Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 23/03/2014 | 1.0 | Add GUI components’ design. | Erdi Gültekin |
| 24/03/2014 | 1.1 | Update GUI components’ design. | Erdi Gültekin |
| 24/03/2014 | 2.0 | Add Controller design. | A. Emre Ünal |
| 24/03/2014 | 2.1 | Update GUI components’ design. | Erdi Gültekin |
| 25/03/2014 | 2.2 | Update Controller interface. | A. Emre Ünal |
| 25/03/2014 | 2.3 | Update Network components. | Deniz Sökmen |
| 25/03/2014 | 2.4 | Update GUI component interface. | A. Emre Ünal |
| 25/03/2014 | 3.0 | Add intro. | Erdi Gültekin |
| 25/03/2014 | 4.0 | Add Controller workflow. | A. Emre Ünal |

**TABLE OF CONTENTS**

Revision History 1

1- Introduction 3

1.1- References 3

1.1.1- Project References 3

2- Software Architecture overview 4

3- Software design description 5

3.1- Graphical User Interface 5

3.1.1- Component interfaces 5

3.1.2- Component design description 5

3.1.3- Workflows and algorithms 6

3.1.4- Software requirements mapping 6

3.2- Controller 7

3.2.1- Component interfaces 7

3.2.2- Component design description 7

3.2.3- Workflows and algorithms 7

3.2.4- Software requirements mapping 8

3.3- Network 9

3.3.1- Component interfaces 9

3.3.2- Component design description 9

3.3.3- Workflows and algorithms 9

3.3.4- Software requirements mapping 9

3.4- Logic [Eren] 10

3.4.1- Component interfaces 10

3.4.2- Component design description 10

3.4.3- Workflows and algorithms 10

3.4.4- Software requirements mapping 10

4- COTS Identification 11

# Introduction

TicTacToe software is a traditional game that can be played by two players. The software has 2 different game types: local and remote game. Local game can be played on the same computer with 2 players. Remote game consists of two remote computers, which are the host and the client, that are used by 2 remote players.

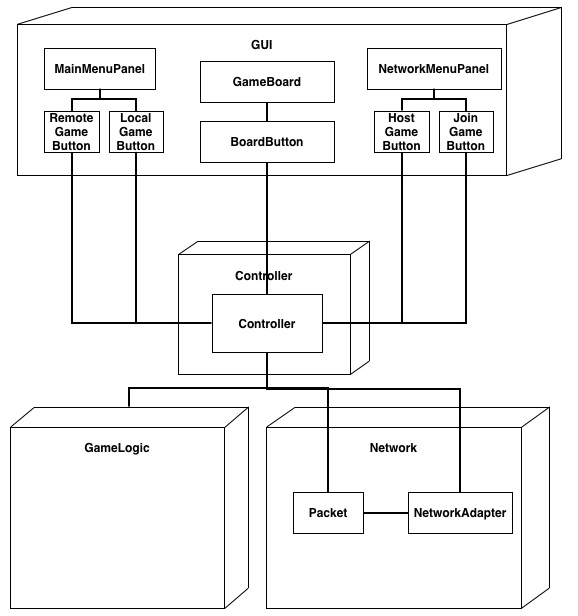
In this Software Detailed Design document, details of TicTacToe software is explained and visualized. The software is formed by 4 components. These are GUI, Controller, Network and Logic.

## References

### Project References

| # | Document Identifier | Document Title |
| --- | --- | --- |
| [R1] | SRS | Software Requirements Specification |

# Software Architecture overview



# Software design description

## Graphical User Interface

### Component interfaces

The user interface of TicTacToe game consists of 4 different components:

1. MainMenuPanel
2. NetworkMenuPanel
3. GameBoard
4. BoardButton

The BoardButton has an interface to set the button text and the button state (enabled or disabled).

No other component defines an interface.

### Component design description

### 

MainMenuPanel: This component creates the main menu of the game. It has two buttons to redirect user, either to a local game or to a network game.  
  
