2014

MIDDLE EAST TECHNICAL UNIVERSITY

DEPARTMENT OF COMPUTER ENGINEERING

Poker Playing Agent

Software Requirement Specifications

REVISION: 1.0

STANDARD: IEEE STD 830 - 1998

A4

HÜSEYİN ALEÇAKIR

HÜSEYİN AYDIN

HEVAL AZİZOĞLU

ZÜLFİYE ARĞIN

PREFACE

This document contains the software requirements specification for the “Poker Playing Agent” software. The document is prepared according to the “830-1998 IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998.”.

This Software Requirements Specification provides a complete description of all the software requirements and views of the “Poker Playing Agent”.

The first section of this document includes scope and purpose of project and gives overall description of it.

The following sections include detailed description and requirements of the project.

Record of Changes

Version Number	Date	Number of Section	A – Added M – Modified D - Deleted	Title or brief description
1.0	30.11.2014			First Release

Table of Contents

Table of Figures

Table of Tables

1 INTRODUCTION

This software requirements specification (SRS) report gives complete description of Poker Playing Agent system. Document contains all specifications for project development with their detailed explanations. In this introduction section, purpose of this document will be described and a list of abbreviations and definitions will be provided.

1.1 PROBLEM DEFINITION

Poker is one of the most popular games among game playing agents research area of artificial intelligence. There exist many literature research and publications on adaptation of artificial intelligence to poker game. However, most of them focus on restricted area of the possible solutions. There is a universal need to examine different artificial intelligence methods and strategies on the same problem to obtain better solutions with the comparison and combination of these. Especially, academicians focus on artificial intelligence research and many commercial game companies interested in developing games played by artificial intelligence systems are facing problems on this scope. Poker Playing Agent project combines many different fields like artificial intelligence of computer science, statistical methods of mathematics, poker and game industries and so on.

1.2 PURPOSE

The purpose of this document is to give a detailed description of Poker Playing Agent system that will be designed and implemented. It will explain the purpose and features of the system, the interfaces of the system, how it will interact with external applications and the constraints under which it must operate. This document will provide a reference to the developers of the system and to the stakeholders who are future computer poker agent developers and researchers of this field.

1.3 SCOPE

The goal of this project is to design Poker Playing Agent (PPA) system. The system will be composed of two major parts, game component and artificial intelligence component.

Game component part of the system will not be implemented. All of the functionality coming from this part is intended to be obtained from a third party software. Game component part is a client software which game will be played on. The required data for artificial intelligent component will be obtained from this part. This data includes all information about current state of the game. This component also provides a graphical user interface to observe the behavior of the agents and for easy understanding for third party users.

The core part of the system, artificial intelligence component will decide what poker action to take based on the current game state. It will interact with the game component while choosing next action. The aim of the artificial intelligence component is to compare and analyze different types of computer poker algorithms.

1.4 LITERATURE SURVEY AND EXISTING SOLUTIONS

Computer poker is a popular field in artificial intelligence, hence there exists many research and publications on this topic. Many methodologies have been generated and tested with the combination of mathematics, statistics and computer science. In this part of SRS general information about this methodologies, how they have been used and their advantages and disadvantages will be discussed. Also, agents created with this methods will be indicated for further discussion and information.

1.4.1 Knowledge Based Systems

Knowledge based systems demands domain knowledge to aid the design of system.

Some famous computer poker bots in the literature are listed below:

- **Turbo Texas Hold'em:** A commercial software product that identifies a vast number of possible scenarios and encodes these decision points within a knowledge-based system
- **SOAR rule-based architecture:** A rule-based expert system for multi-player, limit Texas Hold'em, built upon the SOAR rule-based architecture.
- **Loki:** It is developed by University of Alberta. Loki is one of the first computer poker agents.

- **Poki :** Uses opponent modelling techniques and a formula-based betting strategy that accepts effective hand strength as its main input and produces a probability triple as output. It is also developed by University of Alberta.

In general, there are two different approaches for knowledge based systems: rule based expert systems and formula based strategies. Below details of both can be found.

1.4.1.1 Rule Based Expert Systems

In rule based systems if-then structure is used for various scenarios. Rules also designate many conditions to satisfy game state before it becomes activate. The result of satisfying these conditions generate a probability triple. According to probabilistic distribution, next action of agent is chosen.

1.4.1.2 Formula Based Systems

Formula based strategies are similar to rule based systems with ore generalization**.** Various methods are available for computing. Some of them are explained below.

Hand Strength Computations: Immediate hand rank (IHR) value is computed by ranking the hand relative to the entire set of all possible hands a random opponent may have. IHR value is calculated according to given formula:

$$IHR = (wins + (ties/2)) / (wins + ties + losses)$$

IHR described above provides a measure of hand strength at the current point in the hand, but makes no considerations for future community cards to be dealt. A separate measure, hand potential, can be used to compute the positive or negative effect of future community cards. Positive potential gives the probability of currently holding an inferior hand, but improving to the best hand with future community cards. Conversely, negative potential computes the probability of currently holding the better hand, but falling behind with future cards. Effective hand strength combines the results of immediate hand strength and positive potential:

$$EHS = IHS + (1 - IHS) \times PositivePotential$$

Pot Odds: A further input typically considered within a formula-based approach are the pot odds. The pot odds give a numerical measure that quantifies the return on investment by contrasting the investment a player is required to make to stay in the hand, given the possible future rewards. The equation to calculate pot odds is as follows:

PotOdds = c / (p + c)

c: the amount to call

p: the amount currently in the pot

1.4.2 Monte Carlo and Enhanced Simulation

Monte Carlo simulation involves drawing random samples for choice nodes in the game tree and simulating play until a leaf node reached which payoff value is known. This random path selection and game simulation process, i.e. trial, is repeated many times to make expected values for intermediate nodes converge to a stable set of values which may be considered as real evaluation value at the end. This may require thousands of trial. Accuracy of the agent's predictions about opponents betting actions affects the accuracy of the final evaluation values.

At the beginning of the process, it is known that there are two players who did not fold yet. Here is how this simulation usually conducted:

- Agent predicts what hand opponent may have at this point in the game and assigns this hand to opponent.
- Different simulations are being generated to estimate checking/calling and betting/raising evaluation values. Expected value of a fold action is always zero.
- Agent predicts its own betting action and its opponent's betting actions by generating probability triplets (f, c, r) and randomly choosing one of the actions based on this triple.
- For further community cards, agents predicts that card among currently available cards in the deck.
- Finally, a numerical value for the trial can be determined at either the fold or showdown nodes depending on how much money agent has won or lost during the trial.

Advantages/disadvantages of the Monte Carlo method are explained below:

- A basic game tree search involves exhaustively searching the breath of a tree until a certain cut-off depth is reached. Nodes on the frontier of the cut-off depth are then assigned values by some evaluation function and these values are back propagated to the intermediate nodes of the tree. This type of game tree search works well for games of perfect information, but fails for games involving imperfect information as the missing information ensures that the exact state of the game cannot be known. Monte Carlo

simulation provides an alternative, dynamic procedure to search state space for games with hidden information.

- Monte-Carlo simulation method was successfully used in games that involve chance events, such as backgammon, and games that involve imperfect information such as Scrabble before applying to poker. Hence outcomes of the method was clear in some manner beforehand.

Famous agents developed using Monte-Carlo simulation method:

- **r00lbot:** This poker bot is one of the earliest ones used Monte Carlo simulation to determine actions by conducting almost 3000 trial.
- **Loki-1 and Loki-2:** Agents developed by the University of Alberta CPRG to play limit Texas Hold'em against multiple opponents. Loki-2 combined knowledge based static evaluation function of Loki-1 with the simulation to gain better estimation of expected values. In the implementation of Loki-2, team used weight tables to keep all possible hand combinations of the opponent and fill this tables during trials to better determine according to more possible ones. They labeled this as "selective sampling simulation".
- **Poki:** This agent evolved from Loki. It consisted of both Formula based Strategy and Simulation based Strategy. University of Alberta CPRG then compared the behaviour of the agent with these two different strategies.
- **AKI-RealBot:** An agent developed outside of University of Alberta. Basic strategy of the agent was to exploit the weaknesses of the opponents. This strategy made agent fourth among six players in AAAI.

1.4.3 Game Theoretic Equilibrium Solutions

This procedure includes the computation of the equilibrium solution by using game theoretic principles and application of it to the poker game. Equilibrium solution is defined as a strategy such that if player change it, the result of game will be negatively affected.

Necessary information for the solution can be represented in two form: matrix form (normal form) and sequence form. Detailed information for these form can be found in paper, "Computer Poker: A Review". Matrix form is useful for relative small game such as rock-scissor-paper etc. However, deriving the equilibrium solution of poker game is not possible with matrix form. Therefore sequence form is used for solving equilibrium solution in poker game.

Although sequence form simplifies representation of information about states of poker game, since poker game has so many states, it is still difficult to get solution for the poker game. Some limitation is established for poker game like in case of Rhode Island Poker which reduces the states 10^9 times relatively the case of Texas Hold'em Poker. Nevertheless, this is still not enough

to get the equilibrium solution with linear programming. Abstraction methods will be introduced for dealing with so many states. An abstraction method is called lossless if it does not cause loss of information and lossy otherwise. In abstraction methods, states are divided into equivalent classes and this is called as bucketing or binning. All the abstraction methods use this idea in different ways. Expectation based abstraction is simplest abstraction method which uses expected hand strength for bucketing. In potential-aware automated abstraction not only current buckets are used but also buckets for future rounds are also computed. Both expectation based and potential-aware automated abstraction methods preserve the observation of previous rounds. Therefore they are called as perfect recall abstractions. In imperfect recall abstraction, agent forgets the previous observation so that more resources are used for current bucketing. There are two more lossy abstraction methods. They limit or totally eliminate betting rounds in Texas Hold'em. All of the lossy abstraction are used for solving full-scale poker problem but a solution for the abstract model cannot be said as solution of full-scale poker. Hence the strategies obtained by these models are called as near equilibrium or ε-Nash equilibrium.

