13 december 2019

Software Engineering I
Group project
Game Documentation

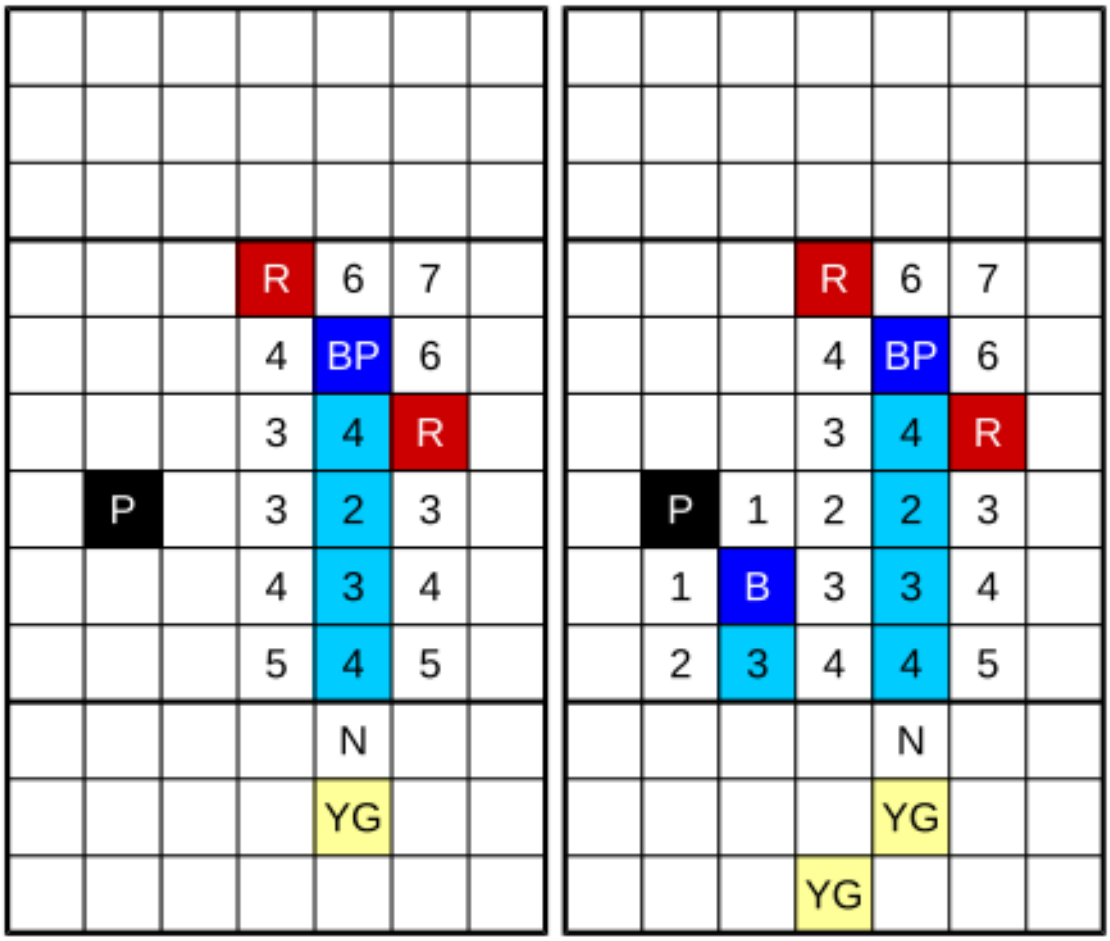
Elif Ozdemir
Assyl Salah
Nikita Zakharov
Vladislav Sorokin
Denis Harbatsenka

Instructors: Aleksander Kosicki, Michal Okulewicz

Contents

1 Introduction

The system organizes matches between 2 teams of 4 players . The game is played in real time on a board, consisting of the tasks area and the goals area. Each team of players needs to complete an identical task, consisting of placing a set of pieces, picked from the tasks area, in the project goals area. The pieces appears at random within the tasks area. The player's view of the tasks area is limited and the shape of the goals needs to be discovered. The game is won by the team of players which is first to complete the project.The architecture of a system and a communication protocol design should focus on solving the effects of asynchronous actions taken by the players.



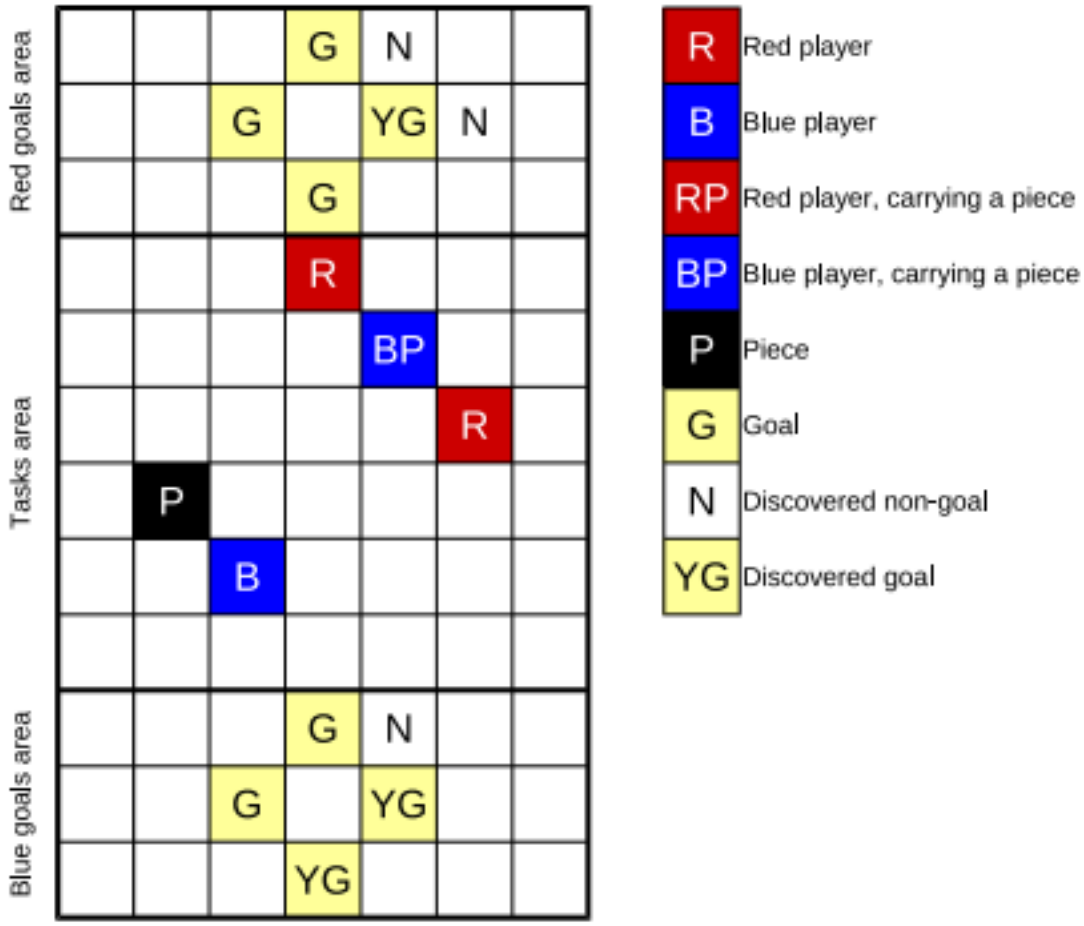


Figure 1: Global state of the game.

2 System requirements

2.1 User stories

As a player I could mute another player so I would not see dummy messages from him/her.
As a host I could create lobby so that players could join to it.
As a host I could configure game properties so that we would enjoy game more.
As a team we can have check that everyone is ready so that we could start the game.
As a player I could check a piece so that I would know what kind of piece it is.
As a host I could configure where goals are so that team could train before the real game.

2.2 FURPS

Functionalities

Game Setup:

1. The game system allows to play the game.
2. The game should be played on a computer.
3. The players should be able to join the game before the game starts.
4. It's possible to play one game at a time.
5. Game Master and Player are the main components of the system.
6. The game consists of two teams and a team consists of 1-4 players.
7. One of the players is the host.
8. System should serve as an intermediary for communication between the players.
9. All the communications between Game Master and Player should be handled by the server.
10. The communication server handles everything at the lower level without processing the content.
11. Game Master should be aware of and able to process all requests sent to him from Player.
12. Game Master should make all the decisions locally according to the requests made to it.

Game Play:

13. The rules that are valid during the game play are described in details in Game rules section of the documentation.

Usability

1. The full documentation is provided.
2. A manual for the user is included in the documentation.
3. Some use-case diagrams will be provided.

