REQUIREMENT ANALYSIS

**Vision**

We envision a board game called Monopoly. This is the third iteration. In our previous iterations we aimed the user to choose number of players between 2-8, play a toss tournament to decide with whom to start the game then play the game until the predefined number of turns are played. Special squares were not defined.

However, in this iteration, we defined special squares and their actions such as “Lots”, ”Railroads”, ”Utility” which can be bought by a player if he/she lands on them. In addition a player must pay rent if he/she lands on a square that doesnt belong to him/her. There are still Jail, GoToJail, IncomeTax and LuxuryTax squares. Players may buy a property randomly depending on the dice value in this iteration.

The game ends when all other players have gone bankrupt and there’s only one player left.

**Problem Statement**

For the first iteration the main goal was to first decide who the game would start with by starting with a dice tournament and then moving on the board.

For the second iteration the issues were, stating the special squares and their actions, controlling the players' financial statement (if a player went bankrupt or not).

For the last iteration in addition to the previous iterations, the goals are to read the input file and organize the squares according to the file automatically, do the special actions on the squares and to finish the game when all of the players except one have gone bankrupt.

To be able to do this we used some methods of object oriented programming, abstract programming.

**Scope**

The Monopoly Game is able to have some special squares, if a player is on a special square he or she has to do the action that it states. Also the game is able to take input from the user about how many players there are going to be and what their names are. In addition deciding on with whom the game should start by doing a dice tournament is possible. User can see on which square the players are after every time the die rolls. Some squares have the ability to get tax from the users. Some special squares can be bought depending on the dice value of the player. If a user buys a property square, a player that lands on it must pay rent unless he/she is the actual owner of the square.Players must pay rent to Utility and RailRoad squares as well. If a user has less money than the tax he or she has to give, he or she goes bankrupt. If all of the users except one goes bankrupt the game ends.

**Stakeholders**

Şükrü Gümüştaş (Programmer)

Atakan Ülgen (Analyst/Programmer)

Hilal Balcı (Scrum Master/Programmer)

Murat Can Ganiz (Customer)

Berna Altınel(Customer)

**Glossary of Terms( Alphabetically)**

Board: Dashboard where the game is played on and which contains squares.

ElectricUtility: A utility square where a player pays the owner 10 times the roll of the dice.

Free Parking: The square where nothing happens, the player waits.

Go Square: Starting point where the users increase their money by 200 Turkish Liras.

Go to Jail Square: The square which sends the player to the Jail Square if they are on it.

Income Tax : The square where players lose 10 percent of their money if they are on it.

Jail Square: The square where the player waits for three turns if they are on it.

Lot Square: The square where the player pays rent to the owner.

Luxury Tax: The square where players lose 75 Turkish Liras if they are on it.

Printer:Prints out the inputs.

PropertySquare: Lot, Utility and RailRoad Squares.

RailRoad: A square where the player pays the owner 5 times the roll of the dice plus 25.

Square: The location where the player is on.

Toss: To roll the dice.

Toss Tournament: A small game where dice are rolled and the player who has the biggest number starts the game first.

WaterUtility: A utility square where a player pays the owner 10 times the roll fo the dice.

**Use Cases**

1. User enters integer number between 2-8

2. User enters the names of the players

3. Toss Tournament begins

4. Players are sorted according to the number on their dice

5. The program asks for the name of the input file

6. Board and the squares are created according to the file.

7. Game starts.

8. Players move according to the number on their dice.

9. Game is played and finished when everyone goes bankrupt and there’s only one player left.

10. Everything is printed on the screen and to an output file.