Corinne Brucks

Dr. Alex Groce

CS 362

June 9th, 2015

CS 362 Test Report

The purpose of this assignment is to assess the status and reliability of two classmate's dominion code. To do this, I used the tests that I had written for assignments 2 and 3 and 4, and I used the results of testing my own dominion code with these tests as a baseline. For assignment 2, I had written 4 unit-tests and 4 card-tests.

For my unit-tests, I tested the initialize game function. Unittest1 tested whether or not initialize game correctly assigned the right numbers to the different kingdom cards. Unitttest2 tested whether or not initialize game would deal the hands correctly. This test was design to catch a bug that I fixed- the bug was that the initialize game had the code for dealing the initial hands commented out. Unitttest3 tested whether initialize game would accept a set of kingdom cards with repeating cards (it should return an error). Unitttest4 tested whether initialize game would accept a negative number of players (it should return an error).

For my card-tests, I tested the smithy card, the village card, the baron card (completely), and the some aspects of the steward cards. My results were submitted as part of assignment 2 and are summarized below:

All unittests were successful (note I ran this again on June 9th , 2015 and unittest1 failed because great_hall was not initialized properly).

All of the cardtests failed. My summary of the results of each follows:

- cardtest1 failed. The use of smithy does not increase the number of cards of the current player. Something with how the cardeffect function is implemented is what is causing this since the unittests for initgame seemed to pass. Note: this is one the functions that I refactored with bugs. It increases the amount of cards by 9 instead of 3. However, the assert still failed when I changed it to check for 9 cards.
- cardtest2 failed. Failed on the first assert. Once again modifying the assert to match the buggy refactored version still failed. (I would replace the asserts with if statement error messages and it failed on all of those)
- cardtest3 failed. This complicated test fails on the first assert. The location of this assert suggests that the hand does contain estate cards. Initially before the dealing cards part of initGame was fixed, it would fail on the first assert in part of the code logic for having

no estates since there were no cards in the player's hand. This suggests that although the hands were dealt correctly, there is a fault with card effect. (When the asserts were replaced with if statement error messages, all of the test conditions were failed).

- cardtest4 failed. This card tests one option for the complicated steward card. It is different from the other test since it used playcard instead of cardeffect directly. It still fails on the first assert though. This suggests that the problem was not simply calling cardeffect when playcard should be called.

My testdominion game tester that I designed for assignment 4 was much more successful, and using it I was able to discover a bug in Scorefor, the fixing of which I covered in part 2 of this assignment in the debugging.txt file.

I choose to analyze Alec Shields and Curtis Minks code for this assignment because of our shared history as group members in CS361. Also, I had used Alec's code for the comparison of input from assignment 4, and Curtis and I did the codereview together over each other's and another classmate's (Will Olsen's) code. As part of this process, I had to modify the Makefile to have targets that used their dominion files and to store all of the outputs of their tests in their own separate result file.

I ran my unittest and cardtest suite on their code. Although we had the same errors and test pass and failures, the unittests had a different amount of coverage. For instance, for unittest1 and unittest2, I got 16.21% coverage but Alec's implementation had only 16.17% coverage, and Curtis's 16.15% coverage. All of the cardtests failed for Alec and Curtis's code indicating that they did

I already described the issues with Curtis's code in the bugreports that I submitted for this assignment. I'm running out of time but would have recapped them here.

When I ran my testdominion game tester than I used to find and test my scoreFor bug on their code, I noticed that neither Curtis or Alec's dominion implementation had this bug fixed. Sure enough, I looked at their code, and the faulty line of code that used the discardcount to index the deck was still in place.

In conclusion, Alec Shield and Curtis Minks' implementations of dominion appear to be almost as stable and reliable as my own dominion code. They have the same errors but gcov reports slightly less coverage. Also, hey are missing my scoreFor fix that improved the output from my randomtester. Because of this, their code is slightly less stable and reliable than mine. Most more testing and bug fixing is needed on all of our implemnetations through.