Team 4 Testing Evidence

(J. Dickerson, D. Desautels, K. Smith, A. Melen)

Contents

Introduction

Our Rat-A-Tat-Cat is a 2 player Java-based game that runs as an embedded applet in any Java-enabled web-browser. At the outset of the project, our team divided work into the several primary components, and then specified their interfaces. This process made working around developer schedules and time-constraints easier as each team member had a largely compartmentalized unit to work on. While this “modular” approach made development easier, it made testing more disjoint and without a unifying procedure. Because of this, team members were responsible for the testing procedures of their own code. No single harness was written that captured all components under a single umbrella. Once individual components were tested by their developer, the system was coupled together and tested for “top-to-bottom” functionality.

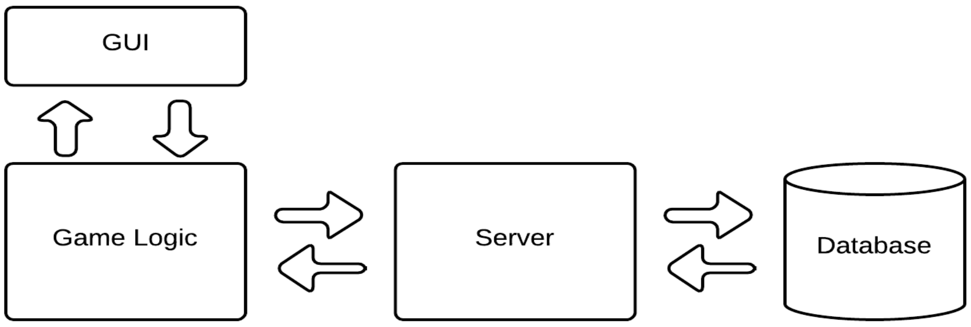


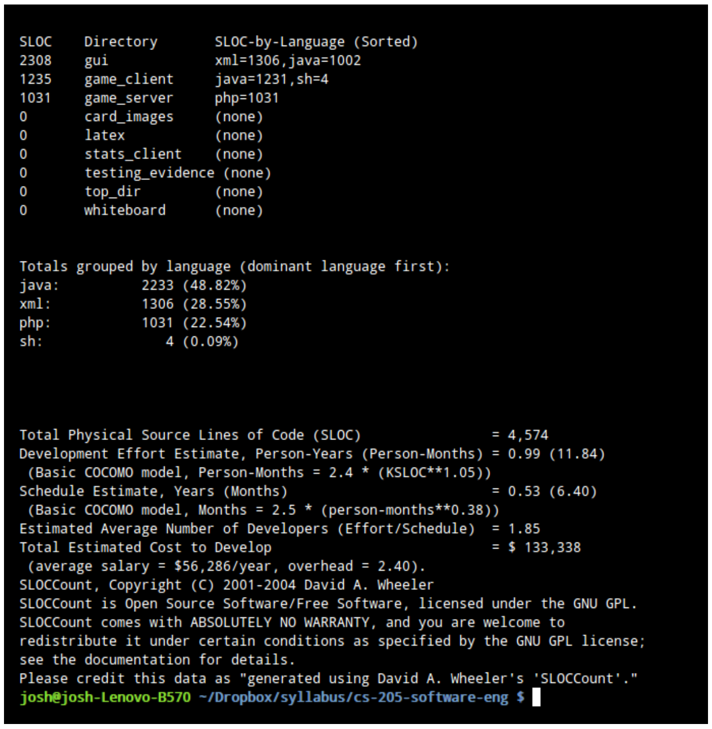
Fig 1.0 – Game System Architecture

On the client side, both GUI and game logic testing was performed, while on the server-side connectivity, database availability, and the Java-to-PHP interface were tested.
Whole system runs were performed where the applet was built, embedded, and run in the Google Chrome browser. During these runs, data was posted to the web API, and added to the database. The leader-board graphs were then inspected, and observed to be changing as gameplay was being simulated. Figure 1.1 shows live usage graphs changing over time as simulation games are played.

Overall System Testing

To get an overview of the system, we used the SlocCount command-line utility to record the size and composition of the codebase.

The output of the SlocCount tool shows a breakdown of the project's lines of code, estimated programmer-time employed, and estimated of labor costs. The lines of code are slightly inflated by included libraries, but nonetheless give a quick indicator of the codebase composition.



```
SLOC    Directory       SLOC-by-Language (Sorted)
2308    gui             xml=1306,java=1002
1235    game_client     java=1231,sh=4
1031    game_server     php=1031
0       card_images     (none)
0       latex           (none)
0       stats_client    (none)
0       testing_evidence (none)
0       top_dir         (none)
0       whiteboard      (none)


Totals grouped by language (dominant language first):
java:          2233 (48.82%)
xml:           1306 (28.55%)
php:           1031 (22.54%)
sh:               4 (0.09%)



Total Physical Source Lines of Code (SLOC)                = 4,574
Development Effort Estimate, Person-Years (Person-Months) = 0.99 (11.84)
 (Basic COCOMO model, Person-Months = 2.4 * (KSLOC**1.05))
Schedule Estimate, Years (Months)                         = 0.53 (6.40)
 (Basic COCOMO model, Months = 2.5 * (person-months**0.38))
Estimated Average Number of Developers (Effort/Schedule)  = 1.85
Total Estimated Cost to Develop                           = $ 133,338
 (average salary = $56,286/year, overhead = 2.40).
SLOCCount, Copyright (C) 2001-2004 David A. Wheeler
SLOCCount is Open Source Software/Free Software, licensed under the GNU GPL.
SLOCCount comes with ABSOLUTELY NO WARRANTY, and you are welcome to
redistribute it under certain conditions as specified by the GNU GPL license;
see the documentation for details.
Please credit this data as "generated using David A. Wheeler's 'SLOCCount'."
josh@josh-Lenovo-B570 ~/Dropbox/syllabus/cs-205-software-eng $
```

