CS362: Test Report 2

Devlin Junker 931636867

May 17th, 2012

1 Test Approach

For this test report, I decided to focus on differential testing after we covered that during class. This seemed like the most logical testing strategy seeing as we have 30 different implementations of dominion in this class. I also used my Buy Card random test that I implemented for the first test report because I already have it working, so there is no reason to leave it out. My test suite file *testdom* first calls the Buy Card test 2000 times with random game states and checks to see if the state is correct after the call of Buy Card. After the Buy Card tests have run, I then play 2000 random games, each with 100 turns in them and produce an output file that can be compared against other tester output files from other games of dominion. Below I have included the random Game Test source code.

```
for (seed = 1; seed <= NUM_TESTS; seed++) {
initializeGame(2, cards, seed, &G);

/* Play Game with NUM_TURNS turns */
for(n = 0; n < NUM_TURNS; n++){
/* Print some Details */
fprintf(output, "[%d, %d] ", n, seed);
printDetails(output, G);

/* Determine Random Move */
op = floor(Random() * 4);
card = floor(Random() * treasure_map);
pos = floor(Random() * G.handCount[G.whoseTurn]);

/*Make Move and Print Details of Move */
switch(op){
case 0:
fprintf(output, "BUY %d (cost: %d) ", card, getCost(card));
```

```
result = buyCard(card, &G);
printResult(output, result);
break;
case 1:
fprintf(output, "DRAW ");
result = drawCard(G.whoseTurn, &G);
printResult(output, result);
break;
case 2:
fprintf(output, "END TURN ");
result = endTurn(&G);
printResult(output, result);
break;
case 3:
fprintf(output, "PLAY position %d ", pos);
result = playCard(pos, -1, -1, -1, &G);
printResult(output, result);
break;

}

fprintf(output, "\n");
}

}
```

This test will check that the expected actions in BuyCard will happen and nothing unexpected changed in the gameState. This is effective if the implementation of others code is similar to mine, however this may not work if someone else decides that they will do something in buycard that I decide should be in a different piece of the code (For Example: When to change the phase that the gameState is in).

2 Test Statistics

I ran the test suite on 3 different team members code, so that is 6000 tests of buycard, and 6000 random games played, each game with 100 turns each. The coverage I was able to get on all of the implementations of dominion was right around 50%. Buy Card and Draw Card and Play Card have all been tested over 50,000 times each. Each of the kingdom cards in my test suite (adventurer, council room, feast, gardens, mine, remodel, smithy, village, baron, and great hall) have been tested about 250 times each.

3 Communication

Each bug report was placed in the bug-reports folder with a small description of the bug and a possible solution to it. In some bug reports, I also included other suggestions for the code. I have included an example below:

```
Infinite Loop in cardEffect for Feast

Best Solution: set x = 0 after printf("None of that card left, sorry!\n")

Other Notes: Might want to check choices to see if valid,
my test program sends -1 for choice1 which is invalid
```

4 Quality of Code

My code has begun to improve, with more bugs being squashed with more tests being run. Differential testing has helped me improve my code, because I can see the difference between my implementation and other peoples and notice when there is something wrong with my code.

5 Bugs Found

The number one bug that I found was the infinite loop bug when playing a feast card, I included the same bug report for each of my testing team that contained the bug. The bug report was simple, explaining to set x = 0 if there are no more cards left in the pile. My random games test found this bug because during the random games, when playing a card, the choices are set to -1. I included this detail in the bug report and suggested that before checking the number of cards left, the game should make sure it is a valid card choice.

The other bug I found in other peoples code was an error in buycard where the number of buys is not decremented after buying.

In my own code, I found a bug in buycard where if the number of coins was equal to the cost of the card, it wouldn't be bought. This bug cropped up in my first differential test because my implementation wasn't able to buy something and the other persons was.

I also found a bug in my adventurer implementation. During differential testing I noticed that in one of the games with my implementation, card number -76493526 was in the hand of the player. After looking through the output of the random games I found that this was placed in the deck after playing an adventurer card. I haven't been able to solve this problem yet, but it was surprising.

6 Future Testers

In the future, more coverage needs to be done of the card effects, this is the least tested part of the implementations because there are a lot of different things that need to be checked based on which card is played. I developed a good infrastructure for running individual types of tests (unit, differential) and being able to include these in an overall test suite. The testsuite is called by running make testdom, the unit tests can be run by calling make testBuyCard and the differential testing can be run by calling make randomTests to produce an output for two different implementations and then using diff to compare the differences in the output files.