Advantages/ disadvantages of this method:

- For the current state of the examination of this strategy it can be said that finding an equilibrium solution is not that easy. Although nodes of game tree are reduced by using abstraction methods, these abstraction methods cause losing information. Therefore solution of the obtained abstract model cannot be used for full-scale game.
- It is also said that this approach focus on limiting agent's own exploitability and then taking advantage of weaker opponents in paper. Therefore this strategy can be considered as defensive strategy. An agent using this kind of strategy may not lose, but also it does not win so much.

Using this method, various computer agents are developed. Some of the explained below:

- **Sparbot:** This bot uses a strategy known as PsOpti which is obtained by using expectation based abstraction with elimination of betting rounds.
- **Hyperborean:** A combination of the PsOpti suite agents, winner of the AAAI Computer Poker Competition in 2006.
- **BluffBot:** This bot uses similar abstraction with PsOpti and finished second in competition that Hyperborean won.
- **Aggrobot:** This bot uses abstraction for no-limit variation of Texas Hold'em.
- **BluffBot2.0:** No-limit version of BluffBot.
- **Adam:** This bot limits Texas Hold'em by abstracting poker game tree and executing fictious play algorithm.
- **SmallBot and BigBot:** These bots use the range of skill algorithm and obtained by reduced betting abstraction 3 bets are allowed and 4 bets are allowed per round respectively.

1.4.4 Exploitive Counter-Strategies

An exploitive strategies counter strategy is one that tries to benefit from opponent's weakness deviating from equilibrium point and leading to higher playoffs. In the first levels of the game, agent is just training itself and construct a model of opponent's behaviour in equilibrium state.

The agent may be allowing itself to be exploitive, however it will also exploit weak opposition for more value than an equilibrium strategy.

All solutions based on trying to exploit their opponents by constructing opponent models. We can divide these solutions into two main category. First one is adaptive strategies that use imperfect game search tree. Second one is static strategies that use game theoretic principles.

Advantages/disadvantages of this method:

- An agent developed according to an exploitive strategy will be able to exploit all different styles of its opponents.
- Opponents may trick the exploiter playing a certain strategy for several iterations. It's the paradox of the exploiter being exploited.

Famous bots developed with using this technique:

- **Brennus:** It is created by Maîtrepierre. Using Adaptive game tree just described Brennus placed 8th out of 17 competitors at the 2007 AAAI Computer Poker Competition.
- **Hyperborean-BR:** The Data Biased Response procedure was used to construct the 2009 version of Hyperborean, the agent which placed second in the 2009 ACPC 2-player limit bankroll competition

1.4.5 Case Based Reasoning

Case-based reasoning (CBR) is a type of lazy learning algorithm, where a set of cases are stored and maintained that encode knowledge of previously encountered situations, along with their solutions. When a new problem requires a solution, a case is created by binding appropriate values to its features.

While applying CBR to Texas Hold'em poker, all cases which includes information about previous hands with solutions and outcomes, are stored. Stored solutions are re-used to aid future betting decisions.

Famous agents which uses this method:

- **CASEY:** began with an empty case-base. As it played more and more games, cases were created and stored in the case-base along with their associated outcome.
- **CASPER:** designed to play Texas Hold'em at a full table consisting of up to 10 players. Its case-base by observing hand histories which involved formula-based and simulation-based versions of Poki. CASPER was shown to be competitive against a range of computerized opponents offered through Poker Academy Pro, including Poki.
- **Sartre:** trained on the hand history logs of the top -Nash equilibrium agents from previous year's AAAI Computer Poker Competitions.

1.4.6 Evolutionary Algorithms and Neural Networks

In evolutionary algorithms, procedures evolve a population over consecutive generations. A member of this population is evaluated by using fitness function. Members found successful according to this function produce offspring by a crossover.

Coevolution keeps separate genetic populations. This means that members from one population can compete against members from another population but evolution must be in their own populations. Pareto coevolution is an example for this approach. PC evolve limit hold'em agents which compete in full ring games with up to 10 players. These members were artificial neural networks that compete against each other over numerous generations.

Famous agents:

- **MANZANA:** A neural network based agent. It is a limit hold'em agent and winner of the bankroll division of the limit hold'em competition at the 2009 and 2010 ACPC.

1.4.7 Bayesian Poker

A Bayesian network is a graphical model that encodes probabilistic relationships among variables of interest. The edges between nodes in the graph represent dependency relationships. Each node within the network has an associated conditional probability table, which allows conditional probability values to be calculated based on the values of the parent nodes. By assigning nodes within the network different values, probabilities can be propagated throughout the network, resulting in a probability distribution over the random variables. Hence, we can use Bayesian network to maximize expected utility.

The network shows the players the final hand of the player. So their opponent figure out whether the player wins or not. The player's action is affected by final hand.

While Bayesian Poker Program has undergone several stages of development, results against other computerized players indicate there is still further room for improvement.

Agents developed using Bayesian Poker methods

Beats:

- A simple probabilistic opponent
- A rule-based opponent
- Some amateur opponents

Loses to:

- Good boots (such as bots developed by U. of Alberta)
- Experienced humans

Famous agents:

- **BPP:** AI researchers Ann Nicholson and Kevin Korb from Monash University have spent many years investigating the use of Bayesian networks within the domain of poker, beginning with five-card stud and later adapting this approach to Texas Hold'em. Korb and Nicholson have designed and applied Bayesian decision networks for heads-up, limit Hold'em resulting in a poker agent called BPP (Bayesian Poker Program). BPP competed in both the 2006 and 2007 AAAI Computer Poker Competitions. In 2006 BPP placed 3rd out of 4 competitors in the limit Hold'em bankroll competition and 4th out of 4 in the instant run-off competition.

1.5 DEFINITIONS, ACRONYMS AND ABBREVIATIONS

Abbreviations, Acronyms	Definitions
AI	Artificial Intelligence
Bankroll	Amount of money that a player have
Big Blind	Player who must put the minimum bet amount to table at the beginning of the hand.
DBMS	Database Management Systems
Dropdown	GUI element which allows the user to choose one value from a list.
GUI	Graphical User Interface
House Rake	The amount of chips that is taken from the pot by the house.

Pot	Total amount of chips that is contributed as potential winnings for a hand.
PPA	Poker Playing Agent
Small Blind	Player who must put the half of the minimum bet amount to table at the beginning of the hand.
SRS	Software Requirement Specification
Stake	Amount of money that small blind and big blind must put table.
UC	Use Case

Table 1: Definitions, Abbreviations and Acronyms

1.6 REFERENCES

The important resources which are useful for understanding the problem definition and existing solutions and guide requirement analysis are listed below:

- IEEE STD 830-1998, IEEE Recommended Practice for Software Requirements Specifications
- Rubin, J. "Computer poker: A review - Department of Computer Science." 2011. <https://www.cs.auckland.ac.nz/research/gameai/publications/CPReviewPreprintAIJ.pdf >
- Heckerman, David. *A tutorial on learning with Bayesian networks*. Springer Netherlands, 1998.
- Korb, Kevin B, Ann E Nicholson, and Nathalie Jitnah. "Bayesian poker." *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence* 30 Jul. 1999: 343-350.
- "AI Strategies for Solving Poker Texas Hold'em - SlideShare." 2014. 31 Oct. 2014 <http://www.slideshare.net/GiovanniMurru/ai-strategies-for-solving-poker-texas-holdem>
- http://www.computerhope.com/jargon/d/dropdm.htm

1.7 OVERVIEW

This SRS document is organized according to IEEE Standard 830-1998 standards and the subsections of it are adapted accordingly. Following section of this specification document will focus on describing the system in terms of product perspective, product functions, user characteristics, assumptions and dependencies. Then, in the third section, specific requirements of the system will be addressed, which will enclose external interface requirements, functional requirements of the system, performance requirements, and other requirements. In the next two chapters, data and behavioral models of the system will be given. These will be followed by a

chapter including the planning and basic schedule of the project and the document will be and with a conclusion.

2 OVERALL DESCRIPTION

In this section of the SRS document, background information about specific requirements of the system will be provided briefly. In addition to this, general factors that affect the product and outline of the functional requirements will be mentioned. In short, this section will mainly give information about product perspective, product functions, constraints, assumptions and dependencies.

2.1 PRODUCT PERSPECTIVE

Poker Playing Agent system will consist of two parts: Client Software and Poker Playing Agent. Client software will provide poker game environment for the agent system. This environment includes poker bot adaptation interfaces, game table and configuration settlement options and all other related game components that usually available in a regular poker game flow. Moreover, game statistics will be provided through this component to train agents and analyze the system performance. Poker playing agent part of the system will contain implementation of different poker bots and will be mainly related with algorithmic background of a poker game. This component will include many separately developed computer poker agents and will be used for comparing the performance of different methodologies and algorithms currently popular and well-known in the literature.