Reliability

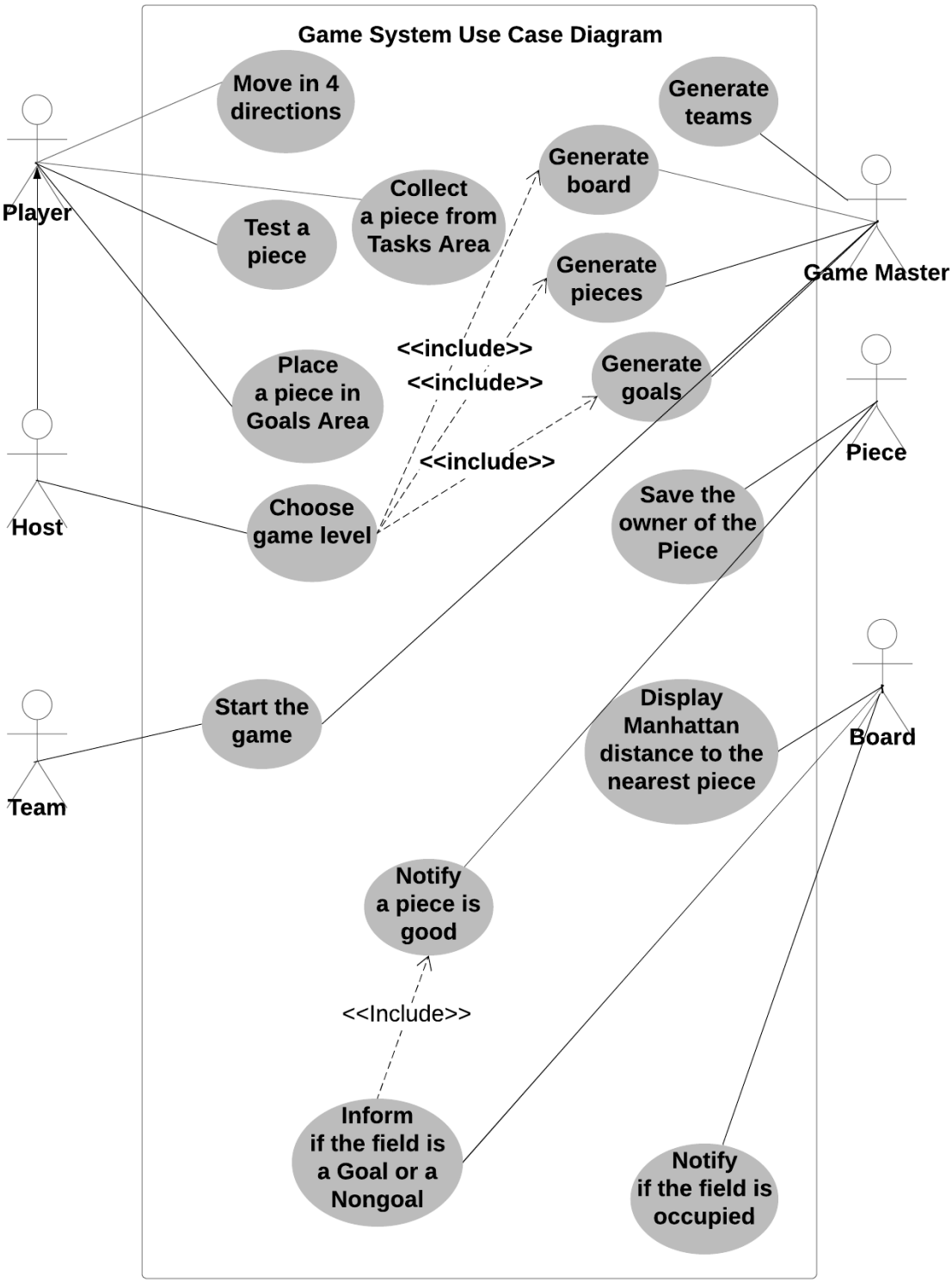
1. Personal data is protected.
2. In case of a failure, user should be provided with an error message describing what exactly went wrong.
3. If there were 5 unsuccessful attempts to send message from server, the player is disconnected.

Performance

1. All team members are supported at the same time.
2. The maximum response time is 1 second.
3. The memory consumption is low

Supportability

1. The game is installable.
2. System should support operating systems like Windows, Linux, Mac.
3. To understand the role of each user in the game, use-case diagrams should be provided.


Game System Use Case Diagram
Move in 4 directions
Generate teams
Generate board
Collect a piece from Tasks Area
Test a piece
Generate pieces
Player
Game Master
<<include>>
<<include>>
Place a piece in Goals Area
Generate goals
<<include>>
Piece
Host
Choose game level
Save the owner of the Piece
Start the game
Display Manhattan distance to the nearest piece
Board
Team
Notify a piece is good
<<Include>>
Inform if the field is a Goal or a Nongoal
Notify if the field is occupied


Communication
Check collision and borders
Player
<<include>>
Make a move
Mute another player
Feedback another player
Game Master
Communication Server
<<extend>>
<<extend>>
Message another player
Board
Piece
Start a game
Update the board
Inform piece type

