Glen Nicol

CS 362

Alex Groce

17 May 2012

Dominion Testing Report 2

Testing strategy: Coverage testing

By calling the 4 major functions, InitializeGame, BuyCard, PlayCard, and Endturn, many of the other functions will be subsequently called and can be tested in a more realistic fashion.

This will be done by simulating games by simply calling these functions a specified number of times per game with multiple game states based on a random seed with various player numbers.

By using grep or similar I can find where cards are played and possible errors are reported in the large test.out file. In order to make sure invalid cards are not introduced I will check for them after each playCard and BuyCard call. Furthermore, a visual inspection will also likely reveal errors that are too hard to come up with checks inside the code.

Increasing coverage:

My initial test was based off of the one given in class. And got an initial coverage of around 29% percent. By looking at the gcov output I found that the end game code was never being called. So I added checks for this in my BuyCard function and in the test as well to catch these conditions after a playCard call. Coverage went up to 43%. Then I realized that the majority of dominion.c is in the cardEffect function for all the specialty cards. I examined the gcov output again and found that when playCard was being called it was never executing the cardEffect function. CardEffect is a very large part of dominion.c. By tuning the test I was able to get correct random options for playCard that successfully entered cardEffect. After this the coverage jumped up to 75 %.

Tester Code important parts:

for (i = 0; i < G.handCount[G.whoseTurn]; i++) {

card = G.hand[G.whoseTurn][i];

cardNumToName(card, cardName); // allow for card name to be easily read in output

printf ("%s ", cardName);

if ((card < 0) || (card > treasure\_map)) {

printf ("\n INVALID CARD IN HAND.\n");

exit(-1);

}

}

printf (": ");

op = floor(Random() \* 3);

card = (floor(Random() \* G.handCount[G.whoseTurn])); // Choose random card from hand

val = floor(Random() \* ((treasure\_map+2)- 1)) ; Choose random card in the current game

switch(op) {

case 0:

printf ("> ENDTURN: ");

r = endTurn(&G);

break;

case 1:

cardNumToName(val, cardName);

if(TEST\_DETAILS)

printf ("> BUY %d%s $%d/%d: ",val, cardName, G.coins, getCost(val));

else printf ("> BUY %s: ", cardName);

r = buyCard(val, &G);

check\_decks(&G);

break;

case 2:

cardNumToName( handCard(card, &G), cardName);

printf ("> PLAY %s: ", cardName);

r = playCard(card, floor(Random() \* G.handCount[G.whoseTurn]), floor(Random() \* ((treasure\_map+2)-1)) , floor(Random() \* G.handCount[G.whoseTurn]), &G);

//r = playCard(card,-1,-1,-1, &G);

check\_decks(&G);

break;

}

printf ("\tError code: %d\n", r);

if(isGameOver(&G)){

printf("End Game conditions met ");

break;

}

**Tuned PlayCard**

r = playCard(card, floor(Random() \* G.handCount[G.whoseTurn]), floor(Random() \* ((treasure\_map+2)-1)) , floor(Random() \* G.handCount[G.whoseTurn]), &G);

The first random number chooses a random card out of the players hand, the second chooses a random out of the current available cards in the game. And the last parameter does the same as the first random parameter. In the case of **Feast** where cards must be taken out of the deck, to avoid an infinite loop a couple lines must be added to choose another random card.

**Steps for grader:**

Use the code in the bigtest directory of my tester-code directory. DomTest.c is the source for the test.

If you are using another dominion.c file other than mine **you must a**dd this line to the first two if/else if within the Feast card effect to avoid an infinite Loop. Around line 670.

choice1 = floor(Random() \*((treasure\_map)-1));

In my Testing I used 5000 Games that are defined in my bigTest.c file. However, I shrunk that down to only 500 for submission. I figured that it would make the files much smaller and easier to grade.

Calling “make testdom” will compile the code and then output the test results into test.out. No other action is required.

**Bugs Caught In my dominion.c**

**1.** When adventurer is played handCount DeckCoun DiscardCount offsets come out to be nonzero indicating a card has gone missing.

<52,60> STATE: 1's turn, DECK:3, DISCARDS: 5, BUYS: 1, cards in hand 5: Copper Estate Adventurer Copper Estate : > PLAY Adventurer: Error code: 0

<52,61> STATE: 1's turn, DECK:1, DISCARDS: 5, BUYS: 1, cards in hand 6: Copper Estate Copper Copper Estate Copper : > PLAY Estate: Error code: -1

Deck decreases by 2, discard is unchanged, hand increases 1. offset -1.

