

Design Document

December 9, 2015

SE 2XA3

Hui Chen

Nareshkumar Maheshkumar

Sam Hamel

chenh43

maheshn

hamels2

Revision History

Table 1: Revision History

Developer	Date	Change	Revision
Hui Chen	Dec 8 2015	Revised section 5 in the Test Report Document	8
Nareshkumar Maheshkumar	Dec 8 2015	Revised 2,3,4.2.4, 4.2.5, 8, 9, 15 and 16 in the Design Document	7
Hui Chen	Dec 8 2015	Revised section 2.1, 2.2.2, 2.2.3 in the Test Plan Document	6
Hui Chen	Dec 8 2015	Revised Non-functional Requirements in the Requirements Document	5
Nareshkumar Maheshkumar	Dec 4 2015	Revised 3.4, 4.8 and 5.8 in the Requirements Document	4
Hui Chen	Dec 2 2015	Revised: Fixed the numbering in 3.3 and 3.4 in the Requirements Document	3
Sam Hamel	Dec 2 2015	Revised: Number the Functional Requirements in the Requirements Document	2
Sam Hamel	Nov 6 2015	Edit and Proofread Revision 1	1
Nareshkumar Maheshkumar	Nov 6 2015	Design Document Revision 1 Finished	1
Hui Chen	Nov 6 2015	Update MIS Section	1
Nareshkumar Maheshkumar	Nov 4 2015	Edit and Proofread Revision 0	0
Sam Hamel	Nov 4 2015	Design Document Revision 0 Finished	0
Hui Chen	Nov 2 2015	Started Design Document	0

Contents

1	Module Guide For No Limit Texas HoldEm Program(NLTP)	6
1.1	Introduction	6
1.2	Anticipated and Unlikely Changes	6
1.2.1	Anticipated Changes	6
1.2.2	Unlikely Changes	6
2	Module Hierarchy	7
3	Connection Between Requirements and Design	8
4	Module Decomposition	8
4.1	Hardware Hiding Module	8
4.2	Behaviour-Hiding Module	8
4.2.1	Texas Holdem Module	8
4.2.2	Pot Module	9
4.2.3	Game module	9
4.2.4	ResourceParser Module	9
4.2.5	AboutFrame Module	9
4.3	Software Decision Module	9
4.3.1	Hand Module	9
4.3.2	Deck Module	10
4.3.3	Player Module	10
4.3.4	AIOpponent Module	10
4.3.5	Card Module	10
5	MIS of TexasHoldEm Module	11
5.1	Exported Access Programs	11
5.2	Interface Semantics	11
5.2.1	State Variables	11
5.2.2	Environment Variables	11
5.2.3	Assumption	11
5.2.4	Access Program Semantics	11
6	MIS of Pot Module	12
6.1	Exported Access Programs	12
6.2	Interface Semantics	12
6.2.1	State Variables	12
6.2.2	Environment Variables	12
6.2.3	Assumption	12

6.2.4	Access Program Semantics	12
7	MIS of Game Module	13
7.1	Exported Access Programs	13
7.2	Interface Semantics	13
7.2.1	State Variables	13
7.2.2	Environment Variables	14
7.2.3	Assumption	14
7.2.4	Access Program Semantics	14
8	MIS of ResourceParser Module	14
8.1	Exported Access Programs	14
8.2	Interface Semantics	15
8.2.1	State Variables	15
8.2.2	Environment Variables	15
8.2.3	Assumption	15
8.2.4	Access Program Semantics	15
9	MIS of AboutFrame Module	15
9.1	Exported Access Programs	15
9.2	Interface Semantics	15
9.2.1	State Variables	15
9.2.2	Environment Variables	15
9.2.3	Assumption	15
9.2.4	Access Program Semantics	15
10	MIS of Deck Module	16
10.1	Exported Access Programs	16
10.2	Interface Semantics	16
10.2.1	State Variables	16
10.2.2	Environment Variables	16
10.2.3	Assumption	16
10.2.4	Access Program Semantics	16
11	MIS of Card Module	17
11.1	Exported Access Programs	17
11.2	Interface Semantics	17
11.2.1	State Variables	17
11.2.2	Environment Variables	17
11.2.3	Assumption	17