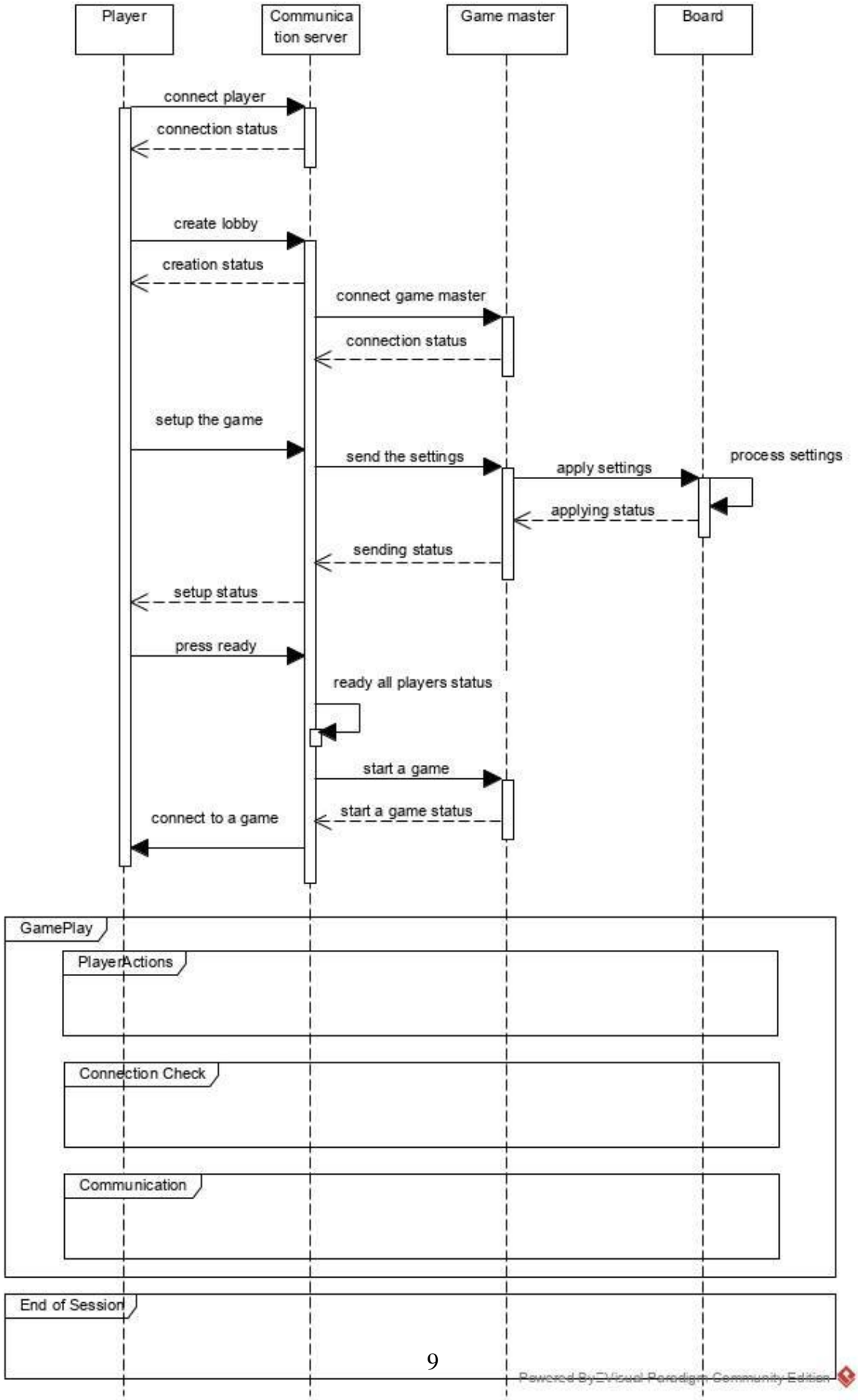
3 Design Documentation

3.1 Architecture design

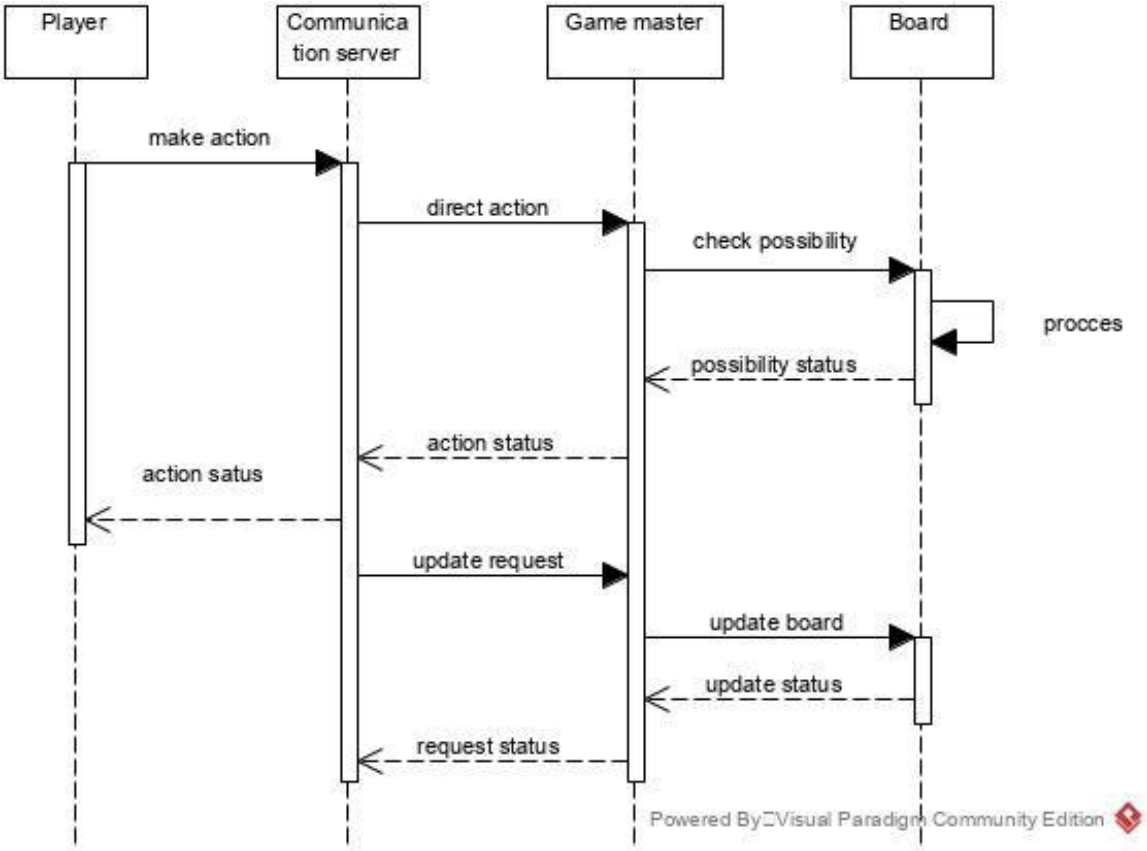
The system consists of three components which are Player, Game Master and Communications Server. Player is an agent playing the game, holds its own view of the state of the game. Game Master generates the board and the shape of the project goals, holds the real state of the game , generates new pieces on the tasks area and generates teams. The game stays under control in real time by the Game Master on regular basis till the end of the game.
Communications Server is responsible for passing messages between Game Master and Players whenever some action is taken by the player like movement or taking a piece. Communication server is sending the message to the Game Master to handle the actions and to send a signal back to confirm the action and update the board. Game Master can communicate with players to restrict them from doing forbidden moves, giving them permission only if the action is valid. Moreover, through the Communications Server players can communicate with each other.

3.2 Communication protocol design

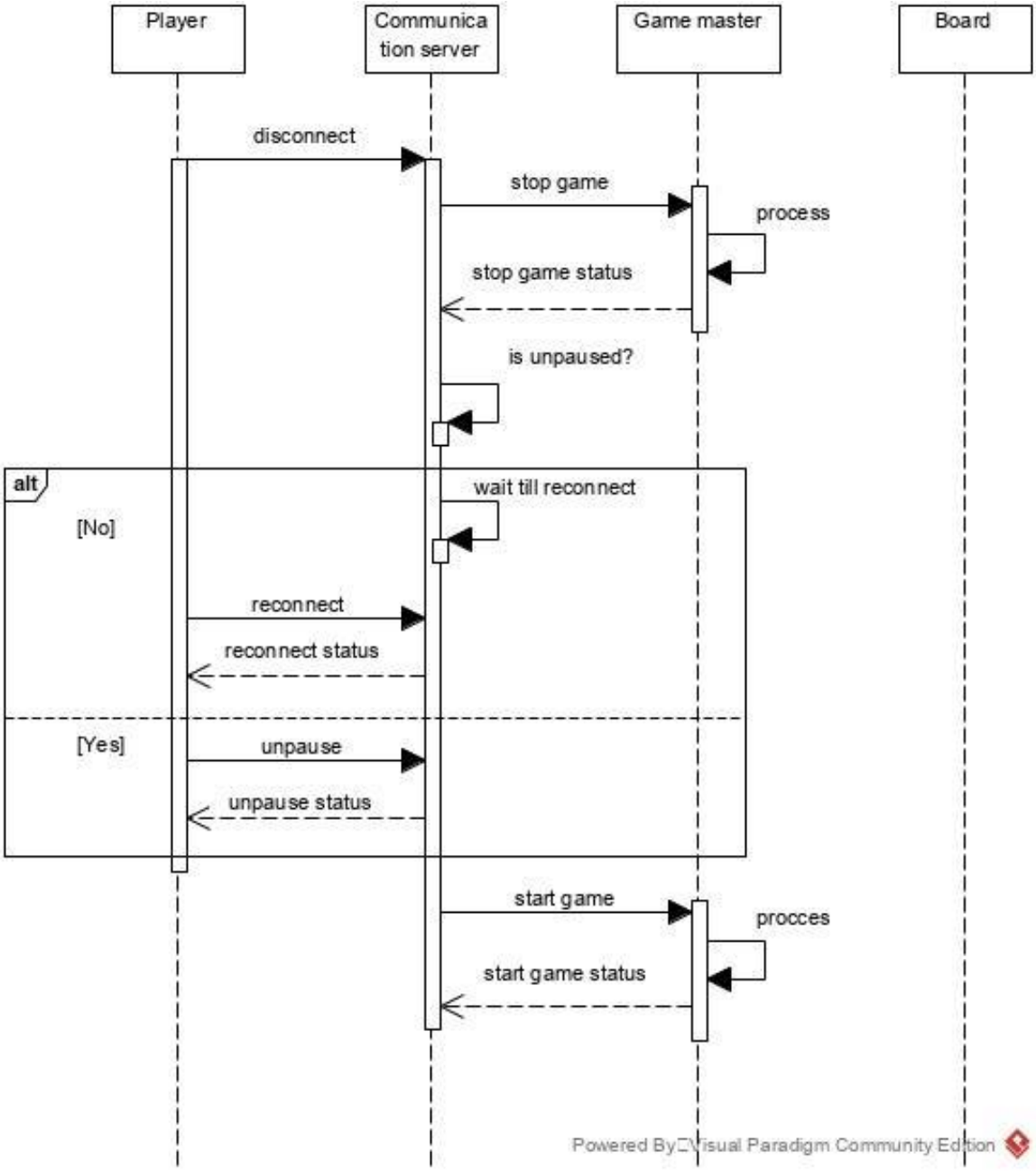
Initialization


Player
Communication server
Game master
Board
connect player
connection status
create lobby
creation status
connect game master
connection status
setup the game
send the settings
apply settings
process settings
applying status
sending status
setup status
press ready
ready all players status
start a game
start a game status
connect to a game
GamePlay
PlayerActions
Connection Check
Communication
End of Session
Powered By Visual Paradigm Community Edition

