The Project Game

Kamil Grabowski, Filip Grajek, Bartosz Jasiński, Ivan Rukhavets

Wersja 1.0

January 15, 2017

Wydział Matematyki i Nauk Informacyjnych Politechniki Warszawskiej

Date	**Author**	**Description**	Version
15.10.2016	Kamil Grabowski, Filip Grajek, Bartosz Jasiński, Ivan Rukhavets	Initial document	1.0

Contents

1 Introduction

2 Requirements

2.1 Functional requirements

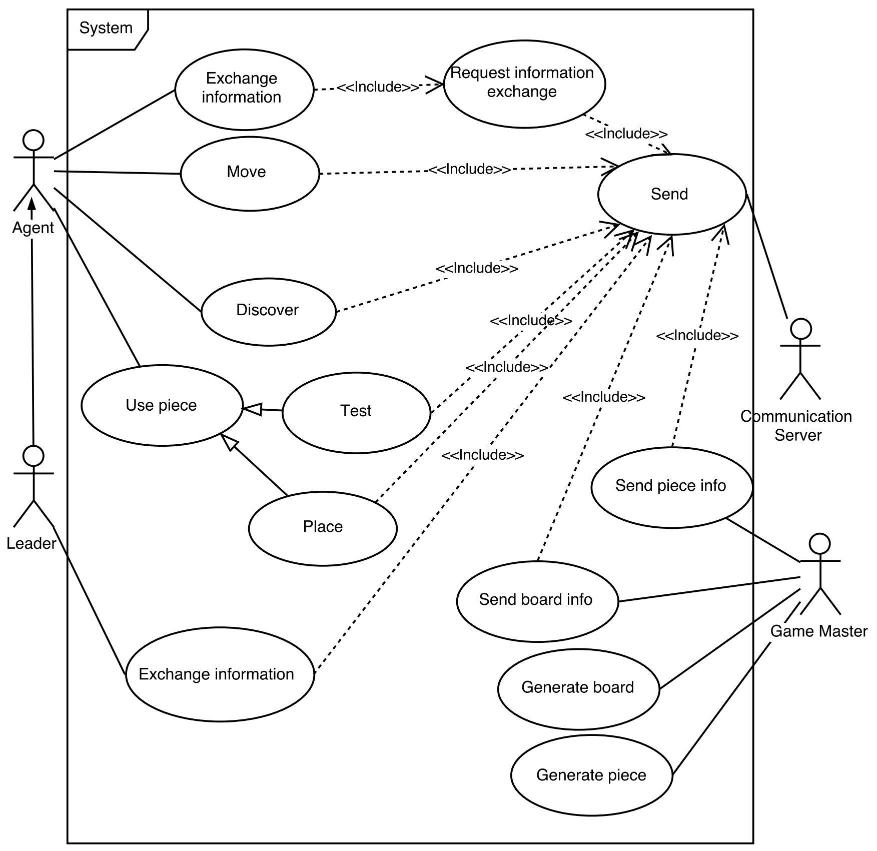

Figure 1: Use-case diagram

In the system there will be four main actors:

1. Game Master
2. Agents

3. Team Leaders
4. Communication Server

The game master is the core of the system. At the start he will generate the board and the first pieces. He is also responsible for managing the board and sending information to the agents during the game, for example he will decide if a move is possible. Because the game master is the only part of the system which knows the exact state of the board, he also has to inform the agents when a team has won.
An agent is an independent program that collaborates with other agents from his team in order to complete the project first. He is able to move on the board, during movement he discovers the state of the board around him. If he walks on a field with a piece, he will collect it and then he can test it or place it in his goal area. All those actions have to be passed to the game master, since he decides the outcome of it. An agent is also able to request an information exchange with other agents. If the request is accepted, both agents share their current board state.
The team leaders are two special agents (one in each team), so they have the same functionality as an agent.Moreover, they can exchange information directly with other agents, without having to ask first.
The agents and the game master are all independent programs, they could be even on different computers. Because of that, we need a communication server. It has only one responsibility: passing messages between all parts of the system.