<57,319> STATE: 1's turn, DECK:11, DISCARDS: 5, BUYS: 1, cards in hand 5: Estate Baron Adventurer Curse Copper : > PLAY Adventurer:

added card 0 back into discard

added card 1 back into discard

added card 2 back into discard Error code: 0

<57,320> STATE: 1's turn, DECK:6, DISCARDS: 8, BUYS: 1, cards in hand 6: Estate Baron Copper Curse Copper Silver : > PLAY Baron: Error code: -1

Offset -1

FIX: Shuffle requires the discard pile to be moved to deck then shuffle can be called. I also added breaks and continues to the shuffle code to ensure a deckCount of 0 was not used in lower code resulting in a -1.

I also found that the built in function, discardCard() does not actually put the card into the discard pile which is why the offset was always -1. Everytime I was discarding the adventurer card it was actually being trashed because the code was not complete. Adding these two lines of code in the trashflag < 1 condition fixes this.

state->discard[currentPlayer][state->discardCount[currentPlayer]] = state->hand[currentPlayer][handPos];

state->discardCount[currentPlayer]++;

**2.** Multiple Cards are not discarded upon play.

* 1. Adventurer played successfully, except not discarded. Detected on <10,504><10,626> <10,806><11,1396>**FIX:**  add discardCard(handPos, currentPlayer, state, 0); before returns.
  2. Feast is not trashed on play, Detected on <13,657>

**FIX:** Add code to trash after gaining card.

if(state->hand[currentPlayer][i] == feast)

x = i;

}

discardCard(x,currentPlayer, state, -1);// trash feast

* 1. Baron is not discarded on successful play. <13,928>**FIX**: add discardCard(handPos, currentPlayer, state, 0); before return 0;

1. Baron doesn't gain an estate when the player has none in his/her hand. <1,1055> But does on <2,950>
2. Gardens are not implemented. Will add code to trash them on play.

**Bugs Caught**

**Ellingsn:** Same bugs as entry 2 from my code above. Adventurer is not Implemented at all. Adding code to trash adventurer when played.

Lines executed: 79.46% of 516

**Omaraa:** Same bugs as 2.1-3 from my section. Detected lines are not the same.

Lines executed 62.76% of 529

**Taylodav:** Same bugs as 2.1-3 from my section. Detected lines are not the same.

Lines executed:59.10% of 533

**Lesliew:** Adventurer was never called in a play roll. I tried changing the test to change the seed but even with 1000 games it never came up. I cannot test if Adventurer is discarded after play, but from code inspection I can tell that it was not. Baron and Feast suffer then same bug due to it being in the master file.

Lines executed:53.47% of 548

**Tangke:** I did not have to insert the code into the Feast card effect because there was already similar code implemented. However, the code does not compile. I will not spend the time to debug one person's code.

**Nguyenta:** Same as 2.1-3 from my section.

Lines executed:77.86% of 515

**Shearini:** Same bugs as 2.1-3 from my section. Detected lines are not the same.

Lines executed:79.78% of 549

**Mcconnjo:** The test did not complete because an invalid card was detected. The test print out is:

<1,423> STATE: 0's turn, DISCARDS: 30, BUYS: 1, cards in hand 5: Adventurer Gardens Copper Province Province : > PLAY Adventurer: An invalid card# -1 was detected in 0's discard Error code: 0

<1,424> STATE: 0's turn, DISCARDS: 351, BUYS: 1, cards in hand 8: Adventurer Gardens Copper Province Province Copper Copper Copper : > PLAY Copper: An invalid card# -1 was detected in 0's discard Error code: -1

I was able to eliminate the invalid card by moving the decrement counter from a post to a pre decrementor. However, it is still adding far too many to the discard pile.

I also found that the implementation never does the moving of cards correctly.

<4,31> STATE: 0's turn, DECK:50, DISCARDS: 38, BUYS: 1, cards in hand 5: Salvager Copper Adventurer Baron Copper : > PLAY Adventurer: Error code: 0

<4,32> STATE: 0's turn, DECK:38, DISCARDS: 42, BUYS: 1, cards in hand 7: Salvager Copper Adventurer Baron Copper Copper Silver : > PLAY Copper: Error code: -1

The changes are highlighted in yellow. Deck goes down by 12, discard goes up by 4 and hand goes up by 2. 4 + 2 -12 = -6

4,234> STATE: 0's turn, DECK:47, DISCARDS: 55, BUYS: 1, cards in hand 5: Adventurer Feast Sea Hag Gardens Baron : > PLAY Adventurer: Error code: 0

<4,235> STATE: 0's turn, DECK:27, DISCARDS: 63, BUYS: 1, cards in hand 7: Adventurer Feast Sea Hag Gardens Baron Copper Copper : > BUY Gold: Error code: 0

Deck decreases by 20, discard increases by 8 and hand increases by 2. 8+2 -20 = -10. This offset is always off by one half of the number of cards that is taken out of the deck. These cards are never put back into play.

I was unable to find beaversource's ticketing system so I emailed mcconnjo directly and described the bug to him.

Lines executed: 80.19% of 530