Contents

I. INTRODUCTION 3
1.1. Contexte : 4
1.2. Objectifs : 4
1.3. Problématiques : 5
1.4. Mots techniques : 5
1.5. Le jeu d'échecs 5
II. ANALYSES 5
2.1. Analyse des besoins 5
2.2. Diagramme de cas d'utilisation 6
2.3. Description: 7
2.4. Analyse du sujet 9
2.5. Architecture 10
III. CONCEPTION 11
3.1. Le modèle de rôle 13
3.2. Le modèle d'interaction 14
3.3. Modèle d'agent 15
3.4. Modèle de service 16
3.5. Le modèle d'organisation ou d'accointances 17
3.6. Le modèle environnemental 18
3.7. Modèle de protocoles et rôles détaillées 18
3.8. Modèle de classes 20
3.9. Diagrammes de sequences 21
IV. IMPLEMENTATIONS 21
4.1. Environnements de travail 21
4.2. Environnement 22
4.3. Outils 22
4.4. Présentation de notre application 22
4.5. Difficultés rencontrées 27
V. CONCLUSION 28

Liste des figures

Figure 1 : Table d'échec – Echec et Mate – Stale Mate 4
Figure 2 : Cas un joueur physique et un logiciel 6
Figure 3 : Cas de deux joueurs physiques 7
Figure 4 : Cas de deux joueurs logiciels 7
Figure 5 : Cas de deux utilisateurs physiques 8
Figure 6 : Architecture du model 11
Figure 7 : Le processus de développement de Gaia et les modèles manipulés. 12
Figure 8 : Modele d'interaction 15
Figure 9 : Modèle d'agent 16
Figure 10 : Interactions entre rôles avec les protocoles 17
Figure 11 : Protocoles (Table-Player) et rôles détaillées 18
Figure 12 : Protocoles (Player-Player) et roles détaillées 19
Figure 13 : Diagrammes de classes 20
Figure 14 : Diagramme de sequence 21
Figure 15 : Illustration alpha-beta pruning (élegage AlphaBeta) 23
Figure 16 : Interface du jeu 25
Figure 17 : Interface de JADE 26
Figure 18 : Interface de la Table 26
Figure 19 : Interface du joueur 27

Liste des tableaux

Tableau 1 : Description des acteurs du systèmes 8
Tableau 2 : Les agents et leurs comportements 9
Tableau 3 : Modèle de rôles Table 13
Tableau 4 : Modèle de rôle Joueur 14
Tableau 5 : Diagramme d'intéracition 15
Tableau 6 : Interactions 15
Tableau 7 : Modèle de services 16

RAPPORT DE PROJET SMA-IA

Enseignant : NGUYEN Manh Hung

Par : Groupe 3

Mohamadou Aminou OUMAROU ALTINE
HAMIDULLAH Yasser
SONFACK SOUNCHIO Serge
DAOUDA saidi kadri

Sujet : IMPLEMENTATION D'UN JEU D'ECHEC AVEC JADE

I. INTRODUCTION

Ces dernières années, la programmation a beaucoup évolué. Alors qu'il n'y a pas si longtemps on ne parlait que de la programmation procédurale. En termes de logiciels, les besoins changent rapidement et ces derniers sont de plus en plus présents dans la vie quotidienne. Comme les logiciels deviennent de plus complexes, l'effort de conception de ceux-ci est en croissance constamment. Les techniques de programmation devaient donc s'adapter, ce qui a mené à la programmation orientée-objet. La programmation OO et ses techniques de modélisation, comme OMT (Object Modeling Technique), permettent d'enrichir la programmation procédurale en y ajoutant des concepts essentiels comme l'encapsulation, l'abstraction et le polymorphisme.

Depuis quelques temps, une évolution de celle-ci s'est mise en place : la programmation OA (Orienté-Agent). En effet, de nous jours, la POO n'est pas toujours adaptée aux besoins des applications. Les systèmes sont souvent distribués sur différentes machines. Ils s'exécutent indépendamment les uns des autres et aussi, ils doivent communiquer entre eux. Souvent, ils sont divisés en sous-systèmes indépendants et chacun d'eux exécutent une partie du travail dans un but commun. La POA propose une façon beaucoup plus naturelle de concevoir ce genre de système.

Dans ce contexte se concentre ce travail pratique de conception d'un système à base d'agent pour une initiation avec un cas plus concret utilisant les techniques avancées.

Ce rapport est le compte-rendu du notre projet intitulé « Conception d'un système multi-agents d'un jeu d'échec » en utilisant le la plate-forme JADE.

1.1. Contexte :

Le jeu d'échec fait partie des jeux qui nécessitent beaucoup de réflexion et de stratégie avancées car beaucoup de choses à prendre en compte. Le jeu consiste à faire face deux joueurs sur une table à 64 carreaux dont 32 claires et 32 sombres. Chaque joueur possède 16 pièces dont il y a les pièces maîtresses et les pions. Le joueur déplace des pièces pour attaquer et faire en sorte que le Roi adverse soit en position qu'on appelle « échec et mate » où il est encerclé est ne peut plus bouger ce qui définit le gagnant. Mais le jeu peut se terminer d'une autre façon à ce que les deux joueurs terminent avec un match nul : le stale mate (pas de déplacement possible mais pas mate), les mouvements trop répétitifs qui n'aboutit à rien et la résiliation (capitulation d'un joueur).