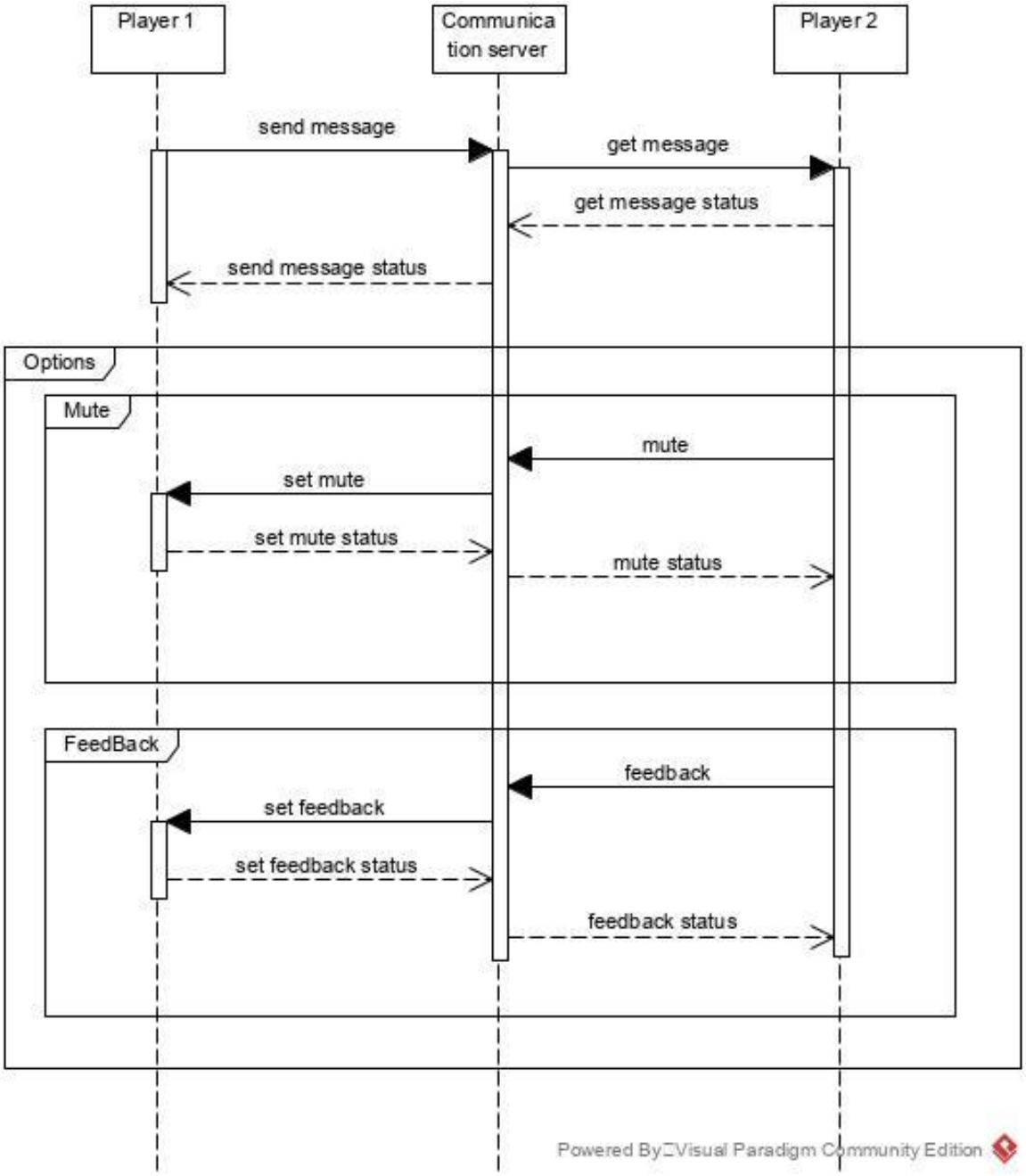
Player actions



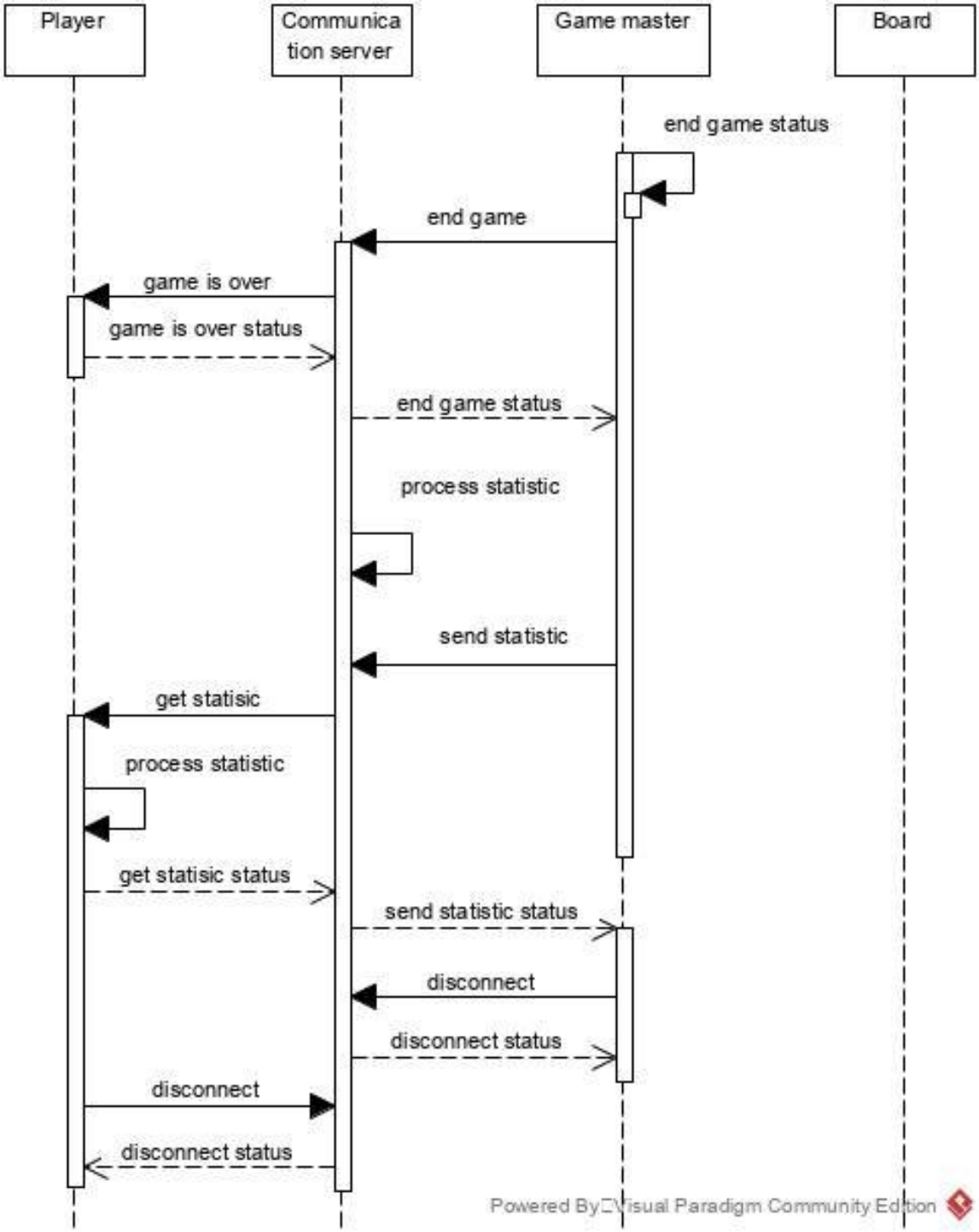
Connection check



Communication



End of session



Type of messages

- Connection message
- Status message
- Communication message
- Setup message message
- Action message
- State message
- Control message
- Update message
- End game message
- Statistic message

These are templates

```
<?xml version="1.0" encoding="UTF-8"?>
    <message type="connection">

        <id unique="true" value="{string}"></id>
        <sender value="{string}"/>
        <recepient value="{string}"/>
        <generation value="{datetime}"></generation>
        <expire value="{datetime}"></expire>

        <status.required value="{boolean}"/>
        <required.id value="{string}">

        <ip value="{IPv4}"/>
        <port value="{int}"/>
        <available value="{boolean}"></available>

    </message>


    <message type="status">

        <id unique="true" value="{string}"></id>
        <sender value="{string}"/>
        <recepient value="{string}"/>
        <generation value="{datetime}"/>
```