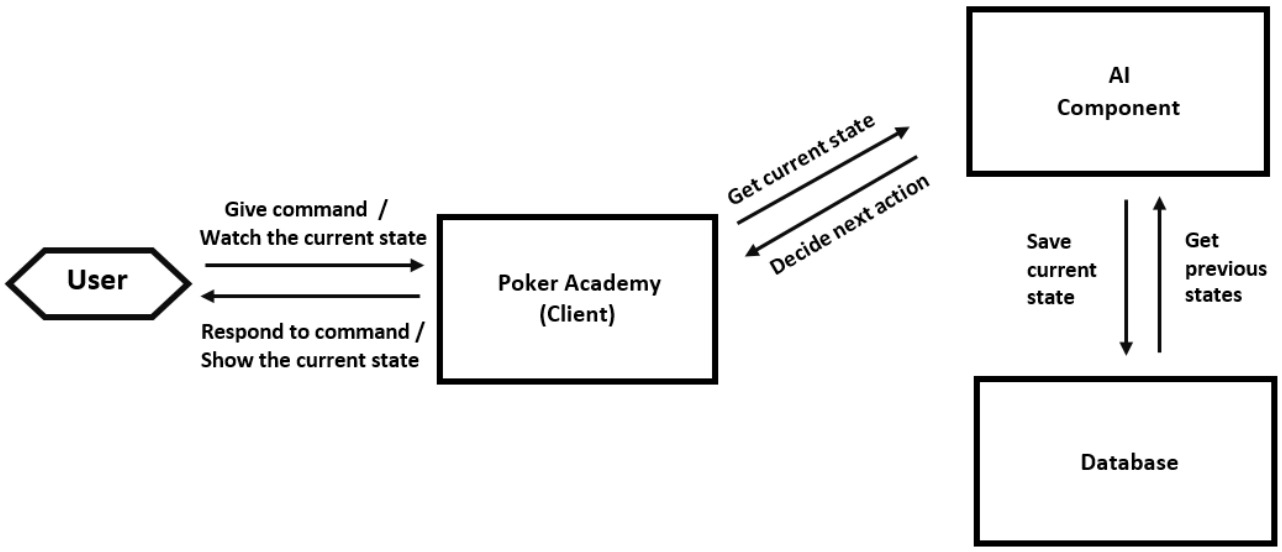

Figure 1: Context Diagram

We will describe how the software operates inside various constraints in the following subsections.

2.1.1 System Interface

As stated previously in the introduction part of this section, Poker Playing Agent system will not be self-contained and it will depend on a third party poker client software to execute the gameplay and obtain the related data from the game. This third party software is expected to provide an easy to use and comprehensive application programming interface and the connection between two parts of the system will be obtained through it.

2.1.2 User Interfaces

User interfaces of this system will be obtained from third party client software since main purpose of this project is to implement the logic at the backend side. Main component, Poker Playing Agent, will work behind the interfaces; it will make decisions during the poker game however users, who observe the game, will not have ability to interact with the agent other than playing against it. Hence there is no need to implement extra user interfaces. All of the demands on user interfaces can be satisfied from an external application. Basic functionalities intended to obtain from user interfaces are listed below:

- User will interact with the system like a desktop application.
- User will not be aware of the implementations of poker agents, i.e. there will not be any functionality in the user interfaces to direct user to agent algorithms.
- User will encounter a game opening menu when she/he enters the application. In this menu game options will be shown to user. User will choose game option by clicking on the naming buttons.
- Game type selection will direct user to a menu where he/she can choose poker game type to play like limit and no-limit.
- In this poker type menu, user can enter an existing table associated with a type or he/she can create a new game table with clicking “Add New Table” button on the interface.
- User can choose which agents will sit on a table with a list like menu. This menu will enable user to select bots from a list or delete already sitting ones from the table.
- There will be an interface to add newly created poker agents to game. This menu will just let agent to be recognized by the game.
- When user clicks “Start Game” button, poker game will start. After this point, a regular poker game will be held between computer poker agents and human player.

- Game table interface will be user-oriented, in other words, table will not be complicated; players, cards and pot amounts will be shown clearly.
- In game table interface player whose turn is currently, big and small blind amount will be indicated.
- Current action options for the human player will be shown as a layout in the game table interface while it is human players turn. Options will be bet/raise, check/call and fold and human player will choose one of these through a checkbox.
- There will be a menu for user to disable human player in the game. After this selection human player will only sit on the game table as an observer.
- There will be statistical information bars for human player on the game table interface.

Mock interfaces related with these functionalities are given below.

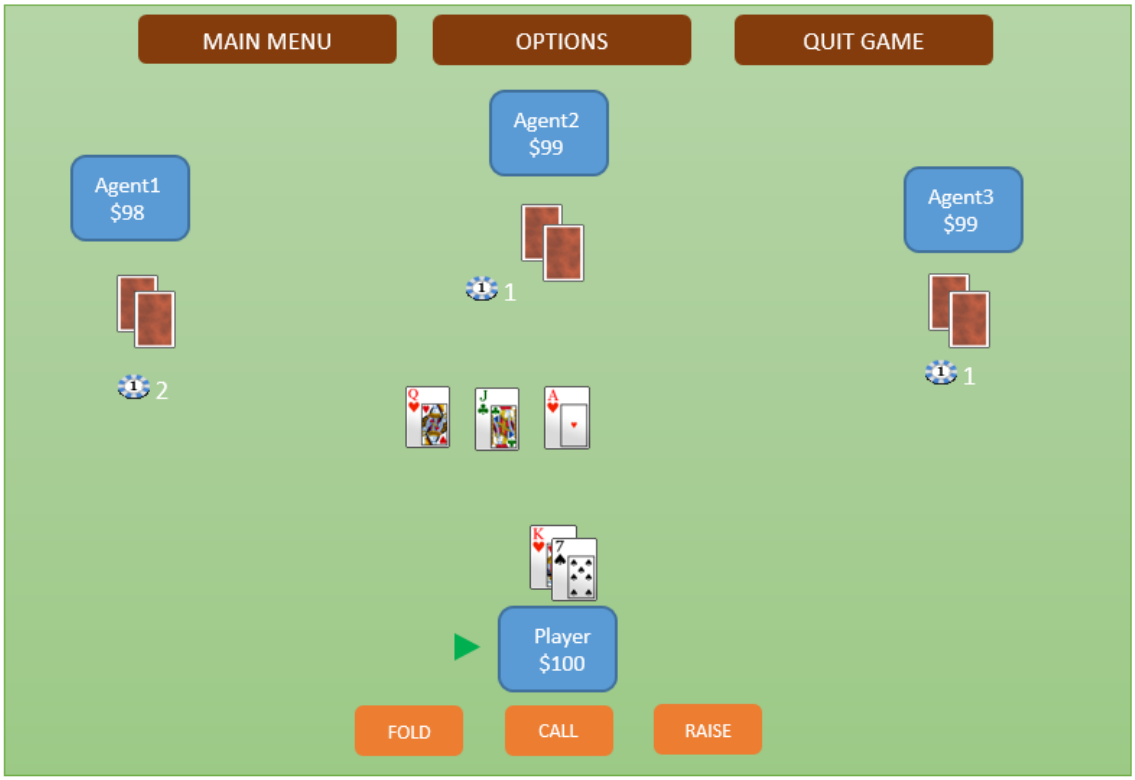

Figure 2: Mock User Interface for Game Scene

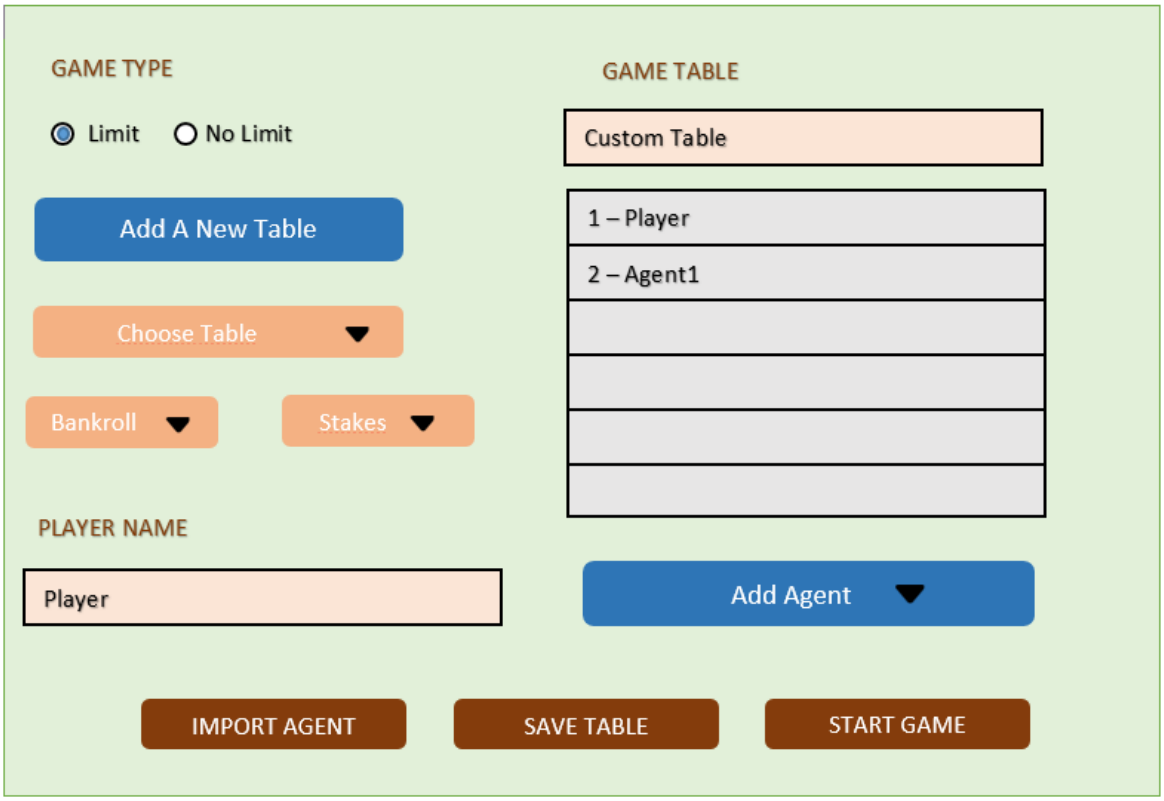

Figure 3: Mock User Interface for Game Configuration

As already stated above, all of these functionalities can be obtained from a third party poker client software. At the end of the initial investigation, it is observed that Poker Academy is the best option at the moment. It satisfies almost all of these expectations and also it offers more such as many previously created and famous poker bots to play against, many statistical tools like hand evaluator. Moreover, user interfaces of the Poker Academy is basic and easy to interact and observe during the gameplay. As a result, Poker Academy will be used for Client Software part of the system during development. Two screenshots from Poker Academy can be found in the following page.