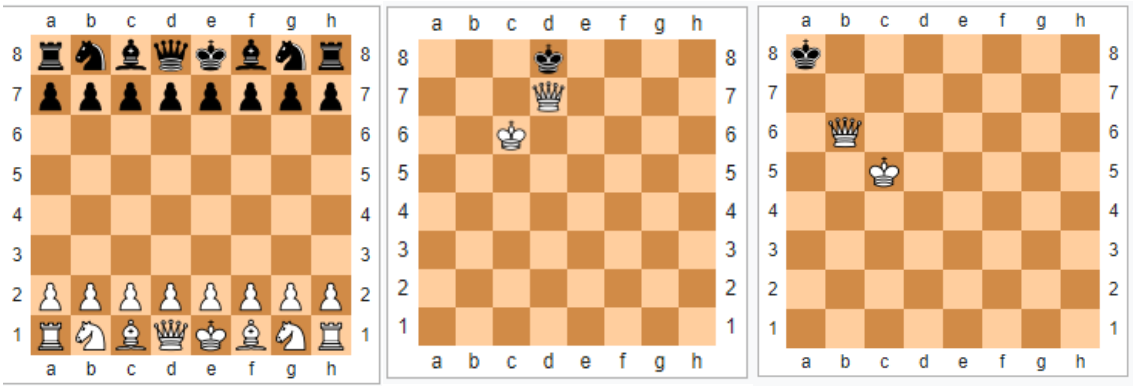

Figure 1 : Table d'échec – Echec et Mate – Stale Mate

1.2. Objectifs :

Dans ce travail pratique l'objectifs est de concevoir un système à base d'agents dans lequel des agents autonomes se communiquent entre eux et commencer à jouer le jeu d'échec. Comme dans la vie réelle les gens cherchent à jouer le jeu d'échec avec d'autres personnes et il y a aussi règlements. Le respect de ses règlements est assuré

par l'arbitre, nous voulons mettre en place un système de jeu d'échecs avec des agents autonomes dont il y a les joueurs et les tables d'échecs.

1.3. Problématiques :

Quels sont les agents à prendre en compte ? Quelles intelligences doivent avoir chacun des agents ? Avec quels moyens peut-on les doter des capacités de communiquer, de jouer, concevoir des stratégies ? Avant tout, comment modéliser ce système et implémenter avec les outils dont on a à la disposition ?

1.4. Mots techniques :

Intelligence artificielle, agent, POO, POA, Système distribués, JADE, Java, MinMax, AlphaBeta MinMax, Théories des jeux, zero sum games, Algorithme de recherche heuristique.

1.5. Le jeu d'échecs

Dans un jeu d'échec il y a au moins : 2 joueurs et une table d'échecs, pas obligatoirement mais il est nécessaire aussi d'avoir des chronomètres. Il y a un format de notation propre aux échecs utilisé pour exprimer les déplacement FEN notation.

II. ANALYSES

2.1. Analyse des besoins

2.1.1 Spécification des exigences

C'est une partie très importante du projet et est le point de départ de tout projets informatiques. Elle nous permet de déterminer les exigence fonctionnelles et non fonctionnelles de notre application. En outre c'est un résumé entre les utilisateurs et les clients avec l'équipe de développement. On définit alors les exigences fonctionnelles comme les fonctionnalités que le système doit réaliser.

En effet nous allons expliquer ce cas à l'aide des diagrammes de cas d'utilisation.

2.2. Diagramme de cas d'utilisation

Ces diagrammes des cas d'utilisation permettent de déterminer les exigences des utilisateurs pour l'application. En d'autres termes, les cas d'utilisation énumèrent toutes les fonctionnalités futures de l'application. L'ensemble des cas d'utilisation spécifie les fonctionnalités complète du système.

- Cas d'un joueur physique et un logiciel
- Cas d'utilisation deux joueurs physiques
- Cas d'utilisation deux joueurs logiciels