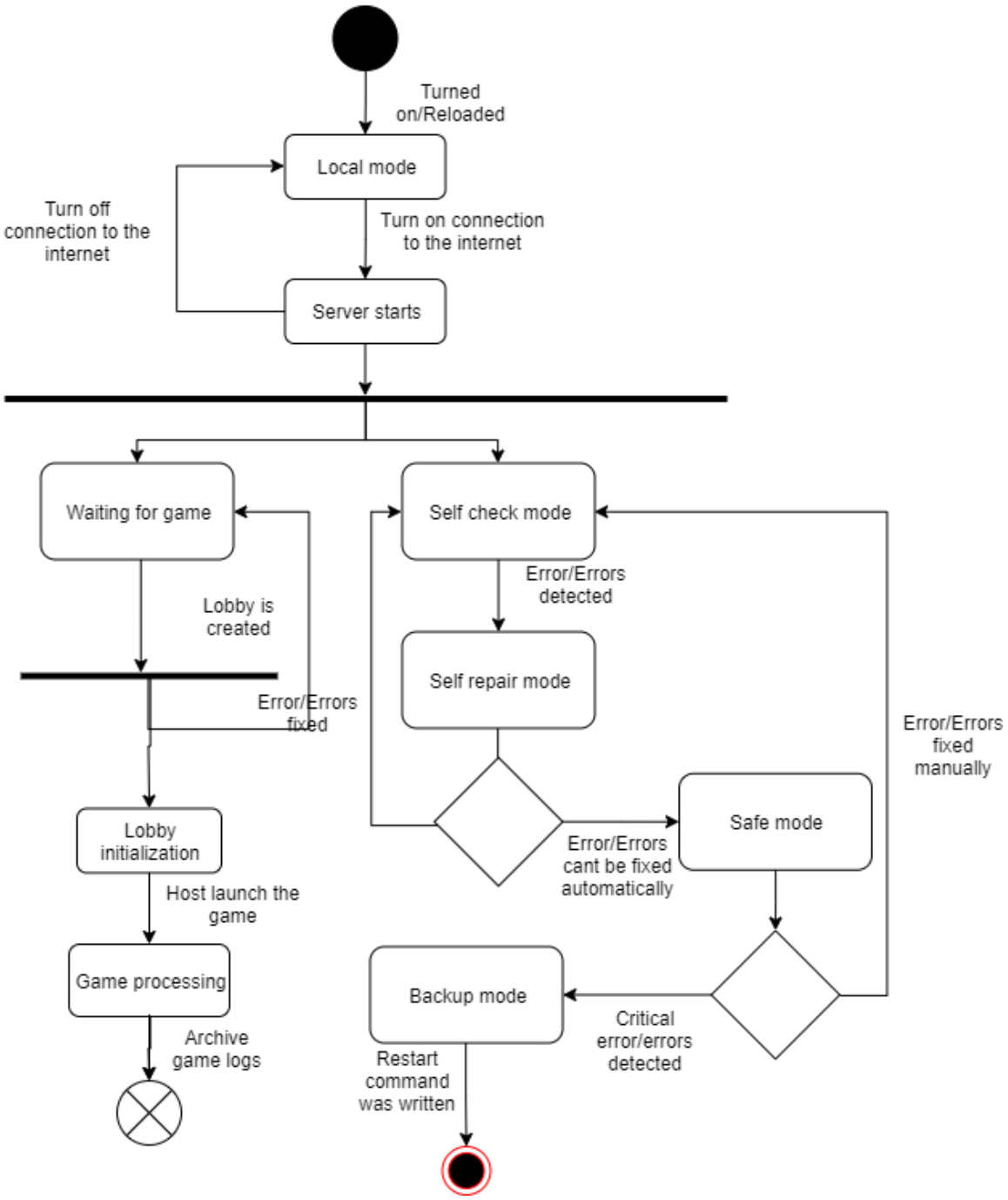
```
    <expire value="{datetime}"/>

    <reference value="{string}"/>
    <status value="{boolean}"/>

</message>


<message type="communication">

    <id unique="true" value="{string}"></id>
    <sender value="{string}"/>
    <recepient value="{string}"/>
    <generation value="{datetime}"/>
    <expire value="{datetime}"/>

    <destinationId value="{string}"/>
    <destinationIp value="{Ipv4}"/>
    <text value="{string}"/>
    <riched value="{boolean}"/>

</message>

<message type="setup">

    <id unique="true" value="{string}"></id>
    <sender value="{string}"/>
    <recepient value="{string}"/>
    <generation value="{datetime}"/>
    <expire value="{datetime}"/>

    <number value="{int}"/>
    <size value="{int}"/>
    <timer value="{time}"/>

</message>


<message type="action">

    <id unique="true" value="{string}"></id>
    <sender value="{string}"/>
    <recepient value="{string}"/>
    <generation value="{datetime}"/>
    <expire value="{datetime}"/>
```

```
    <number value="{int}"/>
    <size value="{int}"/>
    <timer value="{time}"/>

    <type type="action" value="{move|grab|test|place}"/>
    <available value="{boolean}"/>

</message>


<message type="state">

    <id unique="true" value="{string}"></id>
    <sender value="{string}"/>
    <recepient value="{string}"/>
    <generation value="{datetime}"/>
    <expire value="{datetime}"/>

    <state value="{string}"/>
    <destinationId value="{string}"/>

</message>


<message type="control">

    <id unique="true" value="{string}"></id>
    <sender value="{string}"/>
    <recepient value="{string}"/>
    <generation value="{datetime}"/>
    <expire value="{datetime}"/>

    <action value="{start|stop}"/>

</message>


<message type="update">

    <id unique="true" value="{string}"></id>
    <sender value="{string}"/>
    <recepient value="{string}"/>
    <generation value="{datetime}"/>
    <expire value="{datetime}"/>
```

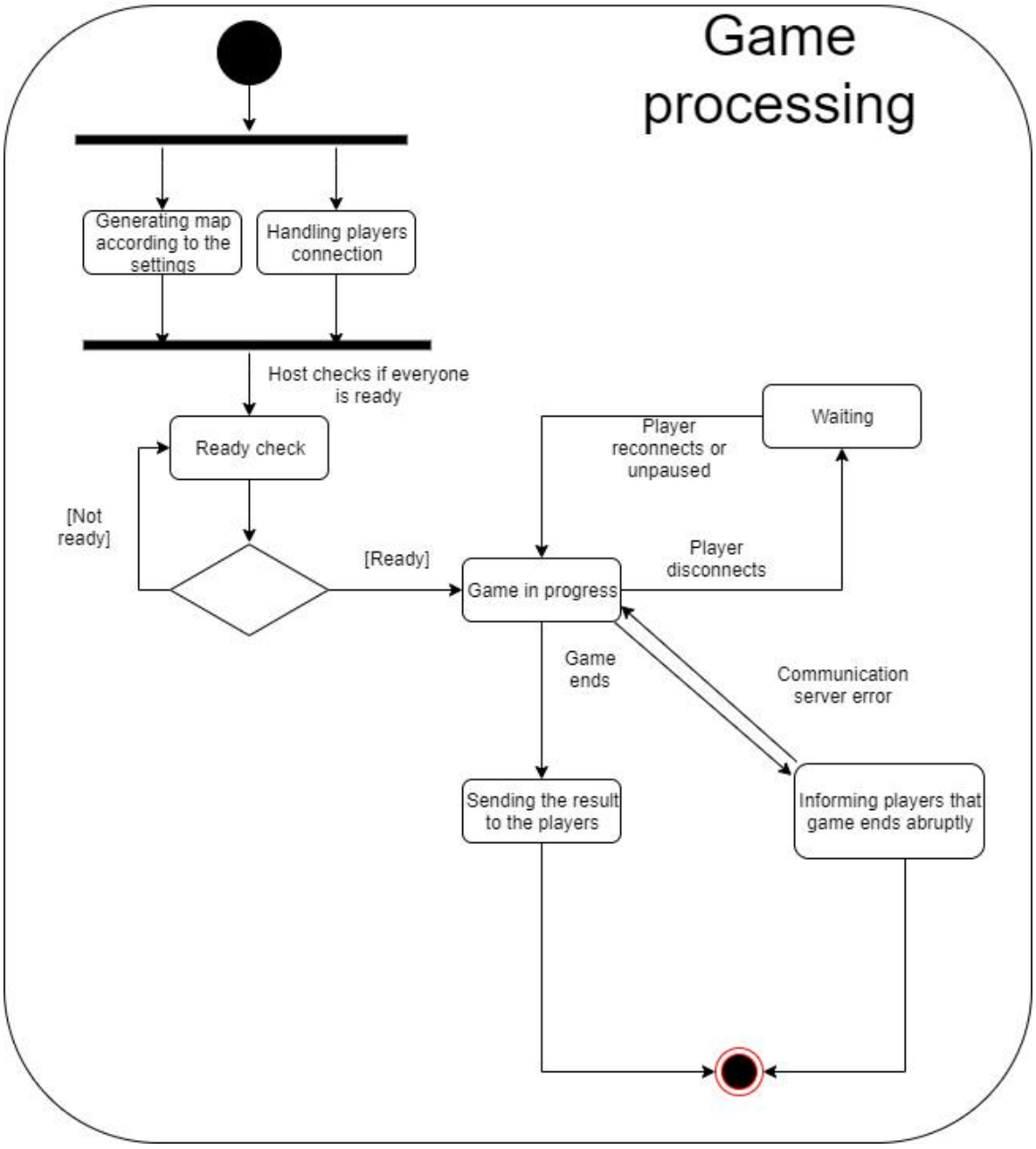
```
    <update.state value="{boolean}"/>
    <board.id value="{string}">
    <position value="{int}"/>

</message>

<message type="game.end">

    <id unique="true" value="{string}"></id>
    <sender value="{string}"/>
    <recepient value="{string}"/>
    <generation value="{datetime}"/>
    <expire value="{datetime}"/>

    <end.flag value="{boolean}"/>

</message>

<message type="statistic">

    <id unique="true" value="{string}"></id>
    <sender value="{string}"/>
    <recepient value="{string}"/>
    <generation value="{datetime}"/>
    <expire value="{datetime}"/>

    <score value="{int}"/>
    <winner value="{string}"/>
    <final.time value="{time}"/>

</message>
```

