# A FREE AND FLEXIBLE POKER ENVIRONMENT

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Daniel George DeBilt

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

June 2010

Fargo, North Dakota

North Dakota State University
Graduate School

**Title**

**A FREE AND FLEXIBLE**

**POKER ENVIRONMENT**

**By**

**DANIEL DEBILT**

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

**MASTER OF SCIENCE**

# ABSTRACT

DeBilt, Daniel George, M.S. , Department of Computer Science, College of Science and Mathematics, North Dakota State University, June 2010. A Free and Flexible Poker Environment. Major Professor: Dr. Kenneth Magel.

This paper introduces a framework and software that allows poker players to create and play original and custom poker games, through a TCP/IP connection, for free. This paper describes how the concept of playing user-created poker games over the Internet is not known to currently exist in a flexible, private, and free environment and also critiques what currently does exist and is available for use. This paper also summarizes the software development process used and the deliverables that ultimately led to a working software application. Future version features and application extensions are also discussed that may enhance the user experience, as well as future research projects.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# CHAPTER 1: INTRODUCTION

Most all poker players know and have tried "5 card draw". That is the game where each player is dealt 5 cards from a standard deck of 52 playing cards, with an option to discard some of those cards later in the game and re-dealt more from the deck to get their hand back to 5 cards again. The overall goal of that game, and any poker game, is to create the best ranking poker hand at the table, which usually ranges from two up to ten players total. Las Vegas Poker & Blackjack, on the Intellivision game system [14], was this author's first introduction to the poker world at an age of six years old. Since that time, the poker world has rapidly evolved over the last few decades and has adapted the basic set of poker rules to create new games with new entertainment value and possibilities.

Poker was said, according to Wikipedia.org [13], to have been found in the 15th century through a game called Pochspiel. Pochspiel was a game that incorporated hand rankings, and the human element known as bluffing. Bluffing, in poker, is the act of a player trying to convince their opponents, who are sitting at the same table, that they have a higher ranking hand in order to get them to fold or surrender their hand. In nearly all poker games, the players are not allowed to see the contents of each other's hands. Playing poker is not considered complicated. Poker games are defined by a set of sequences that must occur in a predetermined order, ultimately leading to a showdown of each player's hand. The showdown, as it is usually called, is where all remaining players show their cards to determine who has the best ranking hand. Normal poker sequences include the ante, dealing cards, betting, and discarding.

## 1.1. Hand Ranks

The hand ranking premise of Poker is simple. A standard deck contains 52 playing cards, which is composed of 4 distinct suits {spades, hearts, clubs, diamonds}, and 13 values per suit {two, three, four, five, six, seven, eight, nine, ten, jack, queen, king, ace}. In poker, the suits themselves have equal rank compared to each other, though the individual card values are ranked in increasing order, from two to ace. If a player were dealt only one card from the deck, their hand rank is determined by that card alone. Things change when a player is playing a poker game that allows players to have more than one card in their hand, which is usually the case. Having the same card value (e.g. two aces) in a hand is usually better than not having the same card value on different cards. It gets more complicated when more cards are allowed in a hand. If all 5 of a player's cards are in contiguous increasing order (e.g. three, four, five, six, and seven), the rank of their hand is called a straight; seven high. There is also a special rank if 5 of a player's cards share the same suit as well, and that is called a flush. A flush is better than a straight. Refer to Poker Hand Ranks (in the Appendix) for all the normal types of poker ranks and how they compared to each other in power. No matter how many cards the game allows any specific player to have, the best hand rank always wins in poker, though ties are possible since a deck of playing cards has multiple cards of the same number and some cards, community cards, can be shared by all players. So for example, if an unusual poker game allowed one player to have 3 cards, yet another player to have 6 cards, the player with 3 cards can still create a better ranking hand than the player with 6 cards.

## 1.2. Ante, Betting, and Actions

In "5 card draw", all players who wish to participate in the game must ante. Ante, is the act of a player declaring their intent to participate in the upcoming deal. The ante is usually denoted by an asset value, such as money, coins, or chips, though the game can be played for free, with no monetary or asset value attached. In poker games that use antes, the players who are going to participate in the next deal, or instance of the game, are dealt a certain number of cards, which depends on the poker game being played. After the players have had a chance to look at their cards and evaluate their hands, a betting round usually ensues. Betting normally starts with the person to the left of the dealer, moving clockwise until every player has had a chance to act. The first player to act has an option to bet, check, or even surrender (fold) their hand, though the player has nothing to gain by folding when no other person has placed a bet yet. If the first player checks, the next player will also have that same option, as long as all players prior have checked. However if the first player makes a bet, all subsequent players must at least equal that bet (known as a call), or else they are forced to fold their hand and be eliminated from the current deal. Players who are calling a previous bet also have the option to raise the bet. The bet and raise amounts in question are dependent on the rules of the game. Each game has its own specific, yet flexible, rules that are usually decided by the players prior to playing. Once all dealing and betting rounds have completed, and at least 2 people are still remaining, they must show their cards and the best ranked hand will be determined the winner. Sometimes the hand does not make it through all the predetermined sequences. This can happen if a player has bet a significant amount of chips, and all other players do not wish to call that bet, folding their hands. The player that had bet that significant amount of chips usually has the option of not having to show their cards to the other players who have folded their hands. When a player does not show their cards, this is called mucking. This allows people to create a bluffing strategy.

## 1.3. Popular Poker Games

As far as types of poker, there are many variations. The most popular and common are the following:

1. Texas Hold'em – this is a poker game where players are dealt two private cards initially. After a betting round, three more cards are dealt on the table face up (suit and value showing) as community cards, which are cards that every player can use in their hand. After another round of betting, a fourth community card is dealt. After yet another round of betting, a fifth community card is dealt. A final betting round takes place, and then it is the showdown, assuming at least two

players have made it that far into the hand. This game is currently the most popular poker game at this time. Most televised and high profile poker competitions showcase this game.

2. Five Card Draw – one of the older, yet still popular poker games. Players are dealt five private cards. A betting round occurs, and then players are allowed to discard a certain number of their cards from their hand. The dealer will replenish the amount of cards they have discarded with new cards from the deck. This allows players a chance at creating a better ranking hand from their initial set of cards. After the dealer has replenished all the discarded cards, a final betting round occurs before the showdown.
3. Seven Card Stud – in this game, the players are dealt two private cards initially. A betting round occurs, and then each player is dealt a card face up in their hand (they cannot actually hold that card, as it must be visible to the other players). The process of dealing a face up card and betting occurs for four total cards. Finally, the last, seventh, card is dealt privately to the players, and a final betting round occurs before the showdown.

4. Omaha Hi/Low – This is similar to Texas Hold'em, except that each player gets 4 private cards. Each player must use two cards from their hand to form a "high" hand, which is basically trying for the best normal poker hand. Each player must also use two, and only two, cards from their hand to form a "low" hand, which is a set of five cards whose face values are not greater than 8, and do not share the same face value as another of the cards in that five card set (e.g. 8-7-5-3-2). The cards used to form the "high" and "low" hands can be shared.

## 1.4. Problem Statement

There are many poker players who belong to a social group that includes other poker players who play their own custom games by their own rules. A social group of poker players will usually consist of friends or peers, and the type of poker games played can vary, and those games aren't necessarily the popular games someone may find on the major poker sites available through the Internet. Some groups play with wild cards, some play games that have different sequences or games that deal the players a different number of cards to create a hand from. As time goes on, some of these social groups dissolve and cease to exist. The big reason for the social groups dissolving is lack of physical access to that group in a reasonable fashion or schedule. Some players have to move for job or other reasons, and can no longer partake in a routine meeting with those same players to play poker. This author has experienced this type of problem when poker playing friends had left after completing college. In most cases, it is not economically feasible to maintain a routine poker playing schedule when the members of the social group no longer live in the same area.

The major problem is that there is no known software that can allow the social groups to continue playing the same custom games, and by their own rules over a network connection, such as the Internet. There is no software package that allows the social groups to all play online in a free, private, and in a flexible environment. Flexible in the terms that they can play the type of custom poker games that they are used to playing when they were all under the same roof and at the same poker table. Even the software packages that a player would have to purchase are still subject to the limitations of the software, which are usually just the most popular poker games. Of the software packages and poker sites that are free, most do not allow a user to create a private table, most do not allow the user to configure the stakes, and none of them allow the user to change the game at the table. Most importantly, all of the available packages do not offer what the social groups would desire the most, custom poker games. Of the newer types of software that are big in today's online gambling circuit, such as PartyPoker.com [0], PokerStars.com [10], and CakePoker.com [11], all offer an impressive visual software package with numerous features, though they all do not meet all the required needs. It may be free to use some aspects of their service, but the tables are not private, the games are limited, and nothing is configurable. As it stands currently, there is nothing available that would meet the needs of a social group of players that enjoy playing poker on a frequent basis, remotely. The assumption is that there are numerous social groups of poker players that would desire this type of software.

## 1.5. **Objectives**

The overall objective is to obtain, extend, or develop software that would meet the needs of not only the social group of poker players that this author plays poker with, but also attempt to meet similar needs of other social groups of poker players looking for the same type of system. This is an important point of view and concept, for the following reasons:

1.) If this software can bring back the frequency of poker playing within any social group, it could keep help maintain their friendships. In this author's personal experiences, the less a person sees someone, and the less they socialize with someone, the more they drift apart. The solution would further reinforce and help maintain those friendships.

2.) It would be free, besides the burden of possible development to introduce an initial version of the software.

3.) It would allow privacy. A group of players would be able to chat and not have to worry about any unknown visitor frequenting the virtual poker table.

4.) It would be flexible enough to allow the social poker group to play their custom games by their own rules, and also not require the players to have to get up and sit down at another table to play a different poker game. The solution could understand a poker game created, as long as it follows the rules of how to create a poker game outlined later in this paper. Most players would never think of being able to play "Blind Man's Bluff" online, with their fellow poker friends.

# CHAPTER 2: LITERATURE REVIEW

Concentration on specific areas of research was employed prior to architecting and creating a potential software application that would be needed to solve the problems to be address. First, given that this application would have to be able to communicate over the Internet, application level protocols or other standards were researched. Secondly, other online poker software applications were critiqued, to get a feel for the visual requirements that could help enhance the user experience. Another area that is always important and a major focus when deploying network applications, is security. If the application was not secure, and open to anyone who knew how to intercede during "private" game play, it would not meet the privacy needs that would need to be solves.

## 2.1. Application Protocol(s)

An application protocol is an important piece in any application being developed which must communicate to separate pieces over a communication medium. Even if an application doesn't use a network for communication, there can be separate components of an application that need to communicate with each other, and a good way to accomplish that could be to define or use existing protocols.

### 2.1.1. **The Art of UNIX Programming**

This online book, "The Art of UNIX Programming" [1], is focused on providing information for UNIX programmers, and how to design software for UNIX systems. Though the book is aimed toward UNIX, some of its general ideas can be applied toward other operating systems and development environments. Focus was placed on the chapters regarding application protocols, given that the project that this paper encompasses would require an application level protocol for communication.

The book argues that when an application protocol can easily be parsed or interpreted by an eyeball, "many good things become easier." Transaction dumps become easier to interpret, and test loads become easier to write. It is agreed with those claims in that it is easier to see things when they are in readable form. An example is using "nvarchar" fields in a database to store data. By using any query analyzer tool, the actual readable data in the field can immediately be seen. However, there is disagreement in some ways, because it can come with a cost. A readable form would be less secure and normally less efficient (more bytes stored and transferred) compared to compressed, non-readable data. Encrypted data may enlarge the size of the data being transmitted, and though it is also not read-able, it would less compromise the security of the application.

The author discusses three of the widely used mail application protocols used on the Internet, which are SMTP, POP3, and IMAP. Then the author transitions to HTTP, IPP (Internet Printing Protocol), BEEP (Blocks Extensible Exchange Protocol), and finally XML based protocols such as XML-RPC, SOAP, and Jabber. The author mentions that there is a developing trend of application protocol design toward using XML. XML is a protocol that can achieve great flexibility and would be an easy protocol to maintain for communication when new, added information needs to be exchanged. An XML interface allows an interface to become more easily backward compatible.

Given that hardware and bandwidth have made great progress in the past few decades, and that the conceptual view of the information that may need to be exchanged between poker players is somewhat minimal, an XML based application protocol would not add any significant performance impact on the project to be developed.

## 2.2. **Visual**

It's important to critique and review current online poker applications because they will give a good base point as to what this project should try to encompass and build on in a visual and utility sense. Not only does it help provide a good idea on what a GUI could or should look like, they can also be used to help gather requirements of the system and GUI. Many of these online poker applications have been around for at least 5-6 years and are considered very popular and the places to play poker, both for free and for real money. Not all of the major poker sites were reviewed, though a handful of the major ones were.

### 2.2.1. **pokerstars.com**

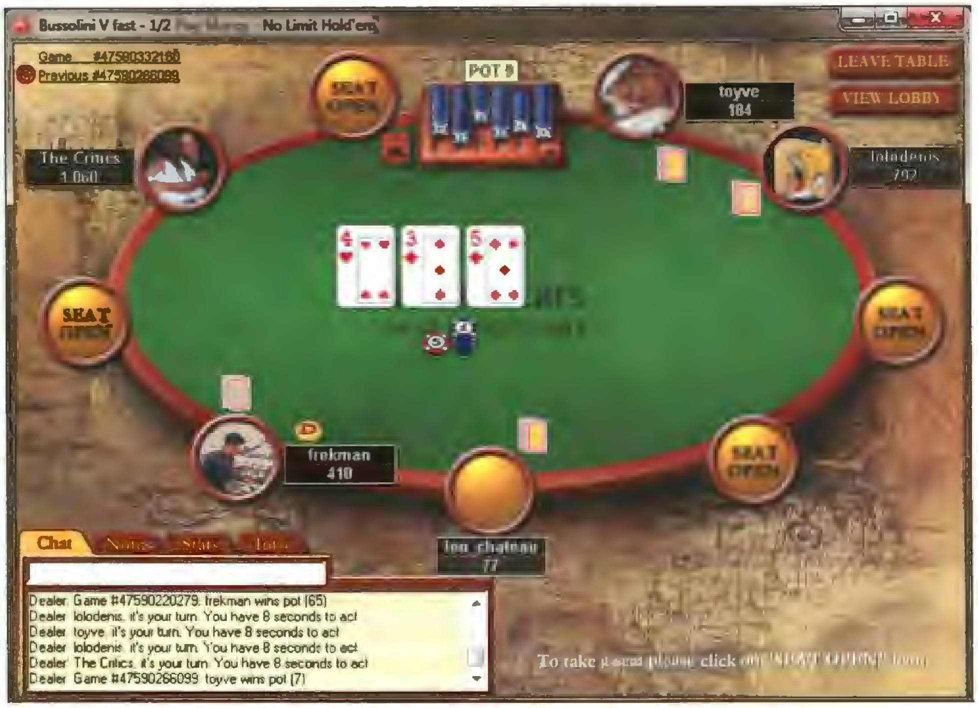
pokerstars.com launched in late 2001, and they are deemed, by themselves, as the world's largest poker site.

> "We are the world's largest poker site, with millions of members worldwide. On December 30 2007, we entered the Guinness

World Record books for having the most players simultaneously playing online at PokerStars - 151,758!"

This record has most likely been broken since then, possibly by the Texas Hold'em game application that can be played on Facebook, created by a company called zynga. The review of that software is coming below.

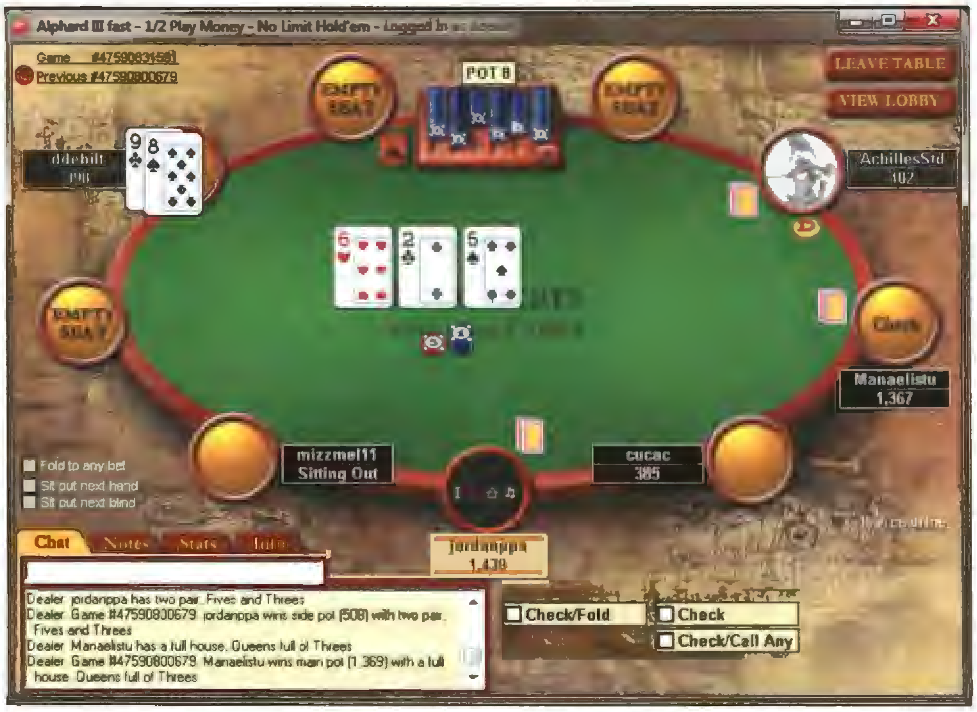
The visual table layout of this site is the most appealing of all the sites reviewed. It looks very clean and non-cluttered. The colors blend in well with the table and seats. It also allows the option to not show avatars. Showing avatars can be arguably distracting and misleading to most other players. This site offers a variety of popular games, including "5-card draw". Figure 1 shows the default layout of their poker tables. Each table can hold up to nine players. The spots reserved for each player is small, which contributes to the layout being open and no cluttered. Other players' cards are displayed smaller than all cards whose values can be seen. The option to chat to other players, as well as messages from the table can be viewed in the lower left portion of the window. Previous history of the table can be viewed through the links in the upper left corner of the window. The history that is viewable goes only as far back as how long the current player has been viewing the table.



```
Dealer: Game #47590220279. frekman wins pot (65)
Dealer: lolodenis, it's your turn. You have 8 seconds to act
Dealer: toyve, it's your turn. You have 8 seconds to act
Dealer: lolodenis, it's your turn. You have 8 seconds to act
Dealer: The Critics, it's your turn. You have 8 seconds to act
Dealer: Game #47590266099. toyve wins pot (7)
```

*Figure 1.* A default partypoker.com table; http://www.partypoker.com.

There are little white check box options in bottom middle of the GUI in figure 2, which are called action intent options. Those white check boxes allow the player to specify an action even when it is not their turn to act. The action that is checked will be the action submitted automatically when it is the player's turn to act. This allows the player to check an action they know they will be performing in the future when it is their turn, and allow them to focus their attention elsewhere and not have to stare and wait for it to be their turn. This is noticeably useful when playing at more than one table or carrying on a conversation with other players and also helps the game execute faster.



