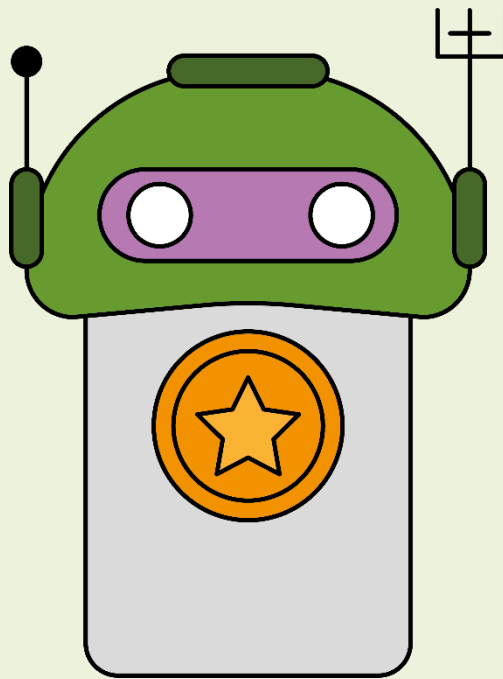


Software engineering project

# BRISCOBOT



Andrea Cecchi - 10491446

Giulia Fasoli - 10536660

## o. Summary

1.	Feasibility study .....	4
1.	Game description .....	4
	Playing cards .....	4
	Gameplay .....	4
	Sum-up .....	5
2.	Game example .....	6
	Pre-game phase .....	6
	Auction phase .....	7
3.	Requirement Analysis and Specification .....	8
1.	Software description .....	8
	Simplification .....	8
2.	Requirements .....	8
3.	Specifications .....	9
	Briscola rules .....	9
	Choosing Bot card .....	9
4.	Design .....	10
1.	Software gameplay and sequence diagram .....	10
	Pre-game .....	10
	Auction .....	10
	Hand .....	11
	Statechart .....	12
2.	Object-Oriented design .....	13
3.	Main scripts .....	13
	AggiornaProbabilità .....	13
	AggiornaRanking .....	14
	AggiornaRankingChiamato .....	14
5.	Coding .....	15
	Framework .....	15
	Code .....	15
6.	Testing .....	17
1.	White Box or Structural Testing .....	17
	Control flow coverage criteria: Path coverage .....	17
2.	Black Box or Functional Testing .....	21
	Testing precondition violations .....	22
	Testing boundary values .....	22
7.	Validation .....	23
1.	Evaluation .....	23

Accessibility ..... 24

## 1. Feasibility study

### 1. Game description

*Briscola a Chiamata*, or *Briscola a Cinque*, is a variant of the classic card game *Briscola* where the number of players is 5. The variant shares with the traditional game the card-taking rules and the values of each card. The game is composed of an arbitrary number of rounds, at the end of each round each player will collect a personal score. The game ends when a player reaches a previously defined score (generally between 10 and 20).

#### Playing cards

The deck of cards has 4 suits (bastoni, coppe, ori, and spade) and 10 cards for each suit (Ace, 2, 3, 4, 5, 6, 7, Jack, Knight, and King). Thus, the deck has 40 cards as shown in Table 1.1.

As summarized in Table 1.2, each card is associated with a value. Ace and 3 are called "Carichi" and are worth 11 and 10 points respectively; King (4 points), Knight (3 points), and Jack (2 points) are called "Figure", while the remaining cards, called "Lisci", have no value. Adding up the scores of the individual cards, the total value of the deck is 120 points.

TABLE 1.1

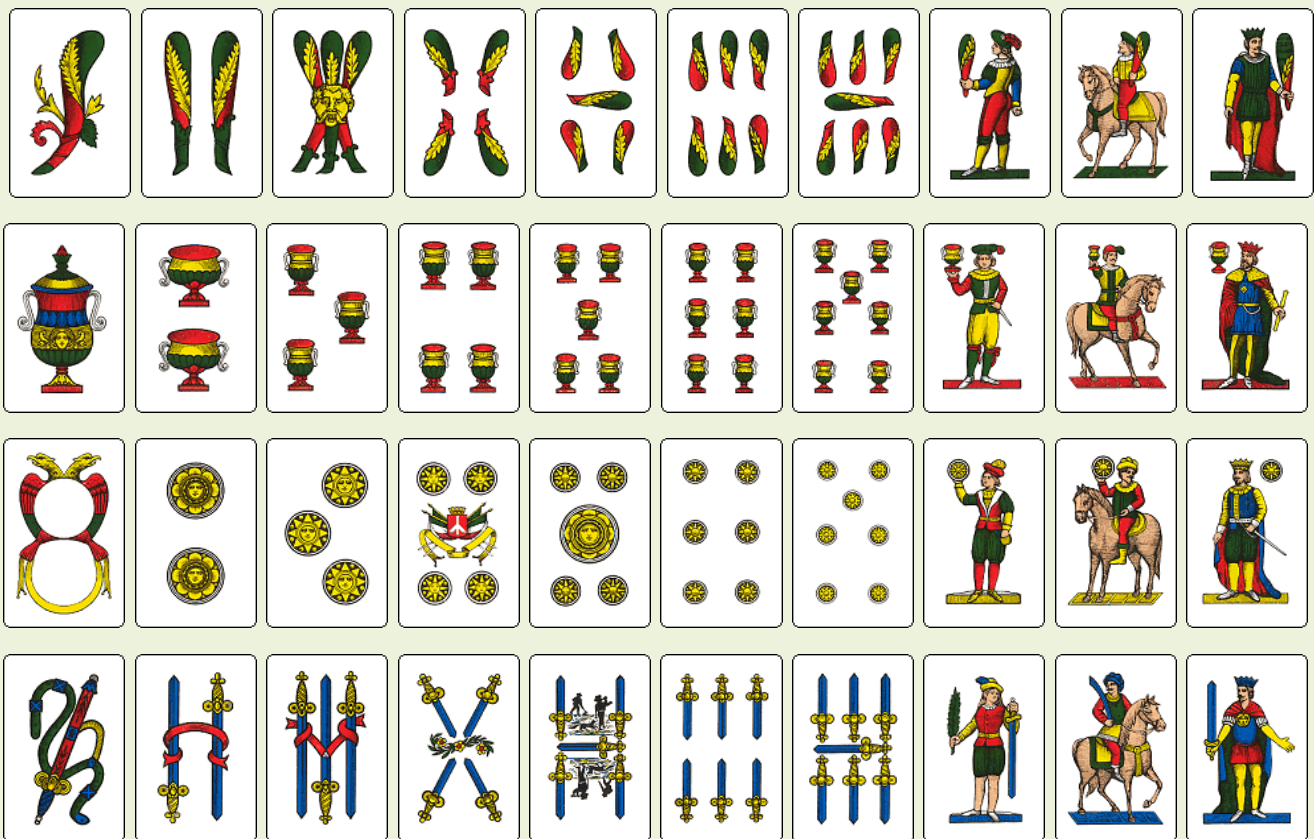


TABLE 1.2

Name	Carichi		Figure			Lisci				
Card	Asso	3	Re	Cavallo	Fante	7	6	5	4	2
Value	11	10	4	3	2	0	0	0	0	0

#### Gameplay

All 40 cards are dealt without initially determining any team: each player has 8 cards and there are no trumps on the table.