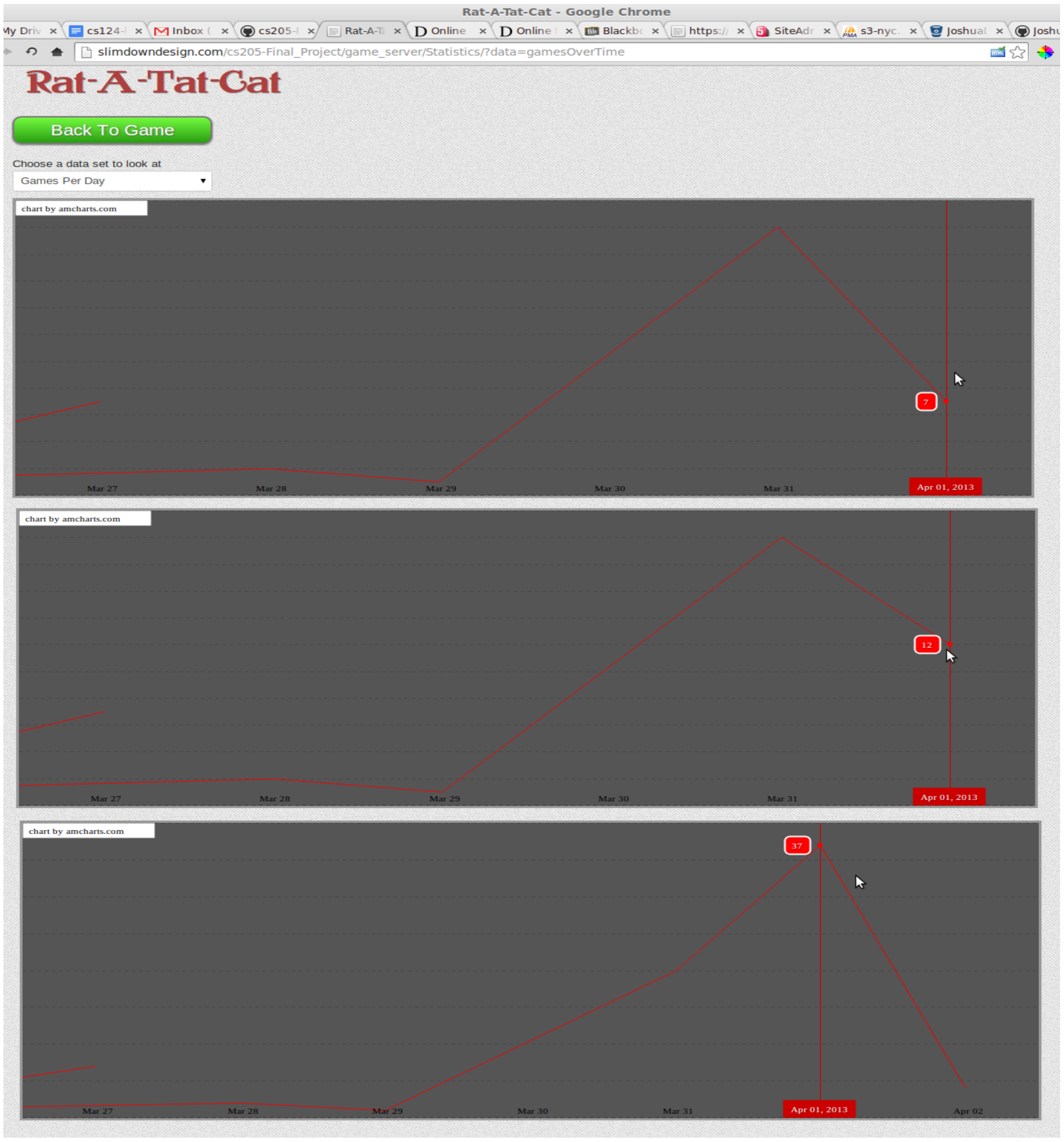
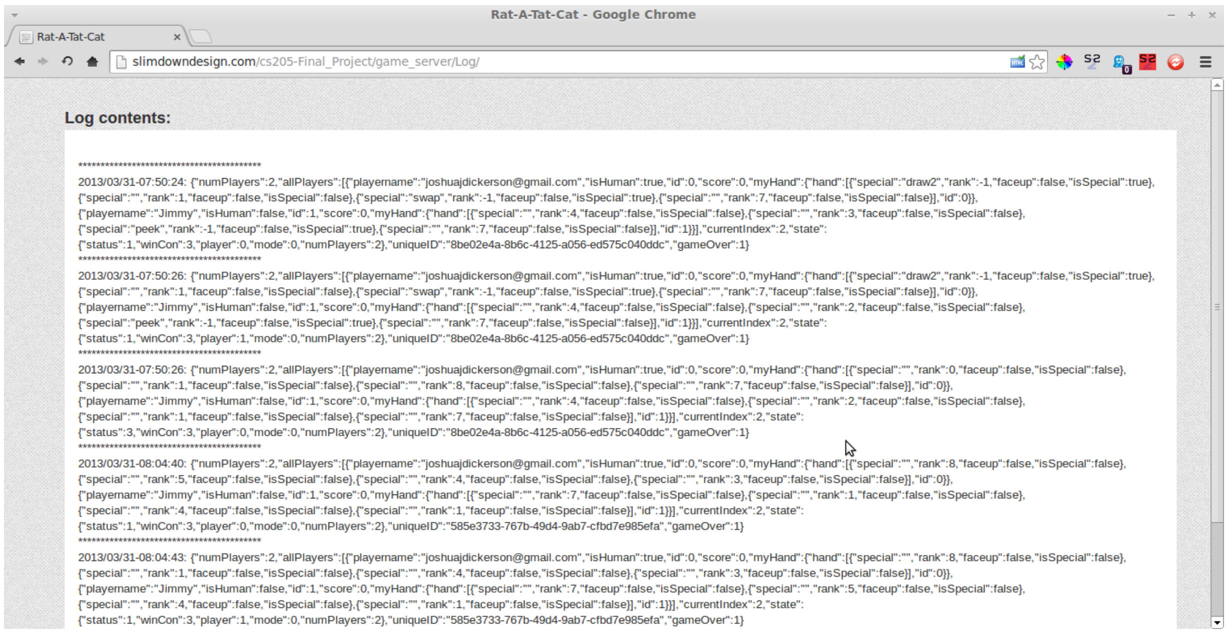