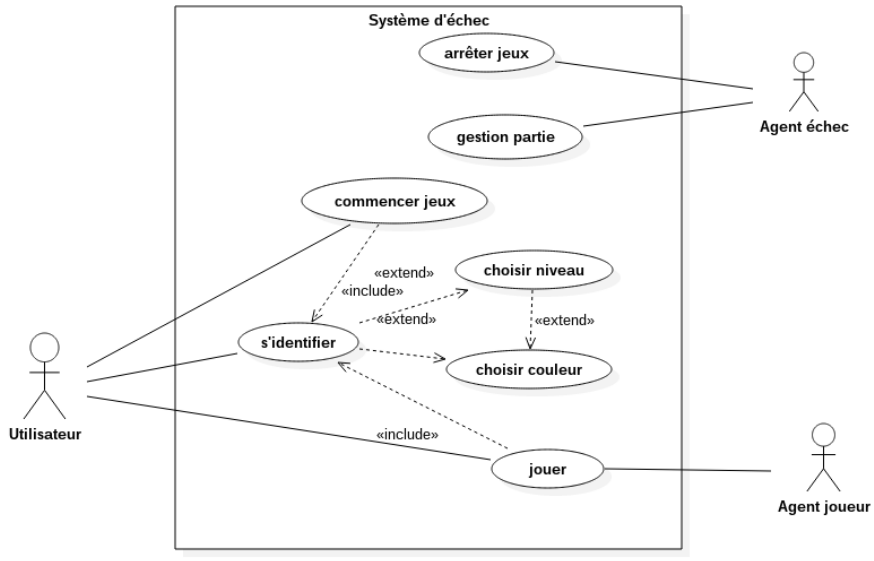

Figure 2 : Cas un joueur physique et un logiciel

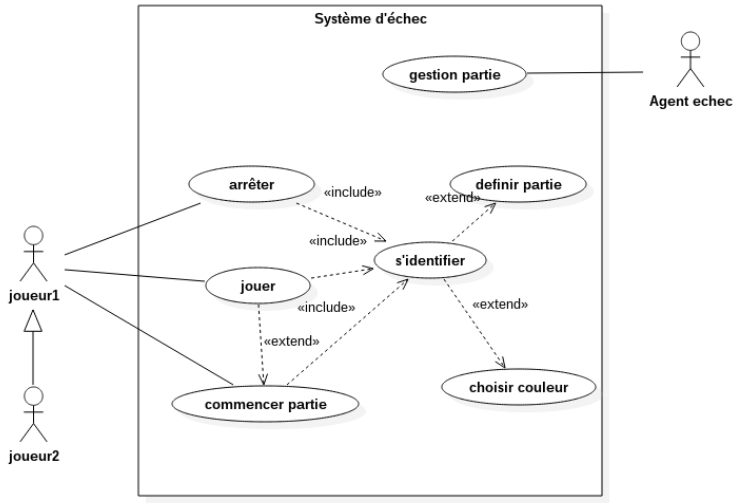

Figure 3 : Cas de deux joueurs physiques

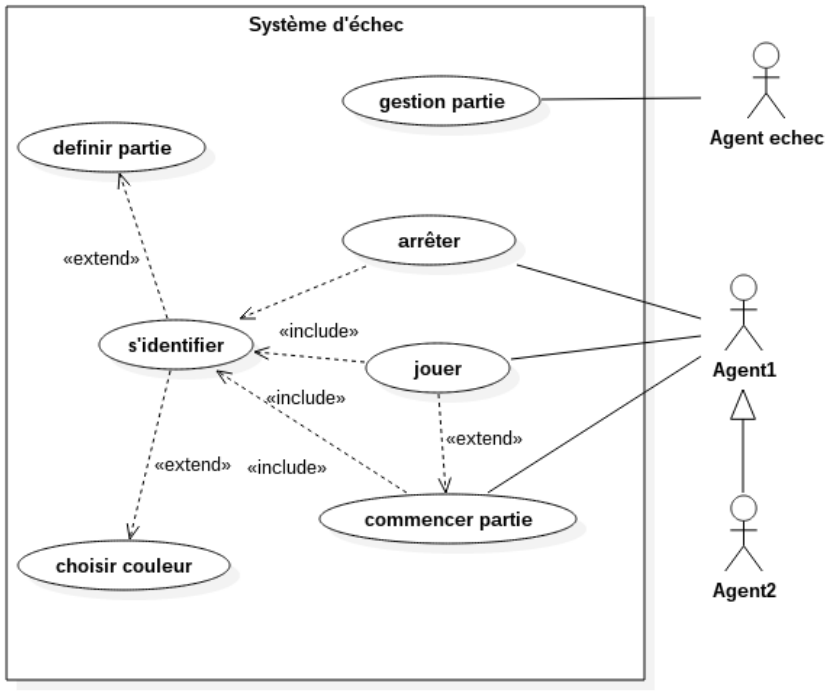

Figure 4 : Cas de deux joueurs logiciels

2.3. Description:

Voyons maintenant la description des cas d'uitilisations.

Tableau 1 : Description des acteurs du systèmes

Acteur	Description des rôles
Utilisateur	représente la personne physique qui interagit avec le système. Il doit avec toute partie, — s'identifier — choisir sa couleur de jeu — choisir le niveau qu'il souhaite jouer — commence le jeu
Agent échec	Il est comme l'arbitre de la partie arrête le jeux dès qu'il y'a un vainqueur, donne la main à tour de rôle à chaque joueur et diffinit la partie de jeux dans le cas ou l'utilisateur n'a rien définis
Agent joueur	Son rôle principal est de joueur avec l'utilisateur dans le but de gagner

TABLE 1 – Description des différents acteur du système

Acteur	Description des rôles
Deux uilisateurs	représentent les deux personnes physique qui interagit avec le système. Ils doivent pour jouer une partie : — s'identifier — choisir chacun sa couleur de jeu — commence le jeu — définir la partie de jeu
Agent échec	Il est comme l'arbitre de la partie arrête le jeux dès qu'il y'a un vainqueur, donne la main à tour de rôle à chaque joueur

TABLE 2 – Description des différents acteur du système

Figure 5 : Cas de deux utilisateurs physiques

Acteur	Description des rôles
Utilisateur	représente la personne physique qui interagit avec le système. Il doit avec toute partie, — s'identifier — choisir sa couleur de jeu — choisir le niveau qu'il souhaite jouer — commence le jeu
Agent échec	Il est comme l'arbitre de la partie arrête le jeux dès qu'il y'a un vainqueur, donne la main à tour de rôle à chaque joueur et diffinit la partie de jeux dans le cas ou l'utilisateur n'a rien définis
Agent joueur	Son rôle principal est de joueur avec l'utilisateur dans le but de gagner

TABLE 1 – Description des différents acteur du système

2.4. Analyse du sujet

Cette partie consiste à identifier à l'identification des agents avec ses comportements et les interactions qu'ils effectuent avec les autres agents et environnement.

Tableau 2 : Les agents et leurs comportements

Type d'agent	Nom	Attributs	Capabilités
Principal	PlayerAgent	disponibilité, couleur, table, panel, type	-Interagir avec les autres joueurs et tables. -Mettre en place une stratégie pour les déplacements.
	TableAgent	disponibilité, table, panel, echequier, listJoueurs	-Interagir avec les joueurs

			-Lancer, maintenir, arbitrer le jeu. -Exécuter les déplacements

2.5. Architecture

Pour avoir en vue global l'architecture de notre model, Voir Figure 2, dans laquelle on a les 4 couches suivantes :

Layer 4 : Interface graphique

Layer 3 : Agents et leurs environnements

Layer 2 : Environnement de développement des agents (JADE)

Layer 1 : Environnement JAVA (JVM)

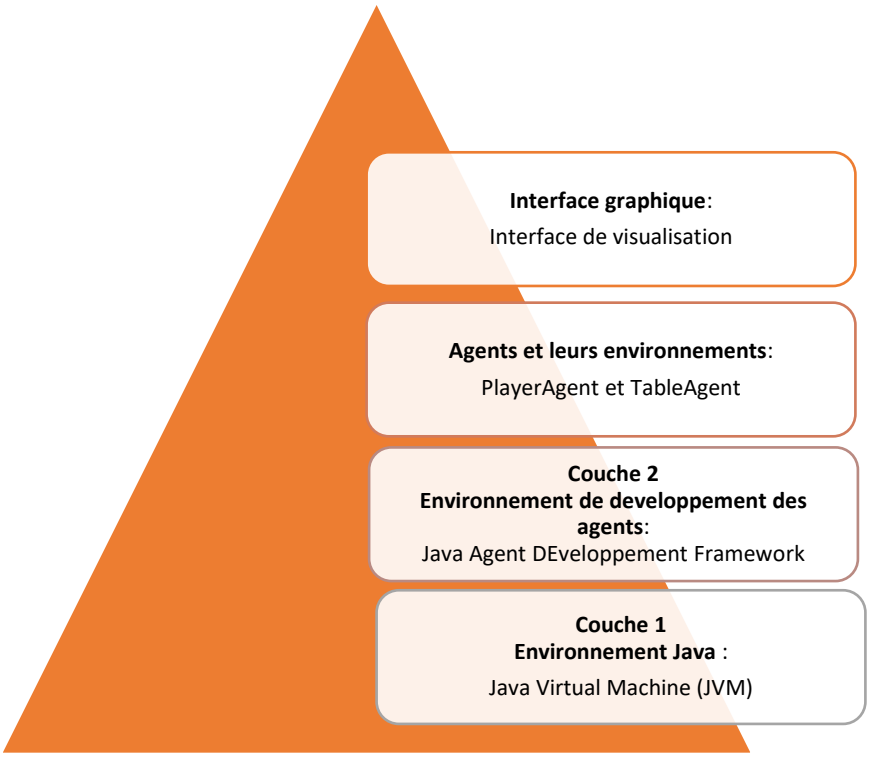

Figure 6 : Architecture du model

III. CONCEPTION

Dans ce travail, on a utilisé la méthodologie GAIA, avec cette approche, on catégoriser les phases de conception comme suit :

