Concurrent Poker Player

Milestone Four

The Parallel Poker Team
Mark Jenne
Ian Roberts
Bennie Waters
Sarah Jabon

October 30th, 2009

Instuctor:
Sriram Mohan
Rose-Hulman Institute of Technology

Revision History

Date	Revision	Comment
10-22-09	1.0	Initial draft
10-30-09	1.1	Revised to address Drew's comments
11-02-09	1.2	Revised to address Sriram's comments

Signatures

Mark Jenne	
Ian Roberts	
Bennie Waters	
Sarah Jabon	

Contents

I Executive Summary **4**

II Introduction **5**

III Background Information **6**

1 System Features **6**
1.1 System Features List 6

2 Supplementary Specifications **12**
2.1 Usability Requirements 12
2.2 Functional Requirements 12
2.3 Performance Requirements 13
2.4 Reliability Requirements 13
2.5 Supportability Requirements 13
2.6 Hardware and Software Interfaces 13
2.7 Documentation, Installation, Legal, and Licensing Requirements 13
2.8 Design Constraints 14
2.9 Training Requirements 14
2.10 Deliverables 14

3 Use Cases **15**
3.1 Use Case #1: Starting the Game 15
3.2 Use Case #2: Setting Player Options 16
3.3 Use Cases and Feature Mapping 17

4 Scenario **18**

IV Documentation Standards **20**

5 Coding Standards **20**
5.1 Reliability Standards 20
5.2 Source Code Standards 20
5.3 Naming Conventions 20
5.4 General Coding Paradigms 21

6 Change Control **21**
6.1 Receiving and Processing Requests 21
6.2 Processing Requests 22

V Artificial Intelligence Engine **23**

7 Neural Network **23**

8 Decision Filter **24**

VI Test Cases **25**

9 Use Case Based Test Cases **25**
9.1 Test Case Set #1: Starting the Game 25
9.2 Test Case Set #2: Setting Player Options 26

10 System Testing **28**
10.1 Neural Network Tests 28
10.2 System Tests 29
10.3 Filter Tests 29

VII Appendix **30**

11 System Features Scale Key **30**

12 Microsoft Coding Guidelines **31**

13 Glossary **31**

References **32**

Index **33**

Part I
Executive Summary

The parallel poker player project involves designing and developing an application that exhibits the benefits of parallel computing [1] over sequential computing [2]. The application will showcase this through a demonstration of different computer-based players involved in a Texas Hold 'Em poker game [17]. Numerous key elements discussed in this document:

- Background Information - Essential information from Milestones One, Two, and Three
- Coding Standards - The conventions which the team will follow when programming
- Change Control - The method of managing suggested requirement changes
- Neural Network Description - A description of the artificial intelligence [15] engine's core element
- Decision Filter Description - A description of the filter function that follows the neural network [16] process
- Use Case Based Test Cases - Test cases based on the use cases
- System Test Cases - Test cases for the neural network [16], general system, and decision filter

The poker application will be a visually stimulating demonstration of the differences between parallel [1] and sequential computing [2].

Part II
Introduction

Milestone Four is the fourth document of the poker application's requirements and specifications. The contents of this document - documentation standards, artificial intelligence engine details, and test cases - draw upon the core elements of previous Milestone Documents One through Three [4] [6] [5], such as user needs, system features, use cases, and supplementary specifications. The test cases were developed from use cases and supplementary specifications from Milestone Two [6] and Milestone 3 [5]. The documentation standards affect the team while working on the prototypes. Based on the test cases, the team will develop an interaction prototype, which will be completed by November 2009. The design of the actual application will be determined by the use cases, supplementary specifications, and the interactive prototype. Once the design is established, the team will create a prototype. This prototype will be fully completed by February 2010. Finally, both the prototype and the incorporated design will be used to create the complete system. The system will be complete by the end of the academic year in May 2010. The objectives of this document are to concisely summarize user interaction with the poker application in terms of scenarios and screen shots. With this core information, it will be possible to develop aspects of the project in fuller detail in the future.

Part III
Background Information

Included here are excerpts from previous Milestone Documents that contribute to this document.

1 System Features

The following section describes all system features that have been approved at this time. Refer to the feature scale in the appendix for information regarding the attribute values for each feature. The developer assignments have been revised for the features and two of the features were divided up per release. The graphical display feature has been divided into two features: a basic display for the first release and an interactive interface for the second release. The ability to support multiple automated players feature has also been divided into two individual features: support for only two players for the first release and support for more than two players for the second release.