Dealer jordanppa has two pair, Fives and Threes
Dealer Game #47590800679 jordanppa wins side pot (508) with two pair, Fives and Threes
Dealer Manaelistu has a full house, Queens full of Threes
Dealer Game #47590800679 Manaelistu wins main pot (1,369) with a full house, Queens full of Threes

*Figure 2.* View of action intent options; http://www.partypoker.com.

### 2.2.2. **pokerroom.com**

pokerroom.com has been around since 1999. No download is required to start playing there, as long as the browser and operating system support java applets. This site is also supported on a smart mobile phone for players on the go, or those who don't mind playing on a phone when there is no access to a normal desktop/laptop with connection to the Internet. This site allows players to play with play money or real money, and offers the most popular poker game types, which includes Texas Hold'em.

The GUI of this site gives a bird's eye view of a virtual poker table, which the player is part of once they join a table and sit down. The GUI also assigns each player an avatar, though each avatar is already pre-assigned to a certain seat number. So if a player does not like their avatar, they can leave that table and re-join later when no other play is currently using the avatar they would like to appear to be to the other players. The cards are easy to read, chat is available on the lower right portion of the GUI, and there are many options a typical poker player would have interest in.

### 2.2.3. **partypoker.com**

The oldest of the major poker sites on the Internet, partypoker.com has been around since 1997, right when Texas Hold'em and Internet usage started to boom. This site allows a player to play with real or fake money. Gambling online with real money is still illegal in most states, though the premise of this paper is focusing on playing the actual game. The cards are easy to read on this site, and this site also has many similarities to pokerroom.com, such as check boxes for future action intent, chat, and a bird's eye view of the table. This site allows the player to choose an avatar, rather than an assigned avatar for a seat. Not only does this site offer poker, but it also offers other casino games such as slots, roulette, bingo, and backgammon.

One of the things that partypoker.com tries to distinguish themselves with, is game fairness. According to their site:

> "Providing a safe and secure gaming environment to our players is vital for the success of PartyGaming. We employ algorithms and practices to ensure total fairness to every player. Our random number generation (RNG) system operates at a high level of encryption and randomness to ensure fairness without exception.
> To that end, we develop and employ state-of-the-art systems that ensure and maintain a collusion-free environment."

### 2.2.4. **fulltiltpoker.com**

According to Wikipedia.org, fulltiltpoker.com launched in June 2004 with the involvement of some of the current big name poker players. It is ranked as the second largest online poker site, behind pokerstars.com. This site has a lot of different poker games to play, such as Texas Hold'em, Omaha H/L, Stud H/L, Stud Hi, and Razz [15], which is increasing in popularity. Even with the large selection of games, 5 card draw is not offered.

The table layout is still similar to the other poker sites reviewed. One unique thing about this site is that a player can change the background image and the table theme. The beach look, which is an available background theme, is more appealing than the traditional look of a dark room which most sites show. This site, as well as most all major poker sites, has an option to show a player the percentages of the sequences that they are still in the hand, and how often they have won during those sequences. This is a very nice feature, it helps a player find out if they are betting too much, too little, or just plain staying in the hand too long when a player should not be. Normally most experienced players keep pace and track of their opponents betting behavior and only stay in the hand when they feel they have a high ranking hand, or a chance to obtain a high ranking hand when more cards are dealt.

### 2.2.5. **cakepoker.com**

cakepoker.com started in 2004. This is another visually appealing site and is still very similar to the rest in terms of features and functionality. In general, all of the poker sites are very similar to each other and they are all lacking major features that would distinguish them from the others. This site does allow players to change their screen names once every 7 days. The advantage of allowing this is that ID tracking software becomes less worthwhile to use. ID tracking software can be used by players to track players they have played against over time. The software can report trends for those players, giving the player who uses that software an advantage.

### 2.2.6. **Texas Hold'em Poker (zynga – Facebook)**

Texas Hold'em Poker, by zynga, is currently the 2nd most popular game on Facebook, which an average of over 28 million monthly users. This could quite possibly be the most used poker software on the planet. zynga, the company which created this software, also has other social games available on Facebook which help advertise this game. This game is also available on other social networking sites, such as MySpace and Twitter.

The only game that can be played on this site, through Facebook, is Texas Hold'em; this is because the application is specific. The table layout is similar to the other sites, though the avatars are the player's actual Facebook profile photo, so in most cases a Facebook user can put a face on a player and know who they are playing against.

### 2.2.7. **Site Comparison**

Table 1 contains a comparison of the most sought after attributes, for this paper's purposes, and their comparison between the sites reviewed above.

| **Site** | **Free** | **Custom Games** | **Private Tables** | **Texas Hold'em** | **Omaha** | **Omaha Hi/Low** | **7 Card Stud** | **7 Card Stud** | **Razz** | **5 Card Draw** |
|---|---|---|---|---|---|---|---|---|---|---|
| Cake Poker | x | | | x | x | x | | | | |
| PartyPoker.com | x | | | x | x | x | x | x | | |
| PokerStars.com | x | | | x | x | x | x | x | x | x |
| PokerRoom.com | x | | | x | x | | x | | | x |
| FullTiltPoker | x | | | x | x | x | x | x | x | |
| zynga poker | x | | | x | | | | | | |

*Table 1*. Poker Site Comparisons.

## 2.3. **Other**

This section contains some other areas of research that cover current and future areas that the framework can cover.

### 2.3.1. **Hiding and Revealing in Online Poker Games**

As mentioned in the introduction, poker is a type of game where a player must make decisions with lack of information. What this means is that a player doesn't know what all the cards are in play, and therefore must deduce if he or she has a better hand than an opponent. This is normally done by taking into consideration what is known, as well as the opponents' behavior and reputation.

"Hiding and Revealing in Online Poker Games" [0], discusses the problems with online poker applications, where players are presented to other players via a handle or avatar. An online player is much more limited with information, usually only to the cards he or she can see, which is their own as well as community cards, if the game supports community cards. A player is unable to physically see the other players, to observe any distress or excitement over their own cards.

The authors also argue that the use of avatars can be more harmful than good. This is because a player usually has the option to change their avatar, and a human poker instinct is to associate betting actions and behavior to what a player looks like. For example, if an opponent player's avatar was a young, beautiful lady in a swimsuit, a male player may be more apt to not take their betting behavior seriously, as compared to an opponent player's avatar being an older, distinguished man dressed in a suit. According to the authors, "This is most problematic where stereotypes are exaggerated."

The authors suggest that if human-like representations are used, such fake human avatars, they "should not convey potentially meaningful cues that are not instigated by user action. Instead, a human-like image should have a range of outputs that match input information."

There is agreement with the author's argument that the use of human avatars may interfere with an online player's ability to understand another player's betting behavior. There are some other clues that may lead an online player to better understand their opponent, and that is the chat interface that is available in all popular online poker software packages. By looking at what that person may say, and the grammar associated with it, a player may get a better feel for that person's intelligence, thoughts, and behavior. There is agreement with the author's, because players do usually make judgments about other people based on the way they talk or look, even the way they virtually look with an avatar.

### 2.3.2. **Dealing Cards in Poker Games**

"Dealing Cards in Poker Games" [5], proposes a new protocol for shuffling and dealing cards in a game of poker. The author argues that it is less efficient to shuffle the deck of cards up front, implying a then predetermined order of dealing, when there is a small number of players in the game or if the game itself only requires a subset of the 52 cards in the deck to be dealt. The proposed protocol for shuffling and dealing, which the author mentions is ideally suited for resource-constrained devices such as PDAs, is letting the players define the card dealt through random number generation and an encryption scheme to verify non-collusion.

The protocol generates and deals a card via a player generating a random number between 1 and 52 and then submitting that number after encryption to all other players in the group. The group the author mentions is analogous to players at a poker table who are playing. If no other player has generated that same number, and no other player has that same number in their current hand, the card is accepted and the card is dealt to their hand through their own local client processing. If there is a collision, each player in the group will have to retry the same deal round again by selecting a random number and submitting it to the group. This removes the need for shuffling cards to a predetermined, yet unknown order, because when a new dealing round begins, it's just a matter of choosing a random number to see if it is already chosen or in existence in the current hand. According to the author, the initial setup of this proposed protocol is 85% more efficient than upfront shuffling, and 66% more efficient overall with both shuffling and dealing in a Texas Hold'em game with 5 players. The comparison was to existing generic shuffle and dealing protocols.

There is agreement with the author that this type of implementation would be more efficient when there are a low number of players, and the poker game itself requires a small amount of dealing from the deck. The author didn't explicitly mention the overall architecture of the game software, but the thought is that this paper implied that it was in a peer-to-peer setting, since each player's client would be submitting a random number for a card to be dealt, where all other client's would have to verify that the card to be dealt to that player is not in existence or requested already. The problem with the protocol is that as the number of players grows, or the type of game changes to something that requires more cards overall, such as "7 card stud", the possibility for dealing collisions will occur, thus decreasing the efficiency of the proposed protocol and actually delaying the dealing of cards much more. Preference was taken for a controller component to manage the dealing of cards, and in a peer-to-peer setting, it could just be a matter of one of the client's hosting the game and controlling the dealing to other players as well as locally. Though with a controller component, it may increase the possibility for collusion from the client hosting the controller component, and some type of 3rd party service may need to be implemented to keep client honesty.

## 2.4. **Security**

Security is an important research topic, because in any poker game, a player absolutely does not want any other player to know what cards they are holding. It is important that a poker application implement a security scheme to reduce any possible chance for messages to be compromised by a 3rd party, which may or may not be participating in the current poker game.

### 2.4.1. A Secure Mental Poker Protocol over the Internet

"A Secure Mental Poker Protocol over the Internet" [4], is focused on eliminating any potential collusion by any player in an online poker game. The authors introduce previous encryption and decryption algorithms, and their drawbacks and opportunities for problems. The premise of the algorithms for encryption and decryptions are that every player (in a client application context) is involved in the dealing and verification process throughout the game. All players (clients) encrypt and decrypt an entire deck based on a large agreed prime number for encryption, though have their secret keys for decrypting. One player deals, and at the end of the hand, they exchange their private keys to ensure that the cards passed around are indeed the correct cards that were in play.

Another type of older algorithm the authors discussed included a "card salesman", who is basically an independent chooser of a crypto scheme and communicates to each player on how they are to encrypt. At the end of the hand, all permutations communicated to each player are checked for fairness. The authors point out that in this type of crypto system, it assumes that the "card salesman" can be trusted, which is not a good assumption to make. Also, cheating can only be detected at the end of the game, and not during the course of the hand.

The authors suggest their own "mental" poker protocol, which is responsible for shuffling and dealing cards one by one. An introductorily summary of this scheme is as follows, from the paper:

1. Alice and Bob agree to choose the same 52 tokens for 52 cards, that are suitable encoding set {1, ..., 52}.
2. Alice and Bob agree to choose the same prime number p, for use in their respective crypto schemes.
3. Alice chooses her encryption and decryption key pairs (one public, one private).
4. Bob chooses his encryption and decryption key pairs (one public, one private).
5. [Shuffling] Alice and Bob each choose a secret random number, encrypt each card in a random order, and send each other the resulting 52 encrypted cards.

6. [Shuffling] Alice and Bob each encrypt the encrypted cards sent by the other, and then sends them all back. On receipt, they both check the double encrypted cards with a different encryption order, if the two sets are not equal, the protocol is stopped. Otherwise, the process continues.
7. [Dealing] When a player needs a card, they draw card "m", where "m" is the card order after the double encryptions. That player sends "m" to the other players after drawing "m" from his or her own set. All other players delete "m" from their set.

When the hand is over, both Alice and Bob reveal their secret random number, where each can check whether or not either had been cheating.

The authors conclude that their protocol is secure, efficient, and suitable for any number of players. Their protocol alleviates the need for a "card salesman."

### 2.4.2. **Keeping Bots out of Online Games**

"Keeping Bots out of Online Games" [3], is a publication regarding software bots. A bot is software that controls an object, or set of objects, in a software application. In the gaming sense, a bot is referred to as a computer robot that is meant to act and sometimes meant to be perceived as a human. Some Bots are considered good, though some Bots are very destructive to a human's enjoyment of a game. In online MMORPGs, such as Ultima Online, bots have been credited with massively destroying economies by their ability to gather materials generate income at an autonomous rate much higher than humans, over a period of time. Bots are an attractive alternative for repetitious games, to gain a competitive advantage over others. The authors quote, "In online card rooms, bots can be used to play games entirely based on their statistical properties, thereby earning money against imperfect human players."

The authors state three requirements to satisfy in distinguishing bots. These requirements for games are to be cost effectiveness, immune to cheating, and preserving human players' enjoyment of a game. The authors give two approaches to eliminating bots. The first approach is the use of software CAPTCHAs, which are Completely Automated Public Turing Test to Tell Computers and Humans Apart. The second approach is the use of hardware CAPTCHAs.

Software CAPTCHAs exist and are extensively used in most online services, games, and other applications to eliminate bots, requiring the user to input a string of characters in a muddled image is common software CAPTCHAs. Most intelligent image reading software applications still cannot accurately decode the characters from a muddled image. The authors argue that introducing software CAPTCHAs into online games could become somewhat disruptive, as it would interfere with the flow of the game if others had to wait until the CAPTCHA test was passed or failed.

Hardware CHAPTCHAs would require physical user input, such as the movement of a joystick, or pressing a key on a keyboard. The authors note that people would be less apt to participate in an online game that required them to update to a new piece of hardware with CHAPTCHA capabilities, in order to play the game. The authors' solution would be to create a new piece of hardware, such as a small keypad for human interaction called a CHAPTCHA token. This new, inexpensive piece of hardware would be required for purchase, but would eliminate the user having to upgrade their existing input devices. The hardware would consist of a random set of characters the user would be required to input on that hardware, in order to validate that a human is indeed participating in the game.

### 2.4.3. **Cheat-Proof Playout for Centralized and Peer-to-Peer Gaming**

In "Cheat-Proof Playout for Centralized and Peer-to-Peer Gaming" [6], the authors didn't focus on poker, though in a general sense it gives background on the different approaches taken to reduce online cheating in both client/server and peer-to-peer software architectures. The authors mention that security and cheat prevention in online games today is highly neglected, mostly because of a performance decrease in the techniques and logic required to effectively prevent cheating, especially for real-time games that require minimal latency for the user experience (e.g. first person shooters, etc.).

Though not relating to poker, for games that require player movement, a technique called "dead reckoning" is used by other clients to guess the location of another client in the game. Average latency and direction are used to guess positioning. An easy way to cheat and invalidate "dead reckoning" is to block a client from announcing its moves for a period of time or post past move announcements to make it appear as if the packets were delayed in the network. This type of issue exists in client/server architectures as well as peer-to-peer. The authors approach to correcting this problem was a simple protocol called Lockstep, and an extension to that called asynchronous synchronization, which improves performance and proves that cheating is not possible.

Lockstep is basically a "stop and wait" protocol, where the announcement (called a frame) of the next player move is given and then only that information is used to make the next turn decision on a client, for other clients. Only one announcement can exist in a given window of time. This means that there is no pattern and thus it must wait until more information is received to update other client's locations, etc, and that multiple announcements will be ignored until a time window has expired that indicates a new announcement can be received from another client. Each client is on the same server controlled time clock for processing messages, thus no client is allowed to process announcements faster or slower. Received announcements are buffered. In a simulation with 75 players, 95% of the frames were stalled at least 10 milliseconds. The authors did not mention the game, the average network latency, or the time window used. The asynchronous synchronization allows each client maintain their own time clock, thus each client does not need to hear from other clients in order to move their player, but still must adhere to the Lockstep protocol when interaction with other clients is required. The interaction can be determined by distance or other factors, as indicated by the game engine. With asynchronous synchronization on top of Lockstep, only 40% of the frames were stalled at least 10 milliseconds, compared to 95% with just Lockstep in place. Performance considerations still need to be taken into account, since the proposed protocols can increase game play latency when a player is interacting with other players, and the game could appear "choppy" depending on move delays.

Though [6] does not specifically relate to a poker game engine, it does bring into consideration some of the types of protocols that could be put into place to prevent online cheating in games. In poker, players do not move, but instead submit actions that must be validated based on their current state in the poker hand. All other players who are currently in the hand do not necessarily need to see the action of another player unless it is their turn to act, since they need information to act upon, such as the amount of chips to call or how many other players are still participating in the current hand. In poker, a Lockstep type of protocol is already in place for live games, because a player whose turn it is to act must act, prior to subsequent players acting.

# CHAPTER 3: RESEARCH APPROACH

After initial research, it was concluded that there is no software that is available, whether purchasable or available for extension, that to solve the problems and meet the objectives mentioned previously. The solution was to build the software that would meet the objectives of this paper.

The solution of developing a software application is feasible to imagine, given the state of current technology and the Internet infrastructure as the communication medium for the solution. The solution envisioned would be a simple client/server application. The clients would be the users (players), and the server would be the table (game controller). Another type of communication and control architecture, such as peer-to-peer, does not appear to be as good as solution in replicating a poker table environment. This is because players can join and drop at any time, and there always needs to be some central control mechanism to control the poker game sequences and determine which player is next to act.

The next step taken, after determining feasibility, was to determine the user requirements. The process model that was followed was prototyping for the GUI and a simple waterfall for the poker logic (since poker rules are already defined and known). The GUI required prototyping, with me as well as others as evaluators on whether or not the GUI met the needs to create the right user experience. Some specific requirements, which are mentioned below, were found throughout the process.

## 3.1. **Solution Requirements**

The first iteration of determining the requirements was done solely through this author's own knowledge of poker. This knowledge also included existing software packages, which were evaluated and explained in the literature review chapter above. Given the popularity of some of those packages, it seemed that gathering GUI and game mechanic requirements from those packages was very beneficial.

The requirements were broken out into specific groups, each a more distinct part of the overall solution, to assist in assuring that all desired requirements were found.

### 3.1.1. **General Requirements**

The following are the general requirements:

- Windows XP / Vista / 7 – The solution must be able to execute on the latest Windows platforms. Windows OS still dominates the personal computer market.
    - The solution was to be developed in C#, which requires the .NET Framework. The full .NET Framework is only supported on a Windows OS, though a Silverlight client does not require a Windows OS. A Silverlight client is not planned for the initial version.
- TCP/IP support – the solution must be able to communicate through an infrastructure or medium that supports TCP/IP. The main essence of this requirement is that the solution must be able to work over the Internet.
- The software must have the ability to interpret and allow known or custom poker games to be played. The software does not need to have any built-in games to play. This general requirement is in place to allow the flexibility into the solution to play custom poker games.
- GUI – The solution must have a graphical user interface for a more enhanced user experience. It is understood that the first release of this

solution may not have a comparable GUI to software packages that already exist, such as partypoker.com, cakepoker.com, etc. Future versions of the game may have an improved interface.

### 3.1.2. **GUI Requirements**

These are general GUI requirements, the actual look and feel of all GUIs was determined through prototyping and from the reviewed poker sites.

#### 3.1.2.1. *Critiqued Poker Sites*

The poker sites that were reviewed and critiqued were used as a foundation for how the initial client GUI should look. All of the critiqued sites, [0], [8], [0], [0], [0], and [12] have their poker table encompassing most of the client space. Each site also has the players visually surrounding the table, which is ideal. Each site also has their chat GUI on the bottom of the client, and the action area on the bottom as well. The action area is the place where the player is able to specify their action when it is their turn to act. Because of their respected popularity, the client GUI created shares the same type of visual locations for the table, player, chat, and action functionality.

