Ellis Levine
Amin Ortiz
Marshall Lang

Requirements Analysis for Project Assignment 2

Problem statement: the application should implement the basic rules of tic-tac-toe using a simple GUI and handle common eventualities, such as a player taking too long during their turn.

Requirements:

- (Input) On start up, the application should display to the user the following options:
  (optional) 1. Single player match (play with the computer).
  2. Two player match (play with another person).
  3. Quit (exit application).
- (optional) If the user selects the single player game option, a match begins against a computer opponent. Else if the user selects two player game, a match between two players begins. Else if the user selects quit, the application ends.
- (Output) Once a match chooses one of the options (single*, two player), the GUI should display the board.
- (Output) A box should display the current player's turn.
- (Output) A box should display the timer.
- (Input) On the current player's turn, if the player selects an empty space on the board, it should be filled with that player's symbol and the next player's turn should begin.
- (Output) If one of the players reaches the victory condition (they fill a row, column, or the diagonal with their symbols), the game should display a message that says that player won. Else if the board is filled up and neither player has won, the application should display a message that a draw has been reached. Else if the timer runs out on a player, the game ends with the player whose timer did not run out being declared the winner.

Introduction: The domain is a Tic-Tac-Toe game that we are preparing to learn more about software engineering principles, specifically those of domain analysis and use case analysis. The goal is to implement a simple Tic-Tac-Toe game between two players using a simple Graphical User Interface (GUI) that implements the basic rules of the game.

Glossary: the game is very simple, so no complex terms should be needed.

General knowledge about the domain: The basic rules for a game of tic-tac-toe are as follows:

- The game is played between two players: X and O.
- The game is played on a 3x3 board.
- The players take turns placing a single X or O (depending on whether they are player X or O, respectively).
- X always moves first.
- Whoever gets three X's or O's in a row wins.
- If all nine squares on the board are filled without either player managing to get three in a row, the game ends in a draw.

Customers and users: the only users will be the people who worked on the project (Amin Ortiz, Ellis Levine, Marshall) and the professor and grader.

The environment: the application will run on a personal computer.

Tasks and procedures currently performed: to be determined. The project document includes a reference to a timer feature being used, perhaps specifically in the case that one player does not take their move in a reasonable amount of time.

Problem statement: the application should implement the basic rules of tic-tac-toe using a simple GUI and handle common eventualities, such as a player taking too long during their turn.

Requirements:

Functional requirements:

- (Input) On start up, the application should display to the user the following options:
  (optional) 1. Single player match (play with the computer).
  2. Two player match (play with another person).
  3. Quit (exit application).
- (optional) If the user selects the single player game option, a match begins against a computer opponent. Else if the user selects two player game, a match between two players begins. Else if the user selects quit, the application ends.
- (Output) Once a match chooses one of the options (single*, two player), the GUI should display the board.
- (Output) A box should display the current player's turn.
- (Output) A box should display the timer.
- (Input) On the current player's turn, if the player selects an empty space on the board, it should be filled with that player's symbol and the next player's turn should begin.
- (Output) If one of the players reaches the victory condition (they fill a row, column, or the diagonal with their symbols), the game should display a message that says that

player won. Else if the board is filled up and neither player has won, the application should display a message that a draw has been reached. Else if the timer runs out on a player, the game ends with the player whose timer did not run out being declared the winner.

Use case: Place a mark on the tic-tac-toe board.

Actors: players, tic-tac-toe system

Goal: to allow one of the basic actions of the game of tic-tac-toe to be played

Preconditions: a match of tic-tac-toe must have started and it must be the actor's turn.

Summary: When a user wants to place mark on the board, a match of tic-tac-toe must be in progress and it must be the user's turn. They then select one of the empty cells on the board, after which their symbol (either X or O) is placed on the cell and the system moves the turn order forward.

Steps:

Actor Actions	*System Responses*
1. Select a cell on the board during their turn	2. Place the user's symbol on the cell, and then moves to the next player's turn.

Postconditions: the board remembers the user's selection by placing a mark in the indicated cell.

Use case: Time out an inactive player.

Actors: players

Goal: Prevent a game from lasting forever by limiting the amount of time alloted to each player's turn.

Preconditions: a match of tic-tac-toe must have started and the player's current turn must have longer than the amount of time alloted to their turn timer.

Summary: If a turn lasts more than the alloted time on the timer, the tic-tac-toe system ends the game and declares the winner to be the player whose turn did not time out.

Steps:

Actor Actions	*System Responses*
1. During their turn, does not select a valid move within the alloted time limit	2. Declares the player whose turn did not time out the winner and ends the match.

Postconditions: the match of tic-tac-toe ends and one player is declared the winner.
Use case: start a match of tic-tac-toe

Actors: player

Goal: to start a match of tic-tac-toe

Preconditions: system must not be in the middle of a tic-tac-toe match.

Summary: when the user wants to start a match of tic-tac-toe, they select this option. The system then initializes an empty board and displays it to the user.

Steps:

Actor Actions	*System Responses*
1. Selects the start a match option	2. Begins a match of tic-tac-toe by initializing a board and then displaying it to the user.

Postconditions: the game board is initialized and displayed to the user.

Use case: restart the board

Actors: player

Goal: allow another match of tic-tac-toe to begin

Preconditions: either a match must be in progress or have ended.

Summary: this option allows the user to reset the board at any point after a game has begun or ended.

Steps:

Actor Actions	*System Responses*
1. Selects the reset option	2. Begins another match of tic-tac-toe by initializing an empty board and then displaying it to the user