1.1 System Features List

<table>
<tr><td>Feature Number
1</td><td colspan="2">Feature
Graphical Display</td></tr>
<tr><td>Status
Approved</td><td>Priority
High</td><td>Effort
Medium</td></tr>
<tr><td>Risk
Low</td><td>Stability
High</td><td>Target Release
First release</td></tr>
<tr><td>Assigned To
Benjamin Waters
Sarah Jabon</td><td colspan="2">Rationale

To provide an attractive display for users who are utilizing the system. This feature is required for the first release as it will provide the means of communicating changes in game state and performance statistics to the user.</td></tr>
</table>

Feature Number 2	**Feature** Interactive Graphical Interface	
Status Approved	**Priority** Medium	**Effort** Medium
Risk Low	**Stability** High	**Target Release** Second release
Assigned To Benjamin Waters Sarah Jabon	**Rationale** To provide an attractive and interactive interface for users who are utilizing the system. The priority is medium because it is desired for the user to be able to interact with the final system. The user interface features are designated to Benjamin Waters and Sarah Jabon due to their development experience of front-end systems.	

Feature Number 3	**Feature** Computational Performance Display	
Status Approved	**Priority** Critical	**Effort** High
Risk Low	**Stability** High	**Target Release** First release
Assigned To Benjamin Waters Sarah Jabon	**Rationale** Constantly provides performance statistics for the analysis of the parallel computing [1], which will reinforce the superiority of parallel computing [1] to the viewer. This feature will be wrapped up in the graphical display of the first release and has a priority of critical because it is necessary to showcase the superiority of parallel computing [1].	

Feature Number	**Feature**	
4	Parallel Computing [1] Based Player	
Status	**Priority**	**Effort**
Approved	Critical	High
Risk	**Stability**	**Target Release**
Low	High	First release
Assigned To	**Rationale**	
Ian Roberts Mark Jenne	Essential for the demonstration of the parallel computing [1] technology in comparison to traditional sequential computing [2]. This feature has a priority of critical because it is a core element of the system. Ian has some experience in working with parallel computing [1] and so will be in charge of the application's usage of and connection to the parallel computing [1] API [14].	

Feature Number	**Feature**	
5	Sequential Computing [2] Based Player	
Status	**Priority**	**Effort**
Approved	Critical	High
Risk	**Stability**	**Target Release**
Low	High	First release
Assigned To	**Rationale**	
Ian Roberts Mark Jenne	Essential for performance comparison between parallel [1] and sequential [2] computing in the application. This feature has a priority of critical because it is a core element of the system. Ian will also be in charge of the sequential computing [2] player to keep it consistent with the parallel computing [1] player.	

Feature Number	**Feature**	
6	Support for Human Player	
Status	**Priority**	**Effort**
Approved	Medium	Medium
Risk	**Stability**	**Target Release**
Low	High	Second release
Assigned To	**Rationale**	
Ian Roberts Mark Jenne	User can interact with the application and play the game with the automated players. The priority is medium because it is not required for the system, but it would be beneficial to the user for demonstration purposes. The target release is the second release; the application will first be developed with only two computer-based players.	

Feature Number	**Feature**	
7	Artificial Intelligence [15] Engine Driving Computer-player Decisions	
Status	**Priority**	**Effort**
Approved	Critical	High
Risk	**Stability**	**Target Release**
High	Medium	First release
Assigned To	**Rationale**	
Mark Jenne Ian Roberts	The automated players need to demonstrate basic analysis and reasoning skills and be able to make informed decisions. The artificial intelligence [15] engine poses a high risk. If the engine is developed and does not yield intended or appropriate results in the decision-making process, the structure may need to be reconsidered with a different reasoning system. Since all of the actions of the automated players rely on this feature of the system, it is of critical importance that it be developed properly despite the level of risk present. The AI [15] engine is also the least stable as it may require alterations or redesign if it does not yield desirable results. Mark has experience in AI [15] development and so will be in charge of the development of and implementation of the AI [15] engine.	

Feature Number 8	**Feature** Support for Two Automated Players	
Status Approved	**Priority** Critical	**Effort** High
Risk Medium	**Stability** Medium	**Target Release** First release
Assigned To Mark Jenne Ian Roberts	**Rationale** The system, particularly the AI [15] engine should be able to support two automated players, one sequential [2] and one parallel [1]. This feature has a priority of critical because it is a core intended function of the system.	

Feature Number 9	**Feature** Support for More than Two Automated Players	
Status Approved	**Priority** High	**Effort** High
Risk Medium	**Stability** Medium	**Target Release** Second release
Assigned To Mark Jenne Ian Roberts	**Rationale** The system should be able to support more than two automated players. This feature has a priority of high because it is a highly desired feature of the final system.	