Figure 1.1 shows the change in games/time graph during and after actual game play.

Server-Side Testing

Evidence of communication during state transitions

To test the exchange of state data between the Java client and the web-server, a view was written to output the contents of a server-side log file. The log file shows the contents of each http-post to the web server, and was used to verify that the Java applet was able to communicate with the server. The data shown below is the post requests being sent during an active game, where each state transition is sent to the server as a JSON object, where it is parsed and stored in the MySQL database.


Rat-A-Tat-Cat - Google Chrome
Rat-A-Tat-Cat
slimdowndesign.com/cs205-Final_Project/game_server/Log/

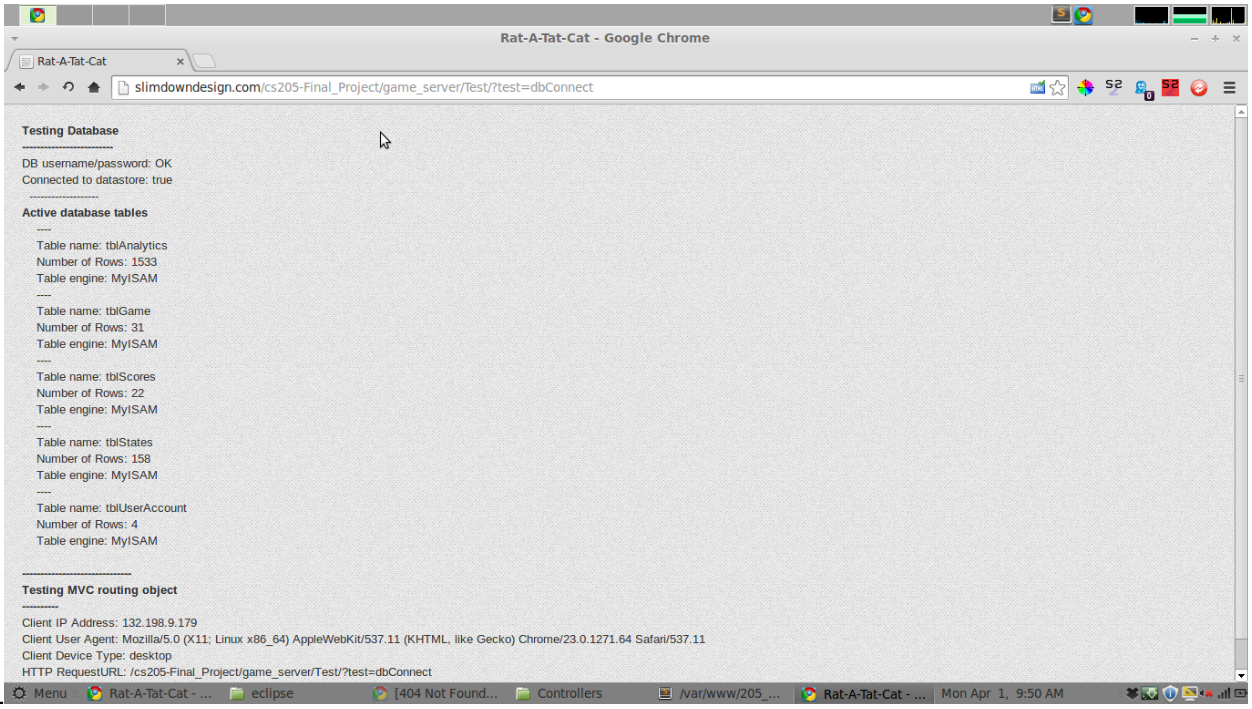
Log contents:

```
******************************************
2013/03/31-07:50:24: {"numPlayers":2,"allPlayers":[{"playername":"joshuajdickerson@gmail.com","isHuman":true,"id":0,"score":0,"myHand":{"hand":[{"special":"draw2","rank":-1,"faceup":false,"isSpecial":true},
{"special":"","rank":1,"faceup":false,"isSpecial":false},{"special":"swap","rank":-1,"faceup":false,"isSpecial":true},{"special":"","rank":7,"faceup":false,"isSpecial":false}],"id":0}},
{"playername":"Jimmy","isHuman":false,"id":1,"score":0,"myHand":{"hand":[{"special":"","rank":4,"faceup":false,"isSpecial":false},{"special":"","rank":3,"faceup":false,"isSpecial":false},
{"special":"peek","rank":-1,"faceup":false,"isSpecial":true},{"special":"","rank":7,"faceup":false,"isSpecial":false}],"id":1}}],"currentIndex":2,"state":
{"status":1,"winCon":3,"player":0,"mode":0,"numPlayers":2},"uniqueID":"8be02e4a-8b6c-4125-a056-ed575c040ddc","gameOver":1}
******************************************
2013/03/31-07:50:26: {"numPlayers":2,"allPlayers":[{"playername":"joshuajdickerson@gmail.com","isHuman":true,"id":0,"score":0,"myHand":{"hand":[{"special":"draw2","rank":-1,"faceup":false,"isSpecial":true},
{"special":"","rank":1,"faceup":false,"isSpecial":false},{"special":"swap","rank":-1,"faceup":false,"isSpecial":true},{"special":"","rank":7,"faceup":false,"isSpecial":false}],"id":0}},
{"playername":"Jimmy","isHuman":false,"id":1,"score":0,"myHand":{"hand":[{"special":"","rank":4,"faceup":false,"isSpecial":false},{"special":"","rank":2,"faceup":false,"isSpecial":false},
{"special":"peek","rank":-1,"faceup":false,"isSpecial":true},{"special":"","rank":7,"faceup":false,"isSpecial":false}],"id":1}}],"currentIndex":2,"state":
{"status":1,"winCon":3,"player":1,"mode":0,"numPlayers":2},"uniqueID":"8be02e4a-8b6c-4125-a056-ed575c040ddc","gameOver":1}