#### Auction

The first player calls a card, i.e., he asks for a reinforcement card for his hand specifying its value, but not the suit: by doing so, he commits himself to reach 61 points by playing (for that hand) in partnership with the holder of the

reinforcement card (unknown by the rest of the players), against the other three. The suits of the card declared, define the trump of the game.

It is generally admitted the practice of declaring a card that is possessed, in this case, one player plays against the other four.

There are two main types of auctions. The first one declares a reinforcement card, the second one declares the minimum score needed to win.

### Card-bidding

The player to the right of the dealer starts the bidding and can either declare a card or pass. The next players, up to the dealer, must pass or declare a card if it is lower than the last card declared. The order of the cards is Ace, Three, King, Knight, Jack, Seven, Six, Five, Four, Two. If two or more players remain in the bid, another round is made between all those who did not pass in the previous round and so on until only one remain.

It is allowed to declare the card with the lowest value, not in one's possession, thus making a "forced" call, increasing however the minimum number of points that the couple constituted by the declarer (*Chiamante*) and the partner (*Chiamato*) commit themselves to reach to win if it is higher than the number of points previously declared.

At the end of the bid, the player who declares the lowest card (or, in the case of a 2, the highest value), defines the suit of the card chosen, which becomes trump.

### Point-bidding

The player to the right of the dealer starts the bidding and can either declare a score or pass. The next player, up to the dealer, must pass or declare a score if it is higher than the last. The possible score starts from 62 points up to a maximum of 120 points.

At the end of the bid, the player who won the auction declares the reinforcement card, whose suit becomes trump.

### Hand

The rules of the game are identical to those of *Briscola*.

Each player deals with a card without being obliged to follow the leading suits played by the first player of the trick. The winner is the player who has played the highest value trump, or if no trumps have been played, the player who played the highest card of the leading suit.

The player who wins takes all the cards on the table and places them face down in a pile in front of him. Each player maintains his/her pile.

The player who won the trick leads the next hand playing another of his/hers remaining cards until the end of each players' cards.

### Score

Each player collects tricks and counts points collected. Partners, who are known by the end of the game, combine their points. Game points are assigned as follows:

If the declarer and his partner have won, they score +2 and +1 points respectively, while the others score -1 points. In case of defeat, the caller scores -2, the partner -1, and the opponent +1.

If a player declares one of his/her cards, by winning (or losing) he will gain (or lose) 4 points, while the gain or loss of the four opponents remains +1 or -1 in case of victory or defeat.

In the case of *Cappotto* (i.e., scoring 120 points) the points lost or gained are doubled.

### Sum-up

The game can be summarized in a bulleted list as:

- The role of the dealer is chosen randomly
- The player on the right of the dealer starts the auction and will be the first one to play when the auction is over
- Each player, in turn, calls or passes

- The auction continues among the players who have not passed. Points are called in ascending order or cards are in a descending order
- The auction ends when the last player is left to call, he becomes the *Chiamante* of the game
- The *Chiamante* declares the chosen card, whose suit defines the trump suit
- The player holding the declared card will thus become the *Chiamato* who will be the partner of the *Chiamante*
- The division of the teams is not known to the players
- The game starts from the first player to the right of the dealer
- The winner of each hand is decided according to the rules of the traditional game *Briscola*
  - The first card played defines the leading suit of the hand
  - Any card of the trump suit wins on the leading suits
  - The hierarchy of the cards is defined by their value, in descending order: ace-three-king-knight-jack-seven-six-five-four-two.
- The winner of each hand plays first the next hand until the cards run out (8 hands total, as many as the cards dealt with each player at the beginning of the game)
- At the end of the game, each player sums up the score of the cards he won with those of the players of his team
- The scores are compared, and the winners are declared. To win the team composed of *Chiamante* and *Chiamato* must reach at least the score decided during the auction phase.

## 2. Game example

### Pre-game phase

One of the five-player shuffles the deck (suppose player 5) and deals the cards giving eight cards to each player. In particular, let's suppose that the distribution of the cards is set as shown in Table 1.3.

TABLE 1.3

Player 1								
Player 2								
Player 3								
Player 4								
Player 5								

### Auction phase

Starting from player 1, being player 5 the dealer, the bidding phase began.

#### Card-bidding

	Player 1	Player 2	Player 3	Player 4	Player 5
Round 1	Ace	Skip	Skip	King	Skip
Round 2	Knight			Seven	
Round 3	Skip			<b>Seven of Coppe</b>	

After 3 rounds Player 4 wins the bidding phase and the card requested is the Seven of Coppe, by default, since no one lowered the auction under the 2 of any suits, the winning point is set to 61.

#### Point-bidding

	Player 1	Player 2	Player 3	Player 4	Player 5
Round 1	65	67	70	75	Skip
Round 2	77	Skip	78	80	
Round 3	81		Skip	83	
Round 4	Skip			<b>83</b>	

After 4 rounds Player 4 wins the bidding phase declaring the winning point: 83. The card declared is freely decided by the player, in this case, the king of *Coppe*.

### 3. Requirement Analysis and Specification

#### 1. Software description

Given 4 real players, who want to play *Briscola a Chiamata*, with physical playing cards, BriscoBot will play the role of the fifth player. The software is designed to collect first game information (players' names, positions, and cards), then auction phase information (player name, card, point-to-win), and lastly playing each hand, receiving as input each player's card and showing on screen all the played card (both players and bot).

#### Simplification

The software will play a simplified version of the game, thus some playing possibilities are not considered.

1. The software won't participate in the auction. Since the auction phase is the most entertaining moment, is left only to the physical player. Moreover, the bid is decided based on three different factors: possible winning hand, possible gained points and possible lost points. The progress of the auction defines the player's strategy that is not considered in this software.
2. Variants from the classical game are not considered. There is a variant of the game called "*A carichi*". This declaration interrupts the bidding, and the caller commits to reaching at least 61 points by himself. The hand will be played without trumps: having started the round according to the starting rules, the peculiarity of this variant is that, after the first card of each hand has been played, each player is obliged to play the same suit as the leading suit. A player can play another card only if he/her has no cards of the leading suit.
3. The winner of the auction must declare the reinforcement card. In the classical game, the Chiamante cannot declare the trump suit for the first playing hand.
4. The system does not consider the sum of victory points over multiple games. Each game is independent of the next.