### 3.1.2.2. *Client*

- All game play must be able to occur and be shown through the GUI
    - This includes allowing the user to specify their poker actions and betting amounts through the GUI
- The GUI must allow a user to communicate to other players who are also connected to the same server.
- Provide the player the ability to specify the IP address and port of the poker server to connect to.

Allowing the users to upload an avatar that other players can see is not planned. A solid argument was provided in [0], which is in agreement, that avatars can be distracting and mislead other players. Though it is assumed that the initial uses of this application will be to provide social groups with a private mechanism to play over a network connection, public use cannot be discounted.

### 3.1.2.3. *Server*

- Provide the user the ability to host a poker game. The host can be referred to as the poker server in subsequent requirements.

- Provide the user, who is hosting a poker game, to select which poker game to play out of a list of created poker games. Once a game is chosen, it will take affect when the next hand is initiated.
- Provide the user, who is hosting a poker game, to specify the starting amount of chips per player, and add an ability to give a player a specific amount of chips at any time.
    - This feature is useful so a player does not have to reconnect or the host wants to configure player chip counts on a player by player basis.
- Provide the user, who is hosting a poker game, to specify the TCP/IP port that the application will use to listen for incoming client connection requests.

### 3.1.3. Game Creation Requirements

- The person who creates the game (game creator), must have the ability to specify the name of the game being created
- The game creator must have the ability to specify the minimum and maximum number of players the game can support

- The game creator must have the ability to specify the number Decks, of cards, that the game uses
- The game creator must have the ability to specify if an ante is used, and if an ante is used, the initial amount of that ante, and additional factors that can change the ante over time, and an expression that can be applied to change the amount of ante over time. Some games refer to an ante as “blinds”, where the ante must be called by all other players on a subsequent betting round. An ability to specify an ante as a “called” or “non-called” ante must exist. Additional factors that can increase the ante over time include:
    a. Number of Deals
    b. Time
- The game creator must have the ability to specify the minimum betting amount, which is enforced during a betting round. The same type of additional factors, as were used in the ante, can also be applied to the minimum bet.
- The game creator must have the ability to specify the maximum betting amount, which is enforced during a betting round. The same

type of additional factors, as were used in the ante and minimum bet requirements, can also be applied to the maximum bet.

- The game creator must have the ability to specify the predetermined order of sequences that occur during the game. The following sequences must be supported:
    a. Ante
    b. Dealing of cards
    c. Betting
    d. Discarding
    e. Changing card visibility
- The game creator must have the ability to specify which players a sequence applies to (e.g. dealing a card to only one player during a sequence).

### 3.1.4. **Game Mechanics Requirements**

The requirements for the game mechanics of the software were clear, given that this author considers himself to be an expert in the poker domain, having played many variations of poker for many years. The game mechanic requirements feed off of the game creation requirements as well.

The common requirements in this area include (for complication purposes, all of the game mechanic requirements are not included):

- If a betting round sequence applies to all players, the game controller must always start with the person to the left of the dealer who has not yet called the last bet amount.
    a. NOTE: In most all cases this is the person to the immediate left of the dealer. However, in some games that incorporate blinds (which is a form of ante, singled out to specific players), those players who already put of blinds have already bet for the upcoming round, and thus the players to their left are the first to act.
- When a player is to act during a betting round sequence, and there is an amount to call, and the player has at least that amount in chips, the player must at least call, or their hand must be folded.
- When a player is to act during a betting round sequence, and there is an amount to call and the player does not have that amount in chips, yet the player has chips, the player must put all their chips into the pot, or their hand must be folded.

- When a player is to act during a betting round sequence, and there is an amount to call, and the player does not have any chips, the player will be skipped.
- A player can only win an amount proportional to what they had bet. This means that the pot of chips can be split in more than one way.

## 3.2. Architecture and Design

A game of poker is an extremely controlled environment. Only one player is allowed to act at one time, and in a predetermined order. All other players who are not allowed to act can only watch and wait until it is their turn to act. The only exception to that is that any player may "stand up" and leave the table at any time, though he or she forfeits any chips that they had currently have in the pot. When a player acts, the action must be validated to ensure that the player did not act incorrectly, given the rules of the poker game being played. Once the action of the player has been validated and approved by the rules of the game, the next predetermined player is up next to make an action.

Given that poker requires this type of controlled environment where only one player is allowed to make an action that must also be validated, the architecture and design of the software being developed requires a controller component. This controller component is responsible for running the game being played, according to the game's sequences. Other responsibilities include understanding the game rules, informing the players who is the person that must make an action, and responsible for validating any action that is made by every player. Figure 3 shows a high level conceptual image of the main components {Client, Controller}.

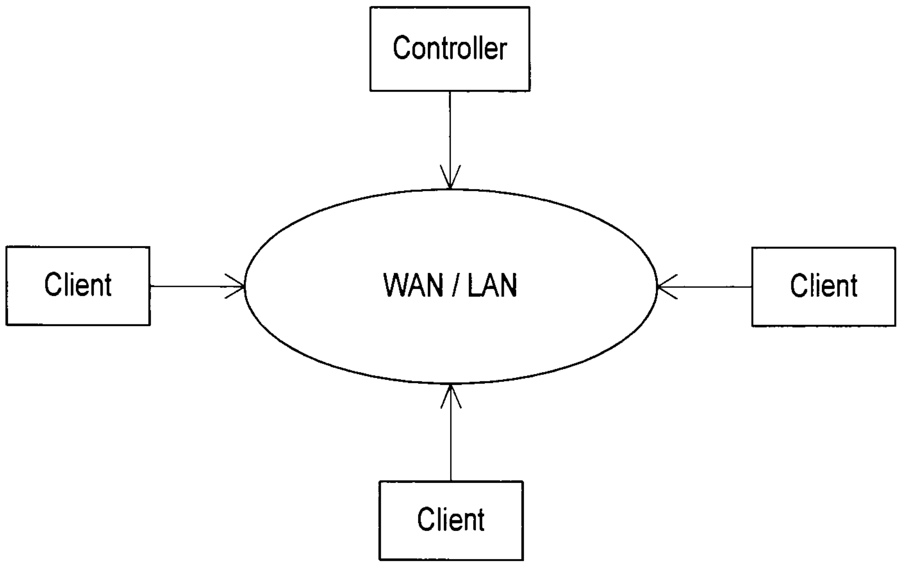


*Figure 3.* High Level Components.

Since the controller component controls the game, it makes sense for the clients to register with the controller in order to play, which makes the controller, in essence, the server in this proposed client/server architecture.

### 3.2.1. **Poker Game Library**

The poker game library is a central repository for all poker related objects which are used by both the clients and the server (game controller) applications. The objects, in most cases, were built abstractly enough so that future enhancements to the application, or the library itself, could be made without compromising much of the architecture and design that was set up.

Taking an object oriented approach, which is an easy way to model pieces of a system, the following types of objects are required:

1. Table – this will be the virtual equivalent to the table in which poker is played on. The significance in the physical poker world is that the table limits the amount of players that can play in a poker game, regardless if the poker game being played can support more. This virtual table will only allow as much as the client GUI can support visually. Figure 4 shows the object structure for the table.
2. Game – The game object represents a modeled poker game, which resides in a created XML file. A game consists of some specified attributes, such as min and max number of players, card decks, ante, min bet, max bet, and sequences. Figure 5 shows the object structure

for the game, change rules, and sequences. Each unique sequence class contains the logic for that sequence, which is constructed from the game definition file.

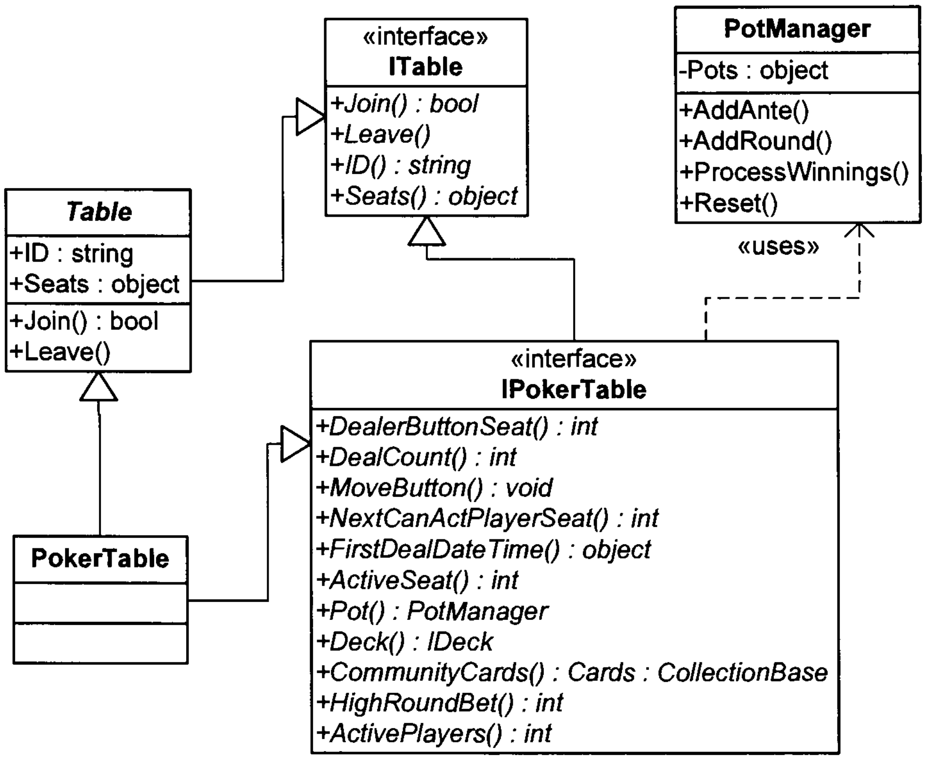


*Figure 4*. Poker Table Object Structure.

3. The Game Sequences – These are the current possible sequences that a user can put into each new game created.

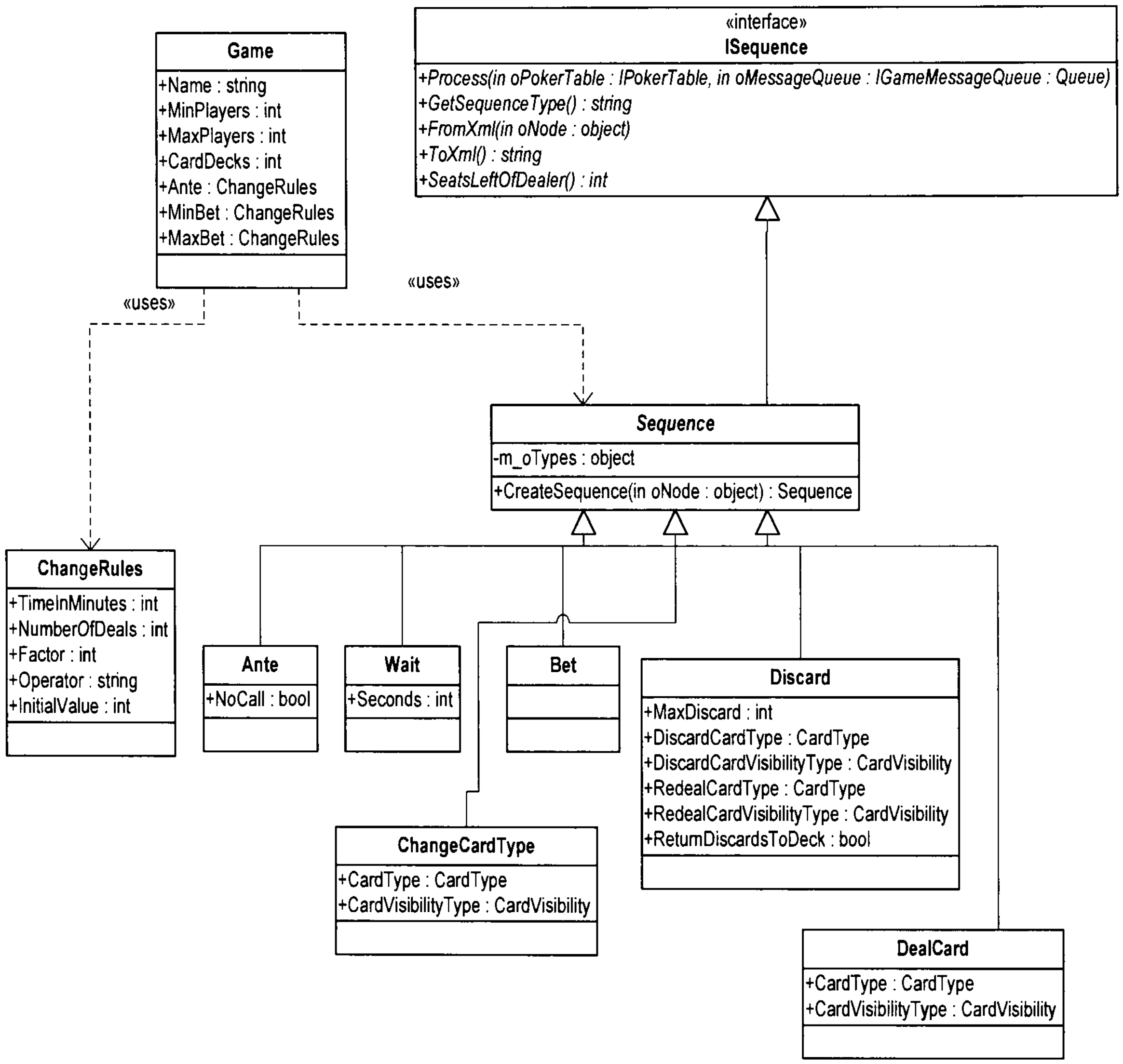
    a. ***Ante*** - The ante is a bet that is made, normally at the beginning of a hand, which is used to put some stakes into the pot. There are two types of Ante, "call", or "no call". A "call" ante means that a player who did not have to ante must put that amount of money into the pot during the next “Bet” round, or else they cannot continue in the hand. A “no call” ante does not need to be matched by any player who was not required to make an ante.

b. ***Bet*** - This is a betting sequence will prompt each player still active in the hand to act. A bet can be directed at a specific player, rather than all active players remaining in the hand.

c. ***DealCard*** - This sequence will deal a card to a player, or all players on the table.

d. ***Discard*** - The discard sequence allows players to remove certain types of cards from their hand, and also the ability to add new cards from the deck to replace the cards that were just discarded.

e. ***ChangeCardType*** - This sequence allows the poker server to change properties of a card in a player, or all players’ hands. For example, in some custom poker games, all players’ cards are dealt face down and flipped up one at a time between betting rounds.



Game
+Name : string
+MinPlayers : int
+MaxPlayers : int
+CardDecks : int
+Ante : ChangeRules
+MinBet : ChangeRules
+MaxBet : ChangeRules
«interface»
ISequence
+Process(in oPokerTable : IPokerTable, in oMessageQueue : IGameMessageQueue : Queue)
+GetSequenceType() : string
+FromXml(in oNode : object)
+ToXml() : string
+SeatsLeftOfDealer() : int
«uses»
«uses»
Sequence
-m_oTypes : object
+CreateSequence(in oNode : object) : Sequence
ChangeRules
+TimeInMinutes : int
+NumberOfDeals : int
+Factor : int
+Operator : string
+InitialValue : int
Ante
+NoCall : bool
Wait
+Seconds : int
Bet
Discard
+MaxDiscard : int
+DiscardCardType : CardType
+DiscardCardVisibilityType : CardVisibility
+RedealCardType : CardType
+RedealCardVisibilityType : CardVisibility
+ReturnDiscardsToDeck : bool
ChangeCardType
+CardType : CardType
+CardVisibilityType : CardVisibility
DealCard
+CardType : CardType
+CardVisibilityType : CardVisibility

*Figure 5*. Game Object Structure.

f. ***Wait*** - This sequence puts a pause in from moving on to the subsequent sequence. This is useful in giving the players ample time to understand what is going on and allowing them to see their hands and make decisions.

4. Player – in order for a game to be played, there must be participation. Only human player support is planned, though future versions could include artificial intelligence for non-human players. Figure 6 shows the object structure for a player.

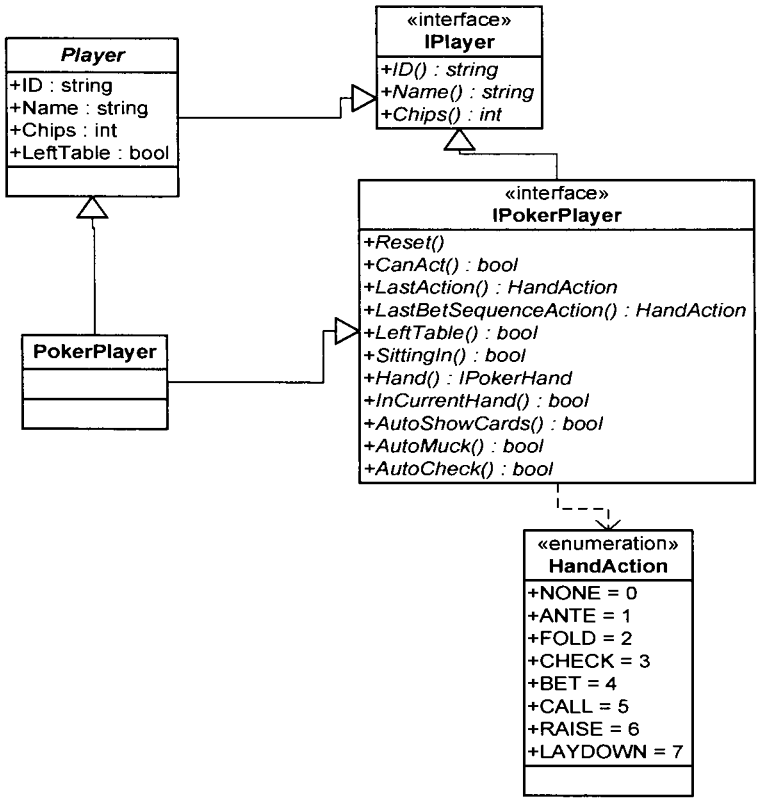


*Figure 6.* Player Object Structure.

5. Deck of Card(s) – Poker requires a collection of cards to be played. Most poker games require a standard deck of cards, which is

composed of 52 individual cards {13 values in 4 suits}. Figure 7 shows the object structure for a deck.

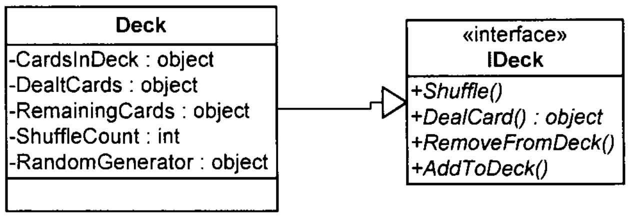


*Figure 7*. Deck Object Structure.