******************************************
2013/03/31-07:50:26: {"numPlayers":2,"allPlayers":[{"playername":"joshuajdickerson@gmail.com","isHuman":true,"id":0,"score":0,"myHand":{"hand":[{"special":"","rank":0,"faceup":false,"isSpecial":false},
{"special":"","rank":1,"faceup":false,"isSpecial":false},{"special":"","rank":8,"faceup":false,"isSpecial":false},{"special":"","rank":7,"faceup":false,"isSpecial":false}],"id":0}},
{"playername":"Jimmy","isHuman":false,"id":1,"score":0,"myHand":{"hand":[{"special":"","rank":4,"faceup":false,"isSpecial":false},{"special":"","rank":2,"faceup":false,"isSpecial":false},
{"special":"","rank":1,"faceup":false,"isSpecial":false},{"special":"","rank":7,"faceup":false,"isSpecial":false}],"id":1}}],"currentIndex":2,"state":
{"status":3,"winCon":3,"player":0,"mode":0,"numPlayers":2},"uniqueID":"8be02e4a-8b6c-4125-a056-ed575c040ddc","gameOver":1}
******************************************
2013/03/31-08:04:40: {"numPlayers":2,"allPlayers":[{"playername":"joshuajdickerson@gmail.com","isHuman":true,"id":0,"score":0,"myHand":{"hand":[{"special":"","rank":8,"faceup":false,"isSpecial":false},
{"special":"","rank":5,"faceup":false,"isSpecial":false},{"special":"","rank":4,"faceup":false,"isSpecial":false},{"special":"","rank":3,"faceup":false,"isSpecial":false}],"id":0}},
{"playername":"Jimmy","isHuman":false,"id":1,"score":0,"myHand":{"hand":[{"special":"","rank":7,"faceup":false,"isSpecial":false},{"special":"","rank":1,"faceup":false,"isSpecial":false},
{"special":"","rank":4,"faceup":false,"isSpecial":false},{"special":"","rank":1,"faceup":false,"isSpecial":false}],"id":1}}],"currentIndex":2,"state":
{"status":1,"winCon":3,"player":0,"mode":0,"numPlayers":2},"uniqueID":"585e3733-767b-49d4-9ab7-cfbd7e985efa","gameOver":1}
******************************************
2013/03/31-08:04:43: {"numPlayers":2,"allPlayers":[{"playername":"joshuajdickerson@gmail.com","isHuman":true,"id":0,"score":0,"myHand":{"hand":[{"special":"","rank":8,"faceup":false,"isSpecial":false},
{"special":"","rank":1,"faceup":false,"isSpecial":false},{"special":"","rank":4,"faceup":false,"isSpecial":false},{"special":"","rank":3,"faceup":false,"isSpecial":false}],"id":0}},
{"playername":"Jimmy","isHuman":false,"id":1,"score":0,"myHand":{"hand":[{"special":"","rank":7,"faceup":false,"isSpecial":false},{"special":"","rank":5,"faceup":false,"isSpecial":false},
{"special":"","rank":4,"faceup":false,"isSpecial":false},{"special":"","rank":1,"faceup":false,"isSpecial":false}],"id":1}}],"currentIndex":2,"state":
{"status":1,"winCon":3,"player":1,"mode":0,"numPlayers":2},"uniqueID":"585e3733-767b-49d4-9ab7-cfbd7e985efa","gameOver":1}
```