Feature Number 10	**Feature** No-Limit Texas Hold 'Em Poker [17]	
Status Approved	**Priority** Critical	**Effort** High
Risk Low	**Stability** High	**Target Release** First release
Assigned To Sarah Jabon Benjamin Waters	**Rationale** Game structure of the application which offers a sandbox environment to demonstrate the parallel computing [1]. All the rules of Texas Hold 'Em poker [17] will be implemented. This feature has a priority of critical because it is necessary to satisfy core requirements of the system as specified by the client.	

Feature Number	**Feature**	
11	Ability to Select Type of Player Types	
Status	**Priority**	**Effort**
Approved	High	Medium
Risk	**Stability**	**Target Release**
Low	High	Second release
Assigned To	**Rationale**	
Sarah Jabon Benjamin Waters	Gives the user the ability to designate automated players using sequential [2] or parallel computing [1]. The priority is high because it is not required for the system, but it is important for the user to interact with the application.	

Feature Number	**Feature**	
12	Ongoing Summary of API [14] Feedback	
Status	**Priority**	**Effort**
Approved	Critical	High
Risk	**Stability**	**Target Release**
Low	High	First release
Assigned To	**Rationale**	
All team members	Provides the CDP team with feedback that they can use to improve the API [14]. The priority of this feature is critical because it is one of our client's core requirements.	

2 Supplementary Specifications

The following is a comprehensive list of all system requirements. They are compiled from the features listed prior and should be sufficient to capture all requirements to which the system must conform.

2.1 Usability Requirements

1. The system must allow the user to start the poker simulation using six clicks or fewer
2. The system should use no language other than English
3. The system will use standard representations of chip value and card type in Texas Hold 'Em [17] for easy recognition
4. The system shall look and feel like standard Poker applications that can be found online, such as at [3] and [13]
5. The system shall be simple enough that it takes a user no longer than two minutes to learn to initiate the game
6. The system should display statistics about the both sequential [2] and parallel [1] players clearly on the screen so the user can view them easily

2.2 Functional Requirements

1. The system shall follow the standard rules of Texas Hold 'Em [17] as specified at [12]
2. The system must display the current game state
3. The system should show animations for bets and card actions
4. The system will allow the user to select the number of hands to simulate
5. The system must display the actions of each computer player as they occur, such as "Player 1 has folded."
6. The system must display the differences in runtime for parallel [1] players and sequential [2] players
7. The system should display the cards graphically
8. The system must allow the user to select the number of players (second release)
9. The system must allow the user to select if each player is parallel [1] or sequential [2] (second release)

2.3 Performance Requirements

1. The system will complete approximately one hand per four seconds. If the engine is still calculating when four seconds is up, then the system will force the engine to stop and the player to make a random move.
2. The system must record how long it takes a player to complete each move
3. After running for the specified number of hands, the system will display statistics in the form of average runtime of each player.
4. The system must not intentionally make the sequential [2] player inefficient
5. The system must use an artificial intelligence (AI) [15] algorithm that functions well in a parallel [1] environment

2.4 Reliability Requirements

1. The system must be stable for as many rounds are required to win a game
2. The system may experience no more than one error per 100 executions

2.5 Supportability Requirements

1. The system must support at minimum two players
2. The system should be designed so that it can be modified or updated by someone who was not involved in the initial design process
3. The developing team is responsible for maintenance until June 2010, when responsibility for maintenance will be transferred to the client

2.6 Hardware and Software Interfaces

1. The system must have a visually appealing graphical user interface
2. The system must use either a Windows [11] form or a Windows [11] presentation foundation (WPF)
3. The system must use Parallel Extensions to the .NET [7] 4 Framework

2.7 Documentation, Installation, Legal, and Licensing Requirements

1. The team members must document their experiences with the Parallel Extensions to the .NET [7] 4 Framework [14] The system's use of .NET [7] code is under a license from Microsoft [9] for Visual Studio [10]
2. The system must have all design considerations and actions documented
3. The system need not include a user manual. However, it is recommended that the system have one

4. The system may be used to its full extent by the client at any time
5. The system will only need a select number of files to run. It will simply be a desktop application
6. The system will not require the user to install any applications

2.8 Design Constraints

1. The system must implement the API's [14] provided by Microsoft's [9] .NET [7]
2. The system must run on a Windows [11] platform

2.9 Training Requirements

1. The system should require no training to use
2. The system should have a "Help" option at the user's disposal

2.10 Deliverables

1. Milestone Documents One to Five
2. Final deliverable, which includes updated versions of all Milestone Documents, client comments, and lessons
3. The poker application
4. Source code
5. Individual engineering journals
6. Reports with feedback on the Parallel Extensions to the .NET [7] 4 Framework

3 Use Cases

Use cases describe the flow of events when a user interacts with the system.