#### 2. Requirements

##### Functional requirements

No.	Description
1	The system must allow string input from a terminal
2	The system must allow the insertion of integers from the terminal
3	The system must allow the selection of cards during the game
4	The system must define its cards according to the cards inserted
5	The system must have an alphanumeric keyboard
6	The system must have a screen
7	No player must be able to see the cards in the system
8	Cards shown by the system must be "Napoletane"
9	The system must follow the rules of the card game " <i>Briscola a Chiamata</i> "
10	A physical player must insert the score and the name of the winner of the bidding phase
11	The system must allow inserting cards from a predefined number of cards
12	The system must allow inserting 8 cards for each player
13	The system must save in memory the score and the name of the <i>Chiamante</i>
14	The system must consider the round as clockwise.
15	The system must play both as " <i>Chiamato</i> " and as " <i>Non-Chiamato</i> "
16	The system must count the winning point at the end of the game
17	The system must define the winning team at the end of the game

##### Non-functional requirements

No.	Description
1	When it is its turn, the system must play a card in 10 seconds
2	The system must have two game modes: " <i>Non chiamato</i> " and " <i>Chiamato</i> "
3	The system must be compatible with Windows
4	The system must have a hierarchy of choices.
5	The system must have a list of decision conditions.
6	The system must not reveal other players' cards
7	The system must not memorize the physical player's cards.



8	The system must save in memory all the cards inserted by the players during the game
9	The system should not choose the BOT playing cards with a random function
10	The system must display the played card as an image on the screen.
11	The system must organize the played card on a game table.
12	The system must display the players' names on the game table
13	The system must display the players' names according to the initial position numbering

### Constraints

No.	Description
1	The system cannot play more than one card
2	There must be 4 physical players
3	Only one computer is needed to play the game
4	Every player must insert the card played in each hand

## 3. Specifications

### Briscola rules

#### Bot cards

To determine the cards of the bot, the system subtracts the 32 cards of the players from the 40 of the deck. To select the cards to subtract, it shows on screen all the cards of the deck and, each player, in turn, selects the cards in his hand.

After selecting and confirming the cards, the system will show the complete deck again to prevent players from discovering each other's cards.

#### Bot game-turn

To make the Bot play when it is its turn, we impose the numbering of the players' positions including the Bot's from 1 to 5 in ascending order from the player to the left of the system, hence the bot during the first hand will always be in the last position.

The numbering of the positions remains fixed during the game, to follow the order of play, another variable considers the turn of the player at each hand. This variable is ordered at each hand.

#### Trumps

The bot does not participate in the auction phase, the increased value of the trump cards is considered only in the choice of cards increasing and decreasing its ranking.

#### Hand conclusion

The winner of the hand is calculated by checking the card on the gaming table. First, the system checks if there are trumps on the table, if they are present, it compares the value of the trumps and assigns the tricks to the higher valued card. If there are no trumps on the table, the system compares the cards of the leading suit and, as before, assign the tricks to the higher figures card of the same suit.

Once the winner is found, the points are assigned to the player, by adding the values of the cards played.

#### Match conclusion

The system marks the player who played the declared card with a flag, at the end of the game the points assigned to the *Chiamante* and the *Chiamato* are added up. The system checks if the total score of the team is greater or equal to the winning point decided during the auction phase.

### Choosing Bot card

To make Briscobot choose the cards to play and thus not have it play randomly, each card the bot has, have a rank that is increased or decreased, comparing a list of condition: the game mode of the Bot (*Non-chiamato* or *Chiamato*), the trump suit, the leading-hand suit, the cards played by other players, the position of the *Chiamato*, the position of the *Chiamante*, and the round of the game.

#### Finding the partner

The last parameter to take into consideration in the decision process is the probability that a player is or is not the partner. To keep track of this probability Briscobot assigns a score to each player, updated at every player's turn. The

value of this probability is set according to a list of conditions that compares the player's card according to the position, in that turn, of the *Chiamante*.

## 4. Design

### 1. Software gameplay

#### Pre-game

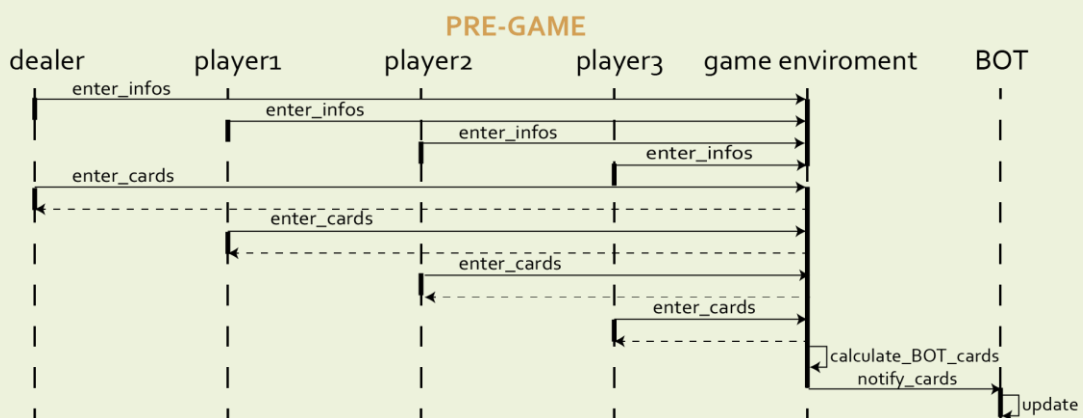
##### Description

The pre-game phase consists of the initialization of players' information (name and position) in a single screen and, after the dealer has shuffled the deck and distributed the cards to the players, of the selection of the 8 cards of each player on a predefined interface. The system notifies the Bot of the cards by subtracting the card selected from the total deck.

##### Bullets

- Enter the names of the players and the position in the table
- Press "conferma"
- Each player in turn selects their cards by clicking on them
- Each player presses "conferma" after checking their card
- The system calculates the BOT's cards by subtracting the 32 cards of the other four players from the total deck

##### Sequence diagram



#### Auction

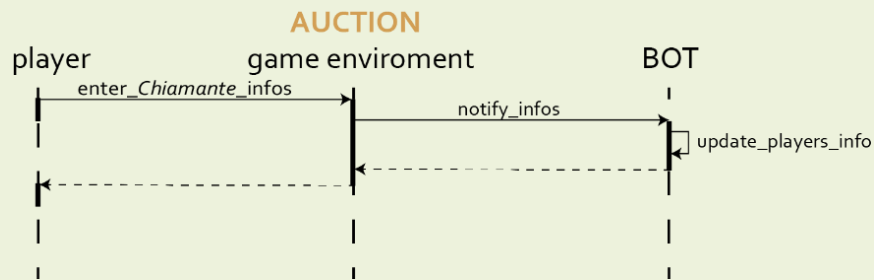
##### Description

The Bot can't be the *Chiamante*, thus it does not participate in the bidding phase. The software waits for the needed information (Chiamante name, declared card, and winning point) on a single screen. Once the information is entered the Bot updates its information about the *Chiamante* player and checks between the given cards the declared one to set game mode (*Chiamato* or not).

