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 |

**TABLE OF CONTENTS**

Revision History 1

1- Introduction [EREN] 3

1.1- References 3

1.1.1- Project References 3

2- Software Architecture overview [EREN] 4

3- Software design description 5

3.1- Graphical User Interface 6

3.1.1- Component interfaces 6

3.1.2- Component design description 6

3.1.3- Workflows and algorithms 6

3.1.4- Software requirements mapping 6

3.2- Controller 8

3.2.1- Component interfaces 8

3.2.2- Component design description 8

3.2.3- Workflows and algorithms 8

3.2.4- Software requirements mapping 9

3.3- Network [DENIZ] 10

3.3.1- Component interfaces 10

3.3.2- Component design description 10

3.3.3- Workflows and algorithms 10

3.3.4- Software requirements mapping 10

3.4- Logic [Eren] 11

3.4.1- Component interfaces 11

3.4.2- Component design description 11

3.4.3- Workflows and algorithms 11

3.4.4- Software requirements mapping 11

4- COTS Identification [ALL] 12

# Introduction [EREN]

Brief description of the software system and the purpose of the document.

This document describes the design of the XXX software system.

## References

### Project References

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

# Software Architecture overview [EREN]

Describe here the top level software components and their interactions/relationships.

Use UML diagrams.

# Software design description

Describe each top level package/component of your software and if necessary sub-components/sub packages.

Use Class diagrams, sequence diagrams and deployment diagrams to illustrate your description.

## Graphical User Interface

### Component interfaces

The user interface of TicTacToe game consists of 4 different components which are MainMenuPanel, NetworkMenuPanel, GameBoard and BoardButton.  
  
MainMenuPanel is designed to catch the user decision about game type. It offers localGameButton and remoteGameButton components to the user.  
NetworkMenuPanel catches remoteGameButton as an input via Controller. It offers joinGameButton and hostGameButton to the user.  
GameBoard can take its input either from MainMenuPanel or NetworkMenuPanel via Controller. It is designed to capture the inputs from localGameButton, joinGameButton and remoteGameButton to start to visualize the GameBoard.

### Component design description

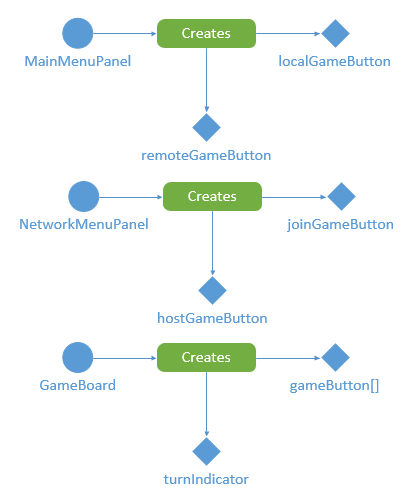
### 

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 can be triggered by local game, join game or host game buttons. It creates the game board which has 9 “BoardButton”s and a player turn indicator. It is also creates pop-up boxes to inform the user about game results and the status of remote player (e.g. disconnection).

BoardButton: This component is used by GameBoard. It includes fields that are required for game such as its status and its sign.

### Workflows and algorithms



### Software requirements mapping

SRS-REQ-101 [R1]: This requirement is handled by MainMenuPanel, GameBoard, BoardButton and relevant other Controller & Logic units.

SRS-REQ-102 [R1]: This requirement is handled by MainMenuPanel, NetworkMenuPanel GameBoard, BoardButton and relevant other Controller & Logic units.

SRS-REQ-103 [R1]: This requirement is handled by MainMenuPanel, NetworkMenuPanel GameBoard, BoardButton and relevant other Controller & Logic units.