3.1 Use Case #1: Starting the Game

Name Starting the Game

Description Describes how the user starts the simulated Texas Hold 'Em [17] game

Actors A potential Microsoft [9] customer or a Microsoft [9] associate who is showcasing the application

Pre-Conditions

1. User has opened application
2. User has specified player options

Basic Flow

1. User clicks "Start" button (alternate flow 1 possible)
2. System begins to simulate automated poker game (alternate flow 2 possible)

Alternate Flows

1. User chooses to exit program
   (a) User clicks "Exit" button
   (b) System exits
2. System does not respond
   (a) System restarts

Post-Conditions The user watches the simulation take place, or the user is unable to watch the simulation because of a system failure.

Other Stakeholders Microsoft's [9] CDP team and other people watching the computer screen for the demonstration

Systems/Subsystems The computer's hardware

Special Requirements

1. The system should not allow the user to start a simulation while a simulation is already in progress
2. The system should notify the user if there is a failure when it starts the game

3.2 Use Case #2: Setting Player Options

Name	Setting Player Options
Description	Describes how a user sets the options for the simulation's players
Actors	A potential Microsoft [9] customer or a Microsoft [9] associate who is showcasing the application

Pre-Conditions

1. Game is functioning properly
2. Parameter menus are functioning properly

Basic Flow

1. User selects "Select Player Types" button (alternate flow 1 possible)
2. User selects the number of players
3. System prompts user to select "Parallel Player" or "Sequential Player" for each computer-based player
4. User selects the type of each computer-based player (alternate flows 1 and 4 possible)
5. User selects "OK" button to indicate a finalized selection (alternate flows 1, 3, and 4 possible)
6. System updates game and changes labels by the players appropriately (alternate flow 2 possible)
7. System returns to main game screen

Alternate Flows

1. User chooses to exit program
   (a) User clicks "Exit" button
   (b) System exits
2. System does not respond
   (a) System restarts
3. User neglects to select one or more types of players
   (a) System displays message to user, stating that the types of players must be selected in order to continue
   (b) System returns to parameter selection menu
4. User cancels the change to player types
   (a) User clicks "Cancel" button
   (b) System returns to main game screen

Post-Conditions The types of players have changed to the user's preferences, or the types of players remain unchanged.

Other Stakeholders Microsoft's [9] CDP team and other people watching the demonstration

Systems/Subsystems The computer's hardware

Special Requirements

1. The system should not allow the user to change the types of players while a simulation is in progress
2. The system should notify the users of the player types they have selected

3.3 Use Cases and Feature Mapping

Each use case incorporates the corresponding features listed in the table.

Use Case	Related Features
#1	1, 8
#2	1, 3, 4, 7, 9

4 Scenario

Mike had a chance to try out the poker application. The application had an empty card table with a large "Start" button in the center with a brief description of how the parallel and sequential players worked in the middle. He clicked on the "Start" button, and a popup appeared. The popup had settings on it, such as the number of players and the number of games. Mike set the number of players to three. He selected Player 1 as a parallel player, Player 2 as a sequential player, and Player 3 as a human player. On the same popup, he set the number of games to one. Once he was done setting his preferences, he clicked the "OK" button.

When the popup closed, the screen had place set for Players 1 and 2 on the left and right sides of the screen respectively. The deck was on the top of the screen, and Mike's place was at the bottom of the screen. All players, including Mike, had a pile of chips next to their hands. Some of Player 1's chips moved into the middle of the table, and Player 1's total amount of money went from \$500 to \$490. The running total of the pot, which was displayed in the middle of the screen, went up to \$10. Then Player 2 contributed \$5 into the pot. As Mike watched, he noticed that the game status bar on the bottom of the screen stated that the big blind was \$10 and the small blind was \$5. The game status bar then displayed the message: "Players have placed pre-flop bets." Then, cards were dealt to all three players.

Three pairs of cards were displayed on the screen. Players 1 and 2 had face-down cards. However, Mike's cards were face-up. He had a five of clubs and a king of hearts. Next to Mike's hand and chips were four buttons: "Call," "Raise," "Check," and "Fold." He noticed the game status bar changed to "Player 1 called. It took Player 1 8.254 ms to make this move." as some of Player 1's chips went to the pot. Player 2 also called. The game status bar also noted that Player 2 called and how long it took for Player 2 to make that move. It was Mike's turn, and he wanted to call. So, he clicked the "Call" button. Five of his chips went into the pot, and his amount of money went down by \$25. Players 1 and 2 checked. The game status bar noted this. Mike checked by clicking "Check."

Then, three cards showed up face up on the top of the screen next to the deck. They were the dealer's cards. The cards were: a three of diamonds, a six of clubs, and a queen of spades. As Player 1's chips went into the pot, the game status bar displayed: "Player 1 raised \$20. It took Player 1 7.345 ms to make this move. The current bet it \$30." Player 2 raised the bet another \$20, as the game status bar noted. Mike wanted to stay in the game, so he called by clicking the "Call" button. It was Player 1's turn, and so Player 1 called. Player 2 checked. Finally, Mike checked by clicking "Check."