When it came to shuffling and dealing cards, the implementation chosen differed from shuffling up front which was the proposed protocol in [5,] where each client submitted a random number for their card. The chosen implementation involves resetting the deck after each hand has completed, populating CardsInDeck above back to a full deck. CardsInDeck is a 2-dimensional array with suits and numbers as a dimension. A value of 1 indicates the card exists in the deck, 0 means the card no longer exists in the deck. The Shuffle() method resets the indices back to 1. On DealCard(), a random number is generated {0...4} for suit, and another {0...13} for number. If a value of 1 exists in that 2-dimensional indice, the card is dealt. If not, random numbers are regenerated until a card is dealt. The algorithm becomes less efficient as more cards are dealt, since more and more indices will have a value of 0, though with current computational power, a card is usually always dealt in less than a millisecond.

6. Card – an individual card, which has a distinct suit {Spade, Heart, Club, Diamond}, and a distinct value {2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K, A}. Figure 8 shows the object structure for a card.

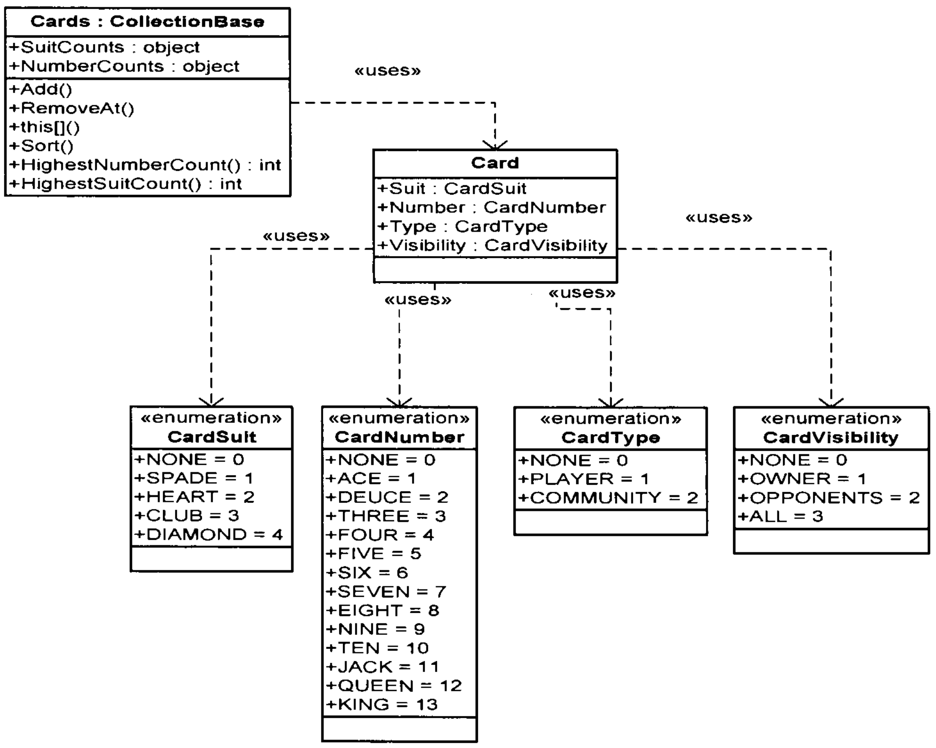


*Figure 8*. Card Object Structure.

7. Hand – The hand is modeled as exactly what is expected that a player would have in terms of cards, during the game, which is a collection of

cards to use. Figure 9 shows the object structure for a hand, which also includes enumerations for the hand powers and the card powers for each individual card number.

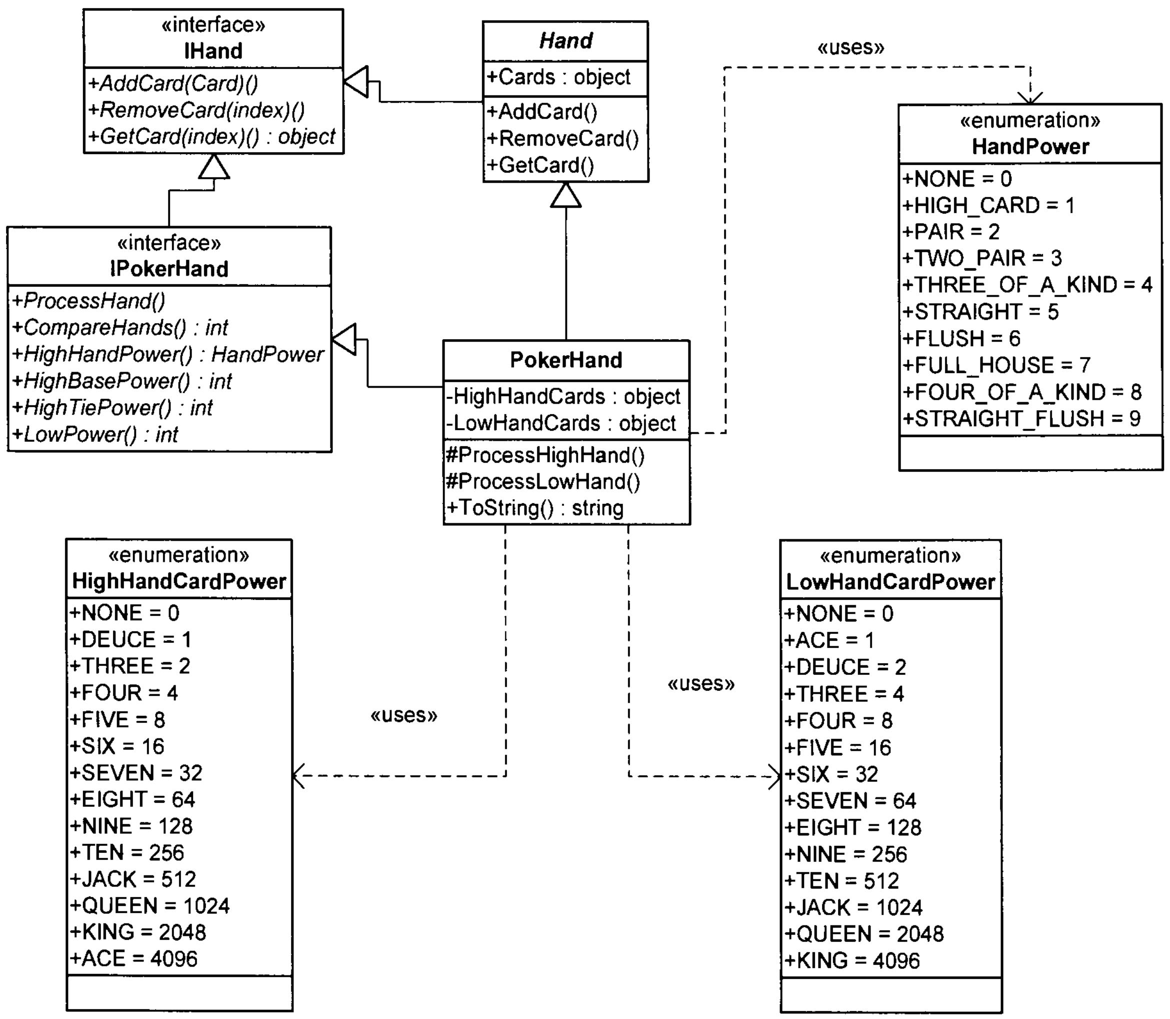


*Figure 9*. Poker Hand Object Structure.

8. Chips – not implemented. Instead, the player object has a "Chips" integer value.

### 3.2.2. **Controller**

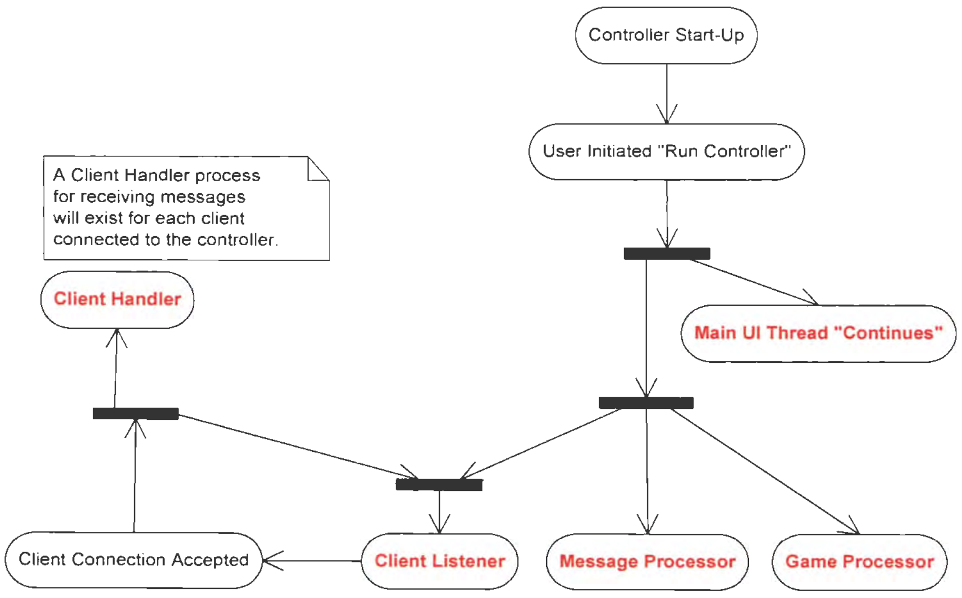
The controller needs the ability to interpret user created games. On initialization, it will load all games that are made available in a default directory. For simplicity, it will look for a “GameFiles” directory in the application’s current directory. Only .txt files will be searched for, and the contents of the file must be valid XML in order for the controller to try and understand it. The user, who started the controller instance, will be able to specify specific options of the controller, through the GUI, such as:

1. The port, on which the controller will listen for incoming client connections.
2. The amount of chips each client will be given once they successfully connect to the controller.
3. An option to give a player a specific amount of chips at any time.
    a. This feature allows players to keep track of their chips over a period of time, and is useful for on-going games that cannot be completed in one sitting.
4. The ability to change the game to be played. This will go into effect after the current game, if any, is done being processed.

#### 3.2.2.1. *Controller Thread / Process Model*

In order to maintain smooth operation of accepting and maintaining client connections, as well as an on-going poker game, multiple processes need to be running asynchronous to each other to alleviate any logic execution blocking that could occur.

Figure 10, illustrates a high level process model of the controller, which gives insight to the specific asynchronous processes that will be running:



Controller Start-Up
User Initiated "Run Controller"
A Client Handler process for receiving messages will exist for each client connected to the controller.
Client Handler
Main UI Thread "Continues"
Client Connection Accepted
Client Listener
Message Processor
Game Processor

*Figure 10*. Controller Process Model.

Once the user initiates the controller to "start" the process of listening for clients and running a game server, three new processes are created, in addition to the process that runs the GUI, which is on the main thread of the controller. The Client Listener process is a process that will continually listen for new client connections, through the .NET thread pool, and will be notified via a callback when a new connection is accepted. Every time a client connection is accepted, a new object that represents the connection, a Client Handler, will become available to the Message Processor for sending and receiving messages.

The Client Handler is responsible for sending and receiving bytes over a socket to the specific client that it was given to handle by the Client Listener process. Any bytes to be sent to the client will be queued to the Client Handler in the form of a message object by the Message Processor. Any bytes received from the client will queued to the Message Processor in the form of a message object. Similar to the Client Listener, each Client Handler will use a callback to get notified when incoming bytes are available to read from the socket, the callback is executed from the .NET thread pool.

The Message Processor is a process that will continually handle messages from the Game Processor and from the Client Handler processes. A Typical usage is the Game Processor needing to inform the players of game information. A message will be queued from the Game Processor. The Message Processor will grab the message and send it to the appropriate Clients through the Client Handler objects. Responses or Requests from the Client Handler objects are communicated back to the game through the Message Processor, where the Message Processor will take the message and decide if it needs to go to the Game Processor, or be distributed to all the other clients through the Client Handlers (such as a chat message). Any message processed by the Message Processor intended for the Game Processor, which had requested a response, will be done through the actual message object itself (such as a monitor / wait / pulse system).

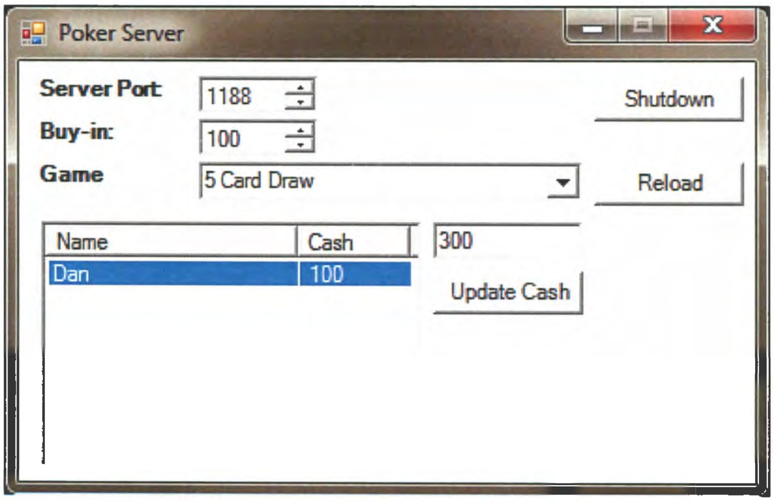
The Game Processor runs the game logic. When it needs to communicate to a client, or set of clients, it will construct a message and notify the Message Processor via a message queue. For messages that require a response, the Game Processor will wait and monitor the message object that it had queued until that message has been given a response, or a timeout had occurred. Once a response is received into the message object, game logic processing will continue. Specifics of the message structure will be explained in the below.

### 3.2.3. **Controller GUI**

It was thought that the GUI of the controller did not require any complication or need to be extravagant in any way for the first release of this application. All that was thought to be required was a simple interface to adjust settings that pertain to allowing a client to connect, such as the listening port for example. There actually is no particular reason for the controller to have a GUI. It could work just as well as a console or service application, where the settings are fed via a configuration file. Figure 11 shows the GUI that allows configuration of the controller component, as well as allowing a host to run a poker server. The GUI has options for the server port, an initial amount of buy-in chips given to each player, start and shutdown of the server, selecting the game to play, as well as an option to update a player's amount of chips at the table at any given time.

#### 3.2.3.1. *Controller Components*

The controller was implemented as Windows application, using Windows Forms in .NET. When the application starts, the PokerServer object starts other threads as noted in the thread process model above. Each connection accepted will be held in a ClientHandler object, which will track the socket used and have access to the message queue, which is handled by the RunMessageProcessor method on PokerServer. When a message needs to be sent to a client, it will call Write on the ClientHandler, which will serialize the request in to bytes and send it over the socket. When a message is received on the socket for a client, it will de-serialize the message into a GameMessage object (discussed later) and queue it to the message queue, to be handled by the RunMessageProcessor method. When the RunMessageProcessor method de-queues a message intended for the RunGameProcessor method, it will set the response which events to the RunGameProcessor that a response has been received that it had been waiting for. Figure 12 illustrates the class diagrams of the controller.


Poker Server

Server Port: 1188
Buy-in: 100
Game: 5 Card Draw

| Name | Cash |
|---|---|
| Dan | 100 |

300

*Figure 11*. Poker Server GUI.

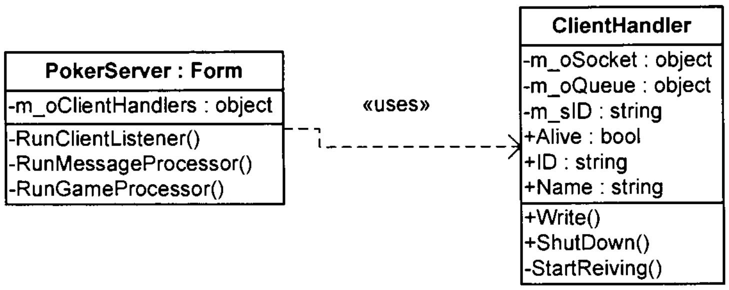


*Figure 12.* Controller Components / Classes.

### 3.2.4. **Client**

The client, just like the poker server, is split into separate functional processes. The client requires processes that will receive and process messages from the server, and at the same time, it must be able to process user input which may or may not generate messages to be sent to the server.

#### 3.2.4.1. *Client Thread / Process Model*

Figure 13 illustrates the client component's process model. When the client starts and the user initiates a connection to a server, the main thread of the application will create two new processes. One process, the Connection Handler thread will handle receiving messages to and from the server (sending is done through the thread wishing to send a message). Like the Client Handler on the server controller component, the Connection

Handler implements a callback when bytes are received over the socket, and the callback is invoked from the .NET thread pool. The Message Processor thread will process all messages that are put onto a queue it watches. The queue is either populated by the Connection Handler process or the Main UI Thread. For example, when the server sends a message to the client, requesting that the client should act on their hand, the Connection Handler receives the message, de-serializes it into a GameMessage (discussed later), and queues it the message queue that the Message Processor watches. The Message Processor will grab it from the queue, process it and display to the user that they should act (through the Main UI Thread). Once the user input is received on the Main UI Thread, a message is created and put onto the same queue. The Message Processor will grab that message and see that it is intended for the server and will serialize the message into bytes and send the message to the server (through the same socket that the Connection Handler process is receiving messages on).

#### 3.2.4.2. *Client Components*

There are not a lot of classes / components in the client executable. The PokerClient is the main form of the application. When the user wishes to connect to a server, a PokerConnect form is created and displayed on top of its parent, the PokerClient form. On a successful connection request to a server, a ClientHandler object is created and held by the PokerClient form. The ClientHandler object, which starts the Connection Handler process to receive messages on, holds a reference to the socket that is used to send and receive messages from the server. Figure 14 illustrates all of the client components.

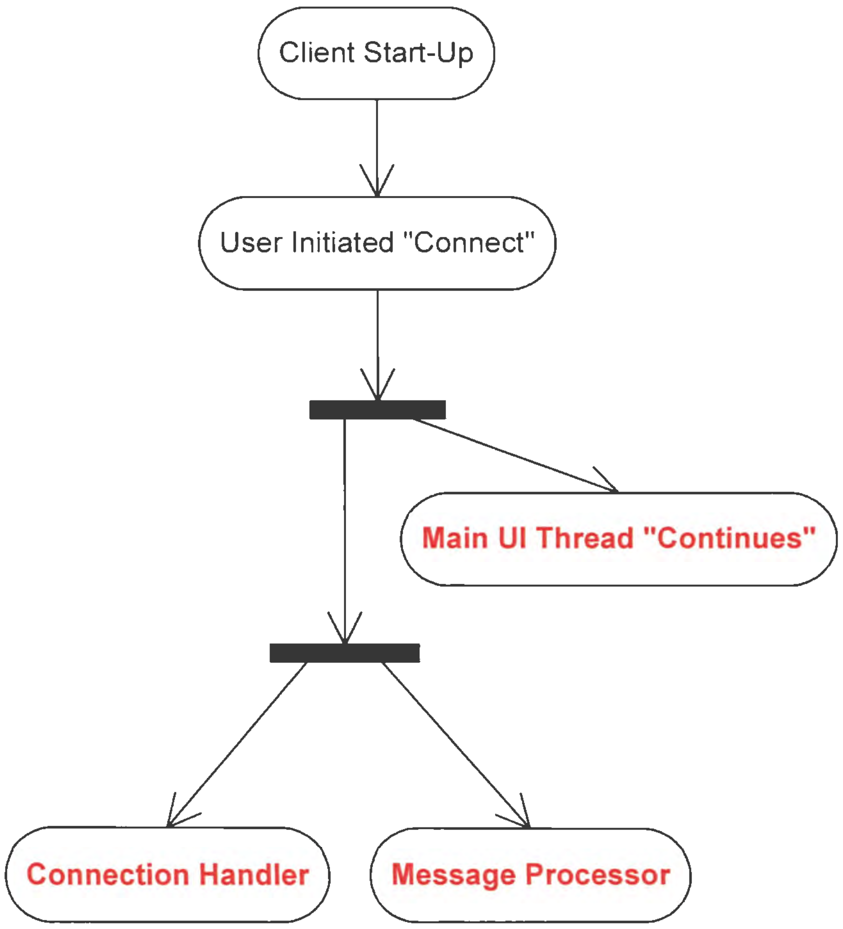


*Figure 13*. Client Process Model.