##### Bullets

- Enter the *Chiamante* player name
- Insert the score and the declared card
- Update the player information: flag\_chiamante, carta\_chiamata, punteggio\_vittoria
- Checks whether it has the declared card.
- Defines the game mode and sets the probability for each player

## Sequence diagram



## Hand

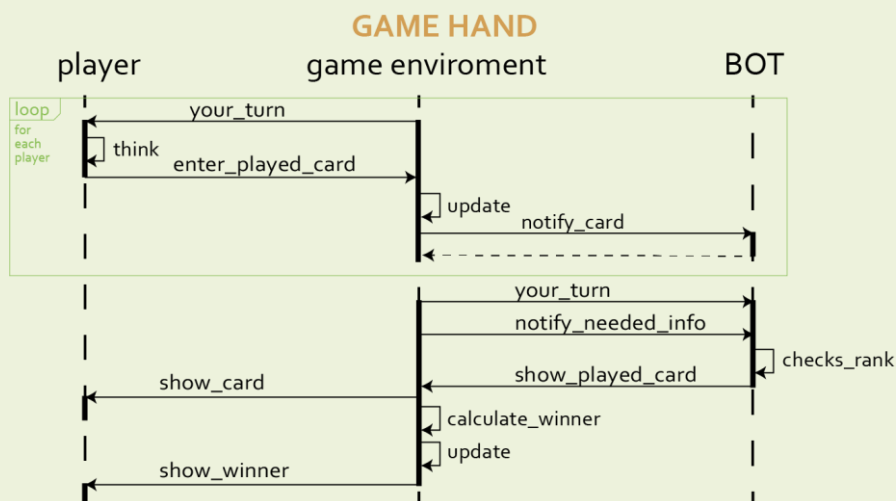
### Description

On each hand, each player, including the bot, plays a card. The decision of the bot's card is taken by the system following a hierarchy of conditions that will be explained in more detail further. At the end of the hand the winner takes all the physical cards on the table, the system also checks the winning card and assigns the player the tricks.

### Bullets

- If it is the system's turn, it plays a card and increases the turn counter
- If it is playerX's turn, the system reads the played card and increases the turn counter
- To check if all the player played their card, the system compares the position of the player with the rearranged array of the turn, if it is in the last position then the hand is concluded.
- To assign the playing hand cards to the winner, the system compares all the cards in the dictionary *carteGiocate* and assign them to the list *prese* of the winner.
- At the end of each round the hand counter is increased by one
- On each hand the system updates the weights of each card and the probability that the single player is the partner.
- When the called card appears on the table, the probability of being a partner is set to 1 for the player who played the card.
- When the hand counter reaches 8 the system stops and adds up the scores of the players and decides the winning team with the relative points.

## Sequence diagram



## Statechart

The four-game phases are linked together to have a better understanding of the overall software procedure as shown in the state chart below.

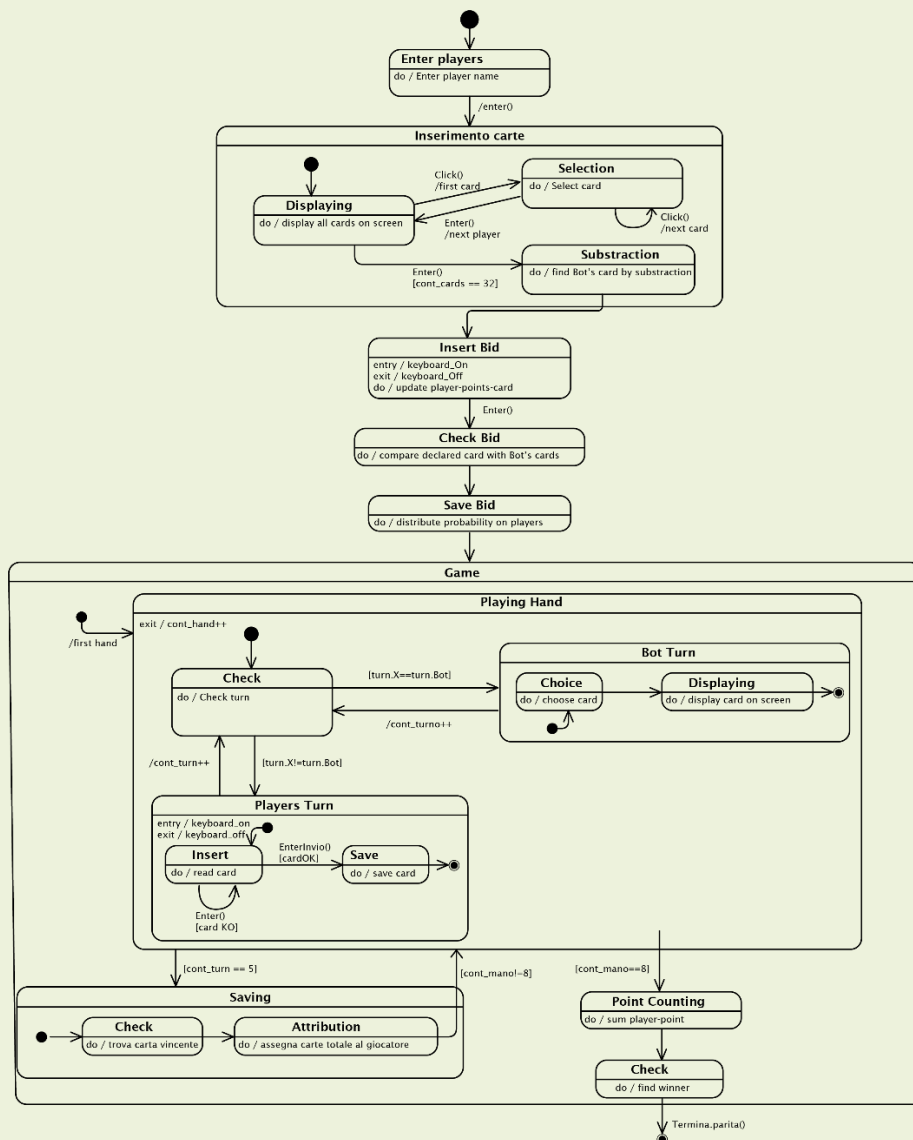
A remark is due to notice. The turn counter can be considered either as an independent variable whose value grows during the game hand from 1 to 5 or as the sorting of a 5-slot array indicating the position of the players changing from hand to hand. To make this concept clearer consider the following example:

Initial table position	
1	Player1
2	Player2
3	Player3
4	Player4
5	Bot

These initial positions are fixed during the game because they show the table arrangement. Let's now assume that the first playing hand is won by Player3. A new array will take into consideration the player's turn by rearranging the positions in a temporary array.