A fourth card showed up face up on the top of the screen next to the other three dealer's cards. It was a four of diamonds. Player 1 raised the bet by \$50, as specified by the game status bar. Player 2 called. Since Mike did not think he had a chance of winning, he decided to fold. So, he clicked the "Fold" button. His cards on the screen flipped over, and the buttons to call, check, raise, and fold became grayed out. Player 1 checked, and the status bar updated correspondingly.

The fifth card was then displayed on the top of the screen. It was a ten of clubs. Player 1 raised \$50. Player 2 called. Then, Players 1 checked. Mike watched as the game status bar updated and the chips went into the middle of the table.

Since the game was over, it was time to show cards. Both Player 1's cards and Player 2's cards flipped over to show the face value. Player 1 had a five of diamonds and a queen of clubs. Player 2 had a three of spades and a two of spades. All the chips in the pot moved to Player 1's side, and Player 1's money total updated. Also, the game status changed to "Player 1 wins."

Mike watched as all of this happened. Then, the statistics information at the bottom of the screen updated. These statistics included measures of efficiency such as the average time to make each move and the average number of possibilities considered. Mike viewed the statistics on the parallel and sequential players. In this case, the parallel player was more efficient. Mike enjoyed working with the poker application, and he decided to play again.

Part IV
Documentation Standards

The coding standards describe how the actual code will be written. The change control describes how requirement changes will be handled.

5 Coding Standards

This section details the coding standards to be utilized through the development phase of this project. The coding standards include reliability standards, source code standards, naming conventions, and general coding paradigms. These standards are partly derived from an internal document at Microsoft [9] about coding guidelines.

5.1 Reliability Standards

- All code must be compiled at the compiler's maximum warning level. A warning can be disabled in an instance where the team deems it appropriate without it having and side-effects on the application. In general, though, the build environment will be set to compile everything with maximum warnings.
- Every data entry input point to a module must perform a full input validation through the use of assertions to ensure that proper values and buffers are being supplied.
- All compiler warnings must be fixed. Warnings will be treated as seriously as errors.

5.2 Source Code Standards

- For clarity and ease of use, there will only be a single class definition per source file. Nested private classes or enumerators, though, are permitted to reside in the same source file as their governing class.
- Code that is no longer needed should be removed from the source base. Do not simply comment it out as leaving dead code behind can lead to confusion and security risks. If necessary the subversion control system can be utilized to recover old fragments.
- Maintain local formatting and code style in source files. When adding new code to an existing source file, the new code should abide by the file's existing coding style.
- Class members will be thoroughly documented. It is important that comments describe the logic behind particular functions, why they are being used, and why they are important.

5.3 Naming Conventions

The following three types of letter casing will be used for different identifiers.

- **Pascal Casing.** Uses an uppercase first character and then uppercases the first letter of subsequent attached words. No underscores are used to separate words. For example: NeuralNetwork

- **Camel Casing.** Similar to Pascal casing with the exception being that the first character of the first word is lowercase. For example: neuralNetwork
- **Loud Casing.** This casing uses all uppercase letters with individual words separated by underscores.

Identifier Type	**Casing**	**Prefix**
Local Variable	Camel	
Global Variable	Camel	g_
Non-Static Member Variables	Camel	m_
Static Member Variables	Camel	s_
Thread-Local Static Member Variables	Camel	t_
Functions and Methods	Pascal	
Functions and Method Arguments	Camel	
Classes	Pascal	
Structures	Pascal	
Interfaces	Pascal	
Enumerators	Pascal	
Enumerator Values	Pascal	
Macros and Constants	Loud	

5.4 General Coding Paradigms

- Keep source code succinct and to the point. Source code needs to be lean and devoid of obsolete or deprecated code. Excessive levels of abstraction and encapsulation need to be avoided as well.
- Strive to use completely verifiable code when possible. Premature attempts at optimization can lead to unsafe code blocks.
- Always handle exceptions properly. Using blocks that catch all raised exceptions and allow program execution to continue are dangerous and usually mask serious program errors.
- Keep memory structures as small with as little redundancy as possible.

6 Change Control

6.1 Receiving and Processing Requests

When our client has a change request, it should be sent via email. Although the client will most likely discuss the change on a conference call, the team must receive an email documenting the change request. Numerous aspects of the requested change should be included in the email. These include:

- A reference to which feature or requirement the client is requesting to change
- A detailed description of the requested change

- On which release that the client expects to see the change
- The importance of this change

With this information, the team will fill out a change request table. This table will have more detailed information about the changes affects on the system. It is important that the team complete this table, as it will aid in the teams decision about whether or not to accept the change. Below is the template for the change request table.

