Challenge - Pairs Game

Emmanuel Miras 40168970@napier.ac.uk
Zoe Wall 40182161@napier.ac.uk
Edinburgh Napier University - Fundamentals of Parallel Systems (SET09109

1 Requirement 2

Within the original implementation the Player Manager process acted as a server and the Controller Manager process acted as a client. In order to implement the required changes, the roles had to be switched. This proved to be quite straightforward, the logic needed for enrollment of the player was abstracted into a new process called Enrol Player, in doing so the Player Manager can be treated as a pure server as it no longer handles the initialisation of the player.

The majority of the logic for making the game work as required in the specification document was already in the system, it was mostly a matter of switching the channels around to support the new client server relationship, while also making sure the logic for each part of the system is abstracted into the correct process.

The data structures contained in the original implementation were sufficient for the functionality of the system. No new data structures were created, but they were edited to support new features that were added (turns, updating player boards)

2 Requirement 3

2.1 Process Network

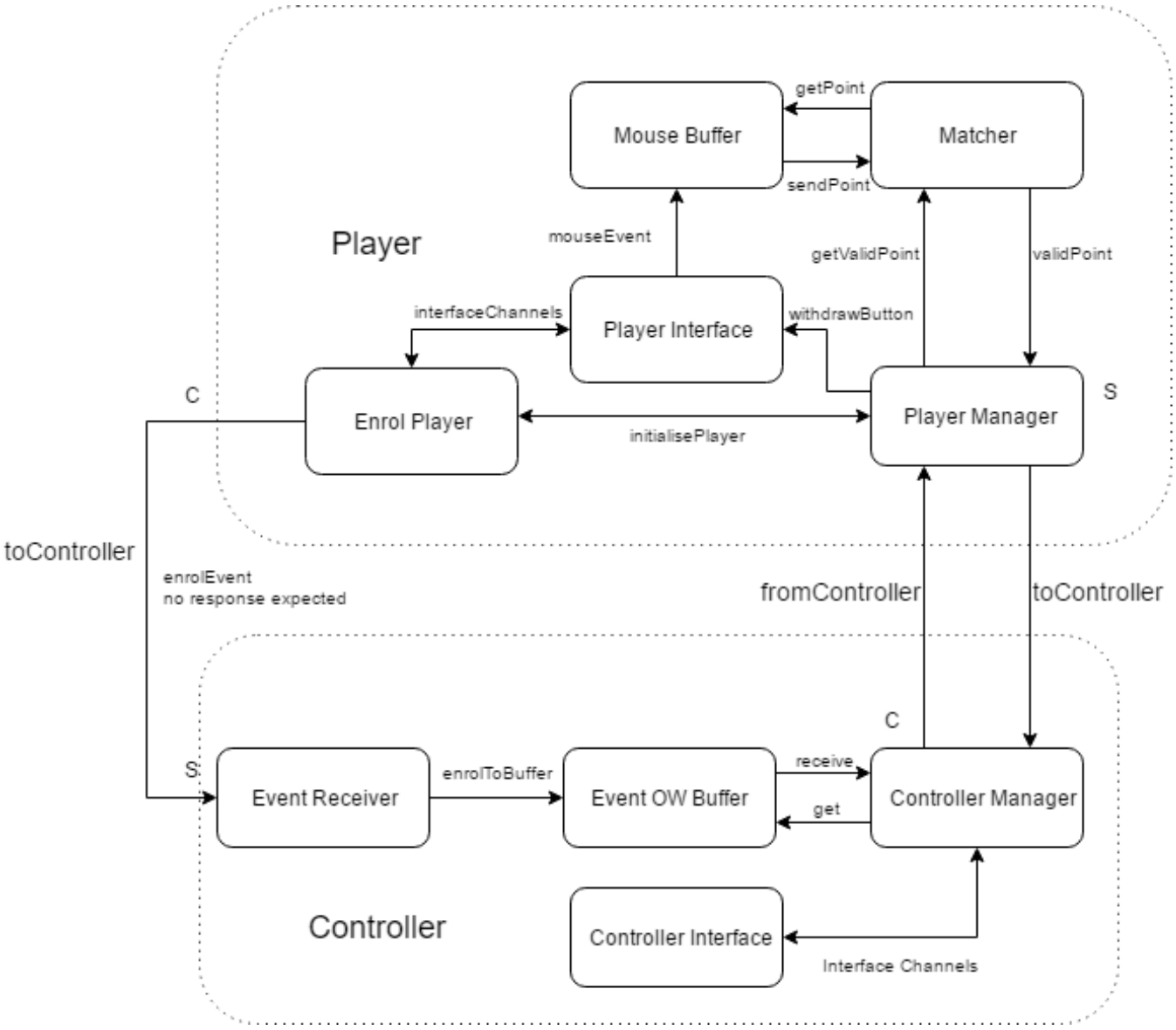

Figure 1: **Process Network** - diagram to show channel connections in the Pairs Game network.