Figure 1.2 - server-log receiving JSON strings during game play

Database Testing

Figure 1.3 shows the results of database connectivity testing. The system is connected to the database, and authentication has succeeded. Active tables are listed with their names, the SQL engine that they employ, and the number of rows present in each. Below the database output, are the results of a query on the PHP Routing object, listing the client's user-agent, IP address, device type, and requested URL.



```
Testing Database
------------------------
DB username/password: OK
Connected to datastore: true
  -------------------
Active database tables
    ----
    Table name: tblAnalytics
    Number of Rows: 1533
    Table engine: MyISAM
    ----
    Table name: tblGame
    Number of Rows: 31
    Table engine: MyISAM
    ----
    Table name: tblScores
    Number of Rows: 22
    Table engine: MyISAM
    ----
    Table name: tblStates
    Number of Rows: 158
    Table engine: MyISAM
    ----
    Table name: tblUserAccount
    Number of Rows: 4
    Table engine: MyISAM

------------------------------
Testing MVC routing object
----------
Client IP Address: 132.198.9.179
Client User Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.11 (KHTML, like Gecko) Chrome/23.0.1271.64 Safari/537.11
Client Device Type: desktop
HTTP RequestURL: /cs205-Final_Project/game_server/Test/?test=dbConnect
```

Figure 1.3 - database connectivity test

Client-Side Testing

Display of GUI development

The GUI has been developed largely separate, but in concert with the game loop. Figure 1.4 shows the main game screen in the NetBeans IDE preview screen. The player cards are displayed below and the opponent's hand is above. The deck is a button which allows the player to click it in order to draw. Next to it is the displayed discard pile. An optional peek button is above the player cards during tutorial mode.