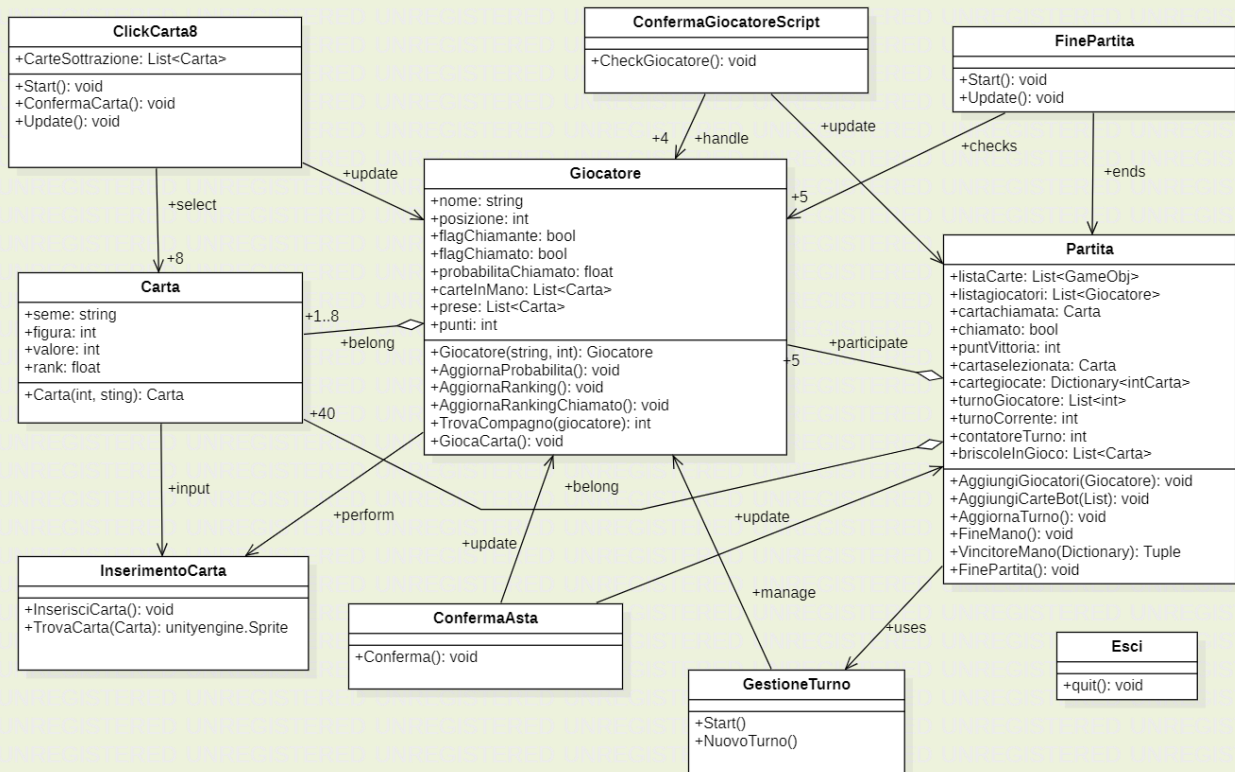
Temp[5] = [3,4,5,1,2]

The hand is concluded when the player's position is in the last slot of the temporary array.



## 2. Object-Oriented design

The architecture is composed of 10 different classes. Only one class is not linked in any way to the others but is an interface class to allow the user to quit the game without using the closing tab. The other classes are linked with methods, mainly to update values then used by the objects. The graphic below shows all the classes of the software with attributes, methods, and connections between them.



## 3. Main scripts

### AggiornaProbabilità

To evaluate the probability that a player is the *Chiamato* a series of parallel ifs with certain conditions have been implemented to update this information at each played card. Each of them has been chosen based on game experience gained over the years. The three conditions that are compared are the card's suit, the card's value, and the player's position concerning the *Chiamante*.

A similar argument can be made for the value that is given to each move, in this case, it is possible also to adjust these values at the end of each game, thus implementing a kind of artificial intelligence.

- The player plays a trump with a value higher than 2 and he is just before the *Chiamante*: -0.3
- The player plays a trump with a value higher than 2 when he is just after the *Chiamante*: +0.1
- The player plays a trump with a value lower than 2 and he is before the *Chiamante*: -0.2
- Player plays a trump with a value lower than 2 and he is after the *Chiamante*: +0.2
- Player is after the *Chiamante*, the caller has played a trump and the player passes it: +0.5
- The player plays a *Carico* and is before the *Chiamante*: +0.3
- The player plays a *Carico*, after the *Chiamante*, who has played a trump lower than 6 -0.1
- The player plays a *Carico*, after the *Chiamante*, who has played a trump higher than 6 +0.3
- The player puts the highest card on the table and he's after *Chiamante*: +0.2
- The player puts the highest card on the table and he's before *Chiamante*: -0.1
- The player is last in hand, *Chiamante* is winning, the player plays a card higher than 2: +0.2
- The player is last in hand, *Chiamante* is winning, the player plays a card higher than 10: +0.5

"if" are parallel between them and autonomous, the entry in one of the previous if does not preclude the entry in one of the following ones. However, some conditions are opposite, so if you enter one you cannot enter the opposite one.

## AggiornaRanking

AggiornaRanking allows the system to choose the card to play when the bot is not the *Chiamato*. Each card of the Bot has an assigned ranking. When it is the Bot's turn, the system uses the data updated up to that point to decide. The card's rank is updated once every hand when it's the bot's turn. A series of "ifs" determine the rank of each card and the system plays the card with the highest rank, when multiple cards have the same rank then the card played is the one with a lower value.

The first discriminating factor is the position of the *Chiamante*, the only player that the bot is sure is an opponent.

The key positions are when he is first or last, i.e., those that most influence the card to be played. If the *Chiamante* is first, he has no control over what the others player will play, so it is a disadvantageous position that must be exploited. Whereas if he is last, he has a position of dominion over the rest of the player, his action can change according to the overall table situation.

The second factor to consider is the Bot's position. The same reasoning made before regarding the *Chiamante* position is used also for the bot's position.

It is crucial to place the *Chiamante* in the harder situation possible each hand, e.g., by trying to collect tricks and put him first-player in the next hand.

The last factor takes into consideration the cards present on the table. It has also been added the possibility to understand who is more likely to be the *Chiamato* among the opponents, through the UpdateProbability function. This value is weighted based on previous game experience and has been adjusted during the testing phase.



## AggiornaRankingChiamato

AggiornaRankingChiamato allows the system to choose the card to play when the bot is the *Chiamato*. The method's logic is the same as the AggiornaRanking: a series of "ifs" define the Bot's card ranking and then the card with higher rank is played and, in the case of equal ranking, the card played is the one with a lower value.

The Bot, precisely knowing the opponents and its partner, will try to help the *Chiamante* by giving him points or trying to let him be the last player in the next hands. Playing the role of *Chiamato* is harder than the other roles due to the need to not let the other players understand too easily the partnership with the *Chiamante*.