Source	The source of the change request, such as the client
Project Aspect	Which part of the project the change falls into vision, requirements, use cases, design, code, or tests
Description	A description of the change
Affected Features	A list of features affected by the change
Affected System Requirements	A list of system requirements affected by the change
Impact on Stakeholders	A description of how the change will affect the stakeholders
Impact on System Functionality	A description of how the change will affect the functionality of the system
Impact on Stabilization	A description of how the change will affect the stability of the system
Final Decision	The final decision of whether or not to implement the change

This change request table will be completed for every change request. The team will carefully analyze each item within the table to make the final decision about accepting the change. The team will make the decision together. These change request tables will be stored in the "Change Control" section within the team's document binder.

6.2 Processing Requests

After the team has made a decision about whether or not to accept the change, the team must update the project documentation. All change request tables accepted or rejected will be kept in a separate section of the teams document binder. This section will be titled "Change Control. Within that section, there will also be a current version of the features and system specifications. The team will update these after each change request. The old versions of the features will be kept within that section, but they will be clearly marked as old versions with the date that they became ineffective.

Part V
Artificial Intelligence Engine

7 Neural Network

Neural networks [16] are computer based simulations of neurons and how they function in living organisms. This model, however, is highly simplified. In its simplest form a neural network [16] consists of discrete layers each composed of nodes. Each node is modeled as a filter of sorts, information is passed to it from every node in the previous layer and the node adjusts this data by a factor specific to each source. This data is then passed to each node in the following layer. In this way many different circumstances can be taken into account. This neural network [16] as described is called a pass forward neural network [16] since the data only moves one direction through the network. This network adjusts to changing circumstances by adjusting the weights that each node applies to the inputs they receive. This model work only when there are a defined number of inputs and outputs to a system. The game state consists of a relatively small number of variables.

1. The cards the player has (Two values)
2. The cards on the table (Three to five values)
3. The bet to call
4. The pot size
5. The amount of money the player has put in the pot
6. The amount of money the player has
7. The set of amounts of money the player's opponents have
8. The set of actions last taken by the player's opponents in that round, such as pass, call, or raise

There are also a defined number out outputs that capture all possible actions.

1. Pass (Y/N)
2. Fold (Y/N)
3. Call (Y/N)
4. Raise (Y/N)
5. Amount to raise (Second release)

Since the game state consists of relatively few variables and neural networks [16] can run in parallel efficiently, neural networks [16] are an ideal artificial intelligence (AI) [15] algorithm for our product.

8 Decision Filter

When the neural network [16] outputs a move, the move must be assessed for legality. The system will check whether the moves are legal or not with a function that will run after the neural network [16]. This function will be referred to as the decision filter, as it takes the neural network's [16] move decisions and filters out all the illegal moves. The filter's inputs will include:

- The neural network's output, which will include a boolean for each action (Pass, Fold, Call Raise)
- The current game state, which will have information about the actions of all players in the round so far, as well as the information about the round as a whole.

Given these two inputs, the decision filter will check the legality of the move with a series of conditional statements. Ultimately, the decision filter will output a final move that will be reflected in the game play. The filter will have a set of legal moves based on the game state to which it can compare the output of the neural network [16].

There are four move booleans: Pass, Fold, Call, and Raise. The neural network [16] will output a value of high or low for each of these four move booleans. If the boolean is high, then the player should perform that action. For instance, if the Fold boolean is high and the Pass, Call, and Raise booleans are low, the player will fold. However, there are three possibilities for the legality of the neural network's [16] output:

1. More than one of the four move booleans are high
2. The move boolean that is high is a legal move given the game state
3. The move boolean that is high is an illegal move given the game state

In case one, the filter will output a random legal move. The team considered outputting a directed move choice. However, a random move will introduce variability into the system that could help train the neural network [16]. In case two, the filter will simply output the neural network's [16] choice. In case three, the decision filter will output the most similar legal move. For instance, if the move is to raise an amount smaller than the big blind, the filter will output the move of raising by the big blind. These legal alternative moves will be programmed into the decision filter. The decision filter will analyze the output of the neural network [16] and output the final move that will be reflected in game play.

Part VI
Test Cases

9 Use Case Based Test Cases

Use case based test cases describe how the user's interaction with the system will be tested.