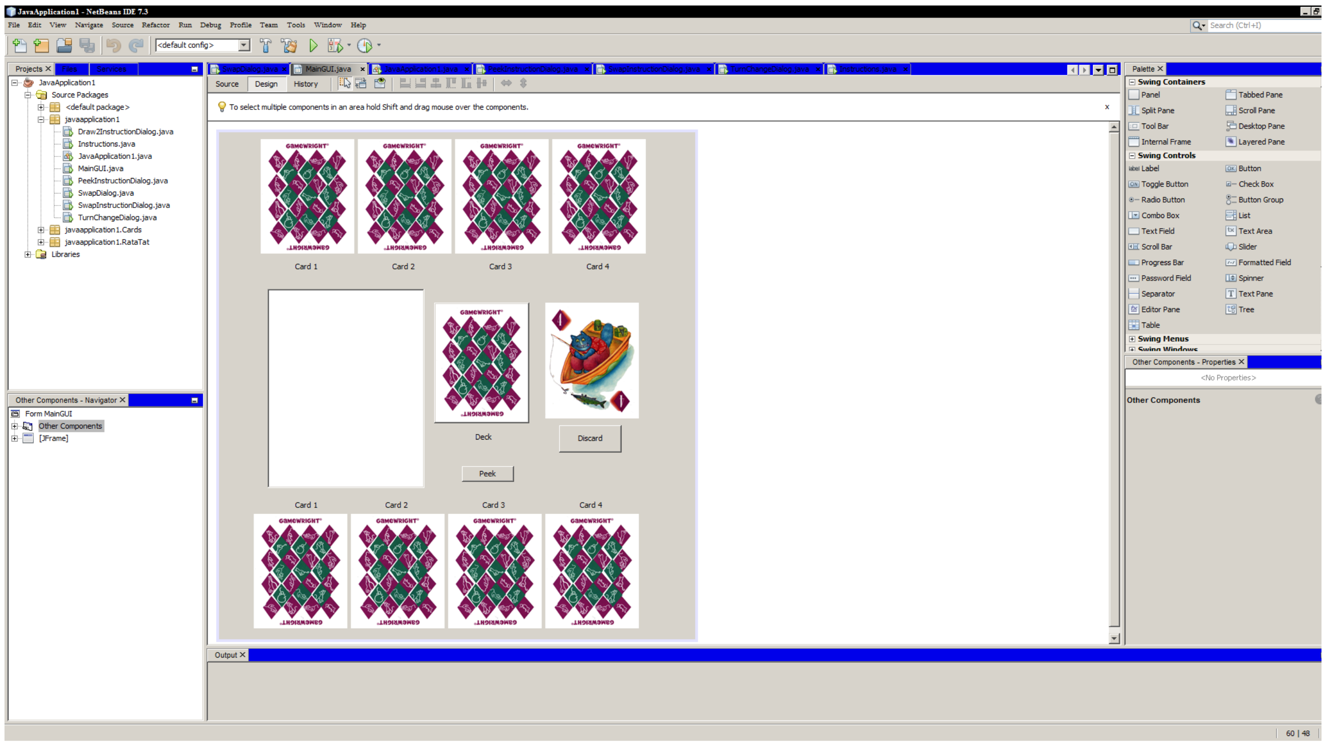
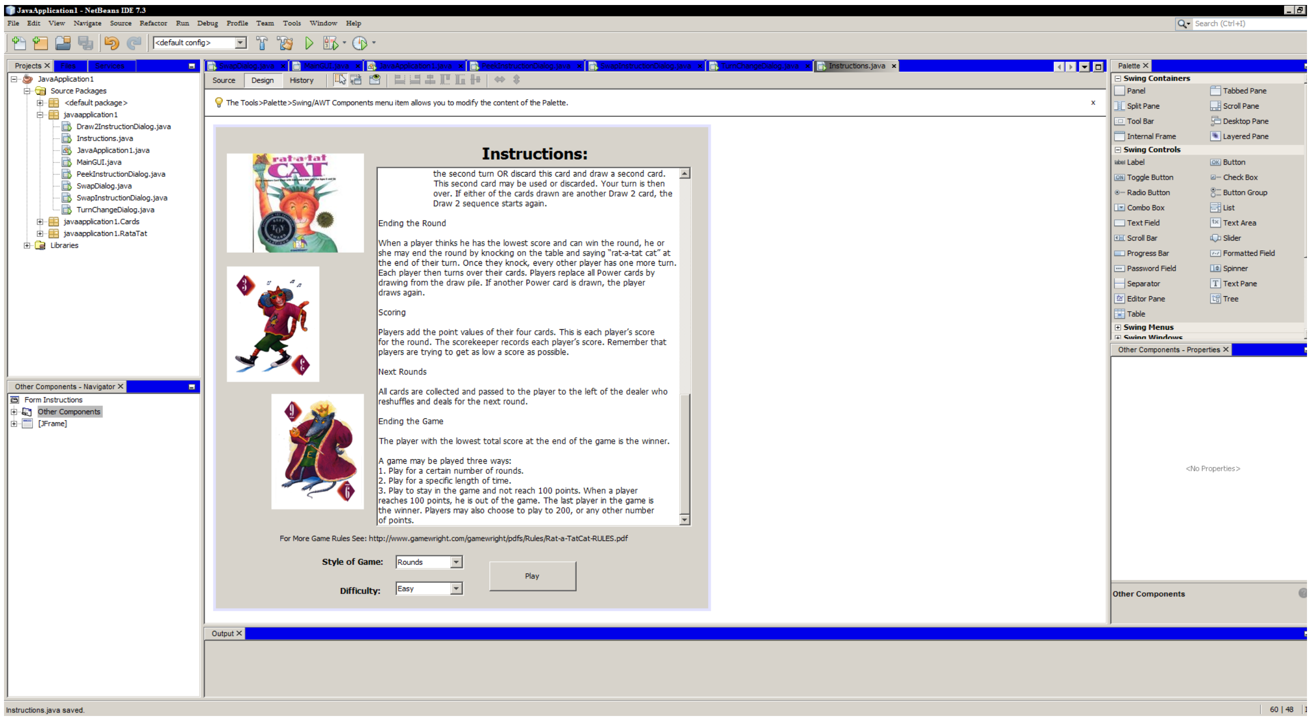


Figure 1.4 – GUI development

Figure 1.5 shows the development process of the introduction/instruction GUI component. The GUI has not yet been coupled to the game loop, and thus was not able to be adequately tested with the rest of the system. System-wide testing was performed using the console version of the game loop.