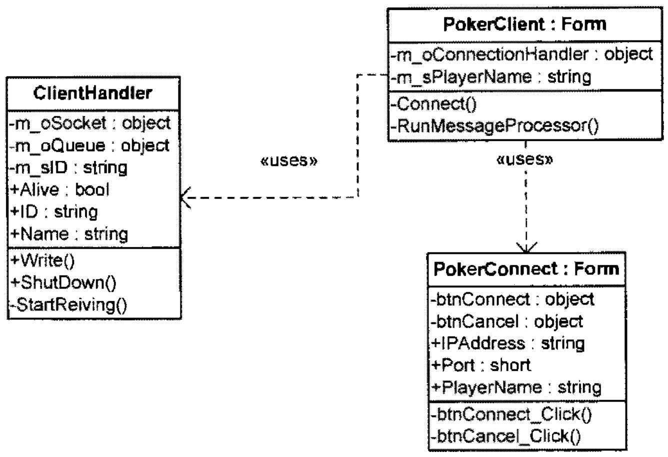


*Figure 14*. Client Components / Classes.

### 3.2.5. **Client GUI**

Creating the GUI was the part of the process that required prototyping. A few different iterations were required in order to make the interface "good enough" to use, even though the understanding was that the GUI would only serve as an initial interface and future releases would enhance and make the GUI more desirable as an interface. The initial GUI was modeled from the critiqued poker sites, through subsequent prototyping was done in coordination with a small group of poker players whom this author players poker with.

On initial startup of the client application executable, the player is shown an empty table. The buttons in the lower left corner of the GUI, which are used for poker actions, are grayed out, and there are no messages in the lower right chat window. The table is empty, though placeholders can be seen where the players will be shown around the table. The only options that are available for the player to do are exit the application by clicking the "Exit (F3)" option on the "File" menu or by clicking the "X" box in the upper right corner on the application window, or connect to a server, which can be done by clicking on the "File" menu in the upper left and going to the "Connect (F1)" option. See Figure 15 for the initial layout of the GUI when the application executable is launched.

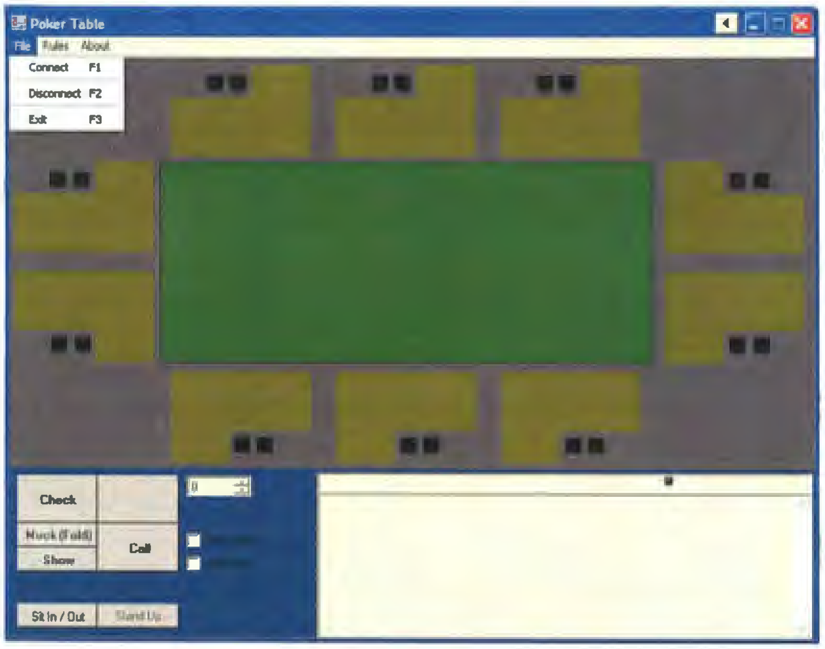


*Figure 15*. Initial GUI on Load.

After clicking on "Connect", the player is able to specify the IP Address and Port of the server to connect to, as well as a name to use for their player. The player is able to cancel the form, which will remove the "Connect" form and the user will be back to the original GUI in Figure 15. Figure 16 shows the form presented to the player, which allows them to connect to a server.

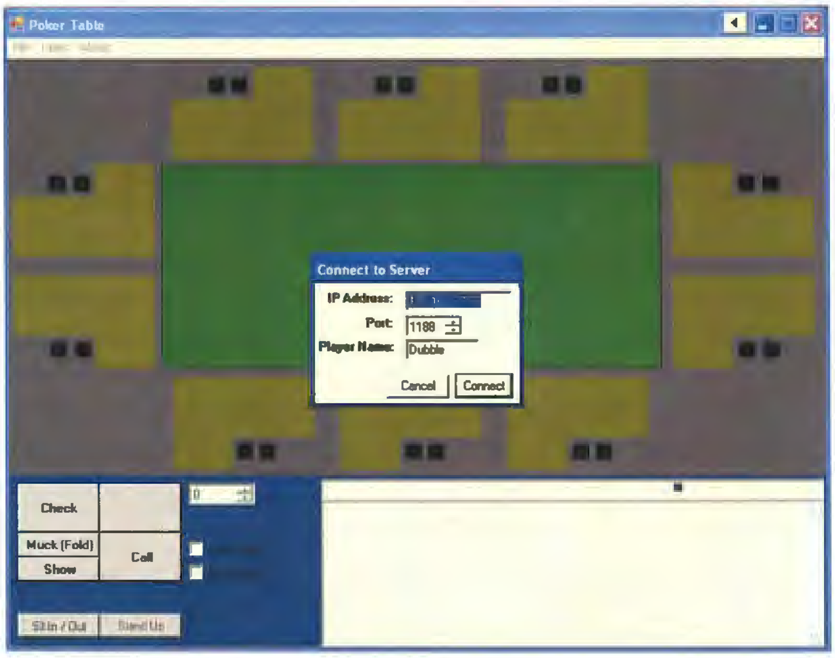
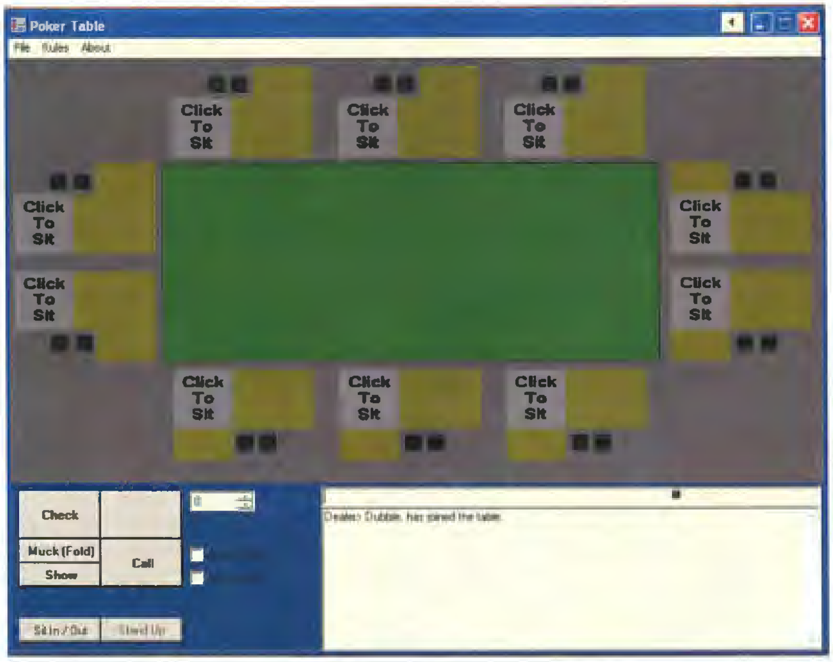


*Figure 16.* Connecting to a Server.

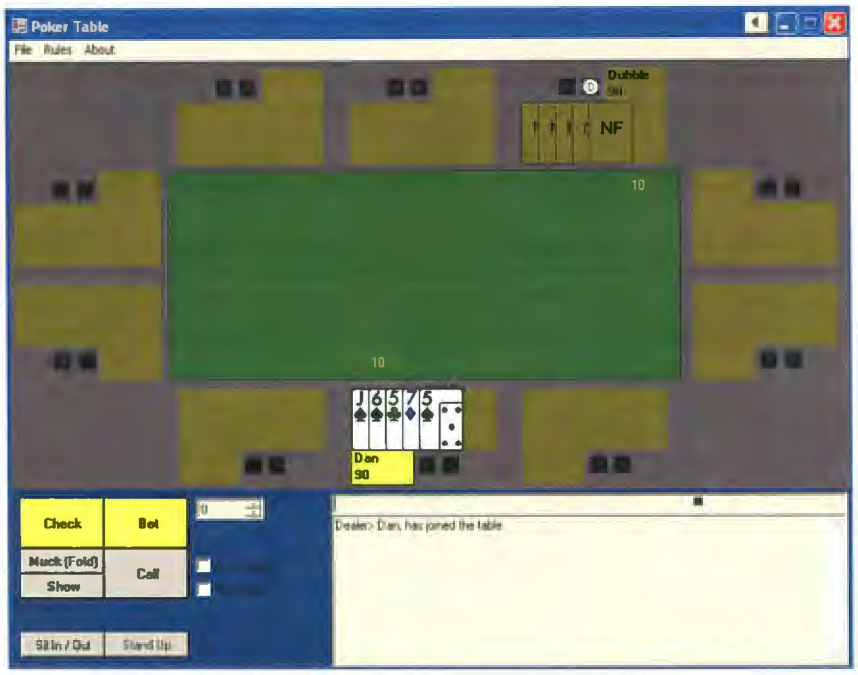
On successful connect, the server will send the player (client) the current state of the table and game. A game may be in progress if other players are currently connected to the server and sitting down to play. If the player is the first to connect to the server, they will be presented with an empty table, and empty seats as is shown in Figure 17. Once successfully connected, the virtual dealer (server / game controller) will send a message to each connected client, indicating the following player has joined the table.



Dealer: Dubbie. has joined the table.

*Figure 17.* An Empty Table.

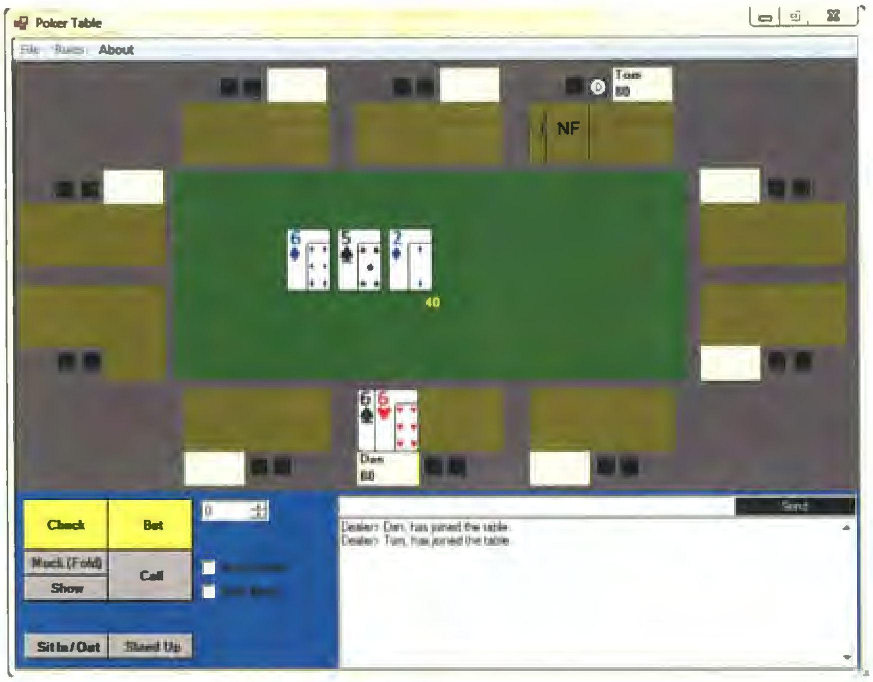
Once a game has started, which is the condition where at least two players are "sitting in"; the cards will be dealt to all players who are "sitting in". Figure 18, shows the view that each player will have when they are dealt cards initially. The player who is next to act will also has their name panel highlighted, as well as the possible actions that they can submit for their turn.



Dealer> Dan has joined the table

*Figure 18*. Actions.

Some games require community cards, such as Texas Hold'em. The figure below, Figure 19, shows an example view of two players playing Texas Hold'em. The three community cards that have been dealt are known as the "flop". Some games also allow the players to discard a portion of their cards during a sequence. 5 card draw is a poker game that allows this. Figure 20, shows a player selecting to discard. When the user is allowed to discard, they can click on their cards. When clicking on those cards, they will become highlighted. If the user wishes to not discard a highlighted card, they can click it again to un-highlight it. Once they have chosen the cards to discard, they can click the "Discard" action button to notify the server of the cards they are discarding. The server will deal out new cards from the deck for the player.



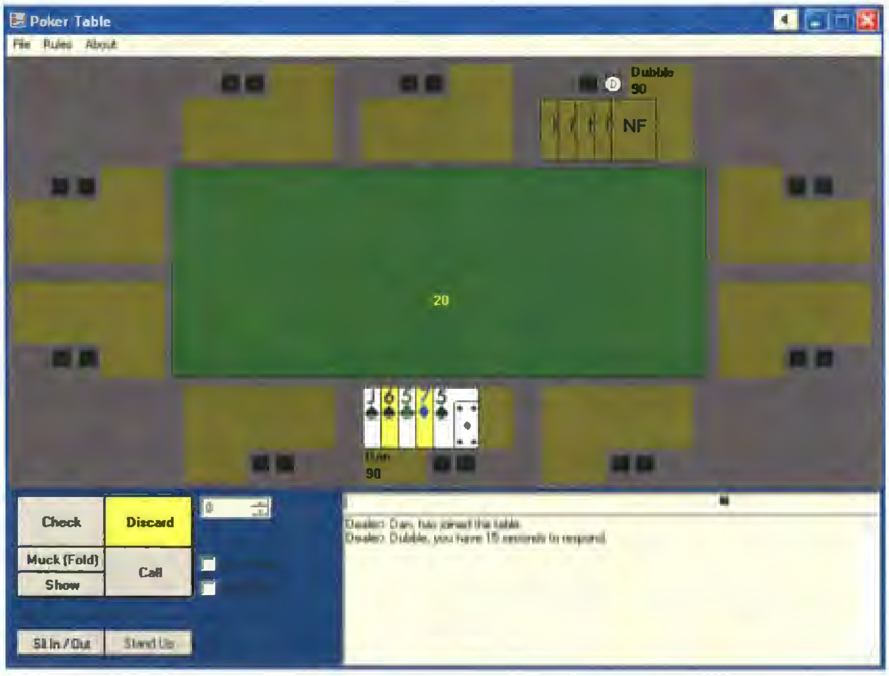
Dealer: Dan, has joined the table.
Dealer: Tom, has joined the table.

*Figure 19.* Community Cards.

## 3.2.6. **Client / Server Layer 5 Communication Protocol**

### 3.2.6.1. *Authentication and Security*

It is assumed that an implementation is not immediately required to authenticate and attempt to keep unknown players or bots out of a private game, as suggested in [3]. To intrude on a game, a user would have to know the IP (Internet Protocol) address and port that the poker server is listening on for incoming connections. From there, the user would need to either have the correct client application or know the protocol explained in the "Messaging" section. If the game's location, the IP address and port, was distributed publicly in order to get a game going, a bot could be used to play in the current implementation if the bot understands this application's messaging protocol.



Dealer: Dan, has joined the table.
Dealer: Dubble, you have 15 seconds to respond.

*Figure 20.* Discarding.

Future implementations or extensions could provide the host with an ability to ban a user, via their IP address, or through a range of IP addresses. Preventing bots is mostly done by requiring the user to enter text, usually based on an image shown on a client. A future extension could randomly pop up an image that would require them to enter a string of text matching that image's text, in order to proceed with the game.

A security mechanism, such as encryption will also not be implemented in the first version of this application. Numerous mechanisms exist, such as suggested in [4] and [6], though the main reasoning for not including it is that each client will only receive, through messaging, the cards that are visible to them, no other cards are sent. However, it is still possible that the other messages to the other clients can be intercepted and interpreted to compromise the information of the game.

#### 3.2.6.2. *Messaging*

This section explains the application level protocol that is used to communicate between the clients and the server. All normal communications, between the dealer and a player are included, and all other messages that weren't added initially for poker game mechanical requirements, were added during the development of the actual application as they were needed. Figure 21 illustrates all of the messaging components created during development.

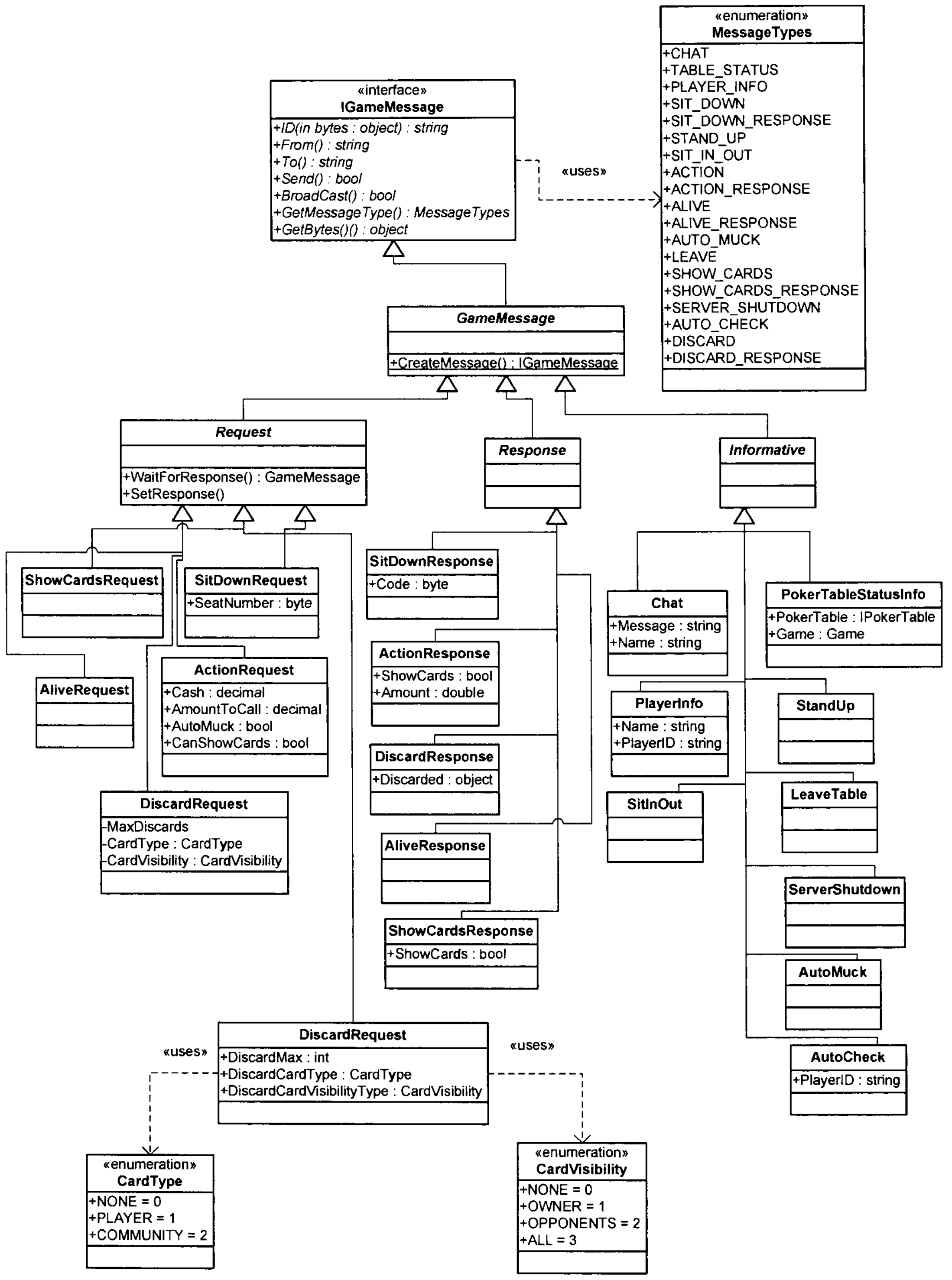


*Figure 21.* Game Message Object Structure.