Figure 4: Screenshot of Game Scene

Figure 5: Screenshot of Game Configuration

2.1.3 Hardware Interface

This system will work as a desktop application without any specific hardware requirement. However a reasonable CPU speed and sufficient amount of memory is expected for the efficiency of computations and smooth flow of user interactions.

2.1.4 Software Interface

Poker Playing Agent system will use a client software which provides interfaces to end users. Currently, Poker Academy is the first option to supply these interfaces such as add AI component, choose game type, choose poker type, choose table, create new table, add bots to game table, load table, choose spectator mode and deal hand.

The system will be able to work on Windows operating system. Hence, Windows OS will be used during development process. MySQL will be used as database server. Java will be used as a programming language while developing computer poker agents. NetBeans integrated development environment (IDE) will be used for developing primarily with Java.

Name	Version	Source
Windows	7 / 8	windows.microsoft.com
MySQL	5.6	dev.mysql.com
Java	5	java.com
Netbeans IDE	8.0.1	netbeans.org
Poker Academy	2.5	poker-academy.com

Table 2: Software for the PPA System

2.1.5 Memory Constraints

Poker playing agent system will make calculations and inferences based on data collected from previously executed operations of this system hence a high capacity for memory shall be provided. Also, data related to each of the poker game played will be kept in the database, hence there will be a need for extra memory for this.

2.1.6 Operations

Details of the system's operations is explained in section 2.1.1 User Interfaces and further information will be given in section 2.2 Product Functions.

2.1.7 Site Adaptation Requirements

Poker playing agent system requires Java 5 to work on an environment hence this requirement shall be satisfied before the adaptation. There is no other specific requirement for site adaptation.

2.2 PRODUCT FUNCTIONS

A general use case diagram that shows all use cases of the system is provided below.

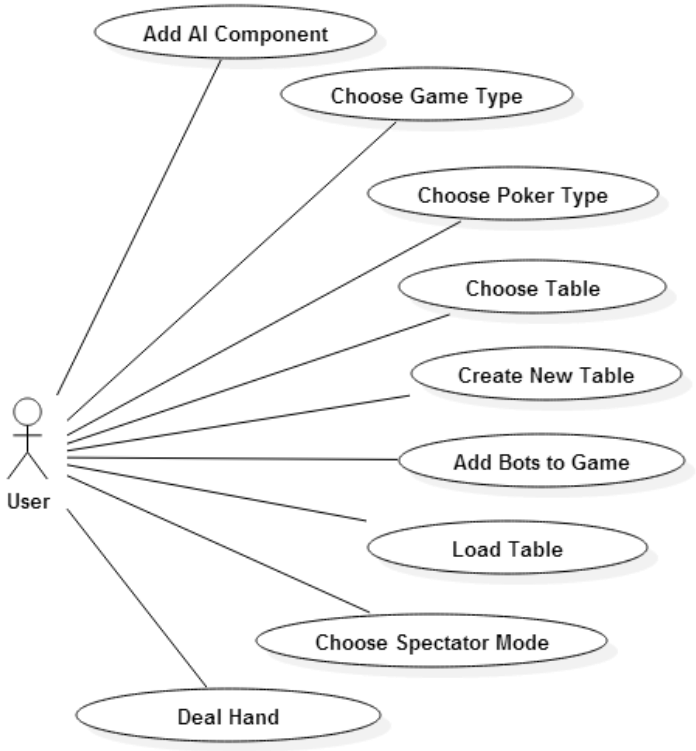

Figure 6: General Use Case Diagram

2.2.1 Add AI Component

Figure 7: UC1 - Add AI component

Users may develop their own artificial intelligence bots. Using this functionality these different poker bots can be added to system. In order to add an AI bot:

- User puts the configuration file of the bot, namely pd file, into the “bots” folder which is located in “PokerAcademyPro2/data/”.
- User clicks “Opponent Manager” button in the game menu.
- User clicks the “Import” button and choose pd file of the bot.
- User clicks “Save” button.

2.2.2 Choose Game Type

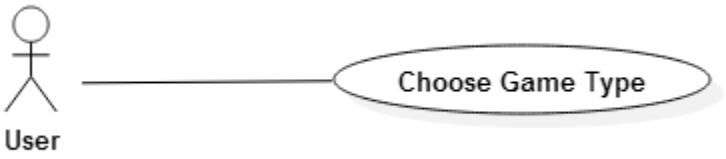

Figure 8: UC2 - Choose Game Type

User can choose type of the poker game by using this functionality:

- After the poker academy is started three different game types at the main menu which are “Ring Games”, “Tournament”, “Poker Academy Online” are shown to the user.
- With “Tournament” option user will be able to attend previously held tournaments. User can join to a tournament by click the button “Tournament”.
- To adapt and play their own artificial intelligence bots, user should be entered in “Ring Games”. User can choose this mode by clicking the button “Ring Games”.

“Poker Academy Online” is not in the scope of the system for the time being.

2.2.3 Choose Poker Type

Figure 9: UC3 - Choose Poker Type

After choosing “Ring Games” option in the main menu, user will be able to choose which type of poker he/she and bots will be play:

- Options which are “Limit” and “No-Limit” are shown to the user.
- There is no need for another action to choose poker type. By choosing table under these options, user also chooses the poker type.
- User may hide or expand the table list of these options by double clicking name of the option.

2.2.4 Choose Table

Figure 10: UC4 - Choose Table

Under the “Limit” and “No-Limit” options, user will be able to choose an existing game table:

- User clicks the name of the table which are created according available bots capabilities, initial bankroll value and number of players.
- User may change these initial configurations of the tables and may join with his/her own bot to the game.

2.2.5 Create New Table

Figure 11: UC5 - Create New Table

This functionality is for the user who wants to create a new table instead of choosing an existing game table. In order to create a table:

- User clicks “Create a new table” button.
- When the table created its name is “Untitled” by default. User may change it by using the table name field.
- User can add bots to the game table, set initial bankroll, stakes and house rake amounts by clicking the dropdown menu of these fields.
- After any changes a “Save” button appears near the table name field. User can click this button if he/she wants to save the configuration of the table.

2.2.6 Add Bots to Game Table

Figure 12: UC6 - Add Bots to Game Table

By using this functionality user can add bots he/she chooses to the table. In order to do this:

- User clicks right button on the player list layout.
- A list of available AI bots on the system is shown to the user.
- User can choose their own bots or another famous bots in the literature to add game configuration.

2.2.7 Load Table

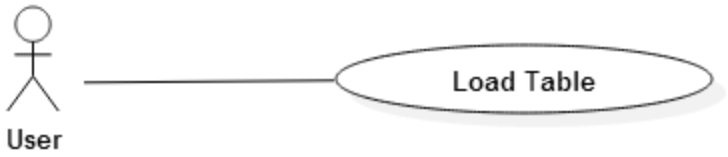

Figure 13: UC7 - Load Table

After a table chosen or the configuration of a new table is done in order to start the game:

- User clicks “Load Table” button or double clicks the name of the table.
- Initial game screen will be created according to chosen bots and bankroll amount and game will be ready to start.

2.2.8 Choose Spectator Mode

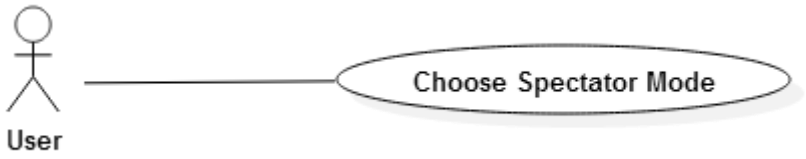

Figure 14: UC8 - Choose Spectator Mode

Game will provide the human player an option to attend the game as an observer or as a player. Initially user will be added to the game table as a player automatically. If user chooses to become an observer:

- User clicks his/her player name and set bankroll from appearing menu.
- User enters “0” as bankroll amount and clicks “OK” button.
- After that if user does not want the game flow to be interrupted, he/she clicks “Options” menu and then “Auto Deal”.

2.2.9 Deal Hand

Figure 15: UC9 - Deal Hand

After game table is available for user, he/she may manage card dealing process:

- If “Auto Deal” option is selected in the game menu, it must be clicked as mentioned previous use case again so that it is closed. Closing these option will enable user to use analytical tools game will provide, such as “Player Statistics”, “Hand Evaluator” for each hand separately.
- After the auto deal is closed user clicks “Deal Hand” button to start a hand in the game. Even if user wants to use auto deal property user must click “Deal Hand” button to start first hand of the game.