9.1 Test Case Set #1: Starting the Game

Scenario Matrix

Scenario Number	Originating Flow	Alternate Flow
1	basic flow	
2	basic flow until 1	alternate flow 1
3	basic flow until 2	alternate flow 1
4	basic flow until 2	alternate flow 2

Test Cases

Test Case ID	Scenario	Description	Condition: User has entered player parameters	Condition: "Exit" is selected	Expected Result
1	1	User starts the game	User has entered the parameters for the AI bots	Invalid	User successfully runs game simulation and views results.
2	2	User exits program during main screen	N/A	User clicks "Exit" button on main screen	Application exits
3	3	User exits program during simulation	N/A	User clicks "Exit" button while simulation runs	Application exits
4	4	Application does not respond	N/A	Invalid	Application restarts

9.2 Test Case Set #2: Setting Player Options

Scenario Matrix

Scenario Number	Originating Flow	Alternate Flow
1	basic flow	
2	basic flow until 1	alternate flow 1
3	basic flow until 4	alternate flow 1
4	basic flow until 4	alternate flow 4
5	basic flow until 5	alternate flow 1
6	basic flow until 5	alternate flow 3
7	basic flow until 5	alternate flow 4
8	basic flow until 6	alternate flow 2

Test Cases

Test Case ID	Scenario	Description	Condition: "Exit" is selected	Condition: "Cancel" is selected	Condition	Expected Result
1	1	User sets player options	Invalid	Invalid	N/A	User leaves menu and starts game
2	2	User exits program during options screen	User clicks "Exit" button on parameter selection screen	Invalid	N/A	Application exits
3	3	User exits program while selecting parameters	User clicks "Exit" button on parameter selection screen	Invalid	N/A	Application exits
4	4	User cancels out of the menu while selecting parameters	Invalid	User clicks "Cancel" button on parameter selection screen	N/A	Application returns user to main screen
5	5	User exits program after selecting parameters	User clicks "Exit" button on parameter selection screen	Invalid	N/A	Application exits
6	6	User tries to only select one type of AI bot	Invalid	Invalid	User only selects one type of AI	Application prompts user to reselect bots
7	7	User cancels out of the menu after selecting parameters	Invalid	User clicks "Cancel" button on parameter selection screen after choosing options	N/A	Application returns user to main screen
8	8	Application does not respond after user clicks "OK"	Invalid	Invalid	User has selected parameters and clicked "OK"	Application restarts

10 System Testing

The following section describes the test cases that will be used to test the features of the system that do not involve the user directly.

The test cases will be defined as the number,a description, the feature to be tested, the relevant inputs to the feature, and the expected result.

10.1 Neural Network Tests

Number	**Description**	**Feature**	**Data 1**	**Data 2**	**Result**
1	Test that the network generates output	7	N/A	N/A	Anything
2	Test that the Network adapts after 50 iterations	7	a Specific game state leading to an illegal action	N/A	A different preferably legal action than originally chosen
3	Test the networks behavior is consistent w/o training enabled	7	A game state	An identical game state a second time	Identical actions taken on both occasions
4	Test that the learned behavior persists between runs	7	A game state	An identical game state after restarting application	The same response

10.2 System Tests

Number	Description	Feature	Data 1	Data 2	Result
1	Test that the game ends if one player has all the chips	10	A game state where one player has all the chips	A hand has completed	The player with all the chips wins
2	Test the player appropriately wins a round	10	A game state	a hand has just ended	The player with the best combination of cars wins as specifies in the rules.
3	A player is removed from the game	9/10	A game state with more than 2 players and one player having no chips	a round has just ended	The player with no chips is removed from the game.
4	Statistics for runtime are reported accurately	3	A computer player has acted	task monitor reports CPU time	Both the games performance metric and the monitoring programs value should be identical

10.3 Filter Tests

Number	Description	Feature	Data 1	Data 2	Result
1	Test that a player cannot raise by more than is in his stack	10	The amount of the player's stack	The raise amount	The player raises the amount of his stack
2	Test that a player cannot raise by less than the amount of the big blind in the first two rounds	10	The big blind	The raise amount	The player raises the amount of the big blind
3	Test that a player cannot raise by less than twice the amount of the big blind in the last two rounds	10	The big blind	The raise amount	The player raises the amount of twice the big blind
4	Test that a player cannot raise a fourth time in a round if there are more than two players	10	How many players	How many raises have been made that round	The player calls

Part VII
Appendix

11 System Features Scale Key

Status - The status of a feature pertains to the progress or current state of the feature during the planning and design stages of the system specified by the project.

Priority - Priority here refers to the importance of the development of a feature based on its significance to the overall system. Of the features listed, their priorities range from medium to critical levels. The critical features are labeled as such because they are necessary for the proper functionality intended of the application as specified by the client.

Effort - The effort attribute of a feature describes the amount of time and resources that need to be designated to the development of that feature based on the level of priority and overall importance to the structure of the system. The critical features compose the base functionality of the entire application and represent the goal of the project as a whole. All of the high priority features are integral to the proper functioning of the application as well, but are not quite of dire importance.