All application messages sent between the client and server will adhere to the following format:

- Byte[0-1]: Length (unsigned 16 bit integer)
- Byte[2...34]: Message ID (GUID)
- Byte[35]: Message Type
- Byte[36...Length-35]: Data (can be in any format – including XML)

Since all messages are represented as objects prior to serialization, and are de-serialized to objects once received, another way to pass messages between the server and clients would be to use .NET's "ISerilizable" interface applied to the message objects. This would allow the objects to be serialized into a stream (sending side), and then de-serialized back into an object from a stream (receiving end). However, a major drawback of using .NET's serialization of objects is that the amount of bytes required to represent the objects is much larger than doing a custom serialization. This can become a problem when a future extension allows hosting a poker server with numerous tables, where each table is constantly sending and receiving information. Also, most hosting centers have byte limits on bandwidth; in case this research is taken further to introduce this into large scale hosting.

Some or all of the messages can use XML as the Data portion of the message if there is actual data to pass. The data bytes, once converted to UTF8 encoding, will be a valid XML document, which could have the following signature:

<Message>

<ID/> - the ID of the message

<Type/> - integer value of the message type

<Data /> - each message request or response will have its own specific data

</Message>

XML would be great for its flexibility and robustness. And it could be anticipated that both the client and server applications will be backward and forward compatible with each other. The game itself decides the level of compatibility. However, since most messages just needed to pass one or two items of significance in this initial version, the minimal amount of data is being passed over the network connection.

The three major categories of messages are Request, Response, and Informative.

### 3.2.6.3. *Informative Messages*

Informative messages do not require an action or response. They are merely an FYI for one or more players involved in the game. Some informative messages are sent periodically by the server, while others can be player-initiated.

***Chat*** – this type of message is initiated by a player, and even by the server (game controller). A human player may send a chat message to communicate with other players on the table. A chat message is displayed for all other players to see. Currently chat messages initiated by players are public, not private. The server can send a chat message informing the players of specific messages, such as who won the last hand and what the winning hands were.

***PokerTableStatusInfo*** – this broadcast informative message is sent periodically by the server to inform each player the current status of the table. This message includes all of the information about the table, which includes all the players who are still sitting at the table, their cards, their action, their cash, their name, the current pot, etc. Everything a player would expect to know about everyone sitting at a table.

***PlayerInfo*** – This message is sent from a client once connected to the server. This message contains the name of the player, and their randomly generated GUID ID, which the client application generates to identify the player.

***SitInOut*** – this message is sent from a client once that player has decided to sit in, or sit out of the next hand. This is basically a toggle. If the player is currently sitting in, and the user submits this message to the server, the server will now flag them as sitting out, and will not deal them in on the next hand.

***StandUp*** – this message is sent from a client when the player has decided to stand up from the table, thus leaving the seat that the player currently occupied, unoccupied. The player is still considered in the same "room" as the table, and can still see chat messages, but since the player is standing up, that player can no longer participate in subsequent gaming until they decide to sit back down at an empty seat.

***LeaveTable*** – this message is sent from a client when the player has left a table "room". A client can initiate this message at any time. When a player sends this message, they will be disconnected from the server that they were previously connected to.

***AutoMuck*** – this message is sent from a client when the player wishes to toggle the automatic mucking of their cards when they have lost a hand. If this toggle is on, when the player loses a hand during a showdown, the cards will automatically be discarded for no one else to see. However, if this toggle is off, the player has the opportunity to show their cards, once prompted by the server through a separate message, for everyone else to see.

***AutoCheck*** – this message is sent from a client when the player wishes to toggle the automatic checking of their action when it is their turn to act and not previous bet has been placed to call. If this toggle is on, and it's the player's turn to act and there is not bet to call, they will automatically check and will not be prompted, by the server, to act.

***ServerShutdown*** – this unfortunate message is sent by the server when it is in the process of shutting down.

#### 3.2.6.4. *Request Messages*

These are message that are initiated by either the client or the server. A response is expected for each request sent.

***AliveRequest*** – this message is sent by the server when a client (player) has not responded to a recent request (the request timed out at the application level for the server). The client will automatically respond to this request upon receipt without human intervention required. Upon receiving a response for this request, the server will still assume the client is up and running.

***ShowCardsRequest*** - If the player does not have the auto-muck toggle on, the server will request the player if they would like to show their cards when the player has either decided to fold, or has won the hand without a showdown required. A showdown is when both players are in the hand at the very end, and no one has folded. Each player must show their hand in the order of "last person to be shows last". If the first player shows their hand, and the other players know they have lost, they can choose to not show their cards. This is useful when trying to utilize bluffing and a long-term personality strategy, where a player may not want the opponent to know their betting style.

***ActionRequest*** – this message is sent by the server when it is time for a player to act {Check, Bet, Call, Raise, Fold, etc}. The message contains some of the options that the player is allowed to do, which is based on what the server thinks their toggles are set at.

***SitDownRequest*** – this message is sent by the client when requesting to sit down at a table. If the server has accepted this player at the seat number requested, a response will be sent indicating that the client can assume they are now sitting at that seat number on the table.

***DiscardRequest*** - In poker games that allow discard sequences, the server will sent a request to the players, asking them which cards they wish to discard. The discard request contains the information, for the client, on the rules of the discard, which includes the max number of cards they can discard, the type of cards they can discard.

### 3.2.6.5. *Response Messages*

Response messages are messages sent to answer a specific request message. An ID within the message itself is used to identify which request the response is intended for.

***SitDownResponse*** – this message is the response sent by the server, in answer to the client's request to sit down on a table, at a certain seat number. The Code is understood by the client through the shared game library logic.

***AliveResponse*** – this message is the response by the client, in response to an AliveRequest message. This response is sent automatically by the client, with no human intervention. This response indicates to the server that the client is still alive and does not need to be removed from the table.

***ActionResponse*** – this message is a response by the client, in response to an ActionRequest message. The Amount value indicates exactly what the action is, based on what the parameters were in the ActionRequest. For example, if the AmountToCall was 0, and this response's Amount was 0, this indicates that the player has made an action of CHECK. If the Amount was greater than 0, this would indicate that the payer had made an action of BET. If the AmountToCall was greater than 0, and the Amount in the response was greater than the AmountToCall, it would indicate a RAISE. If the Amount was equal, and they were both greater than 0, it would indicate a CALL. An Amount equal to 0, when the AmountToCall was greater than 0 and the player still has cash, would indicate a FOLD. An Amount less than the

AmountToCall, when the player has no cash left, indicates the player is still in the game, though "All In" per se.

***DiscardResponse*** – this message is a response by the client, in response to a DiscardRequest message. The response indicates which cards, of the cards in the player's hand, are to be discarded. The cards to be discarded were validated by the client, and will be validated again by the server on response, to make sure they are of the type of cards that can be discarded.

***ShowCardsResponse*** - This is a response by the client, in response to a ShowCardsRequest message. The response indicates whether or not the client wishes to show their cards to the rest of the table.

### 3.2.7. **Poker Game Creation**

For flexibility and robustness, all poker games that can be played with the software must adhere to an XML format, rather than a string or flat byte format where certain offsets and values define property meanings. Through XML, any newer game created with extra attributes and elements will still work for a prior version of the software, as it would remain backward compatible, though lacking the new functionality intended for the later version the game was created for.

A poker game can be constructed through XML in the following way:

```
<PokerGame  name=”name of the poker game “>
        <MinPlayers>
                integer value ranging from 1 to 10
        </MinPlayers>
        <MaxPlayers>
                integer value ranging from 1 to 10
        </MaxPlayers>
        <DecksOfCards>
                integer value ranging from 1 to 255
        </DecksOfCards>
        <Ante> (optional)
                <StartingValue>
                        This integer value is the initial value of the ante,
                        though the value can change depending on the
                        <ChangeRules> element below, if included.
```

</StartingValue>

<ChangeRules> (optional)

<Deals>

integer value ranging from 1 to the maximum Int32 value

</Deals> (optional – this node indicates to change the ante value every time this number of deals has taken place)

<Minutes>

A value in minutes that indicates to change the ante every time this number of minutes passes since the last time the ante value has changed.

</Minutes>

<Expression>

A postfix notation expression, where the variables "value", "deals", and "minutes" can be used in the equation.

```
                Example:  value 2 *
            </Expression>
        </ChangeRules>
</Ante>
<MinBet> (optional)
        <StartingValue>
                This integer value is the initial value of the
                minimum bet, though the value can change
                depending on the <ChangeRules> element
                below, if included.
        </StartingValue>
        <ChangeRules> (optional)
            <Deals>
                integer value ranging from 1 to the maximum
                Int32 value
            </Deals> (optional – this node indicates to change
            the ante value every time this number of deals has
            taken place)
```

```
                <Minutes>
                        A value in minutes that indicates to change
                        the minimum bet every time this number of
                        minutes passes since the last time the ante
                        value has changed.
                </Minutes>
                <Expression>
                        A postfix notation expression, where the
                        variables “value”, “deals”, and “minutes” can
                        be used in the equation.
                        Example:  value 2 *
                </Expression>
        </ChangeRules>
</MinBet>
<MaxBet> (optional – unlimited otherwise)
        <StartingValue>
                This integer value is the initial value of the
                maximum bet, though the value can change
```

depending on the <ChangeRules> element below, if

included.

</StartingValue>

<ChangeRules> (optional)

<Deals>

integer value ranging from 1 to the maximum

Int32 value

</Deals> (optional – this node indicates to change

the ante value every time this number of deals has

taken place)

<Minutes>

A value in minutes that indicates to change

the maximum bet every time this number of

minutes passes since the last time the ante

value has changed.

</Minutes>

<Expression>

```
                    A postfix notation expression, where the
                    variables "value", "deals", and "minutes" can
                    be used in the equation.
                    Example:  value 2 *
                </Expression>
            </ChangeRules>
        </MaxBet>
        <Sequences>
            <Sequence type='<name>' ... />
        </Sequences>
    </PokerGame>
```

All elements and attribute values are constructed with text when a number is not the intended value. This was done for readability and ease of game creation.

### 3.2.7.1. *Sequences*

The sequences are run synchronous and in sequential order that they are defined.

1. **Ante** (<Sequence type='ante' />)

This sequence indicates that players at the table must submit an ante.

***call='yes/no':*** if this attribute is present, this indicates that the other players on the table do or do not need to call the ante being submitted. If this attribute is not included, it defaults to "yes" and all players must match the ante in order to stay in the hand. If call is indicated, or the attribute is not included which would default to "yes", the actual calling of the ante will not take place until a betting round sequence has started. When an ante must be called during the next betting round, the first player to the left of the dealer who hasn't called the ante, is first to act.

2. **Deal Card** (<Sequence type='dealcard' … />)

   This sequence indicates that cards will be dealt from the deck to the players on the table that are still in the hand. The following attributes can be used in the deal card sequence:

   ***cardtype='<type of card>'***: this attribute indicates the type of card to be dealt.

   ***cardvisibility='<visibility of card>'***: this attribute indicates the visibility of card to be dealt.

3. **Bet** (<Sequence type='bet' />)

   This sequence indicates to start a betting round. In a betting round, all players must participate. During a betting sequence a player may have a chance to check, bet, call, raise the bet, or fold. Betting in this game always starts to the player to the immediate left of the dealer.

   - Future release can also define an attribute that allows the game creator to vary the minimum and maximum betting amount within the specific sequence, rather than always use and rely on the MinBet and MaxBet nodes under the <PokerGame> node.

4. **Wait** (<Sequence type='wait'  seconds='<integer>' />)

   This sequence indicates for the game to wait a period of time, in seconds, before the next sequence is executed. This type of sequence gives the players a chance to look at the current status of the table, opponents' hands, etc, before the next sequence will occur.

5. **Discard** (<Sequence type='discard' ... />

   This sequence allows a player(s) to discard current cards in his or her hand, and replace them with new a new card(s) from the deck. The following optional attributes can be used in the deal card sequence:

***maxdiscards='<integer>'***: this value indicates the maximum number of cards that a player can discard from his or her hand. The valid values are 0 (which is the dealer) to 255. This value should be closely tied to the number of decks that are used in the game (e.g. if the game creator is creating a game with one standard deck of 52 cards, and 10 people are playing, a player should normally not be able to discard more cards than are left in the deck, or cause a condition where the last players to discard would not be able to get "new" cards from the deck because they have all been dealt to previous players who have already discarded. There is a card recycle attribute that can deal with this condition (see below).

***discardcardtype='<type of card>'***: this attribute indicates the type of card to be discarded.

***discardcardvisibility='<type of visibility>'***: this attribute indicates the visibility of card that can be discarded.

***redealcardtype='<type of card>'***: this attribute, if included, indicates the type of card that will be dealt back after the discards.

***redealcardvisibility='<type of visibility>'***: this attribute indicates the visibility of card that will be dealt back after the discards.

***returndiscardstodeck='yes/no'***: this value indicates if the discarded cards from each player discarding should be returned to the deck, and available again for dealing. In some games, this option must be available to satisfy a large number of players allowed to discard a large number of cards from a game that only uses a certain number of decks, in order to always make cards available for dealing to all players.

***shuffleondiscardreturn='yes/no'***: this value indicates if the deck will be reshuffled once after all discards have been collected and all re-deals have occurred. If the deck is not reshuffled when discarded cards are returned to the deck, a player may be able to track when a specific card would come available again on deal, though that may be desired by the type of game.

6. **Change Card** (<Sequence type='changecard' ... />)

   This sequence can change a card type. This is useful when a hidden card must be made visible to other players during the course of the game sequences. The following optional attributes can be used in the deal card sequence:

***cardtype='<type of card>'***: this attribute indicates the type of card that can be changed.

***cardvisibility='<type of visibility>'***: this attribute indicates the visibility of card that can be changed.

***newcardtype='<type of card>'***: this attribute indicates the type of card that the changed card will become.

***newcardvisibility='<type of visibility>'***: this attribute indicates the visibility of card that the changed card will become.

***maxchanges='<integer>'***: this value indicates the maximum number of cards in the player's hand that can be changed. The valid values are 1 (which is the dealer) to 255. The changing starts with the oldest card in their hand that matches the cardtypetochange and moves toward the most recent card dealt.

The following common attributes can be applied to sequences by including them within the <Sequence> element:

- ***seatsleftofdealer='<integer>'***: this attribute, if present, indicates which player the sequence applies to. A value of 0 indicates the dealer. A value larger than the maximum possible amount of players for the game is being allowed in case the creator of the game wishes to base this value on some predetermined number, in which the value will be mod by the number of players to determine which player left of the dealer the value applies to. This attribute can be used with:
    - ***Ante***
    - ***Deal Card***
    - ***Bet***
    - ***Discard***
    - ***Change Card Type***
- ****cardtype='<type of card>'***: this attribute indicates the type of card the sequence applies to. If this attribute is not included in the <Sequence> node and is required, the type of card value will be default to "none". Some sequences may have multiple attributes that require a type of a card, the name of the attribute will indicate its usage. The following are the set of valid values for the attribute:

    - "player" – this card is only usable by the player it was dealt to.
    - "community" – this card is usable by everyone.
    - "none" – this card is not usable by anyone.
- ***cardvisibility='<type of visibility>'***: this attribute indicates the visibility of card that sequence applies to. If this attribute is not included in the <Sequence> node, the visibility of card value will default to "none". Some sequences may have multiple attributes that require a visibility of a card, the name of the attribute will indicate its usage. The following are the set of valid values for the attribute:
    - "none" – this card is visible to no one.
    - "player" – this card is only visible to the person it was dealt to.
    - "opponents" – this card is not visible to the person it was dealt to, but is visible to everyone else on the table.
    - "all" – this card is visible to everyone.

### 3.2.8. **Sequences**

The sequences below indicate how the system communicates. Figure 22 is an example of when a client connects to the controller (server). An example of a client request is shown in Figure 23.

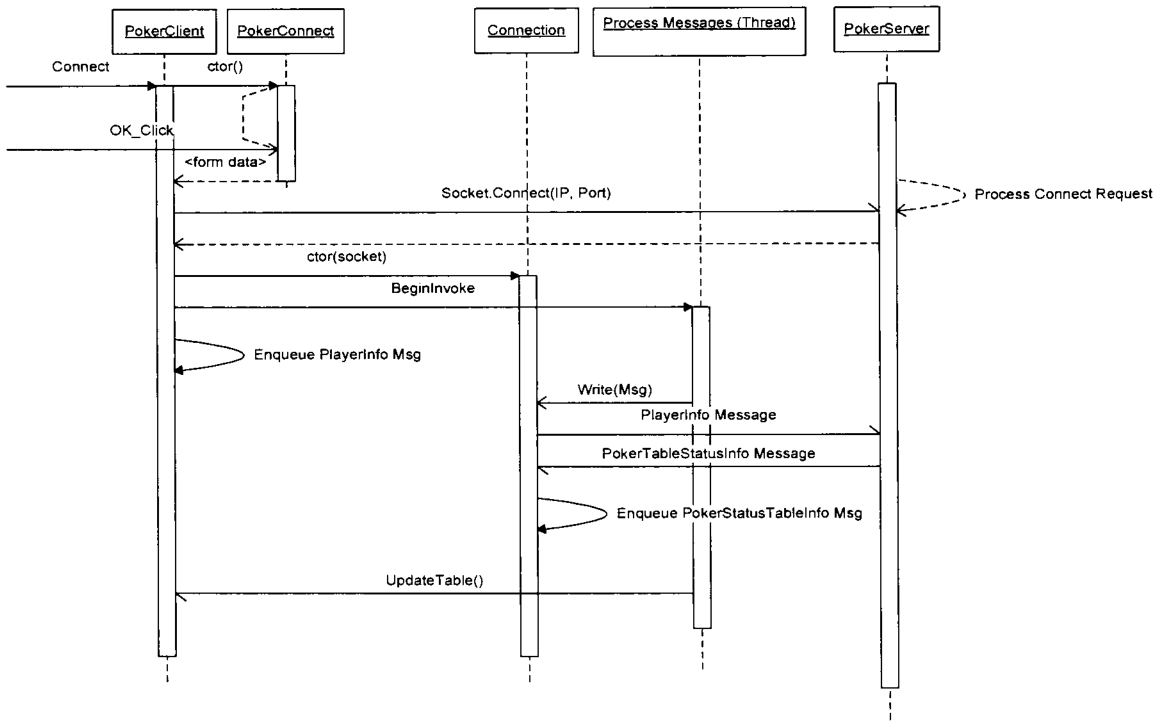


*Figure 22*. Client Connect Request.