2.2 Non-functional requirements

Requirement class	Requirement number	Description
Usability	1	A Read-me file will be added with an instruction how to setup the system.
Reliability	2	The system will continue to work properly even when one or more agents disconnect unexpectedly.
Performance	-	There are no requirements in this class.
Supportability	3	The application logic will be tested using unit tests.
	4	Configuration will be possible through a configuration file.

2.3 Runtime parameters

We have decided to make configuration possible through the game master before starting a game. The player can choose a graphical user interface or a configuration file to do this. When making changes in the menu, the file is automatically overwritten (through xml serialization) to save the settings. When starting a new game, the configuration is always parsed from the file. This way the game parameters are always saved and can easily be exported.
The configuration file will be a xml file, as those types of files are readable by humans and programs. Moreover, through the use of xml schemes we can validate such a file without problems.
The parameters specified in the configuration file are:

1. Board size
2. Task area size
3. Minimum number of players per team
4. Maximum number of players per team
5. Ip address and port of the communication server
6. Number of tasks to complete for victory

If the need arises, new parameters can be added to the file without any problems (like backwards compatibility), which is another advantage of using xml.

3 Design documentation

3.1 Architecture

Game Project architecture consists of fourn main building blocks:

- Game Area
- Agents
- Game Master
- Server

which respectively has folowing components
$\cdot GameArea$:

GArea - area consising game targets which agents have cover with pieces in order to win the game.

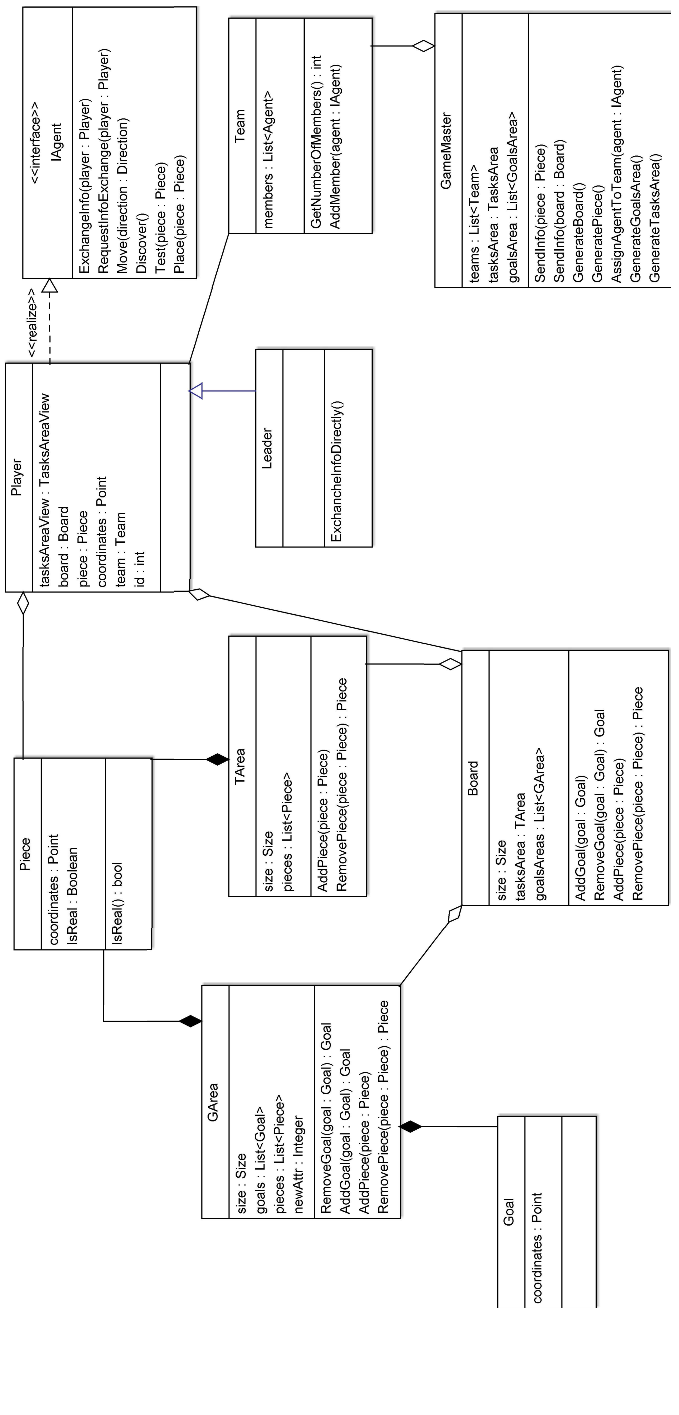

Figure 2: Architecture of system

TArea - area consising pieces which agents raises in order to move them to goal area.