2.3 USER CHARACTERISTICS

The users of system can be divided into two categories: general user who use that product to improve their poker game abilities and poker agent developer who use the product to develop

new poker agents and compare it with other poker agents. General user is expected to have at least intermediate level of poker experience. However, there is no other restriction on general user's educational level. Poker agent developer is expected to have professional level poker experience and also should have know-how on artificial intelligence field and strong programming background. Furthermore, since the system will be in English, both users of the system are supposed to have an adequate level of comprehension.

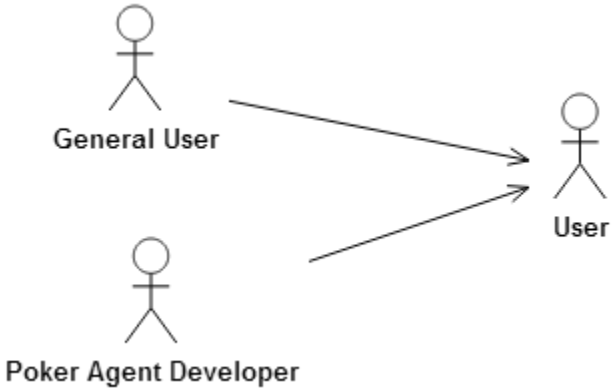

Figure 16: Types of User Diagram

2.4 CONSTRAINTS, ASSUMPTIONS AND DEPENDENCIES

- Huge amount of data will be needed to test some certain types of poker agents, developer will need enormous disk space.
- Java 5 is needed to create computer poker agents and adapt them to the system.
- Development environment should be choose as Windows since executable of Poker Academy is in .exe format compiled for only this platform.
- Poker playing agents should be implemented in Java to direct integration to system. Otherwise a special plug-in should be used to adapt agents implemented in other programming languages.
- DBMS should be capable of receiving and handling large number of requests simultaneously.
- Poker Academy 2.5 requires 128 MB RAM, 200 MB disk space and 800 MHz CPU.

3 SPECIFIC REQUIREMENTS

In this section, external interface requirements, functional requirements, performance requirements and logical database requirements will be discussed. Moreover, there will be information about design constraints and software system attributes.

3.1 EXTERNAL INTERFACE REQUIREMENTS

All the information related with interfaces is given in section 2.1 Product Perspective. Two basic GUIs which show expected properties from a game component and screenshots from chosen game component, Poker Academy, can be found in this section. There are no additional requirements about external interfaces which are needed to be discussed in this section.

3.2 FUNCTIONAL REQUIREMENTS

3.2.1 Add AI Component

3.2.1.1 Introduction/Purpose of feature

The Poker Playing Agent system shall give opportunity poker agent developer to add their own poker playing bots to challenge with other poker bots or real users.

3.2.1.2 Associated functional requirements

Use Case Name	UC-1. Add AI Component
Actors	Poker Agent Developer
Priority	Essential
Trigger	This use case is triggered when user loads poker agent into data folder of client software.
Precondition	• User shall have a bot in JAVA5 programming language • User shall have a project document file per bot
Basic Path	1. User starts to use their system choosing standard ring games. 2. User uses opponent manager to introduce their bots to the PPA system

Alternate Path	1. User starts to use their system choosing standard ring games. 2. User uses opponent manager to introduce their bots to the PPA system
Post Condition	Poker playing agents becomes available within the game and any game configuration includes this agent becomes available.
Exception Path	1. If the bot project document file defective, opponent manager rejects bot. 2. Opponent manager waits user to add corrected bot
Other	
Reference	Section 2.2.1

Table 3: UC1 - Add AI Component

3.2.2 Choose Game Type

3.2.2.1 Introduction/Purpose of feature

The Poker Playing Agent system shall give opportunity both poker agent developer and general user to choose game type at the main menu which are “Ring Games”, “Tournament”, “Poker Academy Online”.

3.2.2.2 Associated functional requirements

Use Case Name	UC-2. Choose Game Type
Actors	Poker Agent Developer and General user
Priority	Essential
Trigger	This use case is triggered when user PPA system.

Precondition	• User shall click the PPA executable
Basic Path	1. User opens the executable file
Post Condition	User is able to connect desired game type
Exception Path	N/A
Other	N/A
Reference	Section 2.2.2

Table 4: UC2 - Choose Game Type

3.2.3 Choose Poker Type

3.2.3.1 Introduction/Purpose of feature

The Poker Playing Agent system shall give opportunity both poker agent developer and general user to choose game type such as Limit and No-Limit.

3.2.3.2 Associated functional requirements

Use Case Name	UC-3. Choose Poker Type
Actors	Poker Agent Developer and General user
Priority	Essential
Trigger	This use case is triggered when user open a table.
Precondition	• User shall have a table
Basic Path	1. User opens the ring games 2. User loads a table which is Limit type
Alternate Path -1	1. User opens the ring games 2. User loads a table which is No-Limit type
Alternate Path -2	1. User creates the ring games 2. User loads a table which is No-Limit type
Alternate Path -3	1. User creates the ring games 2. User loads a table which is Limit type
Post Condition	User are able to connect desired poker type
Exception Path	N/A
Other	N/A
Reference	Section 2.2.3

Table 5: UC3 - Choose Poker Type

3.2.4 Choose Table

3.2.4.1 Introduction/Purpose of feature

The Poker Playing Agent system shall give opportunity both poker agent developer and general user to click the name of the table which are created according available bots capabilities, initial bankroll value and number of players

3.2.4.2 Associated functional requirements

Use Case Name	UC-4. Choose Table
Actors	Poker Agent Developer and General user
Priority	Essential
Trigger	This use case is triggered when user clicks the name of table.
Precondition	• User shall have a table
Basic Path	1. User clicks the ring games 2. User loads a table which is Limit type 3. User set the starting bankrolls 4. User set the stakes 5. User set the house rake
Alternate Path -1	1. User opens the ring games 2. User loads a table with default values
Post Condition	User are able to join table
Exception Path	N/A
Other	N/A
Reference	Section 2.2.4

Table 6: UC4 - Choose Table

3.2.5 Create New Table

3.2.5.1 Introduction/Purpose of feature

The Poker Playing Agent system shall give opportunity both poker agent developer and general user to create their new poker playing environment which is table in poker jargon.

3.2.5.2 Associated functional requirements

Use Case Name	UC-5. Create New Table
Actors	Poker Agent Developer and General user
Priority	Essential
Trigger	This use case is triggered when user clicks create a new table.
Precondition	• User have already selected the one of the game types from the Main Lobby page
Basic Path	1. User selects a game type 2. User selects a No-Limit poker type 3. User clicks a create new table button 4. User sets the new untitled table
Alternate Path -1	1. User selects a game type 2. User select a Limit poker type 3. User clicks a create new table button 4. User sets the new untitled table
Post Condition	User have a new table to join the game
Exception Path	N/A
Other	N/A
Reference	Section 2.2.5

Table 7: UC5 - Create New Table

3.2.6 Add Bots to Game Table

3.2.6.1 Introduction/Purpose of feature

The Poker Playing Agent system shall give opportunity both poker agent developer and general user to add bots to the table. These bots have already imported using opponent manager.

3.2.6.2 Associated functional requirements

Use Case Name	UC-6. Add Bots to Game Table
Actors	Poker Agent Developer and General user
Priority	Essential
Trigger	This use case is triggered when user clicks select opponents menu in the game table
Precondition	• The PPA system have already created bots inside the system
Basic Path	1. User starts to use their system choosing standard ring games. 2. User uses opponent manager to introduce their bots to the PPA system 3. User creates new table 4. User clicks on empty chairs by depressing right-hand mouse button and selects the opponents
Alternate Path -1	1. User starts to use their system choosing standard ring games. 2. User uses opponent manager to introduce their bots to the PPA system 3. User loads table which has been already created 4. User clicks on empty chairs by depressing right-hand mouse button and selects the opponents
Post Condition	The new bot is added the poker table and it is ready to play
Exception Path	1. User starts to use their system choosing standard ring games. 2. User loads table which has been already created

	3. User uses opponent manager to introduce defective bots to the PPA system 4. The game does not start
Other	N/A
Reference	Section 2.2.6

Table 8: UC6 - Add Bots to Game Table

3.2.7 Load Table

3.2.7.1 Introduction/Purpose of feature

The Poker Playing Agent system shall give opportunity both poker agent developer and general user to accessing game table by left-clicking Load Table button.

3.2.7.2 Associated functional requirements

Use Case Name	UC-7. Load Table
Actors	Poker Agent Developer and General user
Priority	Essential
Trigger	This use case is triggered when user pressing Load Table button by Left-Click.
Precondition	• There have been already created at least one table in the system • User choose a table
Basic Path	1. User selects a game type 2. User selects a No-Limit poker type 3. User clicks a create new table button 4. User sets the new untitled table 5. User choose a table by left-click on its name 6. User pressing Load Table button by left-click

Alternate Path -1	1. User selects a game type 2. User select a Limit poker type 3. User clicks a create new table button 4. User choose a table by left-click on its name 5. User sets the new untitled table 6. User pressing Load Table button by left-click
Post Condition	User have a new table to join the game
Exception Path	N/A
Other	N/A
Reference	Section 2.2.7

Table 9: UC7 - Load Table

3.2.8 Choose Spectator Mode

3.2.8.1 Introduction/Purpose of feature

The Poker Playing Agent system shall give opportunity both poker agent developer and general user to give an option to attend the game as an observer or as a player. Initially user shall be added to the game table as a player automatically.