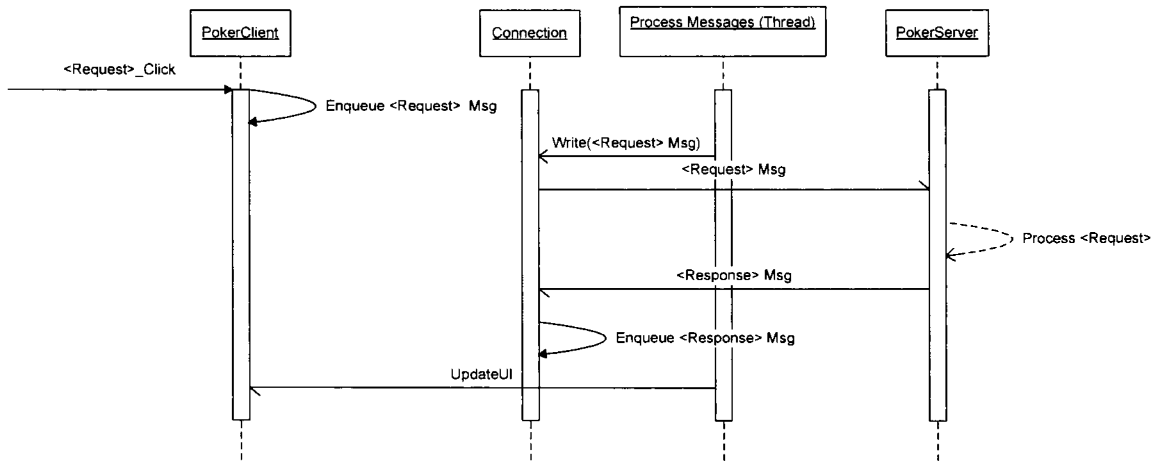


*Figure 23*. General Client Request.

An example of an informative message from the client is shown in Figure 24.

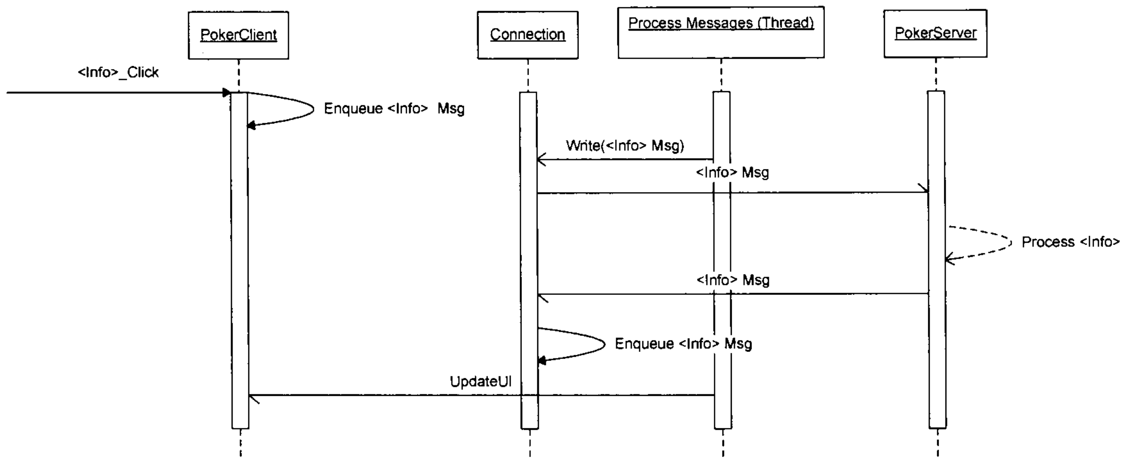


*Figure 24*. Informative Message Sent By Client.

An example of an informative message sent by the controller (server) to one or many clients is shown in Figure 25.

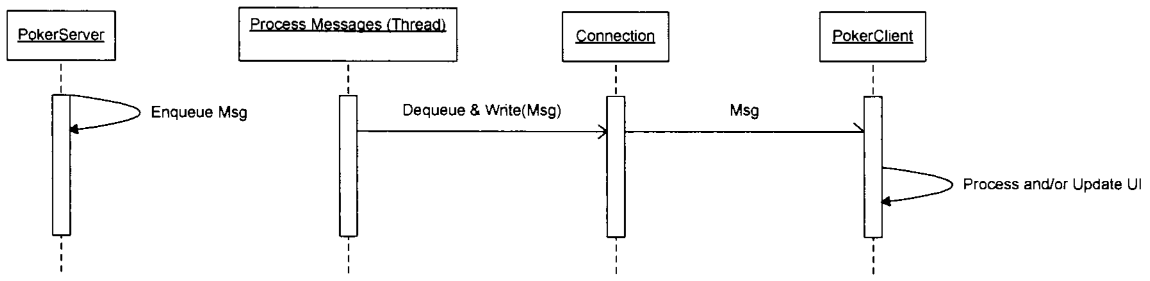


*Figure 25*. Server Message Queue / Send.

An example of when the controller (server) is requesting and waiting for a response from a specific client is shown in Figure 26.

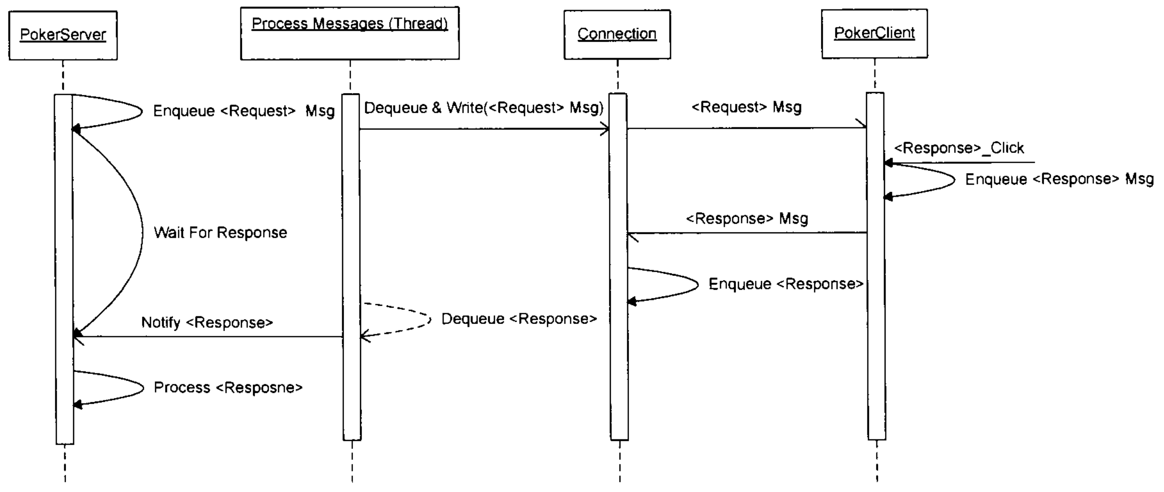


*Figure 26*. Server Request / Response Wait.

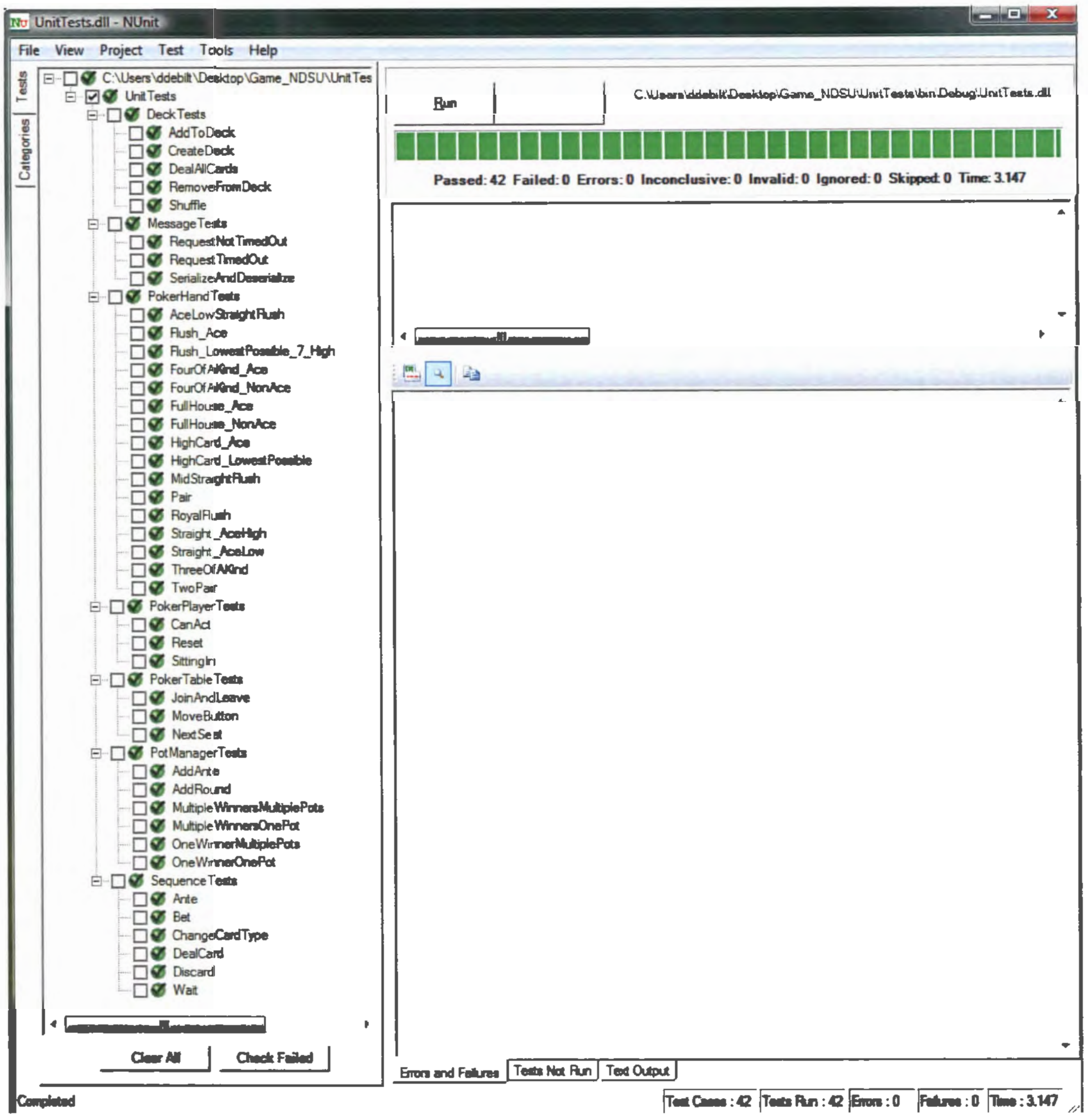
## 3.3. **Testing**

There are two types of testing processes that were done during and after the development of the application. Unit testing was conducted during the development, and after, to ensure that components and algorithms of the application were initially correct and maintained their correctness even through changes for other bugs (problems) that were found during all types of testing. End to end testing was conducted to ensure that the overall application functioned as expected when all components were combined.

### 3.3.1. **Unit Testing**

Tests were created to tests the major aspects and objects of the software created, with the emphasis on the rankings of possible poker hands. Since the software contains asynchronous processes, external dependencies, and object usage cascading, a unit testing framework was used called Moq, http://code.google.com/p/moq/. Moq is a mocking library that allows a tester to mock (spoof) objects under test and their dependencies that they may use. This type of framework allows unit tests to be written for controller, which requires external socket connections to process the game sequences. Most mocking frameworks can only spoof object interfaces or virtual methods. Given that the software developed was designed with interface decoupling in mind, it was easy to use for unit testing.

Unit tests were created to conform to N'Unit standards (attribute-based), and then executed with the N'Unit application. Refer to http://www.nunit.org for more information regarding N'Unit. By looking at Figure 27, it shows the different types of tests that were created and executed during and after the development of the application.



```
C:\Users\ddebilt\Desktop\Game_NDSU\UnitTes
  Unit Tests
    DeckTests
      AddToDeck
      CreateDeck
      DealAllCards
      RemoveFromDeck
      Shuffle
    MessageTests
      RequestNotTimedOut
      RequestTimedOut
      SerializeAndDeserialize
    PokerHandTests
      AceLowStraightFlush
      Flush_Ace
      Flush_LowestPossible_7_High
      FourOfAKind_Ace
      FourOfAKind_NonAce
      FullHouse_Ace
      FullHouse_NonAce
      HighCard_Ace
      HighCard_LowestPossible
      MidStraightFlush
      Pair
      RoyalFlush
      Straight_AceHigh
      Straight_AceLow
      ThreeOfAKind
      TwoPair
    PokerPlayerTests
      CanAct
      Reset
      SittingIn
    PokerTableTests
      JoinAndLeave
      MoveButton
      NextSeat
    PotManagerTests
      AddAnte
      AddRound
      MultipleWinnersMultiplePots
      MultipleWinnersOnePot
      OneWinnerMultiplePots
      OneWinnerOnePot
    SequenceTests
      Ante
      Bet
      ChangeCardType
      DealCard
      Discard
      Wait
```

```
C:\Users\ddebilt\Desktop\Game_NDSU\UnitTests\bin\Debug\UnitTests.dll
Passed: 42  Failed: 0  Errors: 0  Inconclusive: 0  Invalid: 0  Ignored: 0  Skipped: 0  Time: 3.147

Completed    Test Cases : 42  Tests Run : 42  Errors : 0  Failures : 0  Time : 3.147
```

*Figure 27*. Unit Tests.

Looking at the tests, there is a unit test for each specific type of hand that is possible (including when the ace can be a low card). Even though this framework has been created to cover different types of poker games, the tests were created to only cover poker hands possible, which are shared among all poker games. Tests were also created to over the interface methods of the major objects of the system, which are the Deck, Hand, Player, Table, Pot, and Messages.

### 3.3.2. **End-to-End Testing**

This type of testing was more of an ad-hoc approach for the development of this application. Before exposing the application to the potential stakeholders, multiple instances of the client application were executed to simulate actions for each player in each client. This method did uncover problems in both the client and the server applications. When no game mechanical problems were found after this type of testing, other potential users were brought in to help test and provide feedback on the game play and GUI. When other users were involved, they provided feedback on a variety of things, including the GUI. They knew that the GUI was just intended to serve as an easy way to play the game, and would be revamped on future versions of the application as needed, but not for the initial release.

## 3.4. **Deliverables**

The following were defined and developed throughout this paper's research:

1. Requirements of a solution that would allow users to play poker games over a network connection.
2. An extensible client/server software application that allows users to play poker games over a network connection.
3. An XML definition that allows users to define and create custom poker games, which can be played through the client/server application.
4. Unit tests that were created to assist in ensuring a level of quality of the client/server application.

# CHAPTER 4: RESEARCH EVALUATION

Looking back at the restatement of the problem, it appears that the software developed does solve the problems and issues that were presented. Below is a breakdown of the main issues, into groups, in their respective order of importance.

## 4.1. Flexible Poker Games

The main problem, of the set of problems described, was that there was no existing application that allowed a group of poker players to play their own custom poker games, whether purchasable or not. If a poker player wanted to play "Blind Man's Bluff" within their social poker group, they would have to physically meet somewhere in order for the game to proceed. Now, with the application described and developed for this paper, it is possible, and a reality.

### 4.1.1. Flexibility

The application developed allows a user to define a custom poker game within the stated poker rules defined above.

The software package developed includes premade games such as:

1. 5 Card Draw
2. 5 Card Stud
3. 7 Card Stud
4. Texas Hold'em
5. Blind Man's Poker

Even though the above games are the only ones premade and ready to play with the application, the game creation model allows a user to model a custom poker game through XML. As long as the game follows the base defined rules of poker, the game can be created and played. For example, if a user wanted to create a game where each player only received two cards only visible to that player, everyone bets, and then the person with the best two cards wins, the XML would look like:

```
<PokerGame  name="Two Card Madness ">

        <MiniPlayers>2</MinPlayers>

        <MaxPlayers>10</MaxPlayers>

        <DecksOfCards>1</DecksOfCards>

        <Sequences>
```

```
                <Sequence type="dealcard" cardtype="player"
    cardvisibility="player" />
                <Sequence type="dealcard" cardtype="player"
    cardvisibility="player" />
                <Sequence type="bet" />
            </Sequences>
    </PokerGame>
```

In future versions of the application, the set of rules that define a game's mechanics and how it is played in the application can be expanded. For example, in the developed application's initial version, the user is not able to define the value of a particular card number of suit, in respect to the other card numbers or suits in a deck of cards. A future release could allow a game to be created that could define every "Jack" to be worth more than an "Ace", thus a pair of "Jacks" will beat a pair of "Aces". This is a little different than defining a "Jack" as a wildcard, but an example nonetheless.

### 4.1.2. **Privacy**

A user can set up their own poker server and inform anyone, that they wish to allow, the IP address and port to connect to. As long as no one else, whom they do not wish to connect and join their server, knows the IP and port of the poker server application, it can be considered private.

There currently isn't an implemented feature to boot or kick anyone from a table, or ban an IP address from connecting to a server. Future versions of the application can add those extra security measures to ensure that the game remains as private as required.

### 4.1.3. Free

Last, but not least, initially this software package is free. A social group of poker players do not have to pay anyone to use it. The solution was mainly to meet the needs of this author and his group of social poker players.

From the three major points above, which attempt to solve the problem where there is no current way to play custom or well-known poker games in a free and private manner, the application developed makes a great start to fulfilling that need. The functionality is now developed and available, though it can use some enhancements in the future to better the user experience and satisfaction.

# CHAPTER 5: CONCLUSIONS

## 5.1. Evaluation

Considering that the application developed is an initial version, and could be expanded and enhanced, it is felt that the application developed meets the needs of the problem described earlier in this paper. A framework exists that allows users to create their own types of poker games, games that do not currently exist online in a free and flexible environment.

## 5.2. Research Application

The research approach was based on the need for a solution that did not currently exist that met the free, flexible, and private environment. The initial research concluded that there was nothing currently existing that satisfied the requirements and needs. Based on the initial research conclusion, a software application was developed to meet those needs. This type of approach is common in the software industry. When research concludes that there is currently nothing that satisfies the needs and requirements, usually the solution is developed. Or, something is changed or amended to satisfy the requirements and needs. The assumption was that existing applications that could have been amended or changed to satisfy the requirements were proprietary and would not be available for changes. It may also been easier to take the approach of creating a new application from scratch, instead of modifying an existing application whose architecture and design may not have been created robustly and flexible enough to absorb the changes required in a timely manner.

