buying phase and buys as many random cards as they can until they are either out of buys or out of money. Once isGameOver returns true, the game ends and the scores are added up and printed to the terminal. This program gave me some trouble, and for some reasons only certain seeds would get the program to work completely. Some seeds caused the game to go in an infinite loop, and others worked just fine. When I tested it against a classmate's dominion code it did the same thing.

As far as the reliability of my classmates' dominion code, I looked at two people. I tested gibsonri's code with my unittests and my random game tester. The unit tests all passed except for my card test 3 which tested the adventurer. The problem with it was that it should have had the player draw 2 new treasure cards, and they would only draw 1. This is a high severity and priority bug because the adventurer is a card that, at least in my strategy, would play a large part, and only getting 1 treasure card is a big problem. For all of the unit tests for the functions and cards, I got code coverage in the 90%'s for gibsonri's code, just as I had for mine. As I said above, my random dominion game tester didn't work too well, so testing that against gibsonri's code didn't tell me much. Overall, gibsonri's code seemed pretty reliable, except for the bug in the adventurer cardEffect.

I also tested houw's dominion.c code with my unit tests, and his code passed all of the tests and also gave code coverage in the 90%'s. So it's easy to say that their code is pretty reliable according to my tests.

In conclusion, testing dominion was an interesting process and gave me good experience in testing code that I had no part in writing. It was definitely harder to test code that someone else wrote, but that won't be the last time I will be doing that, so it was a good introduction to that. The fact that I have actually played dominion multiple times also made it easier than it could have been. I think something that I need to work on is my random testing since the random test for the whole game didn't go as smoothly as it should have gone. My skills are lacking in that area and I think it's a good skill to have because a random tester seems like a really good way to see how your program reacts to a bunch of different input and it's easier to make random tests than having to input a lot of different data to see how it responds. So overall, it was a good experience working with and testing dominion.