To implement this game strategy the first condition to check is the turn counter with the idea to not let the action of the Bot define its role too early during the game.



## 5. Coding Framework

To be implemented, *Briscola a Chiamata* needed a graphics engine, there are various but the most common are Unreal Engine and Unity, as regards the choice between the two was indifferent as *Briscola a Chiamata* is a simple game for such a framework, many video games now with graphics much more complex use these engines. The choice of one of the two would not have influenced or bound the game in any way.

Unity was the framework chosen for the development of *Briscola*. It is a cross-platform graphics engine that enables the development of video games and other interactive content, such as architectural visualizations or real-time 3D animations.

The choice of C# programming language was consequential to the choice of unity and in general, games are usually created with object-oriented languages. C# and Javascript are the most used and with richer documentation. However, C# is closer to C, a language that we knew better than Javascript, and, not having constraints or limitations, we decided to implement C#.

The last thought has been made also about python because to implement the "intelligence" part of the bot it would have been very interesting to introduce some machine learning to update the weights that have been inserted in the if trees. Future development could include a part of external code that deals only with machine learning and is implemented in python, a language very suitable for data analysis and artificial intelligence. The base written in C# would remain in any case, to exploit Unity.

As ide is used instead of Visual Studio, which has a section entirely dedicated to unity and allows you to connect to debugging and other operations.

The development environment has not, therefore, placed limits in the implementation of the software just because they have used tools with possibilities much greater than the use made

## Code

The code has been fully commented on and indented to increase readability and try to make it understandable at first glance.

It has been thought however to deepen the methods summarizing of the main ones with a small description and a mapping to words that explains the connections between the methods and like they come recalled.

## Unity

In the unity part, the five main scenes were created, which the various classes then refer to. Each scene corresponds to a different screen and represents the graphic interface of the game.

There are five scenes:

- MenuGiocatore: first scene of the game, there are four GameObject with different tags that pass the names of the players to c#
- ScegliCarta: in this scene each player chooses the cards by clicking on them, the images of the cards are stored in the Assets folder, each displayed card has a tag that refers to suit and number, each time you want to create a card you take in input the string contained in the tag and you give it in input to the constructor of the class Card
- Asta: it works exactly like the first scene with the addition of a drop-down menu
- TurnoGioco: in this scene in addition to the inputs are shown the images of the cards played, this operation is done through the instantiate function that receives the coordinates of where you want to place the card and a sprite, ie the image, and creates a GameObject on the screen in the desired position. For the movement of the cards, a force vector is applied to the physical component of the cards via the AddForce function.
- FinePartita: This scene prints who won, receiving it from the FinePartita function, which calculates the score and finds the winner.

## C#

The C# code is commented, every method is explained in the code but, for greater readability, it was decided to list all the function calls in this order: class – method – method calls.

### ConfermaAsta

- Conferma
  - Loads scene TurnoGioco

### Partita

- AggiungiGiocatori
- AggiungiCarteBot
- AggiornaTurno
  - Calls methods:
    - AggiornaRankingChiamato
    - AggiornaRanking
    - GiocaCarta
- FineMano
  - Call methods:
    - AggiornaRankingChiamato
    - AggiornaRanking
    - GiocaCarta
    - FinePartita
- VincitoreMano
- FinePartita
  - Loads scene FinePartita

### Esci

- Application Quit

### Giocatore

- AggiornaProbabilità
- AggiornaRanking
- AggiornaRankingChiamato



- TrovaCompagno
- GiocaCarta

#### CarteSottrazione

- Add
- Remove
- Count

#### InserimentoCarta

- Start
- InserisciCarta
  - Calls methods:
    - AggiornaProbabilità
    - AggiornaTurno
- TrovaCarta

#### Carta

- Carta

#### ConfermaGiocatoreScript

- Loads scene ScegliCarta

#### Clickcarta8

- Start
- ConfermaCarte
  - Calls method:
    - AggiungiCarteBot
    - Loads scene Asta
- Update

#### FinePartita

- Start
- Update

## 6. Testing

The testing phase is crucial during the development of software. The importance to isolate and correct bugs in a program is vital to assure that the final behaviour coincides with the required one.

### 1. White Box or Structural Testing

The white test will focus on the decision-making part of the bot, it will check that all the if-tree is traversed under the right conditions and lead to the expected result without errors.

White testing starts with testing the script `AggiornaProbabilità`. This script is encapsulated in another fundamental method `AggiornaRanking`, because of this layered structure the first script to test is `AggiornaProbabilità`.

#### Control flow coverage criteria: Path coverage

Path coverage is used to test the most important and delicate part of the software: the scripts that guide the bot's choices. Path coverage was preferred as it is the best method of white testing, and it allows to inspect every part of the code.

Since all three scripts have a tree structure, all branches were inspected thanks to a correct sequence of cards.

In addition, the first branches depend on the positioning of the players playing the cards and the number of turns in which the play takes place, so the tests were repeated for a total of 30 different games.

The decision-making part is composed of three different scripts:

- AggiornaProbabilità
- AggiornaRanking
- AggiornaRankingChiamato

Importance was also given to the repeatability of the tests, so each branch was tested at least three times with different cards taken from the same test set.

The first step is to select a test set T which traverses all paths from the initial to the final node of P's control flow.

#### AggiornaProbabilità

AggiornaProbabilità script is composed of twelve "if" with three conditions each one, to test it, all the conditions were satisfied and all the "if" were inspected.

The test sets are divided first concerning the position of the player who plays the card, then by the value of the card and its suit, finally by the number of the turn and the probability that the player who played it is the partner.

#### Test sets 1° IF:

The IF must activate only when a player, that is in a position just before the *Chiamante*, plays a Briscola card with a value higher than 2.

It was tested putting the player in the third position, with the *Chiamante* in position number four. Cards played were 1 *Coppe*, 5 *Spade*, 6 *Bastoni*, 9 *Ori*, and, correctly, only 1 *Coppe* and 9 *Ori* activated it.

#### Test set 2° IF:

The IF must activate only when a player that is in a position just after the *Chiamante* plays a Briscola card with a value higher than 2.

It was tested putting the player in the fifth position, with the *Chiamante* in position number four. Cards used were: 3 *Spade*, 6 *Coppe*, 8 *Bastoni*, 2 *Ori*, correctly, only 3 *spade* and 8 *bastoni* activated it.

#### Test set 3° IF:

The IF must activate only when a player that is in a position before the *Chiamante* plays a Briscola card with a value lower than 2.

It was tested putting the player in the third position, with the *Chiamante* in position number four. Cards used were: 1 *Coppe*, 5 *Spade*, 6 *Bastoni*, 9 *Ori*, correctly, only 5 *Spade* and 6 *Bastoni* activated it.

#### Test set 4° IF:

The IF must activate only when a player that is in a position after the *Chiamante* plays a Briscola card with a value lower than 2.