11.2.4	Access Program Semantics	17
12	MIS of Player Module	17
12.1	Exported Access Programs	17
12.2	Interface Semantics	18
12.2.1	State Variables	18
12.2.2	Environment Variables	18
12.2.3	Assumption	18
12.2.4	Access Program Semantics	18
13	MIS of AIOpponent Module	18
13.1	Exported Access Programs	18
13.2	Interface Semantics	19
13.2.1	State Variables	19
13.2.2	Environment Variables	19
13.2.3	Assumption	19
13.2.4	Access Program Semantics	19
14	MIS of Hand Module	19
14.1	Exported Access Programs	19
14.2	Interface Semantics	19
14.2.1	State Variables	19
14.2.2	Environment Variables	19
14.2.3	Assumption	19
14.2.4	Access Program Semantics	20
15	Use Hierarchy	20
16	Traceability Matrix	21
17	Schedule	22
18	Gantt Chart	22

List of Figures

1	Use Hierarchy Diagram	20
2	Gantt Chart	22

List of Tables

1	Revision History	1
2	Module Hierarchy	7
3	Exported Access Programs for TexasHoldEm Module	11
4	Exported Access Programs for Pot Module	12
5	Exported Access Programs for Game Module	13
6	Exported Access Programs for ResourceParser Module	14
7	Exported Access Programs for AboutFrame Module	15
8	Exported Access Programs for Deck Module	16
9	Exported Access Programs for Card Module	17
10	Exported Access Programs for Player Module	17
11	Exported Access Programs for AIOpponent Module	18
12	Exported Access Programs for Hand Module	19
13	Traceability Matrix Between the Module And Requirements .	21
14	Traceability Matrix Between Anticipated Changes And Re- quirements	21
15	Detailed Timeline	22

1 Module Guide For No Limit Texas HoldEm Program(NLTP)

1.1 Introduction

No Limit Texas HoldEm program follows model decomposition based on the principle of information hiding laid out by Parnas. The system is designed to be adaptable because possible future changes have been accounted for in the form of secrets. This flexibility will allow for easy revision control and will allow developers to quickly modify the program as needed. This design document includes in it a module guide (MG) of the system which gives a brief overview of the modules and allows both designers and maintainers to easily identify the parts of the software. The MG directly relates to the SRS with each requirement being fulfilled by either one module or the conjunction of several modules. For greater detail of the modules and what is needed to implement them there is the Module Interface Specifications (MIS), while the schedule of when these modules will be implemented is included in the Gantt and Pert charts.

1.2 Anticipated and Unlikely Changes

1.2.1 Anticipated Changes

AC1: The specific hardware on which the software is running

AC2: The algorithm used for hand ranking

AC3: The deck shuffling algorithm

AC4: The card distribution methods

AC5: The pot/chip distribution methods

AC6: AI Opponent chip betting methods

1.2.2 Unlikely Changes

UC1: Input/Output devices (Input: File and/or Mouse, Output: File, Memory, and/or Screen).

UC2: The algorithm to calculate hand strength will always use the input from players hand and community card

UC3: The game will always end when a player runs out of chips

UC4: Any change to game state are always display to output devices

UC5: Algorithm used to calculate AI Opponents action are defined using

the parameters defined in the player and pot module

2 Module Hierarchy

This section provides an overview of the module design. Modules are summarized in a hierarchy decomposed by secrets in Table 1. The modules listed below, which are leaves in the hierarchy tree, are the modules that will actually be implemented.

- M1: TexasHoldEm**
- M2: Pot**
- M3: Game**
- M4: Hand**
- M5: Deck**
- M6: Player**
- M7: AIOpponent**
- M8: Card**
- M9: ResourceParser**
- M10: AboutFrame**

Table 2: Module Hierarchy

Level 1	Level 2
Hardware Hiding	
Behaviour-Hiding Module	TexasHoldEm Module Pot Module Game Module ResourceParser Module AboutFrame Module
Software Decision	Hand Module Deck Module Player Module AIOpponent Module Card Module

3 Connection Between Requirements and Design

The design of the system is intended to satisfy the requirements specified in the SRS. The system is decomposed into modules, and the connection between requirements and modules is listed in Table 13.

4 Module Decomposition

4.1 Hardware Hiding Module

Services: Includes programs that provide externally visible behavior of the system as specified in the software requirements specification (SRS) documents. This module serves as a communication layer between the hardware-hiding module and the software decision module. The programs in this module will need to change if there are changes in the SRS.