- Analyse
- Conception architecturale
- Conception détaillée

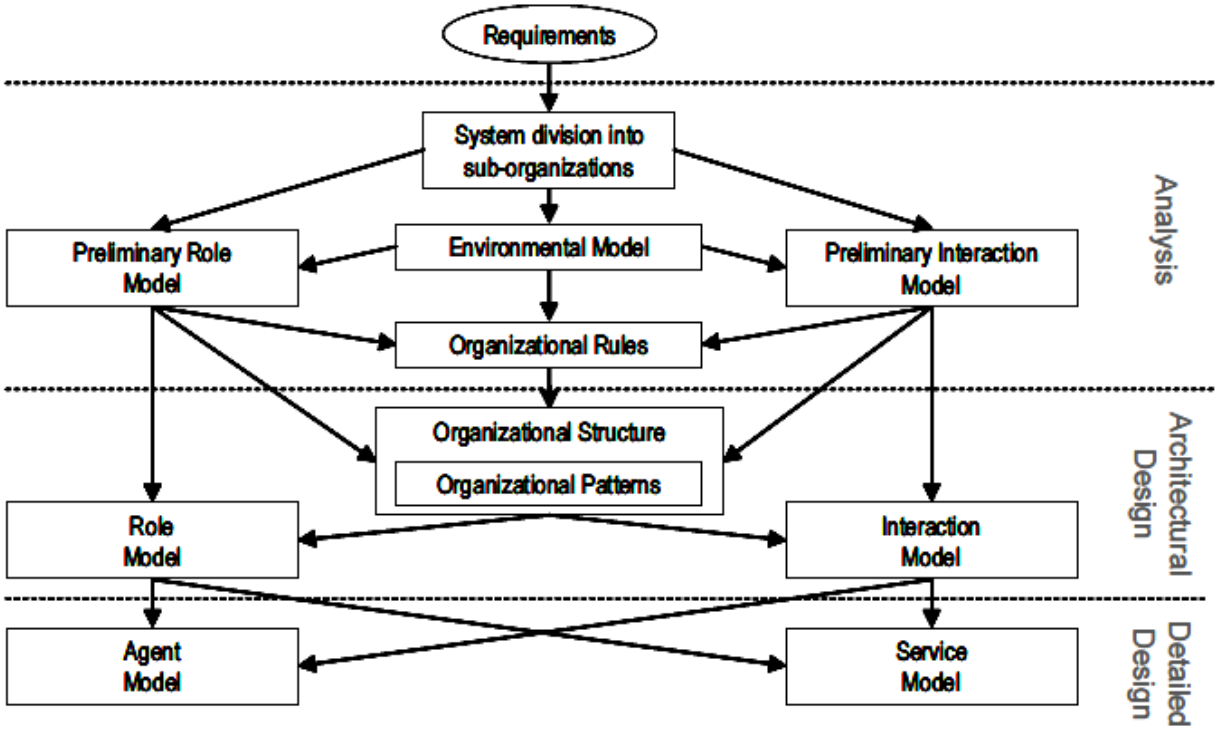

Figure 7 : Le processus de développement de Gaia et les modèles manipulés.

Gaia est une extension des approches d'ingénierie logicielle classique [Wooldridge *et al.*,2000]. C'est une méthode de la seconde génération, par opposition à la première génération que formaient AAII ou DESIRE. Elle est donc plus complète et bénéficie, de plus, d'une large reconnaissance dans le domaine multi-agent. Gaia manipule six modèles d'analyse et de conception différents.

ANALYSE

Cette phase d'analyse consiste à élaborer des modèles préliminaires.

CONCEPTION ARCHITECTURALE

La conception architecturale ou organisationnel, est l'étape qui suit l'analyse dans le processus de développement GAIA dans laquelle on combine les deux modèles de rôle et d'interaction pour avoir une structure organisationnelle.

CONCEPTION DETAILLEE

La conception détaillée est la dernière phase de la conception GAIA, qui permet de finaliser nos modèles puis avoir une vue rapide et globale du système à développer.

3.1. Le modèle de rôle

Qui identifie les différents rôles devant être joués par les agents du système.
C'est une description abstraite de la fonction attendue pour une entité donnée. Outre sa description textuelle, un rôle possède trois attributs – les *responsabilités*, les p*ermissions/- droits* et les *protocoles* de communication disponibles pour le rôle – qui sont regroupés dans un schéma de rôle. Les permissions établissent les ressources auxquelles un rôle a le droit d'accéder. Les responsabilités déterminent la fonctionnalité du rôle en termes de sûreté (*safety*) et de vivacité (*liveness*).
Pour notre cas, le modèle de rôles comporte deux rôles :

Tableau 3 : Modèle de rôles Table

Rôles	Table (Echiquier) ou Arbitre
Description	Gestion du jeu en générale
Protocoles et activités	-INITIATE_GAME_BOARD -BOOK_TABLE -SEND_MAKE_MOVE -TURN_MOVE -END_OF_GAME
Permissions	-Lire : message -Changer : les données du jeu
Responsabilités	
Vivacité---------->	-GAME_INIT -ON_GAME -END_GAME
Sureté---------->	Bonne fonctionnement du jeu.

Tableau 4 : Modèle de rôle Joueur

Rôles	Joueur
Description	Dicte le déplacement des pièces à la table.
Protocoles et activités	-ASK_AVAILABILITY -ASK_BOOK_TABLE -ASK_AID -SEND_MOVE
Permissions	-Lire : message -Changer : les données de déplacement des pièces.
Responsabilités	
Vivacité---------->	-CHECK_AVAILABILITY -MOVE_PIECE
Sureté---------->	Bonne fonctionnement du jeu.

