Amin Ortiz
Ellis Levine
Marshall Lang

Use Case Analysis for Project 3

Domain analysis:

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: the application currently supports a basic implementation of the tic-tac-toe game for two human players on a 3x3 board. Interaction with the system is performed through a simple GUI.

Problem statement: to augment the tic-tac-toe system developed in version 1 with the following functionality:

1. Allow the user to set the size of the board (m rows by n columns).
2. Allow the user to change the number of pieces in a row required to determine a victory.
3. Adding a computer player with basic skill.

Requirements:

- (Input) Before starting a match, the system will allow the user to choose whether they would like to play with a human player or with a computer opponent of basic skill.
- (Input) After choosing what sort of opponent they would like to play against, the system will allow the user to determine the size of the board by entering two positive integers, m and n, that will determine the row size (m) and the column size (n) of the board, as well as allow the user to enter another positive integer (k) that will determine the number of pieces in a row necessary to obtain a victory.

- (Output) When a match starts, the system will display a board with m rows and n columns, as determined by the user's choice of m and n.
- (Output) The first player to get k pieces in a row, horizontally, vertically, or diagonally, will be declared the winner.

Updated Use Case Models

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

Actors: Player, ai player

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 an actor wants to place a mark on the board, a match of tic-tac-toe must be in progress and it must be the actor'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 symbol on the cell, and then moves to the next turn

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

Use case: Time out an inactive actor.

Actors: player, ai player

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

Preconditions: a match of tic-tac-toe must have started and the actor's current turn must have lasted 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

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 the kind of opponent that they would like to play against (human or basic computer). The system then queries the user for the size of the board (m, n) and the number of pieces in a row to determine a victory

Steps:

Actor Actions	*System Responses*
1. Choose opponent	2. Display 'board options' dialog
3. Specify board options (m, n, k)	
4. Confirm selection	5. Remove 'board options' dialog
	6. Initialize and display custom board

Postconditions: the game board is initialized according to the user's preferences and displayed to the user

Use case: start a match with invalid values

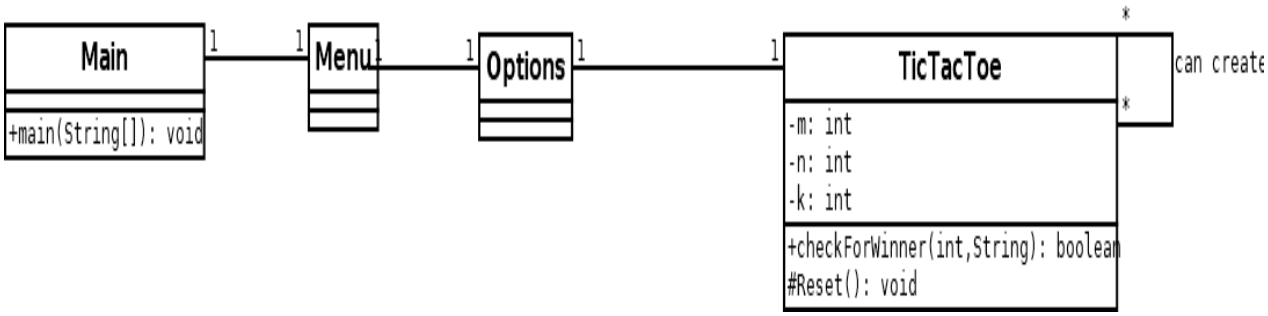
Related use cases:

Extension of:
start a match (extension point: step 3: Specify board options)

Steps:

1. Choose opponent	2. Display 'board options' dialog
3. Specify board options (m, n, k)	
4. Confirm Selection	5a. Indicate that values are invalid
5b. Specify valid board options	
5c. Confirm selection	6. Remove 'board options' dialog.
	7. Initialize and display custom board.

Updated Class Diagram



Sequence Diagram Showing TicTacToe Creation