3.2.8.2 Associated functional requirements

Use Case Name	UC-8. Choose Spectator Mode
Actors	Poker Agent Developer and General user
Priority	Essential
Trigger	This use case is triggered when the player's bankroll set to 0.
Precondition	• User have already joined the table

Basic Path	1. User selects a game type 2. User selects a No-Limit poker type 3. User clicks a create new table button 4. User sets the new untitled table 5. User choose a table by left-click on its name 6. User pressing Load Table button by left-click 7. User set human player's bankroll amount to zero 8. Human player's new status is observer
Alternate Path -1	1. User selects a game type 2. User select a Limit poker type 3. User clicks a create new table button 4. User choose a table by left-click on its name 5. User sets the new untitled table 6. User pressing Load Table button by left-click 7. User set human player's bankroll amount to zero 8. Human player's new status is observer
Post Condition	The human player observe the game
Exception Path	N/A
Other	N/A
Reference	Section 2.2.8

Table 10: UC8 - Choose Spectator Mode

3.2.9 Deal Hand

3.2.9.1 Introduction/Purpose of feature

The Poker Playing Agent system shall give opportunity both poker agent developer and general user to distribute cards to players and manages the action at a poker table.

3.2.9.2 Associated functional requirements

Use Case Name	UC-9. Deal Hand
Actors	Poker Agent Developer and General user
Priority	Essential
Trigger	• This use case is triggered when user clicks deal hand button by left-click or • System automatically deals hands in every step in the game
Precondition	• User have already joined the table
Basic Path	1. User selects a game type 2. User selects a No-Limit poker type 3. User clicks a create new table button 4. User sets the new untitled table 5. User choose a table by left-click on its name 6. User pressing Load Table button by left-click 7. User clicks deal hand button 8. System deals hands
Alternate Path -1	1. User selects a game type 2. User select a Limit poker type 3. User clicks a create new table button 4. User choose a table by left-click on its name 5. User sets the new untitled table 6. User pressing Load Table button by left-click 7. User set human player's bankroll amount to zero 8. Human player's status is observer 9. System starts to deal hands automatically
Alternate Path -2	1. User selects a game type 2. User selects a No-Limit poker type 3. User clicks a create new table button

	4. User sets the new untitled table 5. User choose a table by left-click on its name 6. User pressing Load Table button by left-click 7. User choose option to deal hand by automatically 8. System deals hands automatically
Post Condition	System starts to deal hands
Exception Path	1. User selects a game type 2. User selects a No-Limit poker type 3. User clicks a create new table button 4. User sets the new untitled table 5. User choose a table by left-click on its name 6. User pressing Load Table button by left-click 7. All player left the game 8. System stops to deal hands
Other	N/A
Reference	Section 2.2.9

Table 11: UC9 - Deal Hand

3.3 PERFORMANCE REQUIREMENTS

3.3.1 Agent Performance Metrics

It has been always hard to measure the performance of a computer poker agent, since it has so many unknowns during the game playing and it is hard to decide which action is the appropriate one. Usually a play-and-observe based approach has been adapted for evaluation in the literature. In other words, performance of an agent is being compared with other formerly successful agents or real human players to decide whether it was successful or not. Here, some of the requirements to evaluate performance of an agent will be explained.

- Agent should know where to fold and where to bet like a real human player. It should not play too blindly or too safe.
- Agent should not use the same strategy during a hand, it should make changes between them according to the opponent's moves.
- Agent should beat an average human player.
- Agent should be able to beat some of the known agents in the literature.
- Agent's response time should not exceed an average human's.

3.3.2 Game Component Performance Metrics

- During the game, response time of the default game agents should be ideal.
- Game flow should be easily observable, it should not flow too fast for a human to track.
- This component generates high amounts of data and sends them to artificial intelligence component. These actions should not cause a delay in the game flow.
- Game component should not keep unnecessary data in the memory for efficiency reasons.

3.4 LOGICAL DATABASE REQUIREMENTS

In order to describe the logical database requirements an entity - relationship diagram is provided below.

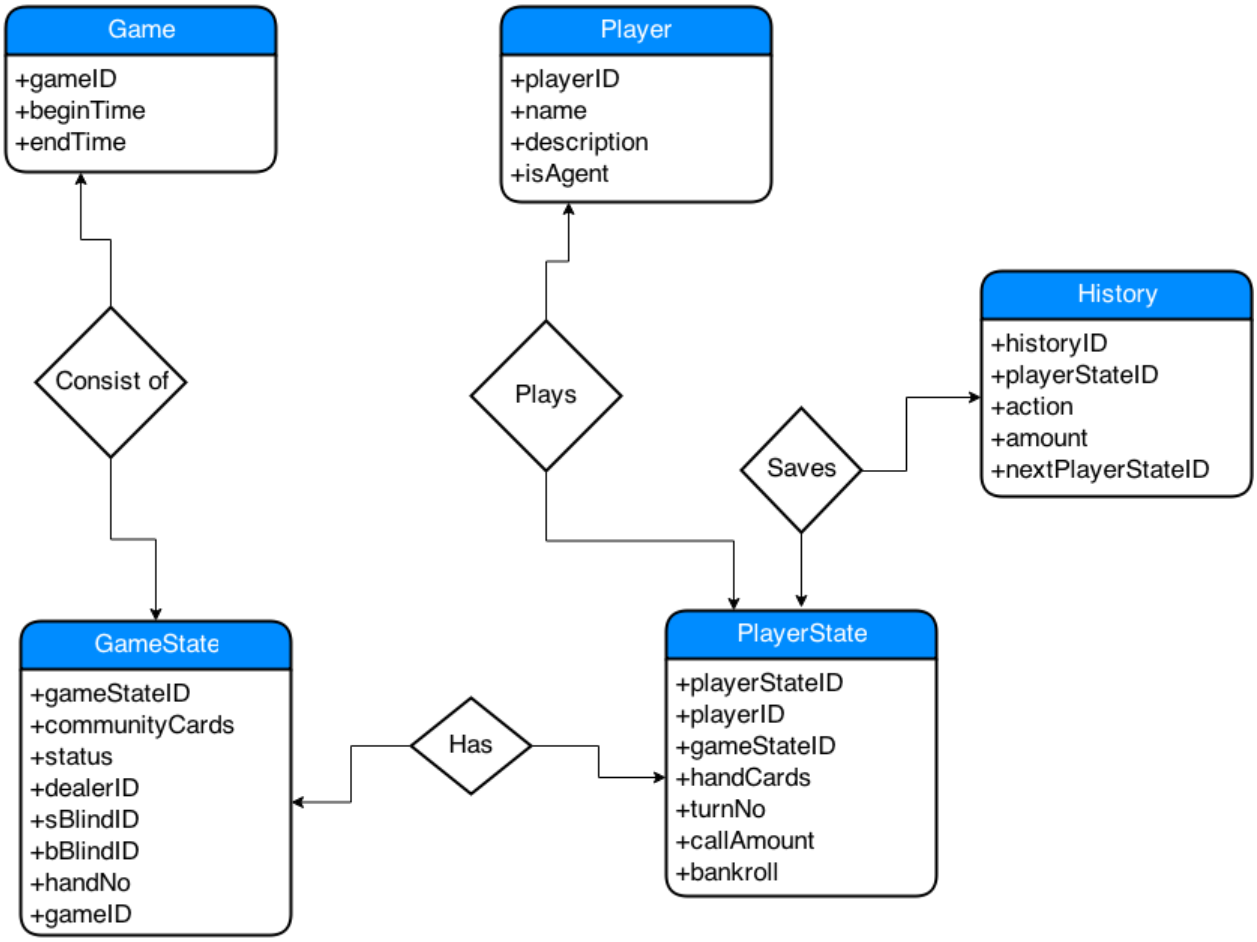

Figure 17: ER diagram

3.5 DESIGN CONSTRAINTS

- Java Programming Language specifically five edition will used as programming language
- UML Standards will be used for the diagrams
- IEEE Standards will be used for the reports
- The system will run as a single standalone system
- Average personnel computer will be enough for our system

3.6 SOFTWARE SYSTEM ATTRIBUTES

3.6.1 Reliability

- Poker Playing Agent system should keep database back-ups in case of failure.
- When system crashes, system should recover user and system information from database back-ups.

3.6.2 Maintainability

- During the design and development stages, requirements and change management should be followed.
- Changes on system should be implemented carefully with using a release control system. Any of these changes should be linked with its request in order to provide a traceable system.
- System complexity should be set minimum with following object oriented design and development concepts.
- All of the design choices should be documented regularly.

3.6.3 Portability

- The system will be portable with JAVA5 Programming language
- The system will run on Windows operating systems
- The system will be compiled integrated development environment such as NetBeans and Eclipse

3.7 ORGANIZING SPECIFIC REQUIREMENTS

3.7.1 System Mode

All the necessary operations of the system are discussed in section 2.1.2.

3.7.2 User Classes

The only user class in the system is Human Player class. Human Player has the capability of adding new agents to the system, creating game table, choosing agents for the game table.

3.7.3 Objects

A class diagram describing the object relationships of the system is provided in Figure X. The main purpose of this diagram is to introduce the basic classes in the system. Therefore, the diagram is not detailed. Parameters and return types of the methods in the classes are not shown. Relationships are shown as associations except Player-Agent generalization and Player-HumanPlayer generalization. Detailed and accurate class diagram will be provided in Software Development Description document.