3.2. Le modèle d'interaction

Qui définit les protocoles de communication entre agents. Dans Gaia un protocole est défini comme par nom, un expéditeur, un récepteur, une description, des entrées et des sorties. Les protocoles définissent les dépendances et les relations entre rôles ;

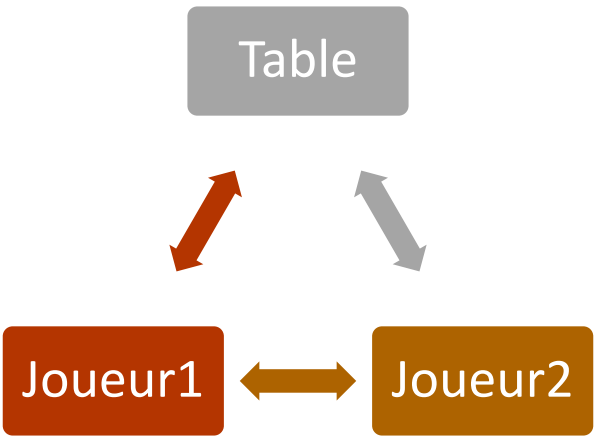

Figure 8 : Modele d'interaction

Tableau 5 : Diagramme d'intéracition

Tableau 6 : Interactions

	Table	Joueur
Table	-	Par message ACL
Joueur	Par message ACL	Par message ACL

3.3. Modèle d'agent

C'est dans le modèle d'agent qu'on attribue les rôles aux différents agents du système.

Il identifie les différents types d'agents et leurs instances

Nous avons deux (2) types d'agents :

L'agent joueur :

Qui a pour but de jouer une partie avec un autre agent (logiciel ou utilisateur) afin d'y gagner.

L'agent table (échec) :

Cet agent a pour fonction d'arbitre, gère la main à tour de rôle entre les joueurs

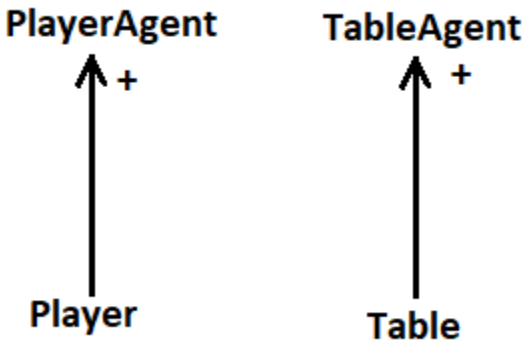

Figure 9 : Modèle d'agent

3.4. Modèle de service

Le modèle de service qui définit, comme son nom l'indique, les différents *services* offerts par le système, et les agents tributaires. Un service est vu comme une fonction d'un agent et possède les caractéristiques suivantes : des entrées, des sorties, des préconditions d'exécution et des postconditions.

Tableau 7 : Modèle de services

Services	Entrées	Sorties	Préconditions	Postconditions
TableAvailability	Informations du joueur	Disponibilité de la table	Il n'y a pas encore de jeu encours	nb_joueur<2
TableBooking	Informations du joueur	Résultat de réservation de table	disponibilité de la table	joueur joignable
TableOnGame	Informations des deux joueurs	Table initial	Les couleurs soient attribuées	Les deux joueurs sont joignables
TableOnGameMove	Informations du joueurs	Nouvel état de la table	Le déplacement soit légal	Le déplacement ne retient ou

				ne met pas le joueur en mate
PlayerPlayingAvailability	Informations du joueur demandeur	Disponibilité du joueur	Il a déjà réservé une table	Il n'a pas encore un jeu encore

3.5. Le modèle d'organisation ou d'accointances

Le modèle d'organisation ou d'accointances qui définit la structure de l'*organisation* grâce à des graphes orientés représentant les voies de communication entre agents. Différent type de relations existent : contrôle, pair (*peer*), dépendance, et d'autres pouvant être définis.