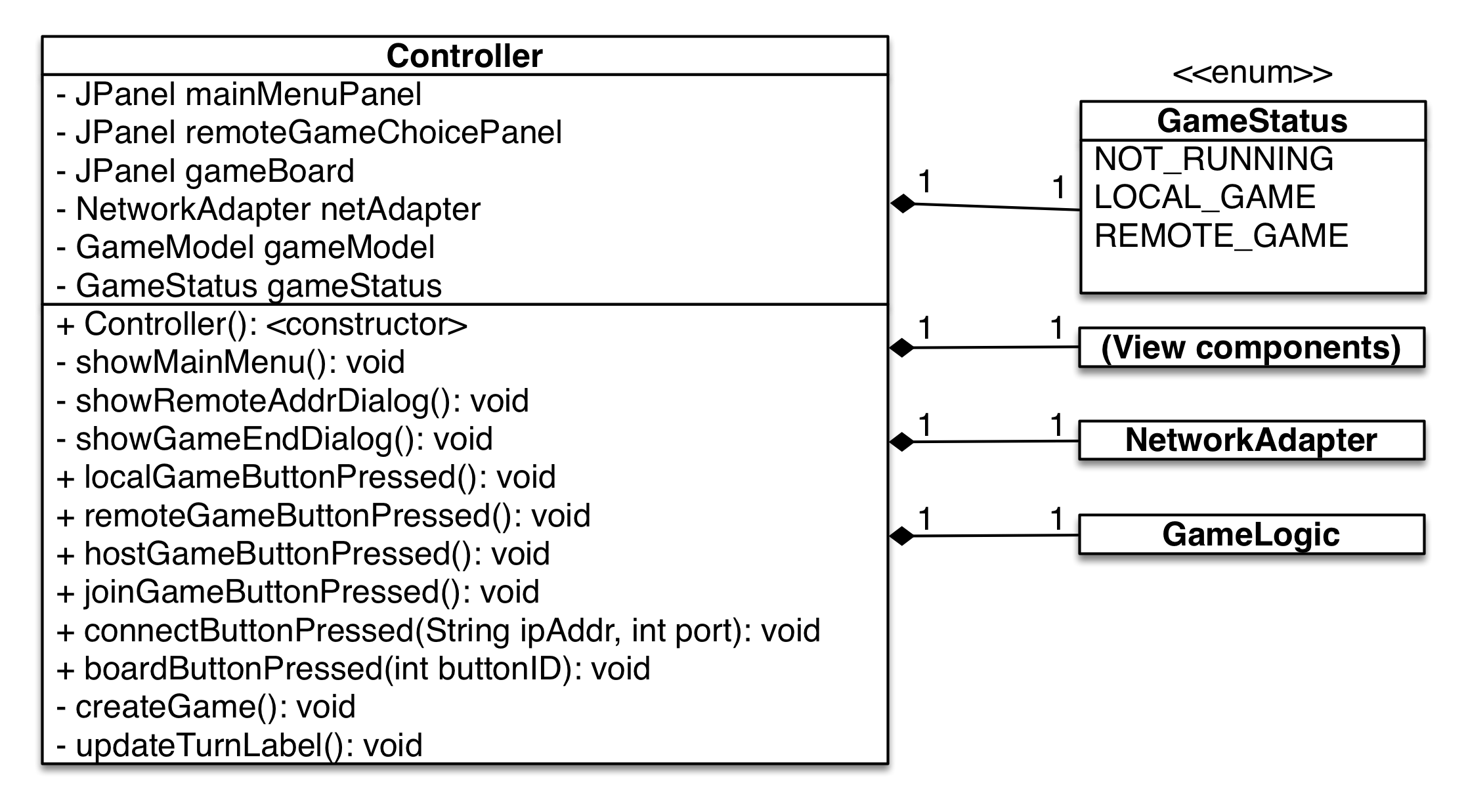
## Controller

### 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

### Software requirements mapping

This component handles the following requirements:

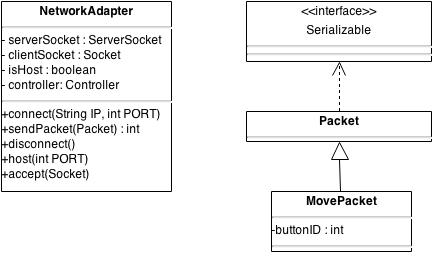
1. SRS-REQ-101 LOCAL
2. SRS-REQ-102 JOIN
3. SRS-REQ-103 HOST
4. SRS-REQ-001
5. SRS-REQ-003
6. SRS-REQ-006

## Network [DENIZ]

### Component interfaces

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

### Component 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 on receive. Carries information about the action.

### Workflows and algorithms

NetworkAdapter itself does not have a task except creating and closing client and server sockets. Any other actions like receiving and sending packets will be applied by the Controller. Therefore the network module does not have its own sequence diagram. Packets and the NetworkAdapter does not have a direct relationship, all of the components of the network module are managed by the Controller.

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

NetworkAdapter handles these requirements listed: SRS-REQ-004, SRS-REQ-005, SRS-REQ-006, 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

Describe the design of the component, Use class diagrams to show the links between sub-components/sub-packages and or classes inside the component.

### Workflows and algorithms

Use sequence diagrams to show the workflows of components/packages/classes inside the component.

Describe algorithms, if possible. An algorithm may be described outside this document, in this case, add the reference to that document.

### Software requirements mapping

List the SRS requirements handled by this component

# COTS Identification

Nothing other than the standard Java v1.7 library.