Board - class which represents board consists one TArea and list of GAreas

·*Agents* :

IAgent - interface for agents, it has basic agent methods

Player - class which implements IAgent, represents usual agents in game

Leader - class which has additional methods characteristic for team leader

Team - class which holds team members useful for communication

·*GameMaster* :

Game Master - class which represents object managing the entire game.

·*Server* :

Server - server of the game its only purpose is to exchange messages between Agents and Game Master

3.2 Messages

3.2.1 Game initiation

After connecting to the communication server, the agent gets a list of games he can join. In order to choose a game, he has to send a message, defined by the following schema:

```
<?xml version="1.0"?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://www.mini.pw.edu.pl/~gameProject/"
   elementFormDefault="qualified">
   <xs:element name="gameConnectAttributes">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="gameId" type="xs:integer" />
            <xs:element name="desiredRole">
              <xs:simpleType>
               <xs:restriction base="xs:string">
                 <xs:enumeration value="Agent"/>
                 <xs:enumeration value="Leader"/>
               </xs:restriction>
              </xs:simpleType>
            </xs:element>
            <xs:element name="desiredTeam">
              <xs:simpleType>
               <xs:restriction base="xs:integer">
                 <xs:minInclusive value="0"/>
                 <xs:maxInclusive value="1"/>
               </xs:restriction>
              </xs:simpleType>
            </xs:element>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

Here is a example of such a message, an agent wants to join the first team in the game with id zero:

```
<?xml version="1.0" encoding="UTF-8"?>
<gameConnectAttributes
   xmlns="http://www.mini.pw.edu.pl/~gameProject/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.mini.pw.edu.pl/~gameProject/
       gameConnectAttributes.xsd">
   <desiredRole>Agent</desiredRole>
   <desiredTeam>0</desiredTeam>
</gameConnectAttributes>
```

3.2.2 Gameplay

Before making every move, agent sends a message with its description to the server. The message is defined by following schema:

```
<?xml version="1.0"?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://www.mini.pw.edu.pl/~gameProject/"
   elementFormDefault="qualified">
   <xs:element name="makeMoveAttributes">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="agentId" type="xs:integer"/>
            <xs:choice>
               <xs:element name="direction">
                 <xs:simpleType>
                  <xs:restriction base="xs:string">
                    <xs:enumeration value="Up"/>
                    <xs:enumeration value="Right"/>
                    <xs:enumeration value="Down"/>
                    <xs:enumeration value="Left"/>
                  </xs:restriction>
                 </xs:simpleType>
               </xs:element>
               <xs:element name="test"></xs:element>
               <xs:element name="place"></xs:element>
               <xs:element name="requestInfoFromAgent">
                  <xs:simpleType>
                     <xs:restriction
                        base="xs:integer"></xs:restriction>
                  </xs:simpleType>
               </xs:element>
            </xs:choice>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

Here is an example of moving up of agent with id 10:

```
<?xml version="1.0" encoding="UTF-8"?>
<moveAttributes
   xmlns="http://www.mini.pw.edu.pl/~gameProject/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.mini.pw.edu.pl/~gameProject/
      move.xsd">
   <gameId>0</gameId>
   <agentId>10<agentId/>
   <direction>Up</direction>
</moveAttributes>
```

And here is a message sent by agent with id 007 when it wants to request information with agent 10:

```
<?xml version="1.0" encoding="UTF-8"?>
<moveAttributes
   xmlns="http://www.mini.pw.edu.pl/~gameProject/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.mini.pw.edu.pl/~gameProject/
      move.xsd">
   <agentId>007<agentId/>
   <requestInfoFromAgent>10</requestInfoFromAgent>
</moveAttributes>
```