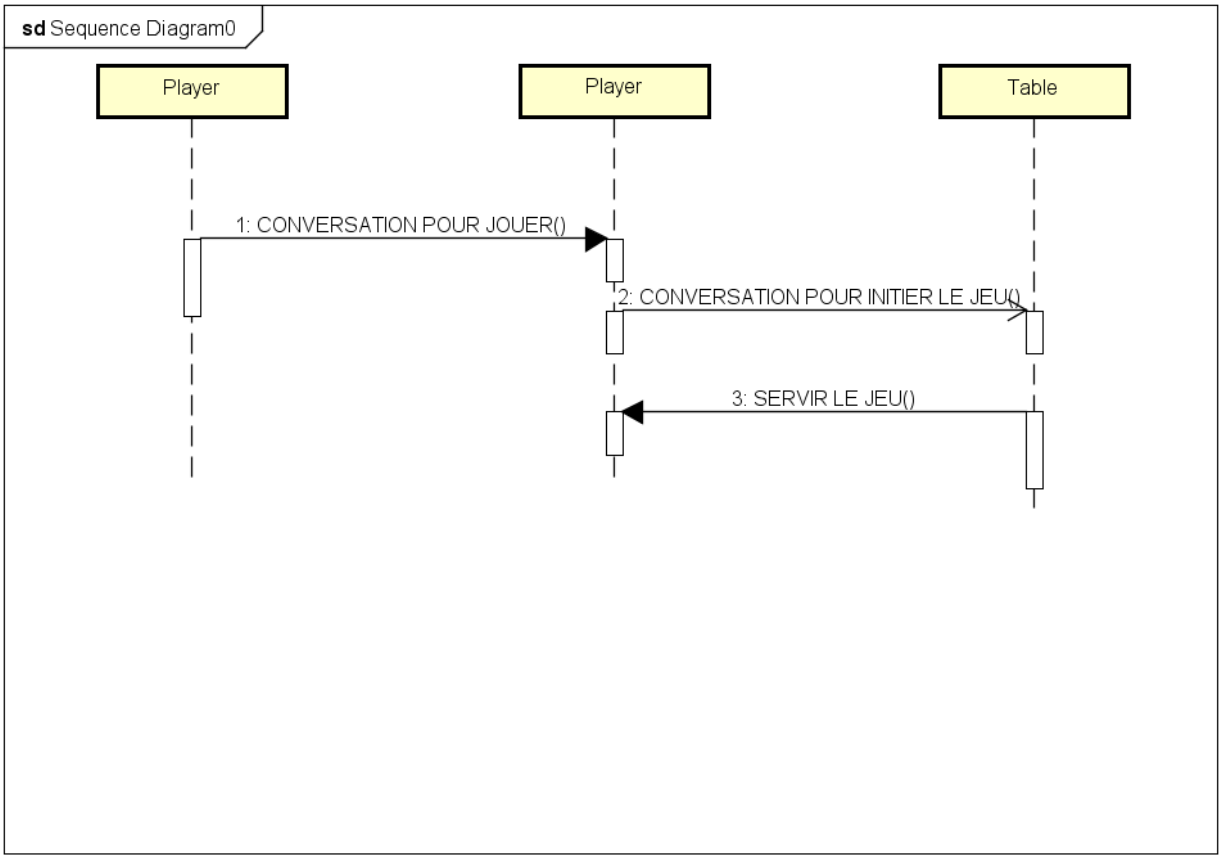

Figure 10 : Interactions entre rôles avec les protocoles

3.6. Le modèle environnemental

Le modèle environnemental qui décrit les différentes ressources accessibles caractérisées par les types d'actions que les agents peuvent entreprendre pour les modifier.

Agents	Player	Table
Action	Enregistrer et désenregistrer service	Enregistrer et désenregistrer service
Ressources	DF (Directory Facilitator)	DF (Directory Facilitator)