JavaApplication1 - NetBeans IDE 7.3
The Tools>Palette>Swing/AWT Components menu item allows you to modify the content of the Palette.
Instructions:
Style of Game:
Rounds
Difficulty:
Easy
Play

the second turn OR discard this card and draw a second card. This second card may be used or discarded. Your turn is then over. If either of the cards drawn are another Draw 2 card, the Draw 2 sequence starts again.

Ending the Round

When a player thinks he has the lowest score and can win the round, he or she may end the round by knocking on the table and saying "rat-a-tat cat" at the end of their turn. Once they knock, every other player has one more turn. Each player then turns over their cards. Players replace all Power cards by drawing from the draw pile. If another Power card is drawn, the player draws again.

Scoring

Players add the point values of their four cards. This is each player's score for the round. The scorekeeper records each player's score. Remember that players are trying to get as low a score as possible.

Next Rounds

All cards are collected and passed to the player to the left of the dealer who reshuffles and deals for the next round.

Ending the Game

The player with the lowest total score at the end of the game is the winner.

A game may be played three ways:
1. Play for a certain number of rounds.
2. Play for a specific length of time.
3. Play to stay in the game and not reach 100 points. When a player reaches 100 points, he is out of the game. The last player in the game is the winner. Players may also choose to play to 200, or any other number of points.

For More Game Rules See: http://www.gamewright.com/gamewright/pdfs/Rules/Rat-a-TatCat-RULES.pdf

Figure 1.5 shows implementation of the different Java Swing components into the GUI.

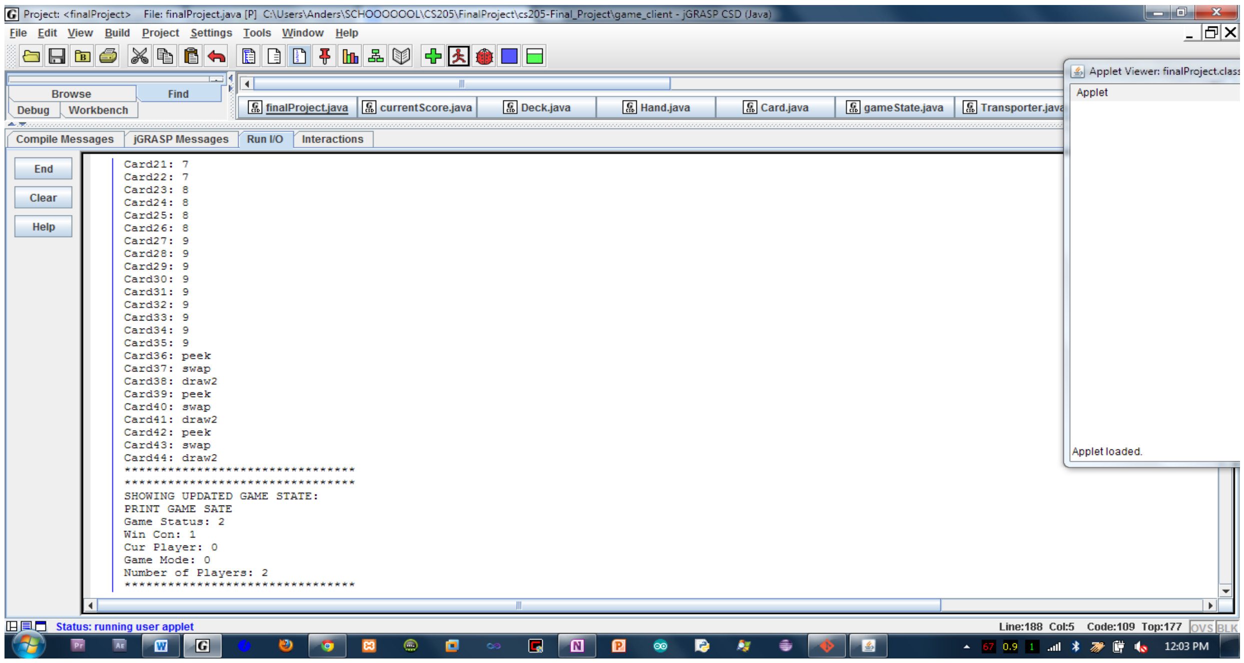
<u>Game Loop Testing</u>

Overview

Game loop testing was performed on both, individual components, and together as a system as a whole. A debug switch was added to the system to provide for a single point to turn testing on or off.