As an answer agent gets message from server described with that schema:

```xml
<?xml version="1.0"?>

<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.mini.pw.edu.pl/~gameProject/"
  elementFormDefault="qualified">


  <xs:element name="response">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="status">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="Allow" />
              <xs:enumeration value="Deny" />
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
        <xs:element name="info" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="field" maxOccurs="unbounded">
                <xs:complexType >
                  <xs:simpleContent>
                    <xs:extension base="xs:integer">
                      <xs:attribute name="x" type="xs:integer"
                          use="required" />
                      <xs:attribute name="y" type="xs:integer"
                          use="required" />
                    </xs:extension>
                  </xs:simpleContent>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

It tells if it allows for move and optionaly sends information from another agent with distance from each known field to a piece. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<responce
   xmlns="http://www.mini.pw.edu.pl/~gameProject/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.mini.pw.edu.pl/~gameProject/
       serverResponceOnMove.xsd">
   <status>Allow</status>
   <info>
      <field x="0" y="0">1</field>
      <field x="0" y="1">0</field>
      <field x="1" y="0">1</field>
   </info>
</responce>
```

After the game is finished, server sends information to all agents about game results with following schema(example attached).

```
<?xml version="1.0"?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://www.mini.pw.edu.pl/~gameProject/"
   elementFormDefault="qualified">
   <xs:element name="gameResult">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="gameId" type="xs:integer" />
            <xs:element name="winningTeam">
              <xs:simpleType>
               <xs:restriction base="xs:integer">
                 <xs:minInclusive value="0"/>
                 <xs:maxInclusive value="1"/>
               </xs:restriction>
              </xs:simpleType>
            </xs:element>
            <xs:element name="team0Score" type="xs:integer"/>
            <xs:element name="team1Score" type="xs:integer"/>


         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<gameResults
```

```
   xmlns="http://www.mini.pw.edu.pl/~gameProject/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.mini.pw.edu.pl/~gameProject/
       gameEnd.xsd">
     <gameId>12</gameId>
     <winningTeam>0</winningTeam>
     <team0Score>100</team0Score>
     <team1Score>99</team1Score>
</gameResults>
```

3.3 Communication protocol

The system will use the TCP protocol to communicate. We have chosen it over UDP, because information sent by the server should always reach the agents.
The messages will be encoded in the XML format and verified through XML schemes. The message parsing system will be written in such a way, that changing to a different format will not be a problem in the future if such a need arises (the parsing will be done by a class which implements a general interface for parsing).

3.4 Special system states

3.4.1 Loss of communication with server - Agent side

When agent loses connection with server we basicly do one thing- on agent's side we try to recconect to server for specified amount of attempts and agent does not preform any moves in game. If it fails, agent recognize that it is not possible to connect and stops atempts of recconection. On the other hand if agent recconects, game restarts from point just befor loss of connection. (czy mozemy ponownie sprobować sie laczyc jesli np. postawimy nowy serwer?)

3.4.2 Loss of communication with server - Game Master side

On the Game Master side we have simmilar behaviour, GM does not perform any actions related to game and tries to recconect. If GM can not reconnect after specified number of attempts Game Master stops its attempts of reconnection. If reconnection is succesful we continue game from point before loss of connection.

3.5 Game related objects

3.6 Game rules

The game is played on a NxM board (size is set in configuration file). The true state of the fields is only known to the Game Master.

The game starts with all agents placed in their team's goals area. The team's goal is to place more pieces in their area than the other team. The game is played in real-time i.e. all agents may move independentely from others.
Possible moves of the Player consist of:

- moving in one of 4 directions
- discovering the contents of 8 neighbouring fields
- testing the picked piece for being a sham
- placing a piece in the goals, in hope of completing one of the project objectives
- request exchange of information with another player

The Agent can learn about fields states only by placing (using) a piece in a given field of the goal area:

- A correctly placed piece results in an information to the Agent that one of the goals of the project have been completed.
- Incorrectly placed piece results in an information that the completed action (getting the piece from the tasks board and placing it in the goals area) has been meaningless, in the sense of project completion.
- Placing a piece which is a sham results in getting no information.