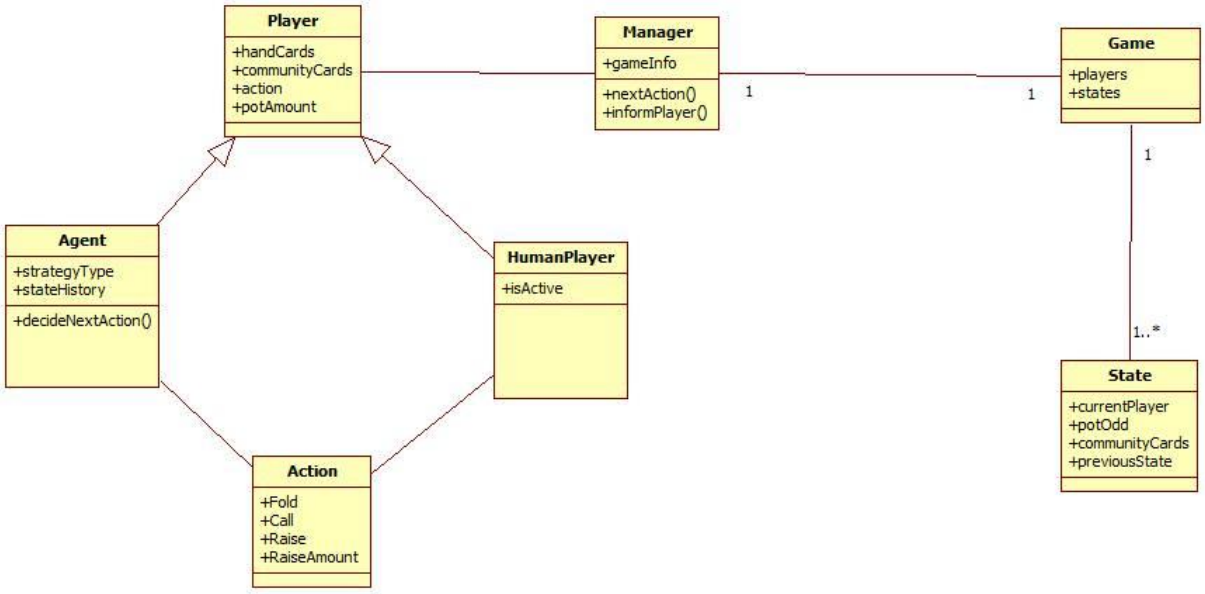

Figure 18: Class Diagram

3.7.4 Features

A sequence diagram that shows user – system interaction is provided in Figure 19.

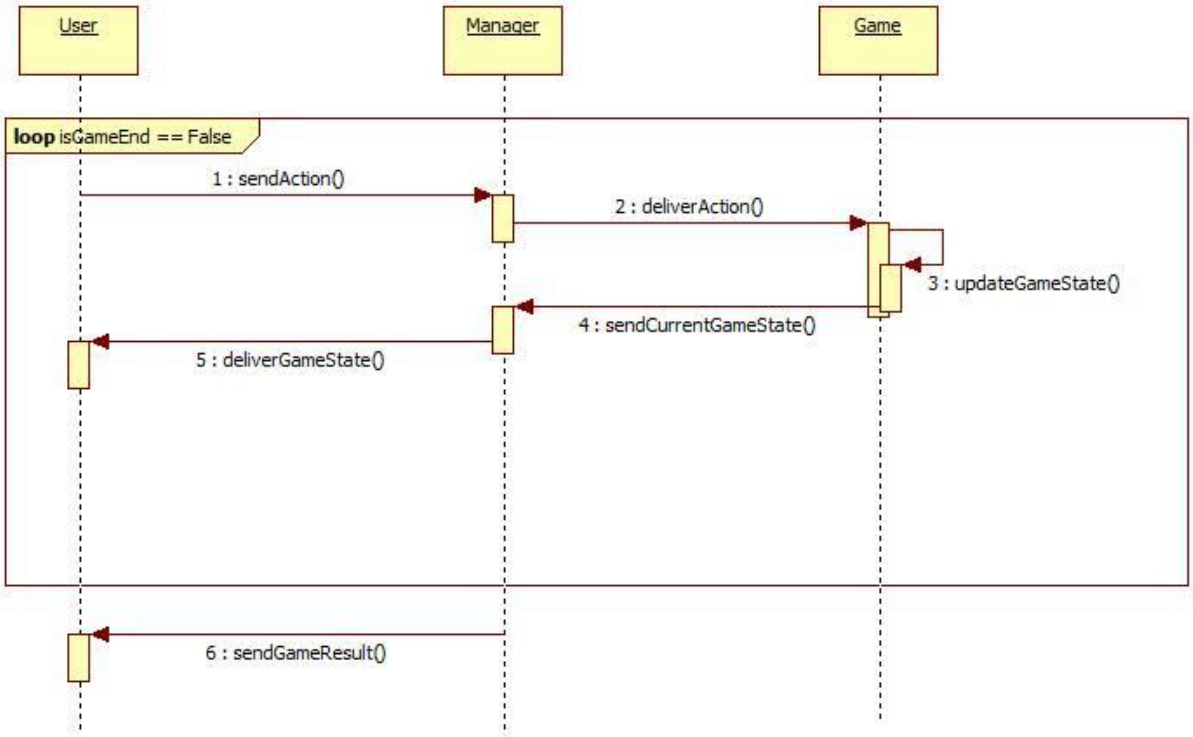

Figure 19: Sequence Diagram

3.7.5 Stimulus

The stimuli of the system are the actions of the players. The flow of the operations in the system is mainly based on these actions. User commands directs game flow during the configurations of the game, but if human player is included to the game, user commands will be a criteria for the flow of the game.

3.7.6 Response

Functions of the system cannot be classified according to their responses.

3.7.7 Functional Hierarchy

This organization scheme is not applicable for the PPA system.

3.8 ADDITIONAL COMMENT

No other requirements are noted.

4 DATA MODEL AND DESCRIPTION

4.1 DATA DESCRIPTION

4.1.1 Data Objects

In this section of the document, the relationships of interconnections and access among entities are specified. These relationships include information sharing and order of execution. Detailed class diagram is provided in Section 3.7.3.

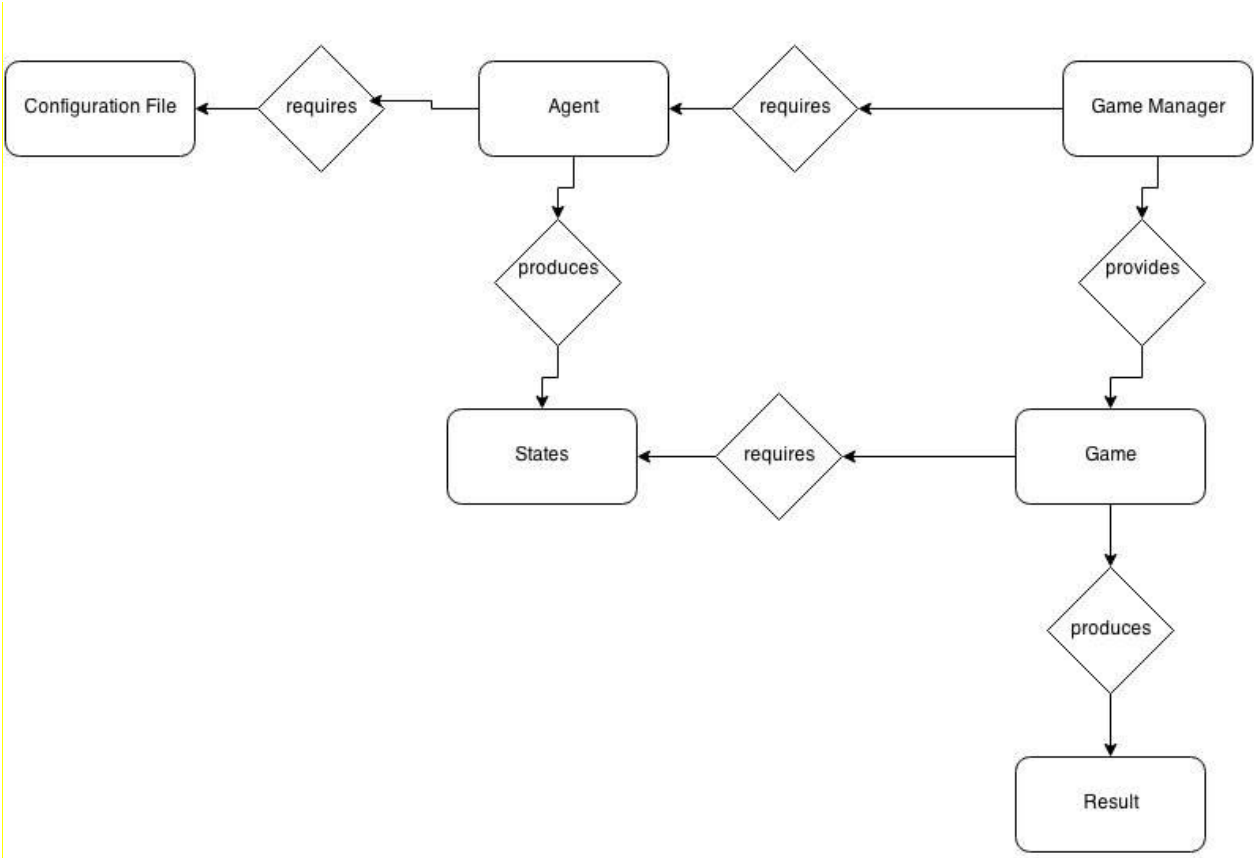

Figure 20: Data Model

5 BEHAVIORAL MODEL AND DESCRIPTION

5.1 DESCRIPTION FOR SOFTWARE BEHAVIOR

Game Playing Agent's workflow basically contains agent creation, adaptation of this agent to system and generation of a gameplay. System starts when a user creates an agent and wants to adapt it to the system. User makes agent recognizable by creating necessary configuration files and integrate them into the data folders. After the agent addition, user chooses poker game type, initial configurations of a game table. Also user may attend the poker game as a human player if he/she wants. After all of these, user starts a poker game. Poker game is executed as a normal Texas Hold-em Poker. When the game ends, user is informed about his/her agent's success during the game with the information of each hand. User is able to observe the process through the user interfaces provided by poker client software.