3.3 Special system states description

Game life cycle in server side point of view

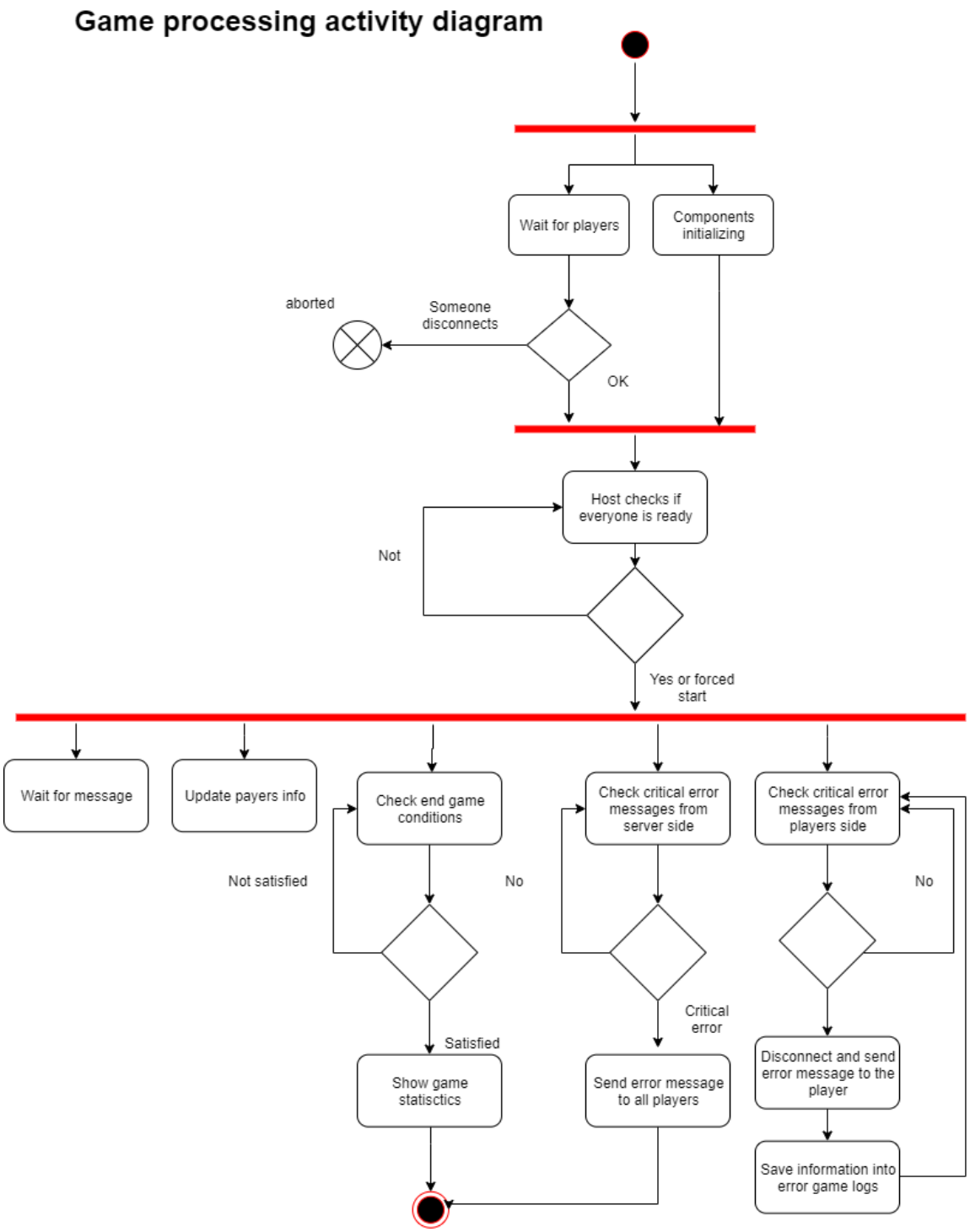


Game Life cycle

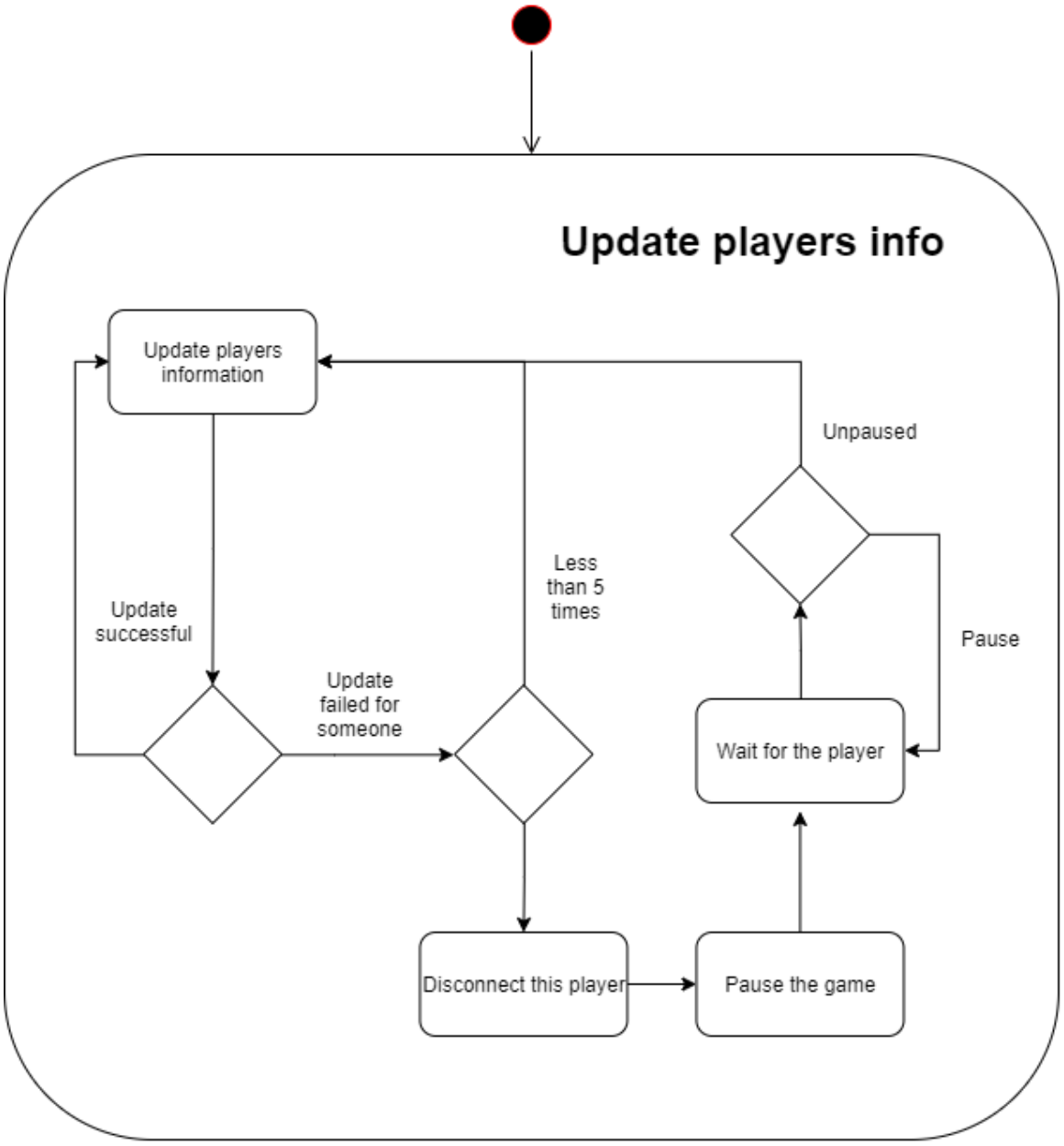

Game processing
Generating map according to the settings
Handling players connection
Host checks if everyone is ready
Ready check
[Not ready]
[Ready]
Waiting
Player reconnects or unpaused
Player disconnects
Game in progress
Game ends
Communication server error
Sending the result to the players
Informing players that game ends abruptly

Game process

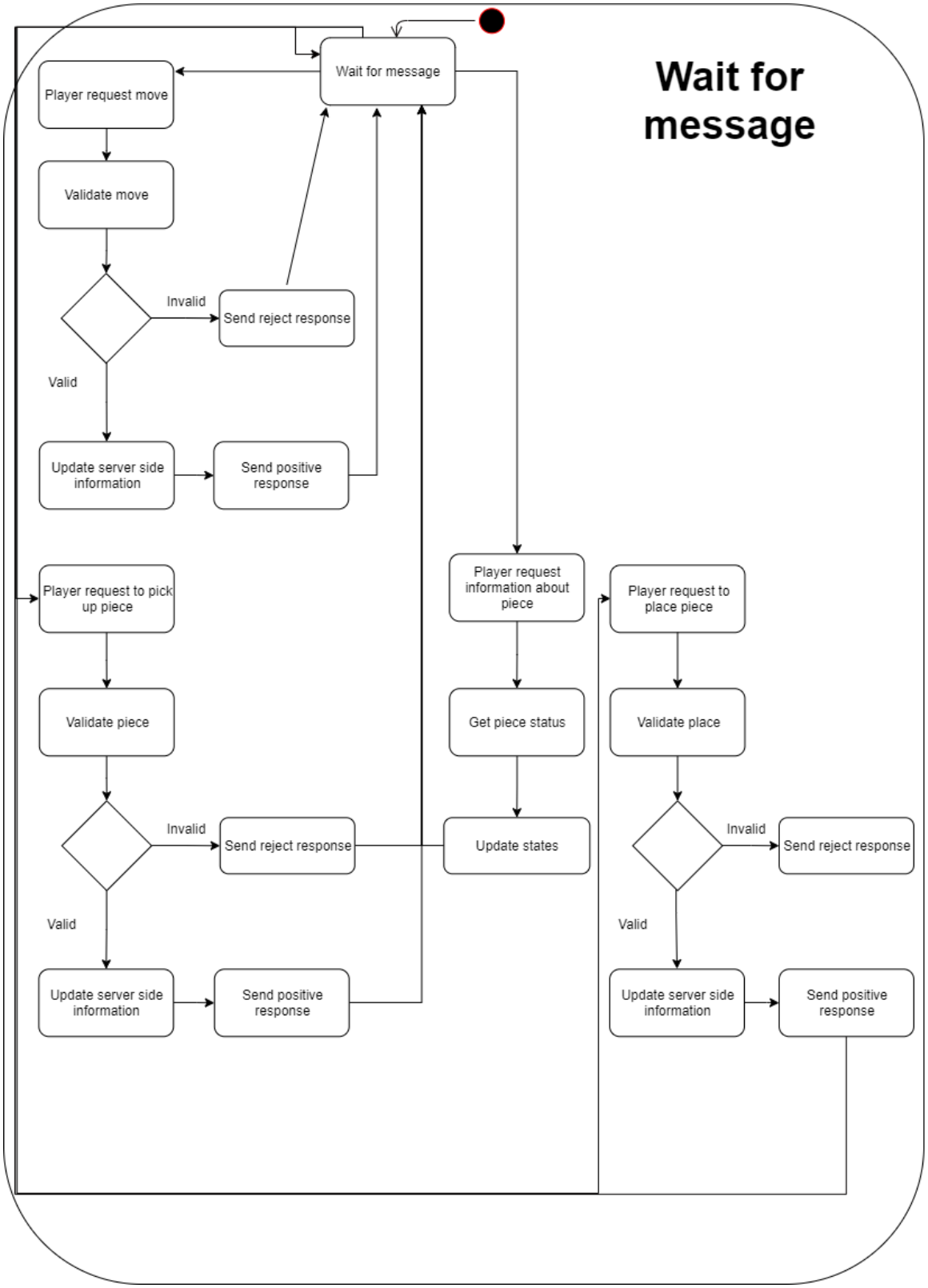
We had one activity diagram and we decided to divide it into three parts , the second diagram and the third was introduced as a state in game process diagram