3.7. Modèle de protocoles et rôles détaillées

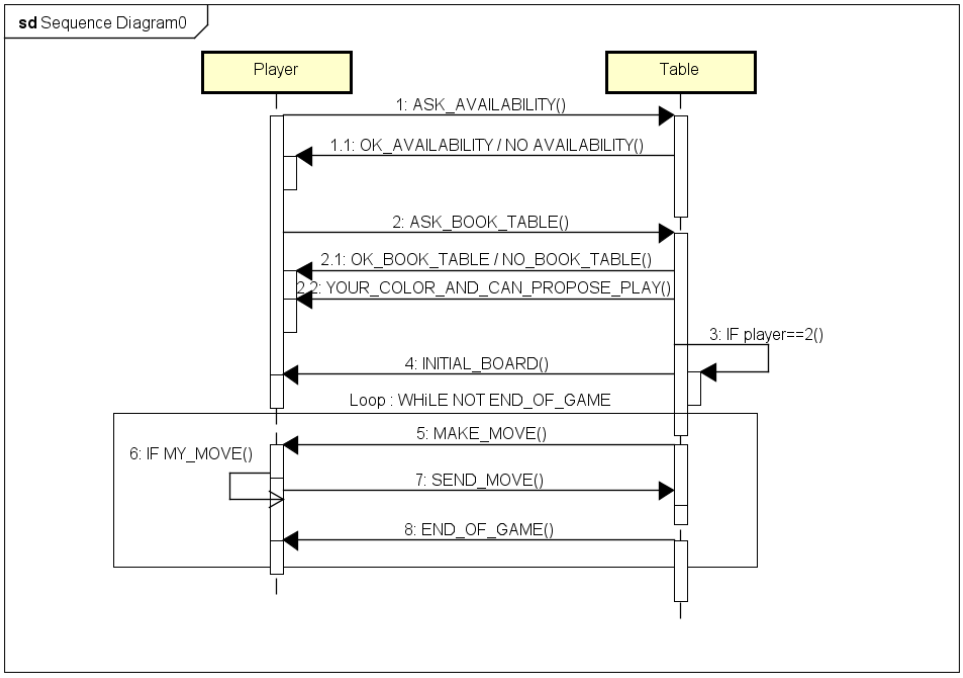

Figure 11 : Protocoles (Table-Player) et rôles détaillées

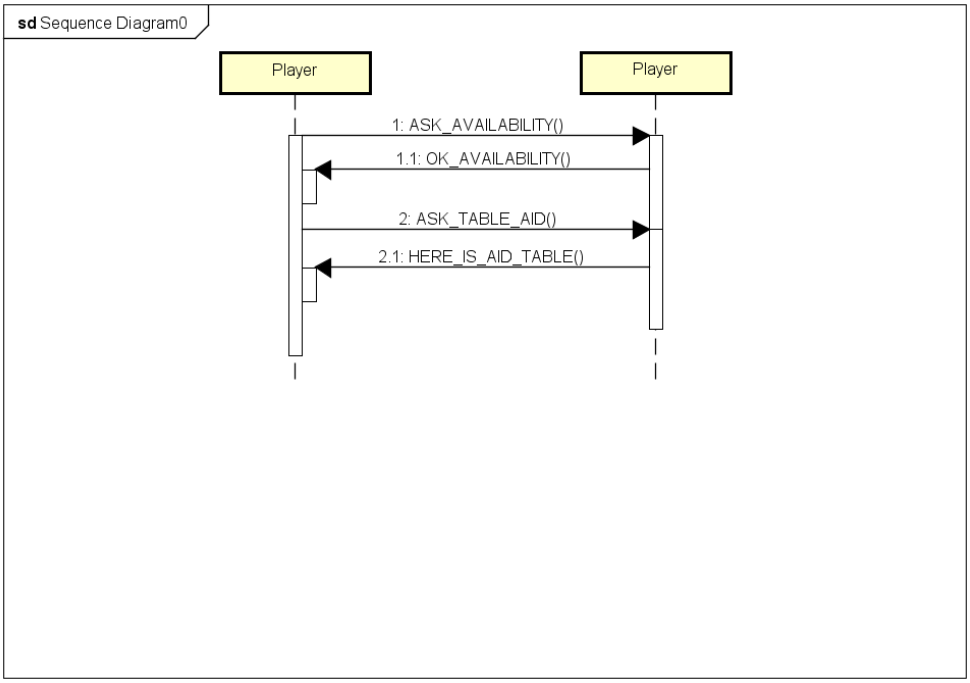

Figure 12 : Protocoles (Player-Player) et roles détaillées

3.8. Modèle de classes

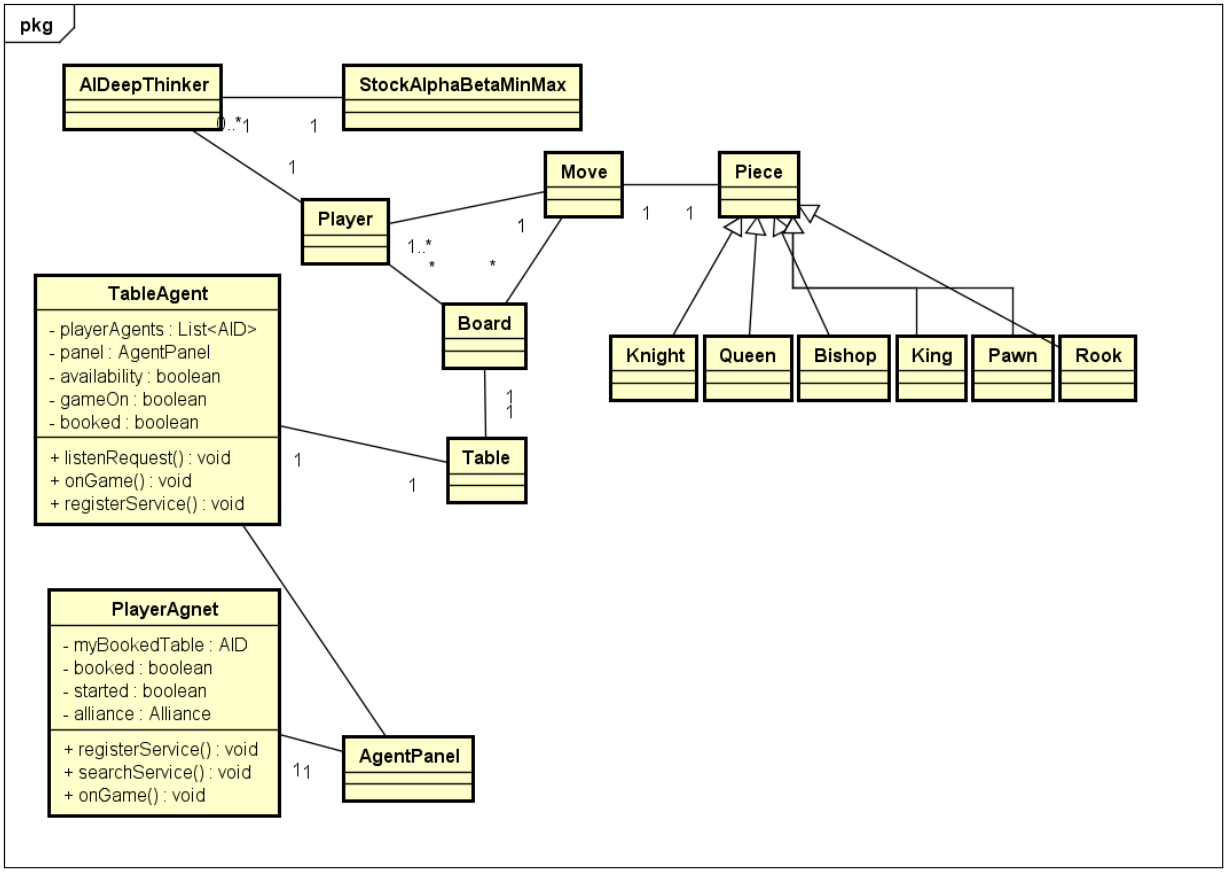

Figure 13 : Diagrammes de classes

3.9. Diagrammes de sequences

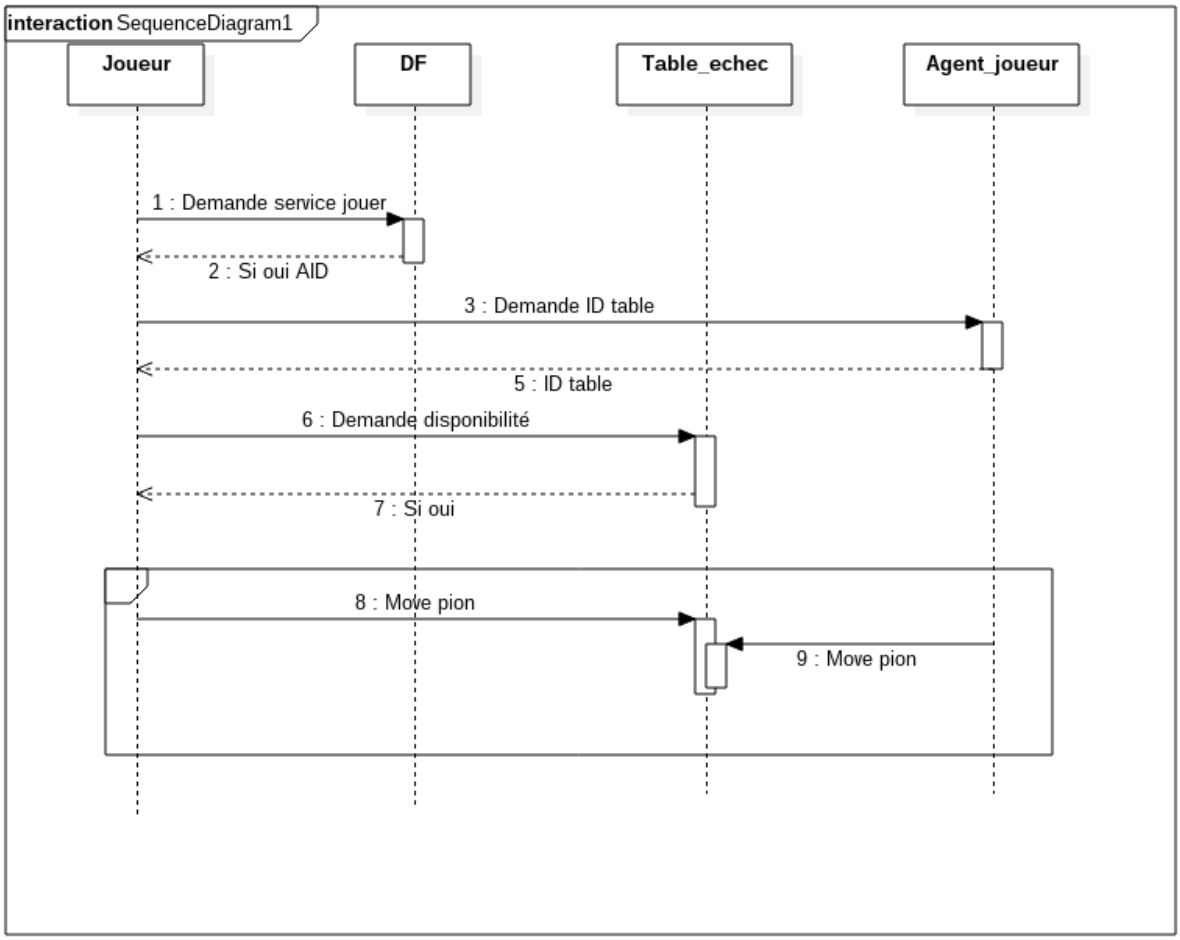

Figure 14 : Diagramme de sequence

IV. IMPLEMENTATIONS

4.1. Environnements de travail

Matériels :

Processeurs : Intel Pentium Inside B590 2.4Ghz x 2

Mémoire RAM : 4Go

Système d'exploitation : Windows 10 64 bits

4.2. Environnement

JADE PLATFORM

Extrait de description sur le site (https://perso.limsi.fr/jps/enseignement/examsma/2005/1.plateformes_3/index-Ferguen.html)

<<Le meilleur moyen pour construire un système multi-agent(SMA) est d'utiliser une plate-forme multi-agent. Une plate-forme multi-agent est un ensemble d'outils nécessaire à la construction et à la mise en service d'agents au sein d'un environnement spécifique. Ces outils peuvent servir également à l'analyse et au test du SMA ainsi créé. Ces outils peuvent être sous la forme d'environnement de programmation (API) et d'applications permettant d'aider le développeur. Nous allons étudier dans cette partie la plate-forme JADE(Java Agent DEvelopment framework).

JADE (Java Agent DEvelopement framework) est une plate-forme multi-agent créé par le laboratoire TILAB [TILAB] >>

IntelliJ Idea

Tout au long de notre de notre travail on a utilisé cet environnement de développement intégrés à sa version open source (community version) version 2018.4.