NetworkMenuPanel: This component can be triggered by network game button on the MainMenuPanel. It presents host game and join game buttons to user in order to set up a network game. Join game option uses pop-up box to get IP address information. Host game option uses pop-ups to inform the user about status of remote play (e.g. player is connecting).

GameBoard: This component is created whenever a game is started. The game board consists of 9 instances of the BoardButton class and a turn indicator. It also creates pop-up dialogues to inform the user about game results and the status of remote player (e.g. disconnection).

BoardButton: This component is instantiated by the GameBoard. It is the most basic element of the TicTacToe game. It gets enabled/disabled depending on the game’s status. The text of the button is set according to players’ turns.

### Workflows and algorithms

### Software requirements mapping

This component handles the following requirements:

1. SRS-REQ-101 LOCAL
2. SRS-REQ-102 JOIN
3. SRS-REQ-103 HOST

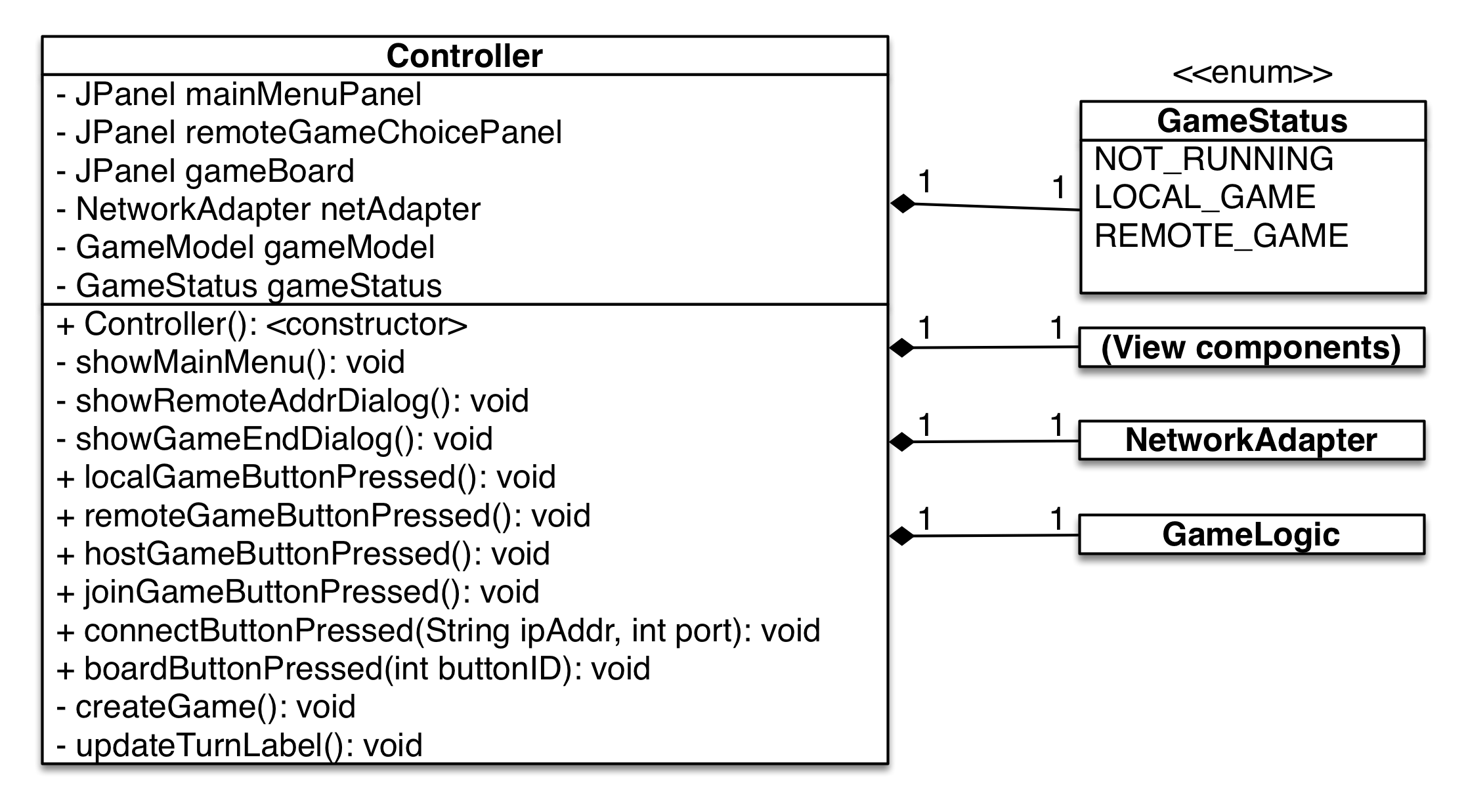
## Controller

### Component interfaces

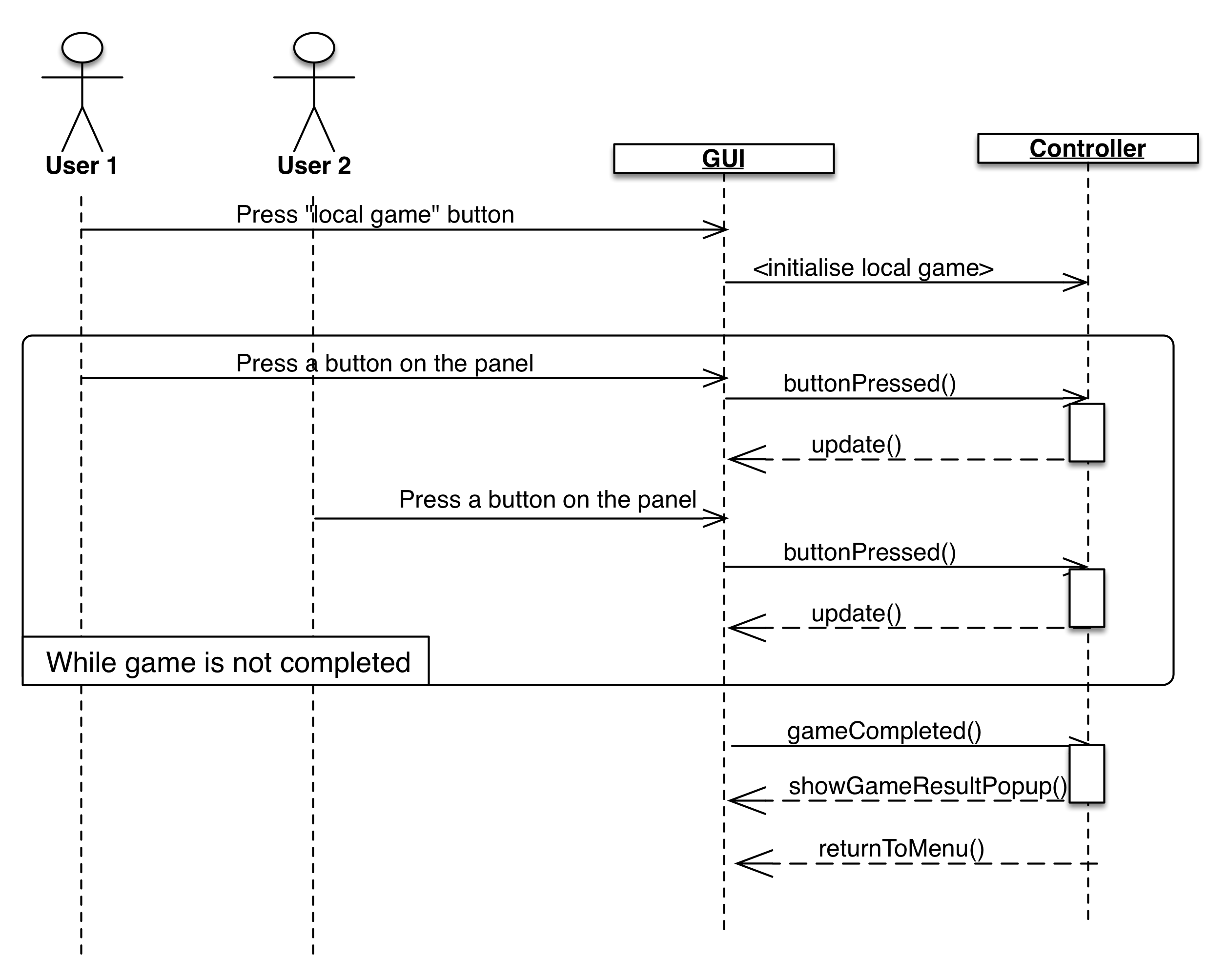
Controller class creates, uses, manages and destroys each and every other component part of the TicTacToe game software.