Game play


Update players info
Update players information
Update successful
Update failed for someone
Less than 5 times
Unpaused
Pause
Wait for the player
Disconnect this player
Pause the game

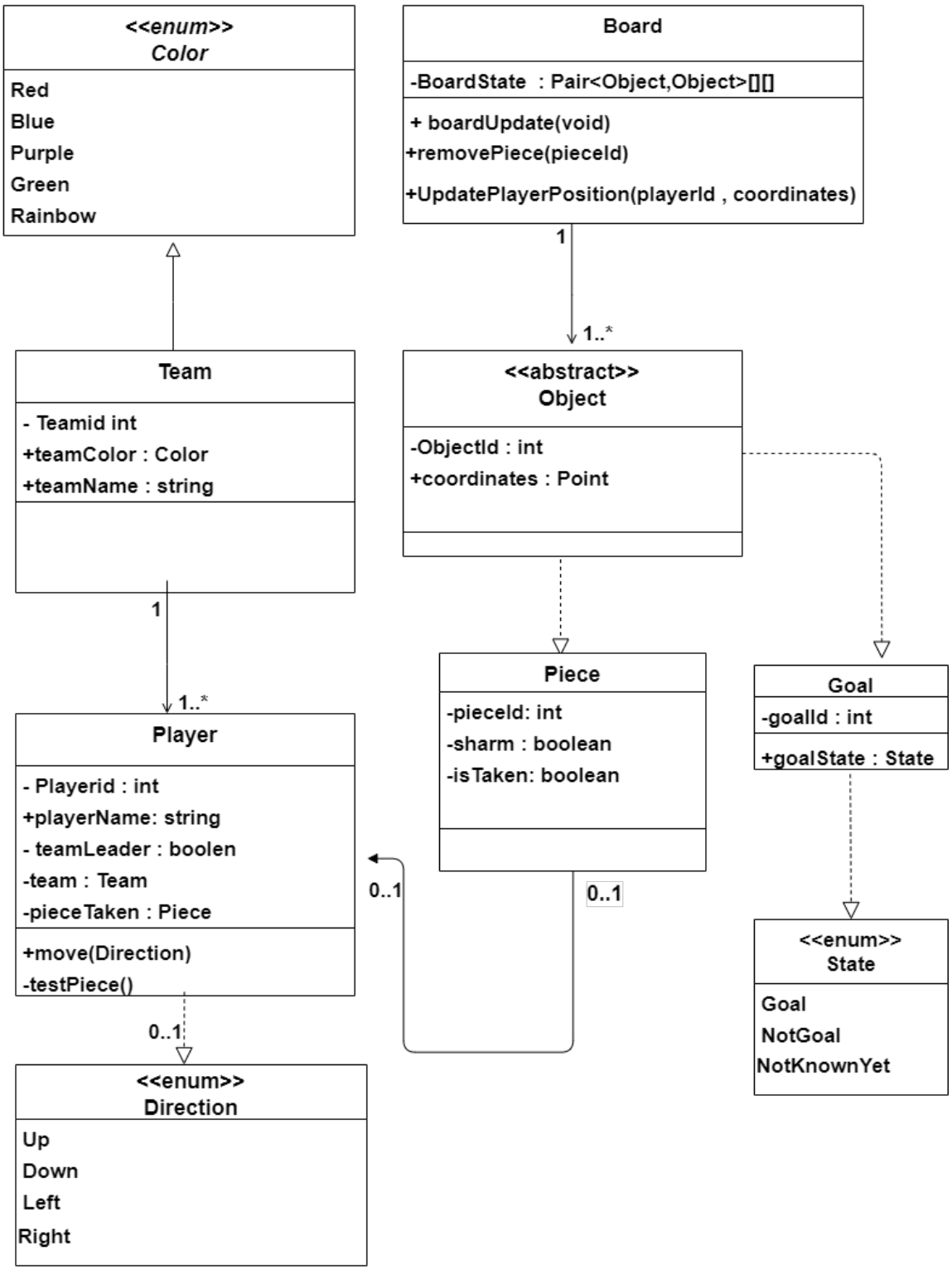
Game termination



3.4 Game related objects

- **Board** - Object which is generated in the beginning of the game and handled game along by Game Master. It provides the players with the information of the entire current state of the game and whenever some changes occur Game Master updates the board.
- **Player** - The participant of the game. As a game object is used for collecting the information for each player, its current state and all achievements.
- **Host** - The player which handles the launch of the game when all the players are ready. Its role is to create a lobby , choosing game level and start the game.
- **Team** - The collection of players no more than 5 and only 2 teams playing at the same time. Each team has its own team leader which is chosen by all the players. The color of the team can be chosen by team leader to be Red, Blue, Purple, Green and Rainbow.
- **Piece** - The object which is used as award in the game appearing randomly in the board and supposed to be discovered by the players as soon as possible. Piece has two types: 'good' and 'sham'. This object can have different states depending on whether its picked by the player or not and etc.
- **Goal** - The part of the Board, which is different for each of the teams. The player can interact with it by bringing the found pieces. The information about the current state of the Goal is saved in this object.