## 5.3. **Future Work**

Expanding features for this application is an exciting area. There are many items that could be done to further enhance the user experience.

### 5.3.1. **Game Features**

A way to indicate wild cards could be added to the poker engine and to allow players to indicate which cards in the deck are wild in the game XML definition, whether by suit or number. Even more flexibility could be added to the sequences that allow betting rules to change as well, such as ante, minimum bet, and maximum bet.

### 5.3.2. **Artificial Intelligence**

A great addition to the application would be to add the ability to include non-human players. This would come in handy when no humans are available to play, or there is a minimum requirement of players for a certain game type. The largest effort involved for this future work would be to make sure the AI is fine tuned and adjustable.

### 5.3.3. **Enhanced GUI**

The application developed does not have a comparable GUI, as compared to the five poker sites included in the literature review section. However, it should be noted that the GUI developed is a first generation GUI, just enough to meet the needs of a prospective user. It does not have avatars, it does not have configurable backgrounds, and it does not have any images of chips, which are a common item of poker.

Future work could easily improve the GUI, without affecting any of the core game functionality of the application. An improved GUI would improve the user experience, and perhaps enhance the application enough to compete on a global level for users of other online poker applications. Other enhancements include an ability to provide instructional, tips, and game-play information to the user as an educational and learning opportunity (e.g. If the player has a certain hand ranking, give that information to the user, so that they can correlate and understand the ranking system much faster).

WPF (Windows Presentation Foundation) could be used to enhance the GUI. The user experience lies with the GUI, and WPF comes with great ways to easily control a GUI through XAML and easy ways to animate objects. A web interface, perhaps spawned from a java applet or the usage of Silverlight, would be another way to enhance portability of the application. Users wouldn't need to install the application on any different computer, but instead just rely on the browser and OS supporting the platform. Silverlight applications are only allowed to connect to certain ports on a destination server, and must be validated by a policy server (listening on Port 943).

An example of a policy that allows Silverlight applications to connect to a server is as follows (showing the actual port range):

```
<?xml version="1.0" encoding="utf-8" ?>
<access-policy>
    <cross-domain-access>
        <policy>
            <allow-from http-request-headers="*">
```

```
                    <domain uri="*"/>
                </allow-from>
                <grant-to>
                    <socket-resource port="4502-4534"
                    protocol="tcp"/>
                </grant-to>
            </policy>
        </cross-domain-access>
    </access-policy>
```

Silverlight can connect to a WCF service with a binding that does not listen on the ports specified above. But since WCF is not a true full dual-plex communication and connection oriented architecture, it would be much harder to push and pull updates to and from the clients and server. This could be solved by having the clients host their own WCF service host when the client is launched, and give this information to the server. Though it the client user would have to make sure their firewall supported this new incoming connection.

A mobile interface would be an interesting spin-off as well. Cell phones are become faster and more robust ever year. Within the next 10 years, a cell phone could be a very powerful personal computer, capable of handling applications as computers can today. Building applications with the mobile mentality in mind is a good thing to consider.

### 5.3.4. **Game Creator**

Currently the application requires any new poker game type to be developed by manually creating the betting and sequence structure with XML. To do this, the user must create an XML file and input the needs of the poker game.

A game creator application or even a feature of the client or server application itself could allow a user to create a poker game by indicating betting and sequence structure. The feature could then save the users newly created game into the XML format desired.

### 5.3.5. **New Game Engines**

New game engines could be designed and created to allow other types of games that are played with a standard deck of cards. Hearts, pinochle, crazy eights, rummy, the list is endless. A game engine could be modeled in XML, just as poker game types are currently modeled. The application could load up all game type models on load and allow the users to pick the games they want to play, just as they do now with the poker game types available.

Another option is allowing the current poker game engine to allow wild cards, configurable card values, or even new hand ranks such as 5-of-a-kind.

### 5.3.6. **Odds**

Everyone who plays poker wants to know what their chances are of winning. If a player is playing Texas Hold'em against nine other players, and they are dealt a pair of aces, they may think you have a great shot of winning. However, the fact that they are playing against nine other players, rather than just two or three, greatly reduces their chance to win if everyone decides to stay in the hand and not fold. Future work, could add an odd calculation system, to give the user a visual representation of what their chances are of winning, depending on what cards have been seen by the user, and how many people are still left in the hand.

### 5.3.7. **Security**

Currently the application has not security scheme. Due to the assumed low, and private usage of the developed application, security is not much of a concern at this point. However, if popularity increases, the addition of security to keep games private and cheating prevention may be a new requirement.

Many application level protocol security schemes exist, including a newly proposed scheme, which was included in the literature review. An easy to implement encryption protocol, such as MD5, could be put into place to provide enhanced security. The client and server would both need to know a secret key in order for this protocol to be implemented.

### 5.3.8. **Commercial**

If most or all of the previous future work items are completed, there could be a business potential to the application. Millions of users play online poker, for real money daily. The poker sites themselves take a cut of the money circulating on the tables, known as a rake. Most sites have a rake percentage that is not too noticeable and does not cut into the profits of the users. Most sites employ a 1-2% rake percentage. If millions of users play online daily and this application has the potential to see a few thousand of those users, a few percentage of the money exchanging hands could be significant.

In order to be a commercial application, the application would require extensive modifications, most of which I won't mention because it would require extensive research to figure out what all of them would be. One of the more important items is that it would need to be certified by independent quality assurance firms. To pass those rigorous tests, perhaps thousands of automated component and end-to-end tests would need to be created, and thousands of hours of testing would need to be put in place to ensure the system works as expected.

# BIBLIOGRAPHY

1. Raymond, Eric Steven, The Art of Unit Programming, 2003 http://www.faqs.org/docs/artu/index.html (February 20, 2009)

2. Golder, Scott A. and Donath, Judith S. (2004): Hiding and revealing in online poker games. In: Proceedings of ACM CSCW04 Conference on Computer-Supported Cooperative Work 2004. pp. 370-373.

3. Golle, Philippe and Ducheneaut, Nicolas (2005): Keeping bots out of online games. In: Lee, Newton (ed.) Proceedings of the International Conference on Advances in Computer Entertainment Technology - ACE 2005 June 15-15, 2005, Valencia, Spain. pp. 262-265.

4. Zhao, W., Vadaharajan, V. and Mu, Y. (2003). A Secure Mental Poker Protocol over the Internet. In Proc. First Australasian Information Security Workshop (AISW2003). Adelaide, Australia. CRPIT, 21. Johnson, C., Montague, P. and Steketee, C., Eds. ACS. pp. 105-109.

5. Golle, Philippe (2005): Dealing Cards in Poker Games. In: Proceedings of the International Conference of Information Technology: Coding and Computing (ITCC'05) – Volume 1. April 2005. pp. 506-511

6. Baughman, Nathaniel E., Liberatore, Marc and Levine, Brian Neil (2007). Cheat-Proof Playout for Centralized and Peer-to-Peer Gaming. In IEEE/ACM Transactions on Networking (TON) – Volume 15. Issue 1 (February 2007). pp 1-13.

7. PartyPoker.com. http://www.partypoker.com

8. PokerRoom.com. http://www.pokerroom.com

9. FullTiltPoker. http://www.fulltiltpoker.com

10. PokerStars.com. http://www.pokerstars.com

11. Cake Poker. http://www.cakepoker.com

12. zynga poker. http://apps.facebook.com/texas_holdem

13. “Poker.” http://en.wikipedia.org/wiki/Poker (February 23, 2009)

14. “Intellivision.” http://en.wikipedia.org/wiki/Intellivision

15. Razz (poker game). http://en.wikipedia.org/wiki/Razz_(poker)

# APPENDICES

## 7.1. No Limit Texas Hold'em Game XML

```
<PokerGame name="No Limit Texas Holdem">

        <MinPlayers>2</MinPlayers>

        <MaxPlayers>10</MaxPlayers>

        <DecksOfCards>1</DecksOfCards>

        <Ante>

                <StartingValue>10</StartingValue>

                <ChangeRules>

                        <Deals>5</Deals>

                        <Minutes>20</Minutes>

                        <Expression>value 2 *</Expression>

                </ChangeRules>

        </Ante>

        <MinBet>

                <StartingValue>20</StartingValue>

                <ChangeRules>

                        <Deals>20</Deals>
                <Minutes>20</Minutes>

                <Expression>value 2 *</Expression>

        </ChangeRules>

</MinBet>

<Sequences>

        <Sequence type="ante" seatsleftofdealer="2" />

        <Sequence type="ante" seatsleftofdealer="2" />

        <Sequence type="ante" seatsleftofdealer="1" />

        <Sequence type="dealcard" cardtype="player"

cardvisibility="player" />

        <Sequence type="wait" seconds="3" />

        <Sequence type="dealcard" cardtype="player"

cardvisibility="player" />

        <Sequence type="wait" seconds="5" />

        <Sequence type="bet" />

        <Sequence type="wait" seconds="3" />

        <Sequence type="dealcard" cardtype="community"

cardvisibility="all" />
```

```
        <Sequence type="wait" seconds="1" />
        <Sequence type="dealcard" cardtype="community"
cardvisibility="all" />
        <Sequence type="wait" seconds="1" />
        <Sequence type="dealcard" cardtype="community"
cardvisibility="all" />
        <Sequence type="wait" seconds="3" />
        <Sequence type="bet" />
        <Sequence type="wait" seconds="1" />
        <Sequence type="dealcard" cardtype="community"
cardvisibility="all" />
        <Sequence type="wait" seconds="3" />
        <Sequence type="bet" />
        <Sequence type="wait" seconds="1" />
        <Sequence type="dealcard" cardtype="community"
cardvisibility="all" />
        <Sequence type="wait" seconds="3" />
        <Sequence type="bet" />
        </Sequences>

</PokerGame>
```

## 7.2. 5 Card Draw Game XML

```
<PokerGame name="5 Card Draw">

        <MinPlayers>2</MinPlayers>

        <MaxPlayers>6</MaxPlayers>

        <DecksOfCards>1</DecksOfCards>

        <Ante>

                <StartingValue>10</StartingValue>

        </Ante>

        <MinBet>

                <StartingValue>10</StartingValue>

        </MinBet>

        <MaxBet>

                <StartingValue>40</StartingValue>

        </MaxBet>

        <Sequences>

                <Sequence type="ante"/>
        <Sequence type="dealcard" cardtype="player"
    cardvisibility="player" />
        <Sequence type="dealcard" cardtype="player"
    cardvisibility="player" />
        <Sequence type="dealcard" cardtype="player"
    cardvisibility="player" />
        <Sequence type="dealcard" cardtype="player"
    cardvisibility="player" />
        <Sequence type="dealcard" cardtype="player"
    cardvisibility="player" />
        <Sequence type="bet" />
        <Sequence type="discard" discardcardtype="player"
    discardcardvisibility="player" redealcardtype="player"
    redealcardvisibility="player" maxdiscards="3"/>
        <Sequence type="bet" />
    </Sequences>
</PokerGame>
```

## 7.3. **5 Card Stud Game XML**

```
<PokerGame name="5 Card Stud">

        <MinPlayers>2</MinPlayers>

        <MaxPlayers>10</MaxPlayers>

        <DecksOfCards>1</DecksOfCards>

        <Ante>

                <StartingValue>10</StartingValue>

        </Ante>

        <MinBet>

                <StartingValue>10</StartingValue>

        </MinBet>

        <MaxBet>

                <StartingValue>40</StartingValue>

        </MaxBet>

        <Sequences>

                <Sequence type="ante"/>

                <Sequence type="dealcard" cardtype="player"

        cardvisibility="player" />
        <Sequence type="dealcard" cardtype="player"
    cardvisibility="all" />
        <Sequence type="bet" />
        <Sequence type="dealcard" cardtype="player"
    cardvisibility="all" />
        <Sequence type="bet" />
        <Sequence type="dealcard" cardtype="player"
    cardvisibility="all" />
        <Sequence type="bet" />
        <Sequence type="dealcard" cardtype="player"
    cardvisibility="all" />
        <Sequence type="bet" />
    </Sequences>
</PokerGame>
```

## 7.4. **7 Card Stud XML**

```
<PokerGame name="7 Card Stud">
    <MinPlayers>2</MinPlayers>
    <MaxPlayers>7</MaxPlayers>
```

```
<DecksOfCards>1</DecksOfCards>

<Ante>

        <StartingValue>10</StartingValue>

</Ante>

<MinBet>

        <StartingValue>10</StartingValue>

</MinBet>

<MaxBet>

        <StartingValue>40</StartingValue>

</MaxBet>

<Sequences>

        <Sequence type="ante"/>

        <Sequence type="dealcard" cardtype="player"

cardvisibility="player" />

        <Sequence type="dealcard" cardtype="player"

cardvisibility="player" />

        <Sequence type="dealcard" cardtype="player"

cardvisibility="all" />
```

```
            <Sequence type="bet" />

            <Sequence type="dealcard" cardtype="player"

        cardvisibility="all" />

            <Sequence type="bet" />

            <Sequence type="dealcard" cardtype="player"

        cardvisibility="all" />

            <Sequence type="bet" />

            <Sequence type="dealcard" cardtype="player"

        cardvisibility="all" />

            <Sequence type="bet" />

            <Sequence type="dealcard" cardtype="player"

        cardvisibility="player" />

            <Sequence type="bet" />

        </Sequences>

</PokerGame>
```

## 7.5. **Blind Man Poker Game XML**

```
<PokerGame name="Blind Man Poker">

        <MinPlayers>2</MinPlayers>
        <MaxPlayers>10</MaxPlayers>

        <DecksOfCards>1</DecksOfCards>

        <Ante>

                <StartingValue>10</StartingValue>

        </Ante>

        <MinBet>

                <StartingValue>10</StartingValue>

        </MinBet>

        <MaxBet>

                <StartingValue>40</StartingValue>

        </MaxBet>

        <Sequences>

                <Sequence type="ante"/>

                <Sequence type="dealcard" cardtype="player"

        cardvisibility="opponents" />

                <Sequence type="bet" />

        </Sequences>

</PokerGame>
```

## 7.6. **Poker Hand Ranks**

The ranks below are listed in order of descending rank. When examples are given below, the number indicates the card value, the letter indicates the card suit {s=spades, h=hearts, c=clubs, d=diamonds}.

### 7.6.1. **Straight Flush**

The Straight Flush requires 5 cards. All 5 cards used must be the same suit. All 5 cards combined must form a contiguous sequence that spans 5 numbers.

***Example***: {4h-5h-6h-7h-8h}

The highest ranking Straight Flush is a {10-J-Q-K-A} of any suit (considered a Royal Flush in Poker lingo).

***Tiebreaker***: If multiple players have a Straight Flush, the Straight Flush with the highest card value wins. If multiple players have the same highest card value, it is a tie, and the pot, if any, is split among those players.

***Example:*** {7s-8s-9s-10s-Js} beats {2h-3h-4h-5h-6h}, since J has a higher card value than 6.

{7s-8s-9s-10s-Js} ties {7h-8h-9h-10h-Jh}, since both hands have the same highest card value, which is a J.

### 7.6.2. **Four-of-a-Kind**

A Four-of-a-Kind requires 4 cards. All 4 cards must have the same card value.

***Example***: {4s-4h-4c-4d}

The highest ranking Four-of-a-Kind is an {As-Ah-Ac-Ad}.

***Tiebreaker***: If multiple players have a Four-of-A-Kind, the Four-of-a-Kind with the highest card values wins. If multiple players have the same highest card values (which can exist when the 4-of-a-kind is made up of community cards), the highest 5th card, if exists, is used to determine the winner. If a 5th card does not exist, or is the same, it is a tie, and the pot, if any, is split among those players.

***Example***: {8s-8h-8c-8d} beats {4s-4h-4c-4d}, since an 8 has a higher card value than 4.

### 7.6.3. **Full House**

A Full House requires 5 cards, where 3 of the 5 cards must have the same card value, and the other 2 cards must have the same card value.

***Example***: {9s-9h-9c-3s-3h}

The highest ranking Full House is an {A-A-A-K-K}, where the cards can be of any mix of suits.

***Tiebreaker***: If multiple players have a Full House, the Full House with the highest card values of the cards where 3 must match, wins. If multiple players have the same highest 3 card values, the player with the highest card values of the cards where 2 must match, wins. If the players still tie, the pot, if any, is split among those players.

***Example***: {Ks-Kh-Kc-6s-6h} beats {Js-Jh-Jc-2s-2h}, since the K, in the set of Ks, has a higher card value than a J.

{Qs-Qh-Qc-7s-7h} beats {Qs-Qh-Qc-5s-5h}, since the 7, in the pair of 7s, has a higher card value than a 5.

### 7.6.4. **Flush**

A Flush requires 5 cards, where all 5 cards used must have the same suit.

***Example***: {As-Js-8s-7s-3s}

***Tiebreaker***: If multiple players have a Flush, the Flush with the highest card value wins. If multiple players have the same highest card value, the pot, if any, is split among those players.

### 7.6.5. **Straight**

A Straight requires 5 cards, where all 5 cards used must form a contiguous sequence that spans 5 numbers.

***Example***: {7h-8s-9s-10d-Jc}

***Tiebreaker***: If multiple players have a Straight, the Flush with the highest card value wins. If multiple players have the same highest card value, the pot, if any, is split among those players.

### 7.6.6. **3-of-a-Kind**

A 3-of-a-Kind requires 3 cards, where all 3 cards are the same number.

***Example***: {Kh-Kd-Ks}

***Tiebreaker***: If multiple players have a 3-of-a-Kind, the 3-of-a-Kind with the highest card value wins. If multiple players have the same highest card value, up to two remaining cards in each player's hand is used to determine the winner. The player with the highest card(s), which are not the same numbers as the cards that were used to create the 3-of-a-kind, is determined the winner.

### 7.6.7. **2-Pair**

A 2-Pair requires 4 cards, where 2 cards of the 4 are the same number, and the other 2 cards of the 4 are the same number, but different than the other 2 cards. The condition where 2 cards are the same number is called a Pair.

***Example***: {6s-6d-2h-2c}

***Tiebreaker***: If multiple players have a 2-Pair, the Pair with the highest card value of both pairs determines the winner. If multiple players have the same highest card value, up to one remaining card in each player's hand is used to determine the winner. The player with the highest card, which is not the same numbers as the cards that were used to create the 2-Pair, is determined the winner.

### 7.6.8. **Pair**

A Pair requires 2 cards, where both cards are the same number.

***Example***: {7s-7d}

***Tiebreaker***: If multiple players have a Pair, the Pair with the highest card value determines the winner. If multiple players have the same highest card value, up to three remaining cards in each player's hand is used to determine the winner. The player with the highest card(s), which are not the same numbers as the cards that were used to create the Pair, is determined the winner.

### 7.6.9. **High Card**

A High Card requires only one card.

***Example***: {Ks-8d-6h-2c-4h}

***Tiebreaker***: If multiple players have a High Card, the High Card with the highest card value determines the winner. If multiple players have the same highest card value, up to four remaining cards in each player's hand is used to determine the winner. The player with the highest card(s), which are not the same numbers as the cards that were used to create the High Card, is determined the winner.