It was tested putting the player in the third position, with the *Chiamante* in position number two. Cards used were: 2 *Coppe*, 10 *Spade*, 6 *Bastoni*, 1 *Ori*, correctly, only 2 *Coppe* and 6 *Bastoni* activated it.

#### Test set 5° IF:

The IF must activate only when a player that is in a position after the *Chiamante* that plays a card with a value lower than the one played by the *Chiamante*.

It was tested putting the player in the fifth position, with the *Chiamante* in position number four. Cards used were: 1 *Coppe*, 5 *Spade*, 6 *Bastoni*, 9 *Ori*. correctly, only cards higher than the others activated it.

#### Test set 6° IF:

The IF must activate only when a player that is in a position before the *Chiamante* plays a card with a value higher than 10.

It was tested putting the player in the fifth position, with the *Chiamante* in position number four. Cards used were: 4 *Coppe*, 8 *Spade*, 2 *Bastoni*, 3 *Ori*, correctly, only 3 *Ori* activated it.

#### Test set 7° IF:

The IF must activate only when a player that is in a position after the *Chiamante* that plays a Briscola  $\leq 6$  and the player plays a card with value  $\geq 10$

It was tested, putting the player in the second position, with the *Chiamante* in position number one. Cards used were: 6 *Coppe*, 1 *Spade*, 4 *Bastoni*, 3 *Ori*, correctly, only 3 *Ori* and 1 *Spade* activated it.

#### Test set 8° IF:

The IF must activate only when a player that is in a position after the *Chiamante* that plays a Briscola  $\geq 6$  and the player plays a card with value  $\geq 10$

It was tested putting the player in the third position, with the *Chiamante* in position number one. Cards used were: 10 *Coppe*, 1 *Spade*, 3 *Bastoni*, 4 *Ori*, correctly, only 3 *Ori* and 3 *Bastoni* activated it.

#### Test set 9° IF:

The IF must activate only when a player that is in a position after the *Chiamante* and plays the highest card in general

It was tested putting the player in the fifth position, with the *Chiamante* in position number two. The cards used were: 8 *Coppe*, 10 *Spade*, 2 *Bastoni*, 1 *Ori*, correctly, only the highest cards activated it.

#### Test set 10° IF:

The IF must activate only when a player that is in a position before the *Chiamante* and plays the highest card in general

It was tested putting the player in the second position, with the *Chiamante* in position number four. The cards used were: 10 *Coppe*, 6 *Spade*, 1 *Bastoni*, 3 *Ori*, correctly, only the highest cards activated it.

#### Test set 11° IF:

The IF must activate only when a player, last in hand, that puts a card with a value higher than 2 when the "*Chiamante*" is winning

It was tested putting the player in the fifth position, with the *Chiamante* in position number four. Cards used were: 2 *Coppe*, 8 *Spade*, 3 *Bastoni*, 10 *Ori*, correctly, only 10 *Ori*, 8 *Spade*, and 3 *Bastoni* activated it.

#### Test set 12° IF:

The IF must activate only when a player, last in hand, that puts a card with a value higher than 10 when the "*Chiamante*" is winning

It was tested putting the player in the fifth position, with the *Chiamante* in position number two. Cards used were: 1 *Coppe*, 7 *Spade*, 10 *Bastoni*, 3 *Ori*, correctly, only 3 *Ori* and 1 *Coppe* activated it.

### AggiornaRanking

Since it is possible to run each track only when certain conditions occur, each test set is divided into three macro-categories depending on the position of the *Chiamante*. The first three sets are: T1, T2, T3: T1 encloses all test sets with the *Chiamante* in the first position, T2 contains all tests with the *Chiamante* between position 2 and position 4, finally T3 contains all test sets with the *Chiamante*'s position equal to 5.

At this point a second division of the tests has been made according to the position of the Bot:

- the test set T1 is divided into T1.1 and T1.2, T1.1 for the bot in the second position and T1.2 for the bot in the fifth position
- the test set T2 is divided into T2.1, T2.2, and T2.3, T2.1 for the bot in the first position, T2.2 for the Bot between the second and the fourth position, and T2.3 for the bot in the fifth position
- the test set T3 is divided into T3.1, T3.2, and T3.3, T3.1 for the bot in the first position, T3.2 for the Bot between the second and the third position, and T3.3 for the bot in the fourth position

Now the test sets are composed only by players' cards and all branches can be run without error and with correct outputs. For formal simplicity, only an example case is reported, highlighted in Figure 7.1.

Let's suppose that Player2 won the bidding phase, and he declared the Knight of *Spade* as the winning card. Let's now suppose that the previous hand was won by Player1. Then a set of cards as shown in Table 6.1 triggers all the conditions considered in the highlighted branch.

TABLE 6.1


Playe1	Player2	Player3	Player4	Bot
				



FIGURE 6.1

### AggiornaRankingChiamato

As for AggiornaRanking, the testing of the method can be done systemically. All the branches are followed depending on the occurrence of a different condition.

First, the game turn is considered. The possible sets are T1 when the game turn is less or equal to 3, and T2 when the game turn is greater than 3.

The two main sets differ in the second partition. Set T2 has an additional condition regarding the position of the *Chiamante* while set T1 considers just a particular case of the Bot's position.

Set T1 is partitioned into three subsets: T1.1, T1.2, and T1.3 depending on the sum of the values of the cards on the game table.

Set T2 is more complex, a further condition divides the possible situations concerning the *Chiamante* position defining sets T2.1, T2.2, and T2.3. In particular:

- T2.1 considers the case in which the *Chiamante* is neither first-player nor last-player of the playing hand
- T2.2 considers the case in which the *Chiamante* is the first one who plays
- T2.3 considers the last remaining case in which the *Chiamante* is the last player

For each subset, another partition is taken into consideration: Bot's position. As for the *Chiamante* also the Bot can play as first-player, fifth-player, or first or fifth player, defining the sub-subsets: T2.1.1, T2.1.2, T2.1.3, T2.2.1, etc

The final parts of the branches are less homogeneous, the conditions consider the cards present on the game table (values and suit), the card played by the *Chiamante*, and in some cases the game turn.

It is possible to find a set of cards that can trigger each branch's conditions. For formal simplicity, only an example case is reported, highlighted in Figure 6.2.

Let's suppose that Player<sub>1</sub> wins the auction phase declaring the King of *Ori* and a winning score of 61 points. Let's further suppose that four hands were already played, then the set of cards represented in Table 6.2 perfectly triggers the designated branch.

TABLE 6.2



Player1	Player2	Player3	Player4	Bot
				



Figure 6.2

## 2. Black Box or Functional Testing

Black box testing is a testing methodology that assumes an external point of view concerning the code: no information is considered concerning the internal functioning of the application. Contrary to what happens with the White box, the focus is on the result and not on the processes that lead to it.

Another difference is the criteria for the partitioning of the software, Black Box criteria are based on the module's specification.