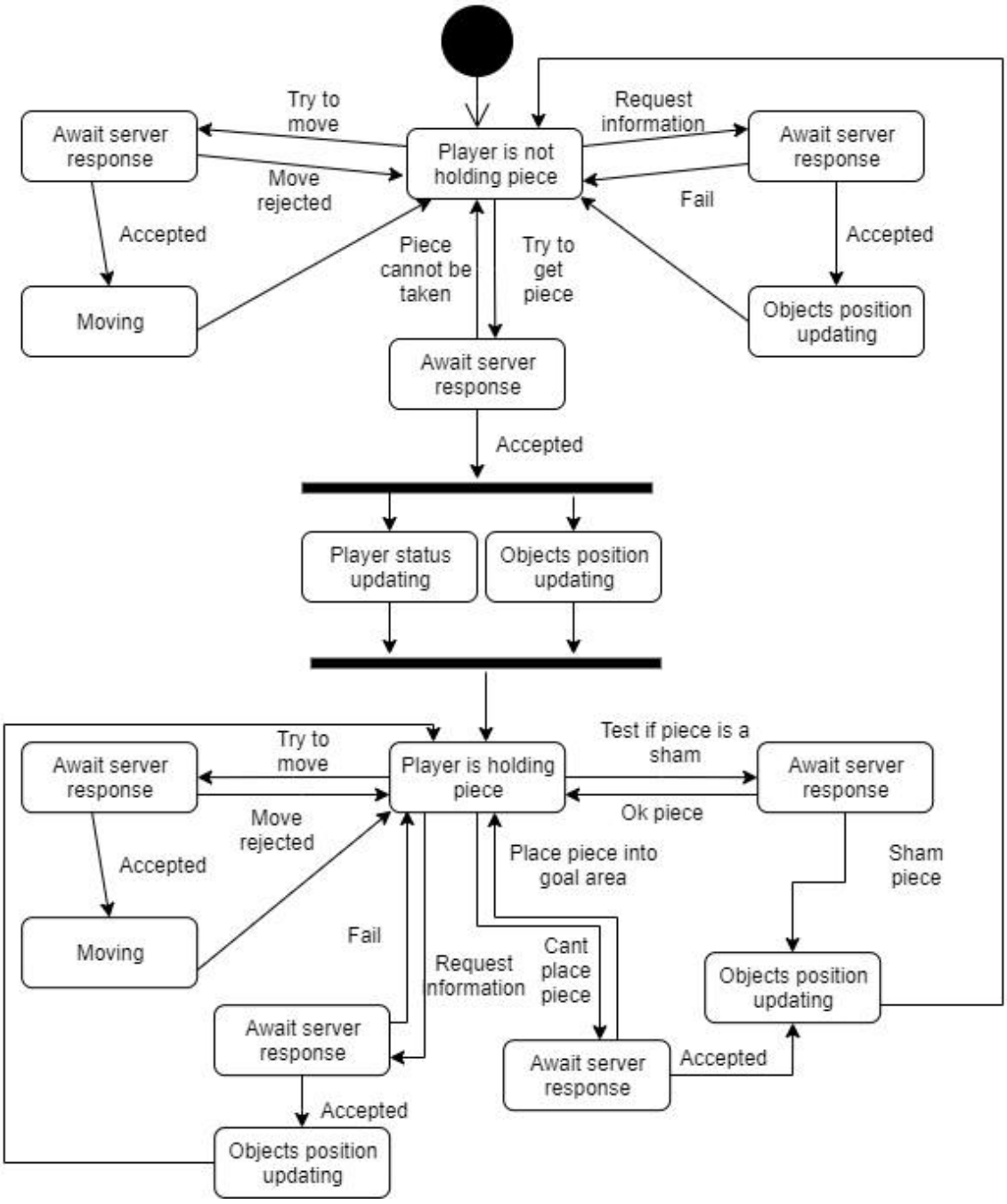
Game Element Class diagram

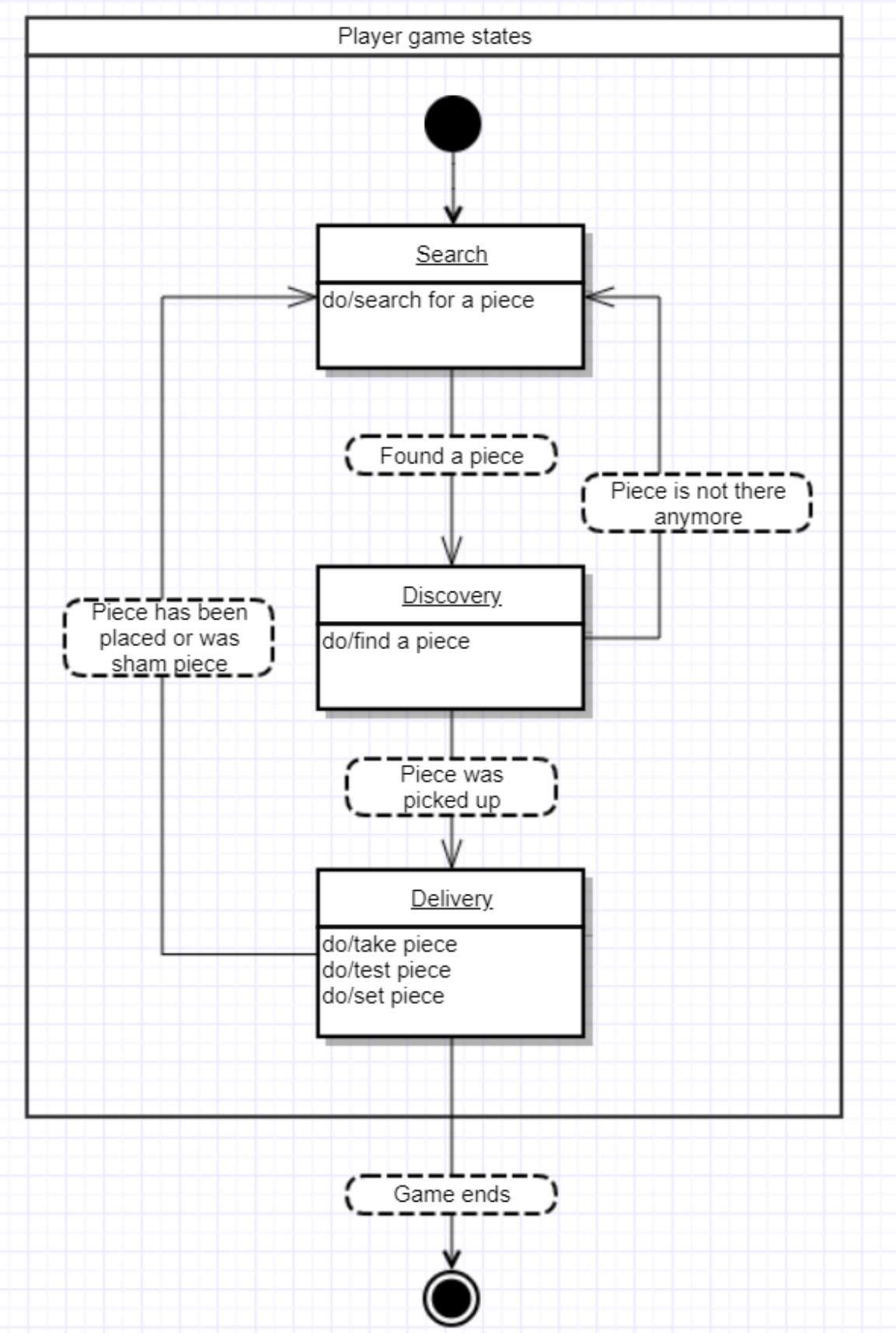


3.5 Game rules

- The Host of the game creates lobby for the Players and after choosing the difficulty level starts the game when all of the team members are ready.
- After the launch of the game Game Master generates the Board, randomly places Pieces and Players.
- Players are placed in the Goal area of their Team.
- The initial goal of each Team is to bring some exact amount of Pieces to the Goal area and put them in the right places.
- Pieces are dropped every 30 seconds and the previous ones disappear after this time expires if they are not approached by player till that moment.
- Each Player can distinguish the content of 8 neighboring fields, other parts of the Board are hidden.
- Player steps out of the goal area and finds out the Manhattan distance to the nearest Piece.
- Player is not allowed to step to the other players field and go to the Goal area of the opposite Team.
- Upon approach a Piece a Player can choose either to bring the piece back without testing it or check if the piece is good or sham.
- If Player decides to check the piece and it appears to be sham he can leave it on the board, can tell all the team members that this piece is sham, or destroy it.
- If the piece is brought to the Goals area by the Player, he/she needs to place the piece in the right field. If the Player places a piece correctly, he/she will be notified that one of the goals have been completed.
- If the Piece was placed incorrectly, the Player will be notified that they discovered a non-goal.
- If the Piece turns out to be sham, the player gets no information.
- The team that finds the right Pieces and puts in the right fields first wins and the game finishes.
- The game is won by one only one team.
- The game is about completing the goals.

Description of the board and player's states during game




Player game states
Search
do/search for a piece
Found a piece
Piece is not there anymore
Discovery
do/find a piece
Piece has been placed or was sham piece
Piece was picked up
Delivery
do/take piece
do/test piece
do/set piece
Game ends

3.6 Game process

- The server starts loading and if it is in operate condition Players are able to connect.
- Each player has opportunity to choose, whether to join a game or create own lobby, so this Player becomes a host.
- When players are loaded in a lobby, then host can start a game if all Players are ready and there are more then 2 Players in each team. Number of Players in each team has to be equal.
- When host presses a button to start the game Game Master generates the Board, randomly places Pieces and Players and all the players are getting loaded on a play board. Then host can start the timer of 30 seconds before the players can move or this timer starts automatically after all the players are loaded.
- Game starts. In order to win a Team need to bring some exact number of true pieces in the goal area faster then their opponents. The real state of the piece is known only to the Game Master and in order to check it player should bring that piece in the goal area. Then team gets a point if piece appears to be true.
- Players need to put true pieces in the correct places and fill all the gaps of the goal area.
- Pieces are dropped every 30 seconds and they disappear if they were not approached by any Player during some stated period of time.
- Player can see only 8 blocks around, other parts of the Board are hidden from the Player and need to be discovered in order to realise if there is a piece over there. However, player knows the distance to the nearest piece which is written in the blocks surrounding each player. It can only be seen by Player.
- There is no chance for player to step on the field of other Player as to step in the opposite team's Goal Area. It is forbidden.
- Whenever Player approaches Piece there can be two decisions to be made. The first is to take Piece without checking it and bring to the Goal Area. The other is to check the piece where it was found to know if it is shame or not. In the first place Player is risking to bring a sham Piece and spend time meaninglessly, but at the same time if this Piece, actually, appears to be true Player saves time.

- If Player decides to check the Piece and it appears to be sham, then it can be left where it stays. Player can inform other Team members that Piece that was found is sham, so others wouldn't spend time on that Piece.
- When Player brings True piece to the Goal area it is crucial to put in the correct place. If everything done correctly all the Team members get notified that one of the Goals was completed. If the Piece was place incorrectly, the Player will be notified that the placement haven't advanced them closer to reaching the Goal. If the Piece turns out to be sham, the player gets no information.
- Players play till one of the following cases take place. The first one is one team completes the Goal Area and they win a game and it gets successfully finished.
- The second option is that one of the teams surrenders and the other team wins. And the last option is that all the Players disconnect from the server, in this case lobby closes, game process terminates.

3.7 Game Interactions

Players can communicate with each other without any restrictions. Communication is needed for acknowledging the current state of the game. For instance, if one player finds out the field in the Goals area to be wrong, the other player in a team cannot see that in his own profile, so this information should be exchanged by communication server.Communication between Players occur without permission of Game master. Interaction between Players and Game Master are meant to be on request-response basis. So if a Player needs to interact with the Game Master, then it sends the response and Game Master makes up some request on this response.