Figure 3 presents a state diagram from the point of view of an agent.
Agent actions have following ramifications and constrains:

- The piece is picked up by a Agent which moves into the piece's field first.
- Agent cannot move into a field occupied by another player.
- Observing a field (either by discovering or entering it) results in receiving information about the Manhattan distance to the nearest piece.

The Game Master may place a new piece to any field at any moment. The game finishes when one all pieces were collected by teams. The team which has collected more pieces(not sham) wins.

3.7 Game interactions

Each agent before moving sends information about move to the Game Master. The GM if the move is acceptable (no other agents on field, etc.) tells the agent that it's OK and updates game info. Else it sends deny to the agent.
The agent may request information from another agent. If it succeeds, it receives all info known to that agent.
Figure 4 presents activity diagram for an agent and Game Master.

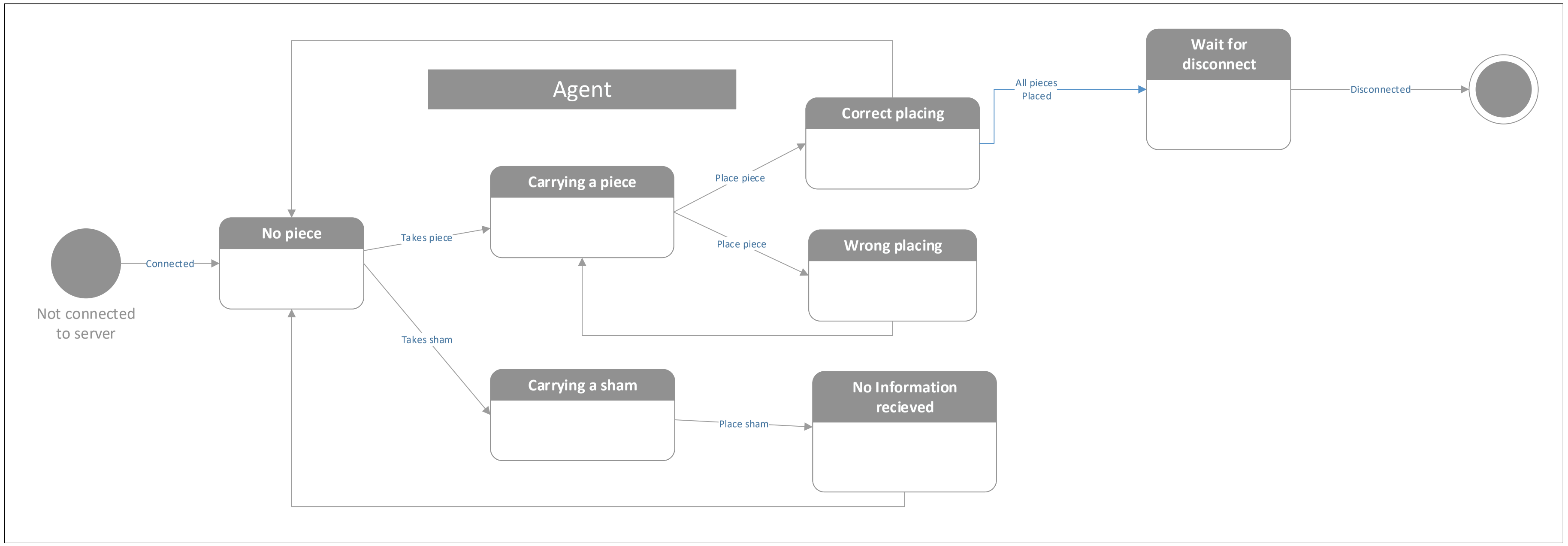