The modules identified are:

1. Players name input
2. Input Players cards
3. Auction data input
4. Game phase
  - a. Card input

- b. Card played
- 5. End of the game

### Testing precondition violations

To test the robustness of the program in presence of unexpected inputs, some tests are developed to violate the preconditions set. Every scene has its tests

- Players name input
  - Input test set: one or more empty fields
- Input Players cards
  - Input test set: choose less than eight cards
  - Input test set: choose more than eight cards
  - Input test set: choose two times the same card with different players
- Auction data input:
  - Input test set: one or more empty fields
  - Input test set: in "*Inserire punteggio*" put a value greater than 120 and less than 61
  - Input test set: "*Chiamante*" name does not exist
  - Input test set: in "*Figura Carta chiamata*" put a value higher than 10 and lower than 1
- Game phase
  - Input test set: in "*Inserimento Carta*" put a value higher than 10 and lower than 1
  - Input test set: in "*Inserimento Carta*" put a letter or a symbol
  - Input test set: "*Inserimento Carta*" empty
- End of the game
  - Input test set: the total sum of the points more than "*punteggioChiamato*" to see if the victory is awarded to the right team
  - Input test set: the total sum of the points less than "*punteggioChiamato*" to see if the victory is awarded to the right team
  - Input test set: random cards played by players during the Game Phase, to control that every card has its correct value, if the sum is equal to the correct one the test is passed, if not, the test is not passed

### Testing boundary values

These tests test the boundary of the interval when input can belong to the interval

- Players name input
  - Input test set: Two or more equal names
  - Input test set: player name = BOT
- Input Players cards
- Auction data input:
  - Input test set: in "*Inserire punteggio*" put a value equal to 120 and equal to 61
  - Input test set: in "*Figura Carta chiamata*" put a value equal to 10 and equal to 1
- Game phase
  - Input test set: in "*Inserimento Carta*" put a value equal to 10 and equal to 1
- End of the game
  - Input test set: the total sum of the points equal to "*punteggioChiamato*" to see if the victory is awarded to the right team

All the tests have been done at least five times and all have been passed correctly without errors, there is only one problem that has not been remedied, the only solution implemented is to ask for further confirmation of the card insertion, i.e., if a player makes a mistake inserting another existing card during the playing hand, the system does not notice that the card is wrong, because it cannot know if it is or not, being anyway the card valid.

Future development could implement a script that permits modification of the input of a wrong card.

After the black-box tests, the whole software has been tested, both the intelligence part of the bot and all the inputs and outputs

## 7. Validation

Software validation focused on the Bot's ability to play, and the verification that the software's behaviour matches the required one.

Due to the lack of possibilities derived from the pandemic situation during the time in which the validation phase should have been carried out, this section will collect the feedback from the evaluation that was made, the evaluation test we would have loved to do, and the possible upgrades to the software.

### 1. Evaluation

The evaluation process is significant during the software creation as a further chance of improvement.

The evaluation of the card's played by the Bot is based on assigning a score to each play according to a scale ranging from 1 to 5, the higher the score, the more the play can be considered correct.

To do this, 10 games were played with a total of 8 different people to collect as many opinions as possible.

The level of the chosen players is comparable, always remembering that the bot must play as a mid-level player, trying to make the right plays without ruining the game to other players.

As a second method of evaluation, we took the ranking and the total score made by the bot after the 10 games and it has always been in line with the other players, indeed, the bot has never arrived last.

This result is affected by the fact that, in long games with players of the same level, the one who never wins the bidding phase is also the one that more likely won't arrive last.

Just before each game was given to the players, with the explanation of each vote, to have the fairest possible evaluation, each evaluation was done a posteriori, knowing the Bot's role (*Chiamato* or not). Just before each game Table 7.1 was given to the players, with the explanation of each vote, to have the fairest possible evaluation, each evaluation was done a posteriori, knowing the Bot's role during the game (*Chiamato* or not).

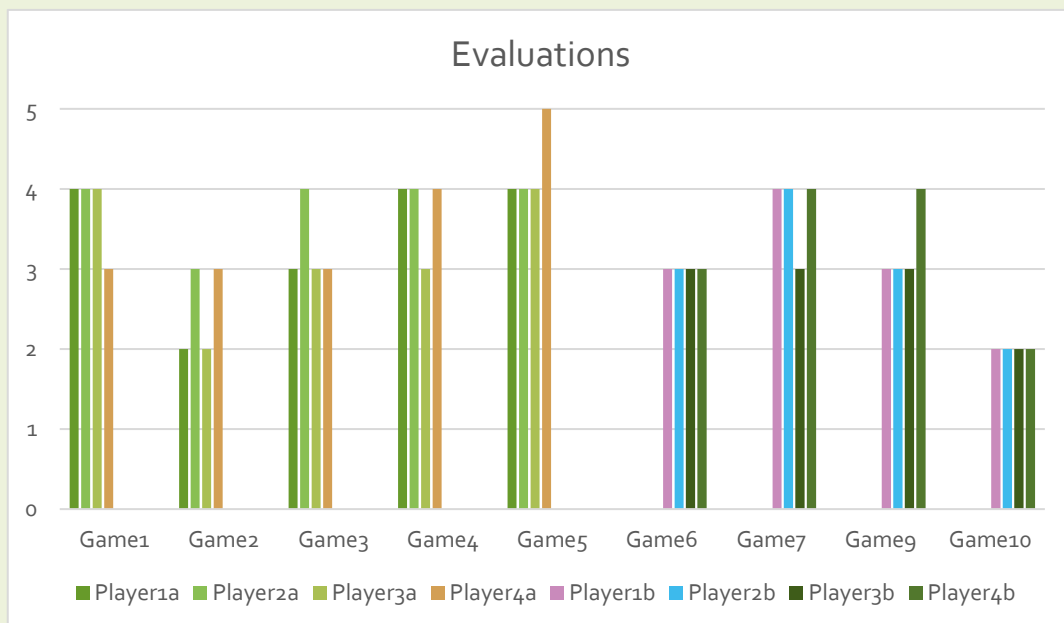


FIGURE 7.1

TABLE 7.1

Mark	Explanation
1	Played completely wrong
2	Bad play but not overly damaging
3	Wrong play but does not harm the game
4	Played well but could play better
5	Perfect play

All the scores, as reported in Figure 7.1, were collected, then the average was calculated, and a vote was given to the overall game. The average grade at the end of the 10 games was 3.3. This value could be improved but can be considered as a beginner player.

After this analysis, it is possible to say that Briscobot satisfies the requirements.

### Accessibility

Being *Briscola a Chiamata* a card game played mostly by elder people, to complete the evaluation work, it would have been important to have an elder group try it out to see if they could play it without problems.

One of the most critical situations could be how long the cards stay on the ground before they disappear when the bot plays a card, the time given to the players should be evaluated by them as well. The size of the cards and lettering should also be evaluated to see if they are big enough or not.