Individual Component Testing

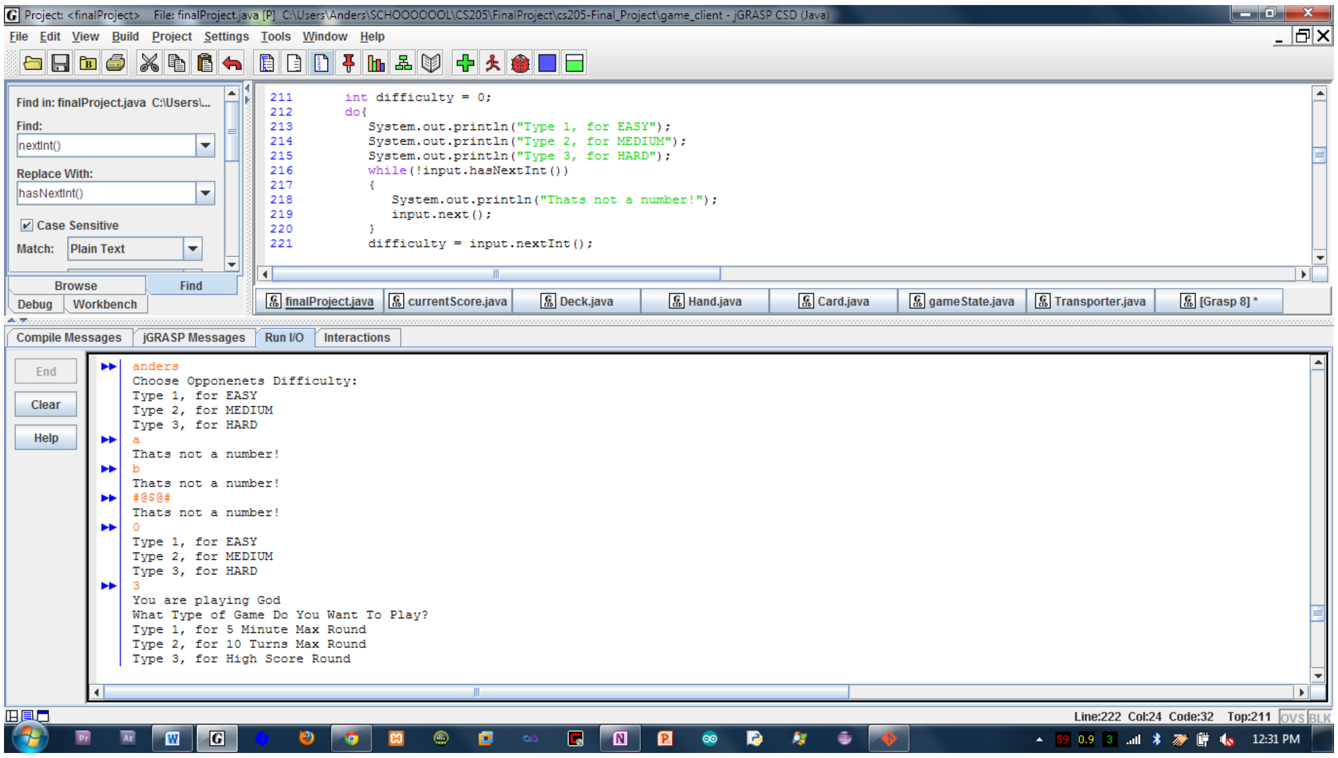
Figure 1.6 shows an un-shuffled deck object output to the console. This output was viewed to ensure that at the opening of each game, all card objects were present, and available for manipulation.



```
Card21: 7
Card22: 7
Card23: 8
Card24: 8
Card25: 8
Card26: 8
Card27: 9
Card28: 9
Card29: 9
Card30: 9
Card31: 9
Card32: 9
Card33: 9
Card34: 9
Card35: 9
Card36: peek
Card37: swap
Card38: draw2
Card39: peek
Card40: swap
Card41: draw2
Card42: peek
Card43: swap
Card44: draw2
********************************
********************************
SHOWING UPDATED GAME STATE:
PRINT GAME SATE
Game Status: 2
Win Con: 1
Cur Player: 0
Game Mode: 0
Number of Players: 2
********************************
```

fig 1.6 – output of deck object

The deck object contains an array-list of card item, each of which is printed to the screen using their toString() method.



```
int difficulty = 0;
do{
   System.out.println("Type 1, for EASY");
   System.out.println("Type 2, for MEDIUM");
   System.out.println("Type 3, for HARD");
   while(!input.hasNextInt())
   {
      System.out.println("Thats not a number!");
      input.next();
   }
   difficulty = input.nextInt();
```

```
anders
Choose Opponenets Difficulty:
Type 1, for EASY
Type 2, for MEDIUM
Type 3, for HARD
a
Thats not a number!
b
Thats not a number!
#@$@#
Thats not a number!
0
Type 1, for EASY
Type 2, for MEDIUM
Type 3, for HARD
3
You are playing God
What Type of Game Do You Want To Play?
Type 1, for 5 Minute Max Round
Type 2, for 10 Turns Max Round
Type 3, for High Score Round
```

Figure 1.7 – user input validation

Overall Game Loop Testing

User input was simulated to automate input validation testing. The system would run through randomized inputs, and compare expected output to actual output. Figure 1.7 shows a portion of this process. After user inputs were tested, http-output was enabled, and the content sent to the server was logged for accuracy (figure 1.2).

Conclusion

Like the writing of documentation, our testing procedure was often an afterthought, not considered to be integral during other phases of development. Because much of our software is not written with unit-testing in mind, the harnesses we developed could easily have become as complex as the game itself. Constrained by time, our testing harness was made to test only the most integral components and interfaces of the game. If done over, we would have developed a testing strategy before any code was written, and implemented the testing-harness early in the process.