5.2 STATE TRANSITIONS DIAGRAM

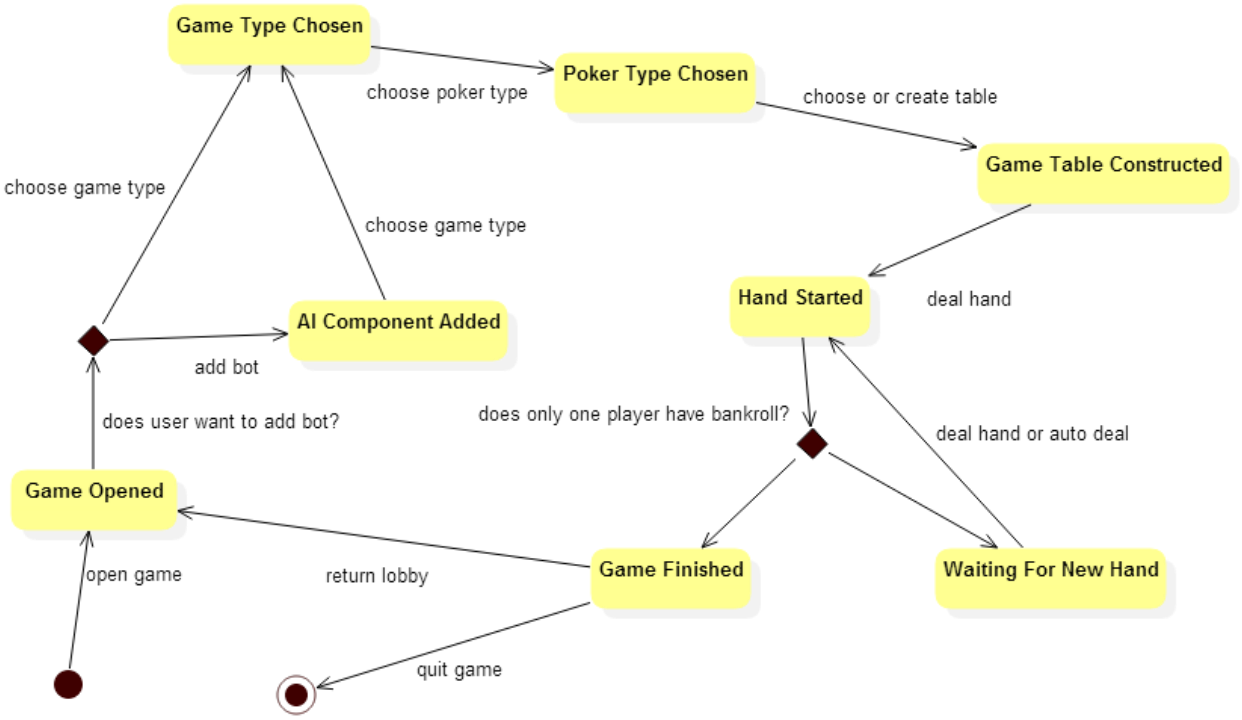

Figure 21: State Diagram

6 PLANNING

In this part of the document, the structure of the team responsible from the project, the basic schedule, and the process model will be presented.

6.1 TEAM STRUCTURE

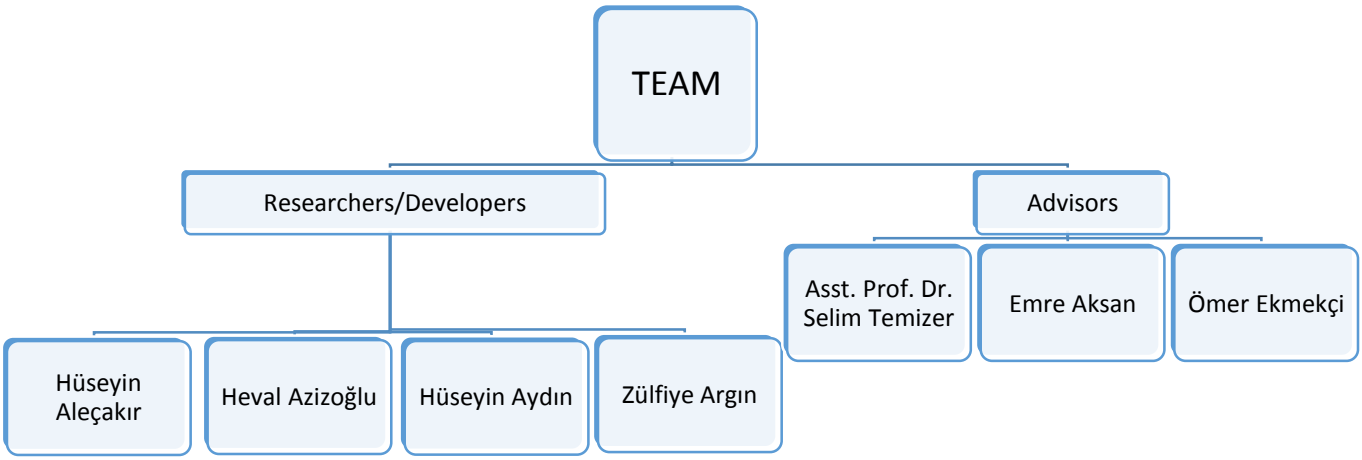

Figure 22: Team Structure

6.2 ESTIMATION (BASIC SCHEDULE)

This project is planned to be finished by June 2015. Process will be divided into parts according to major requirements of the project and after each part ends, there will be a revision session to catch up what has been done and what should be done next.

First, a literature research will be held about major methodologies and techniques in the poker playing agent development. This first literature research will be a base to the further and deeper surveys which will be done during the lifetime of the project. This research part will approximately take 4 weeks.

After the research, poker game client software and big data generation and usage solutions will be investigated. A client which enables developers to integrate agents easily and provides a well-done poker game environment will be looked for. 2 weeks will be used for this part.

During the rest of the lifetime of this project, algorithm design, implementation and testing processes will be done one after another as a cycle. Various number of algorithms will be developed and implemented while creating agents according to most popular and accepted algorithms in the literature currently. Each time, an algorithm will be chosen according to their technical requirements and a deep research will be done about that algorithm then it will be implemented. An exhaustive testing and observing process will also be held after the implementations due to the nature of the project. Each of these cycles will take approximately 3-4 weeks.

6.3 PROCESS MODEL

In this section of the document, software development processes will be discussed in detail since making a decision on the methodology conforming the needs and requirements of a software project is extremely important while developing a software product.

First methodology which can be used is Waterfall. In this method, every phases of the software development follows each other. Once a phase is done, there is no revision process to make improvements on that phase. Hence all of the decisions and designs should be ready at the very beginning of the project and they should be followed thoroughly. Waterfall methodology is the most traditional and rule-based method and it is hard to adapt it into the large scale, constantly evolving projects.

The other most basic methodology should be discussed is Agile Software Development. This methodology is based on iterative and incremental development idea hence it is suitable for changing requirements and it provides sustainable development. Scrum is widely used and well-practiced process model in the agile development. In Scrum, development process divided into

parts, called sprints. These sprints usually takes no longer than 2 weeks to enhance incremental development. During each sprint, each team member is assigned with a number of tasks which will make improvements on the product. These tasks, how team handles them and whether the processes are being executed properly is checked with regular meetings in Scrum methodology. Sprint planning, Daily Stand-ups, Grooming and Retrospective Meetings are the most basic communication and interaction ways inside the team.

At the first sight, Scrum may be seen as it is not well correlated with research projects however in this point, it is proper to investigate the nature and the requirements of the project. In Poker Playing Agents, team will implement separate algorithms but each algorithm may indicate a clue to improve previously implemented algorithms hence there could be a need to go back and redesign and implement. Furthermore, testing processes of each of these algorithms will be combined, i.e. after each implementation, all of the agents created so far will be tested. Any defects or irrationalities may lead us to return back to our agent's implementations. Hence, Scrum methodology seems to be the most proper method to follow during the development of Poker Playing Agent project. Hüseyin Aleçakır is assigned to be Scrum Master in this project. He will coordinate the members and inform them about the process. Regular sprint plannings, grooming and retrospective meetings will be held, however daily stand-ups are not appropriate for this team since all member already have busy schedules. Product and Sprint backlogs will be created in Trac system for team members to easily track the process and measure their performance.

7 CONCLUSION

This software specification document is prepared to give information about Poker Playing Agent project and its requirements. Interaction between users and the system is discussed. Moreover, system's basic functionalities and which tools and technologies will be used during the development process are described in detail. Team members, their responsibilities on this project and how it is planned to conduct this process is also described in this specification document.

8 APPENDIXES

There is no available appendix for this SRS document.

9 INDEX