Risk - This attribute represents the likelihood that a feature will cause unintended results, such as cost overruns or delays. A low risk represents a lack of potential for undesirable events.

Stability - The stability of a feature reflects the probability that the feature will change or the team's understanding of the feature will change throughout the development of the feature. The stability of a feature relates to the risk that feature presents. If a feature causes unintended results, it will very likely be unstable and need to be redesigned.

Target Release - This records the intended version of the product or system in which the feature will first appear and usually reflects the priority of and effort delegated to that particular feature.

Assigned To - Refers to who within the development team will be responsible for the implementation of the feature. Given that we have four members on this project team, each with capabilities maybe not seen in the others, the features must be distributed for development to the most capable person or persons within the team.

Rationale - The reason or reasons for intended implementation of a feature can be required by the system, mandated by the client, or just a feature decided upon to better the user experience in using the system.

12 Microsoft Coding Guidelines

13 Glossary

- C# - an object-oriented language that can be used to create a wide variety of applications, services, and tools [8]
- Microsoft Corporation - A multinational corporation that has five business segments: client, server and tools, online services business, Microsoft business division, and entertainment and devices division [9]
- .NET Framework - A platform build by Microsoft that provides a common set of APIs and a consistent programming model [7]
- Neural Network - A network made up of artificial neurons that mimic the properties of biological neurons [16]
- Parallel Computing - using multiple resources simultaneously to solve a computation problem [1]
- Sequential Computing - solving a computational problem with sequential processing [2]
- Texas Hold 'Em Poker - A popular card game with multiple players. Players can use any combination of five community cards in combination with their own cards to create the best hand according to numerous rules. [17]
- Visual Studio - an integrated development environment produced by Microsoft [10]

References

[1] Blaise Barney. What is Parallel Computing?, 2009. https://computing.llnl.gov/tutorials/parallel_comp/#Whatis.

[2] Barney Blaise. Overview: Sequential Programming, 1995. http://www.hku.hk/cc/sp2/workshop/html/parallel-intro/ParallelIntro.html#sequential%20programming.

[3] Pogo Games. No Limit Texas Hold'Em, 2009. http://casino-games.pogo.com/games/no-limit-texas-holdem-poker.

[4] Sarah Jabon, Mark Jenne, Ian Roberts, and Bennie Waters. Milestone One, 2009.

[5] Sarah Jabon, Mark Jenne, Ian Roberts, and Bennie Waters. Milestone Three, 2009.

[6] Sarah Jabon, Mark Jenne, Ian Roberts, and Bennie Waters. Milestone Two, 2009.

[7] Microsoft. .NET Framework Overview, 2008. http://www.microsoft.com/net/Overview.aspx.

[8] Microsoft. Getting Started with Visual C#, 2009. http://msdn.microsoft.com/en-us/vcsharp/dd919145.aspx.

[9] Microsoft. Microsoft, 2009. http://www.microsoft.com/en/us/default.aspx.

[10] Microsoft. Microsoft Visual Studio, 2009. http://www.microsoft.com/visualstudio/en-us/default.mspx.

[11] Microsoft. Windows, 2009. http://www.microsoft.com/WINDOWS/.

[12] PokerListings.com. Official Texas Holdem Rules and Game Play, 2009. http://www.pokerlistings.com/poker-rules-texas-holdem.

[13] PokerStars.com. The World's Largest Poker Site, 2009. http://www.pokerstars.com/.

[14] Wikipedia. Application Programming Iinterface, 2009. http://en.wikipedia.org/wiki/Application_programming_interface.

[15] Wikipedia. Artificial Intelligence, 2009. http://en.wikipedia.org/wiki/Artificial_intelligence.

[16] Wikipedia. Artificial Neural Networks, 2009. http://en.wikipedia.org/wiki/Artificial_neural_network.

[17] Wikipedia. Texas Hold 'Em, 2009. http://en.wikipedia.org/wiki/Texas_hold_%27em.

Index

.NET Framework, 14

, 13

Application Programming Interface (API), 13, 14
Artificial Intelligence (AI), 4, 9, 13, 21

Change Control, 19
Coding Standards, 18
Concurrency Development Platform (CDP), 11

Decision Filter, 21

Executive Summary, 4

Glossary, 29

Introduction, 5

Microsoft, 14–16

Neural Network, 21

Parallel Computing, 4, 8
Part: Appendix, 28
Part: Artificial Intelligence Engine, 21
Part: Background Information, 6
Part: Documentation Standards, 18
Part: Executive Summary, 4
Part: Test Cases, 23

Sequential Computing, 4, 8
Supplementary Specifications, 12
System Features, 6
System Features Scale Key, 28
System Testing, 25

Table of Contents, 3
Title Page, 2

Use Case Based Test Cases, 23
Use Cases, 15