2.2 Channel Interaction

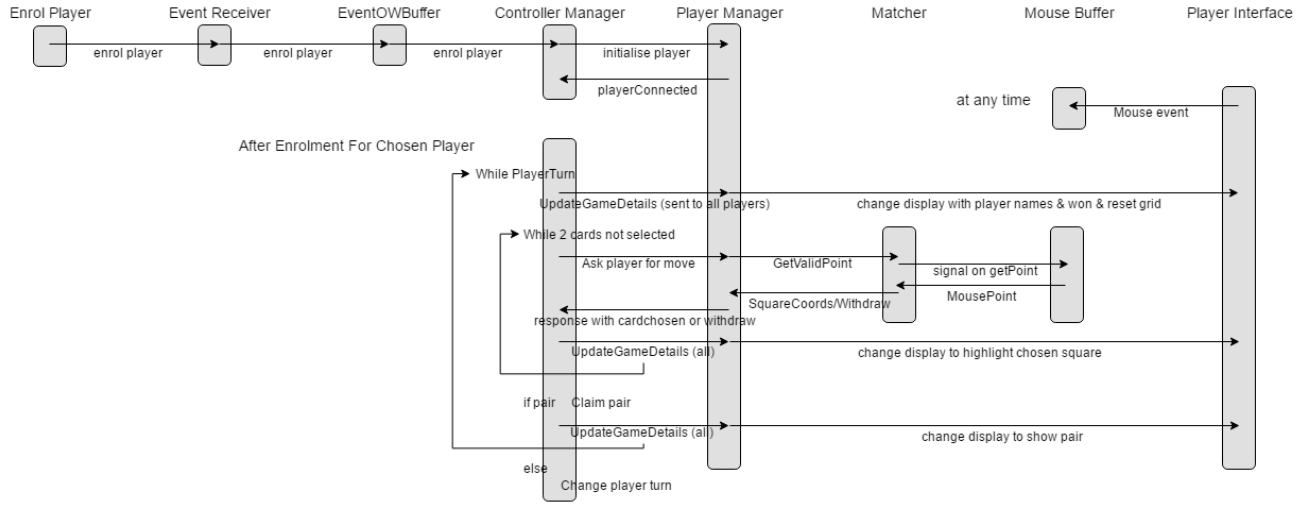

Figure 2: **Channel Interaction Sequence** - diagram to show the sequence of interactions between channels.

2.3 Explanation

The above diagrams detail a modified version of the Pairs Game system to add the functionality of taking turns and dynamically updating every players interface to match the game state.

Turns To implement player turns, the Controller Manager must keep track of every player initialised within a list. To initialise a player, a modified EnrolPlayer object, which contains the location of the player sending the request and the player id, is copied to the Controller Manager. Once the player has been enrolled they are added to a list of current players, which the Controller Manager can iterate through to request for card selections and send a map of square coordinates for updating the game state.

Updates The main change to the communications structure is to modify the behaviour of both the Player Manager and Controller Manager. The proposed solution is to have multiple channel inputs to the Player Manager that are allocated dynamically once the player has been enrolled. In doing so, each of the Player Managers can act as pure servers. This means that the Controller Manager can issue requests to every Player Manager to update their interfaces without the risk of dead-lock. With the current structure, this causes an issue with the enrollment of players as they must initiate the connection to the Controller Manager therefore acting as clients. To solve this, the enrollment logic from the Player Manager is abstracted into another process which can act as a client. An enrollment request can therefore be copied through an event handling system using an overwriting buffer to the Controller Manager client that obeys the client-server pattern.

3 Requirement 4

3.1 Group H

Functionality and ease of playing the game The game implementation by Group H had all of the required features as requested in the specification document and was fully operational, although there were a couple of issues.

Issues with the game:

- Withdrawal from the game not working.
- Clicking on the same card twice would register as a pair.

When playing the game, it was easy to understand what was going on, as relevant information about turns and points where available on the interface. The "next turn" button was removed, allowing for simpler interaction with the interface. There was a time delay after a turn ending, allowing each player enough time to memorize the cards selected, the game could benefit from a shorter time delay, because it gave the impression that the game had frozen.

Communications structures and interactions The communication structures and interactions of the system as portrayed in the particular version of the design document provided, are not very clear, many processes are encapsulated, hiding information that is essential for understanding how the system works. There is a description available, that attempts to explain how the system would behave when running, but there is no channel interaction diagram to visually aid that description. Finally, there is no information available specifying why or how particular design choices were made.

Use of Data Structures An agent was used to handle the turns in the system, the agent is "assigned" to the current player's turn, meaning that the player sends data to the controller through the agent, there is no mention of how the agent is internally structured.

For updating a player's board while an other player is currently playing, an "update" object is sent throughout the system, this seems like an efficient way to update each player's board.

3.2 Group J