Controller class provides an interface for the buttons in the GUI. The buttons use the interface provided by the Controller as their event listeners. For example, the event listener for the localGameButton instance in the MainMenuPanel instance is the Controller::localGameButtonPressed() method. Thus, we can generalize this principle for every Controller::[\*]Pressed() method by concluding that they each represent an interface for a specific button.

### Component design description



### Workflows and algorithms

The Controller manages the whole software. It contains the main method, instantiates and manages the lifetime of every other component accordingly. The workflow of the controller for a local game is shown in a high-level fashion below:

### Software requirements mapping

This component handles the following requirements:

1. SRS-REQ-101 LOCAL
2. SRS-REQ-102 JOIN
3. SRS-REQ-103 HOST
4. SRS-REQ-001
5. SRS-REQ-003
6. SRS-REQ-006

## Network

### Component interfaces

The Network component basically consists of two parts: Network Adapter and the Network Packets.

### E:\Downloads\Untitled Diagram(1)(1).jpgComponent design description

NetworkAdapter: Controls all of the network flow and provides all network-related actions in the game. It can host or connect a game, send packets through the network to the peer or listen to the peer to receive a packet.

Packet: Serializable network packets, which are serialized and sent through the network and deserialized when received. Carries information about the action.

### Workflows and algorithms

NetworkAdapter is responsible of creating, opening and closing client and server sockets. The Controller will initiate actions like receiving and sending packets. Therefore, the network module does not have its own sequence diagram. Packets are created by the controller and forwarded to the NetworkAdapter for transmission.

After creating the NetworkAdapter, if the game is hosted, it will block the game until it accepts a connection. Receiving and sending a packet will also block the thread until a packet is arrived or a packet is sent successfully.

### Software requirements mapping

This component handles the following requirements:

1. SRS-REQ-004
2. SRS-REQ-005
3. SRS-REQ-006
4. SRS-REQ-007

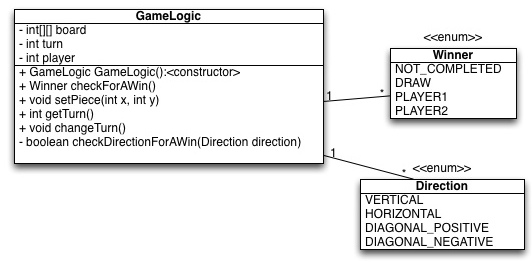
## Logic [Eren]

Repeat the pattern for each component.

### Component interfaces

Describe the interfaces of the component and input output data

### Component design description



### Workflows and algorithms

GameLogic Algorithm

initialize the game

initialize the turn

while the game is not over

     let the player (wrt the turn) pick a square

     switch the turn

Is Game Over Algorithm

for all directions

if 3 cells of the direction are occupied by the same player

return true

return false

### Software requirements mapping

This component handles the following requirements:

1) SRS-REQ-001

2) SRS-REQ-002

3) SRS-REQ-003

# COTS Identification

No commercial library, other than the standard Java v1.7 library, is used.