Secrets: The contents of the required behaviors.

Implemented By:

4.2 Behaviour-Hiding Module

Services: Includes data structure and algorithms used in the system that do not provide direct interaction with the user.

Secrets: The design decision based on mathematical theorems, physical facts, or programming considerations. The secrets of this module are not described in the SRS.

Implemented By:

4.2.1 Texas Holdem Module

Services: Displays GUI

Secrets: GUI implementation

Implemented by: NLTP

4.2.2 Pot Module

Services: Keeps track of chips in pot, and distributes chips to players

Secrets: Method of distributing chips

Implemented by: NLTP

4.2.3 Game module

Services: Manages the state of the game

Secrets: How the game transitions between states

Implemented by: NLTP

4.2.4 ResourceParser Module

Services: Parses the rules and game info from text files

Secrets: The values of the parsed text stored as string variables

Implemented by: NLTP

4.2.5 AboutFrame Module

Services: Sets text for Rules and App Info(Help) page

Secrets: GUI and implementation for menu contents(Rules and App info)

Implemented by: NLTP

4.3 Software Decision Module

4.3.1 Hand Module

Services: Keeps track players cards, and calculates hand ranking

Secrets: Algorithm for calculating ranking of cards

Implemented by: NLTP

4.3.2 Deck Module

Services: stores, shuffles and distributes cards

Secrets: Method of card distribution

Implemented by: NLTP

4.3.3 Player Module

Services: lets user fold/bet

Secrets: Method of betting/folding

Implemented by: NLTP

4.3.4 AIOpponent Module

Services: Lets AI fold/bet

Secrets: Algorithm for calculating whether to bet/fold

Implemented by: NLTP

4.3.5 Card Module

Services: object that represents playing cards

Secrets: Card Contructor

Implemented by: NLTP

5 MIS of TexasHoldEm Module

5.1 Exported Access Programs

Table 3: Exported Access Programs for TexasHoldEm Module

Name	In	Out	Exceptions
displayError()	int	-	Exception*
paint()	Graphic	-	-
save()	-	text file	FileNotFoundException
load()	text file	-	FileNotFoundException
actionPerformed()	ActionEvent	-	-

* all possible exceptions will be included, including but not limited to, FileNotFoundException and NullPointerException

5.2 Interface Semantics

5.2.1 State Variables

5.2.2 Environment Variables

path: Location of the directory to be searched for when saving or loading the save file
imagePath: Location of the directory to be searched for images of cards

5.2.3 Assumption

load() cannot be called without save() being called once before

5.2.4 Access Program Semantics

Main will populate the game window by instantiating the frame that is within TexasHoldEm module which will add all the components such as buttons, and uses paint() to populate the window with image of cards from environment variable imagePath and other simple graphics and colour for background. The save() will look for a save file in path. If it exists, it will overwrite it, if not, save() will create a new file. Load() will look for a save file in path and load all necessary data into the game and use Game module to modify the data for the affected modules. actionPerformed() will read the button input and call methods within Pot module based on its corresponding button.

displayError():

Input - Parameter of int type

transition - Displays an error corresponding to the input

6 MIS of Pot Module

6.1 Exported Access Programs

Table 4: Exported Access Programs for Pot Module

Name	In	Out	Exceptions
anti()	-	-	-
call()	-	-	-
raise()	int	-	-
fold()	-	-	-
getBet()	-	int	NullPointerException
getPot()	-	int	-
roundEndEvaluate()	-	-	-

6.2 Interface Semantics

6.2.1 State Variables

pot:int

bet:int

player1:Player

player2:Player

6.2.2 Environment Variables

6.2.3 Assumption

6.2.4 Access Program Semantics

anti uses the Player module to modify both players' chips and also modifies the state variable pot by the same amount taken from player's chips. Call() and raise() uses the Player module to modify corresponding player's chips by the amount read by TexasHoldEm module. Fold() uses the Game module to modify the corresponding playerFolded parameter and also uses Player module to modify the amount of chips lost or gained. roundEndEvaluate()

uses Player module to modify the amount of chips and uses the Game module to initialize a new round.

getPot():

Input - No input required

Output - Returns the value corresponding to the parameter pot

getBet():