Figure 3: State diagram of agents

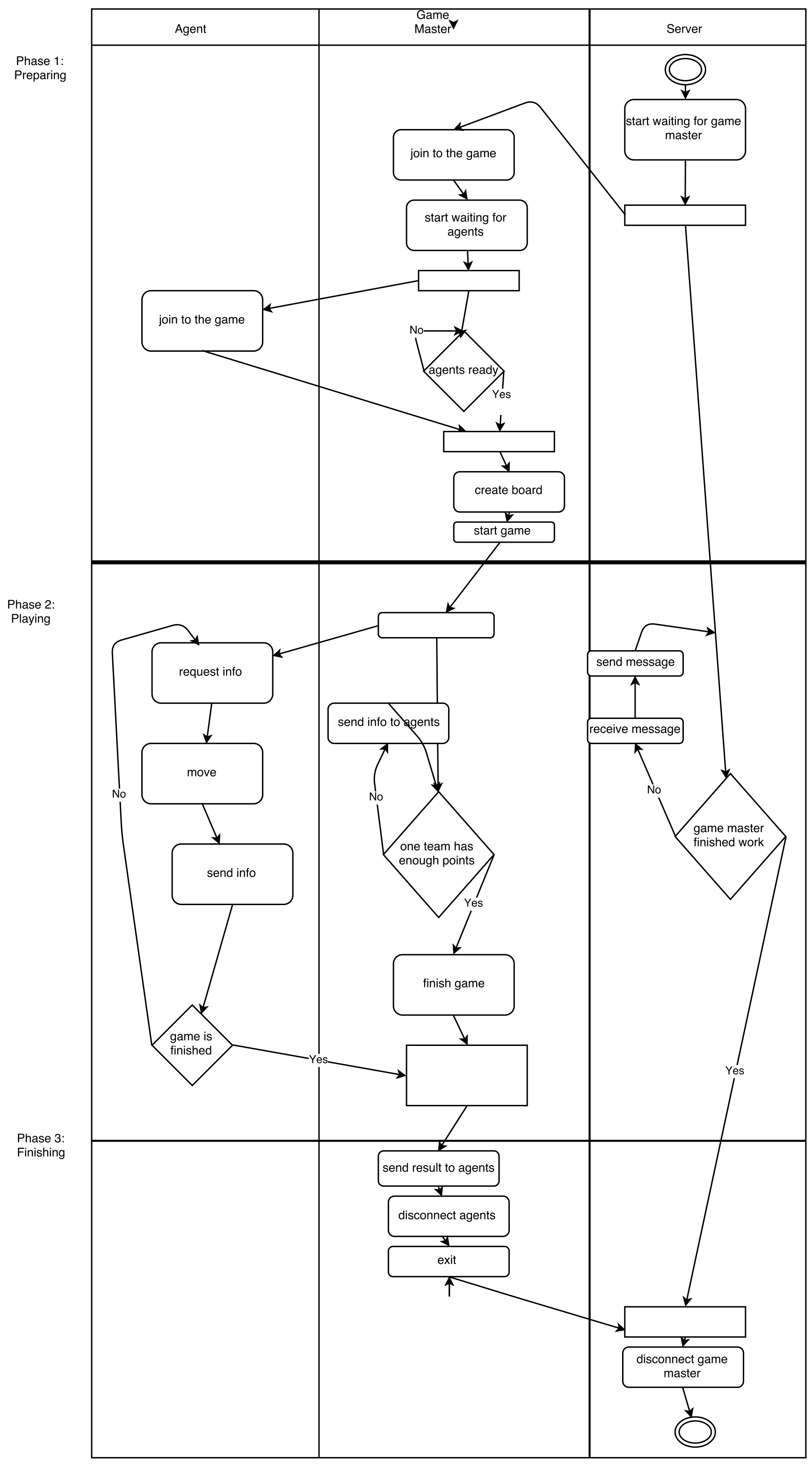

Figure 4: Activity Diagram

3.8 Configuration

3.9 Ways of handling basic program errors

3.9.1 Loss of connection

Loss of connection with Agent

Agent side
When Agent loss connection with Game Master he does not perform moves in game and tries to reconnect with server.
Game Master side
When Game Master recognizes that he can not comunicate with Agent, he only preserves Agent state, from other agent's point of view it looks like disconnected Agent does not perform any moves.

Loss of connection with Game Master

Agents side
Agents no longer make moves and they only pings Game Master to check if he reconnected, after succesful reconnection they resume game.
Game Master side
Game Master tries to reconnect to Server, after specified number of atempts to reconnect GM should give up.

Loss of connection with Server

Agents side
They stop making their moves and start reconnection atempts.
Game Master side
The same on Game Master side, GM preserves game state and tries to reconnect to server.

4 Glossary