Input - No input required

Output - Returns the value corresponding to the parameter bet

7 MIS of Game Module

7.1 Exported Access Programs

Table 5: Exported Access Programs for Game Module

Name	In	Out	Exceptions
playerSwitch()	-	-	-
deal()	-	-	-
nextCard()	-	-	-
isEndGame()	-	boolean	-
roundEnd()	-	-	-
player1Folded()	-	boolean	-
player2Folded()	-	boolean	-

7.2 Interface Semantics

7.2.1 State Variables

currentPlayer:int

Deck: list of Cards

player1: Player

player2: Player

endGame: boolean

player1Fold: boolean

player2Fold: boolean

7.2.2 Environment Variables

7.2.3 Assumption

7.2.4 Access Program Semantics

roundEnd() will call upon methods in Pot to evaluate, distribute the pot to the players and modify the parameters Deck, endGame, currentPlayer and each Player's hand by initializing the corresponding parameters as their default value or distribute new cards to the players.

playerSwitch():

Input - No input required

transition - modify currentPlayer between 2 values

deal():

Input - No input required

Output - Populates the Player hand variable

nextCard():

Input - No input required

Output - Modifies both players' hands by adding the same additional card to each hand

isEndGame():

Input - No input required

Output - boolean value corresponding to the value of parameter endGame

player(1/2)Folded():

Input - No input required

Output - boolean value corresponding to the value of their respective parameters

8 MIS of ResourceParser Module

8.1 Exported Access Programs

Table 6: Exported Access Programs for ResourceParser Module

Name	In	Out	Exceptions
readFile()	String	String	IOException*
parseHelp()	-	String	IOException
parseRule()	-	String	IOException

8.2 Interface Semantics

8.2.1 State Variables

8.2.2 Environment Variables

help: The parsed help textpage stored as a string

rule: The parsed rule textpage stored as a string

8.2.3 Assumption

8.2.4 Access Program Semantics

ResourceParser will read a textfile using BufferedReader() and parse the textfile. Using parseHelp(), it will store the parsed textfile of the help page into a string variable help. Using parseRule(), it will store the parsed textfile of the rules page into a string variable rule.

Input - Text File

Output - Help and Rule textfiles are parsed and output is stored inside string variables

9 MIS of AboutFrame Module

9.1 Exported Access Programs

Table 7: Exported Access Programs for AboutFrame Module

Name	In	Out	Exceptions
startOnHelp()	-	-	-
startOnRule()	-	-	-

9.2 Interface Semantics

9.2.1 State Variables

9.2.2 Environment Variables

9.2.3 Assumption

9.2.4 Access Program Semantics

AboutFrame will use ResourceParser() and use the text stored inside the variables, help and rule from ResourceParser(), and via startOnHelp() and

startOnRule() from AboutFrame, display the parsed text in textboxes

Input - User selection

Output - Rules and Help text displayed in textboxes

10 MIS of Deck Module

10.1 Exported Access Programs

Table 8: Exported Access Programs for Deck Module

Name	In	Out	Exceptions
setDeck()	-	-	-
Shuffle()	-	Deck	-

10.2 Interface Semantics

10.2.1 State Variables

Deck: list of cards

10.2.2 Environment Variables

10.2.3 Assumption

Card module is created before Deck module

10.2.4 Access Program Semantics

setDeck:

Input - No input required

Output - Populates the Deck with Cards

Shuffle: **Input** - No input required

Transition - Modifies the Deck so that it is randomized

Table 9: Exported Access Programs for Card Module

Name	In	Out	Exceptions
getRank()	-	int	DNE
getSuit()	-	char	DNE

11 MIS of Card Module

11.1 Exported Access Programs

11.2 Interface Semantics

11.2.1 State Variables

rank:int
suit:char

11.2.2 Environment Variables

11.2.3 Assumption

11.2.4 Access Program Semantics

Get:

Input - No input required

Output - return value of corresponding parameter

12 MIS of Player Module

12.1 Exported Access Programs

Table 10: Exported Access Programs for Player Module

Name	In	Out	Exceptions
getChips()	-	-	-
gainChips()	int	-	-
loseChips()	int	-	EMPTY
getHand()	-	Hand	NullPointerException
setHand()	Hand	-	-

12.2 Interface Semantics

12.2.1 State Variables

chips:int

Hand: list of Cards

12.2.2 Environment Variables

12.2.3 Assumption

getHand() can only be called after setHand()

12.2.4 Access Program Semantics

Get:

Input - No input required

Output - Each Get method will return value of the corresponding parameter

setHand:

Input - Input will take parameter Hand

Transition - Modifies the state of Hand

(gain/lose)Chips:

Input - Input will take parameter of int type

Transition - Modifies the state of chips

13 MIS of AIOpponent Module

13.1 Exported Access Programs

Table 11: Exported Access Programs for AIOpponent Module

Name	In	Out	Exceptions
getAction()	-	-	-

13.2 Interface Semantics

13.2.1 State Variables

13.2.2 Environment Variables

13.2.3 Assumption

getAction is only called when it gets to its turn in the Game module

13.2.4 Access Program Semantics

getAction will determine the move the AI will take by evaluating the current state of the board and its current Hand, and will call upon the Game module for its action.

14 MIS of Hand Module

14.1 Exported Access Programs

Table 12: Exported Access Programs for Hand Module

Name	In	Out	Exceptions
evaluate()	Hand	int	-
getStrength()	-	int	-
myHand()	-	String	-

14.2 Interface Semantics

14.2.1 State Variables

hand: list of Cards

strength: int

14.2.2 Environment Variables

14.2.3 Assumption

evaluate() is only called when roundEnd() or player1Folded() or player2Folded() from game is called.

14.2.4 Access Program Semantics

getStrength():

Input - No input required

Output - Returns value of parameter strength

myHand():

Input - No input required

Output - Returns the Cards in hand

evaluate():

15 Use Hierarchy

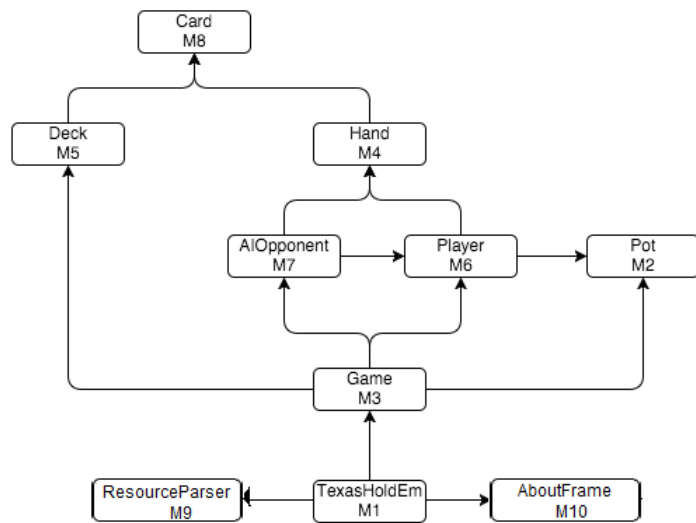


Figure 1: Use Hierarchy Diagram

16 Traceability Matrix

Table 13: Traceability Matrix Between the Module And Requirements

Requirements	Modules
R1	M9,M10
R2	M1,M9,M10
R3	M2,M6,M7
R4	M2,M3
R5	M6,M7
R6	M4,M8
R7	M3
R8	M2
R9	M2
R10	M2,M3

Table 14: Traceability Matrix Between Anticipated Changes And Requirements

Anticipated Changes(AC)	Modules
AC1	N/A
AC2	M4
AC3	M5
AC4	M8
AC5	M2
AC6	M7

17 Schedule

Table 15: Detailed Timeline	
Task	Finish Date
Texas Hold 'Em Module	November 10
Game and Deck Modules	November 11
All other modules	November 12
Revision 0 Demonstration	November 19
Final Demo	December 1
Peer Evaluation and December 3	
Final Doc	December 8

18 Gantt Chart

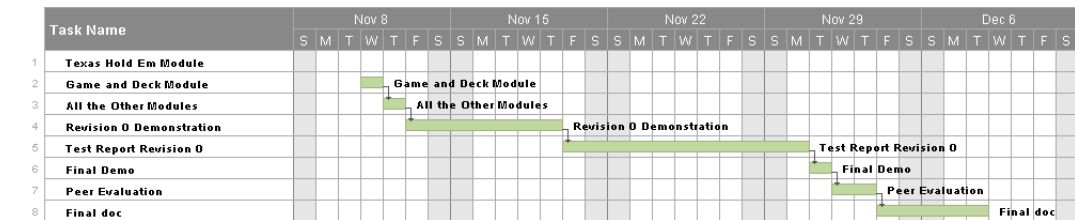


Figure 2: Gantt Chart