4.3. Outils

Librairies graphiques : Swing, Awt (avec Java Native Interface (pour embelir)).

4.4. Présentation de notre application

Les comportements des agents

PlayerAgent

Comportements

Il a deux comportements dont :

(01) CyclicBehaviour : pour la recherche des services et les écoutes de messages de communication entrant.

(02) TickerBehaviour : qui envoi à chaque 5sécondes pour les envoyer le déplacements (pour gérer le jeu).

Intelligence

En intelligence artificielle et en théorie des jeux, l'**élagage alpha-bêta** (abrégé élagage αβ) est une technique permettant de réduire le nombre de nœuds évalués par l'algorithme minimax.L'algorithme minimax effectue en effet une exploration complète de l'arbre de recherche jusqu'à un niveau donné, alors qu'une exploration partielle de l'arbre est généralement suffisante : lors de l'exploration, il n'est pas nécessaire d'examiner les sous-arbres qui conduisent à des configurations dont la valeur ne contribuera pas au calcul du gain à la racine de l'arbre. L'élagage αβ nous permet de réaliser ceci. Plus simplement, l'élagage αβ évite d'évaluer des nœuds dont on est sûr que leur qualité sera inférieure à un nœud déjà évalué, il permet donc d'optimiser grandement l'algorithme minimax sans en modifier le résultat.

Les deux joueurs blanc et noire comme définit dans la théorie de minimax, le blanc qui cherche à maximiser les points et la noire qui cherche à minimiser.

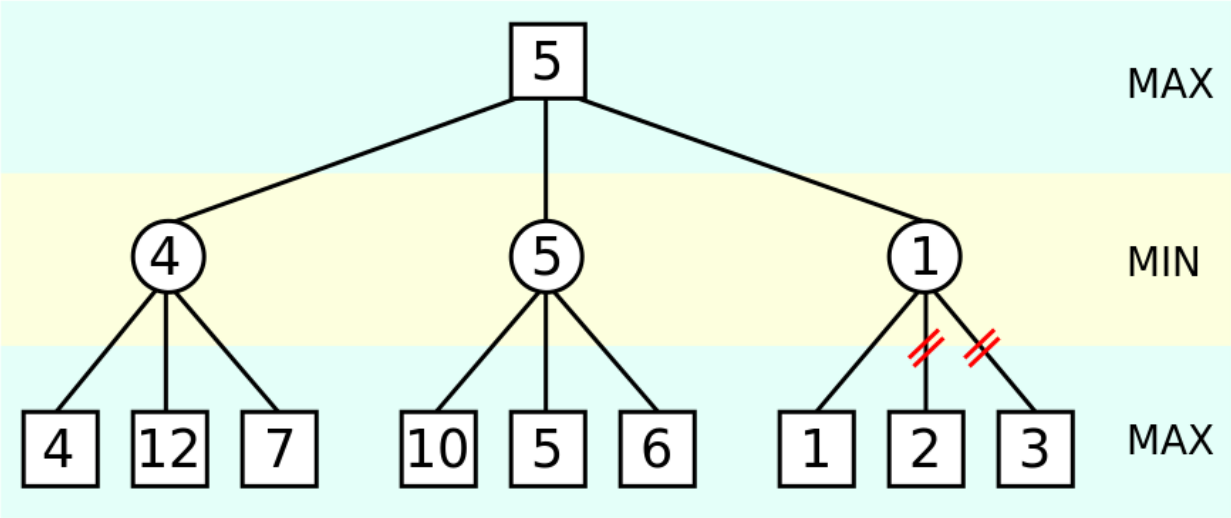

Figure 15 : Illustration alpha-beta pruning (élegage AlphaBeta)

Le joueur blanc choisi depuis la tête de l'arbre en faisant le maximization. Pour l'implémenter on a créé deux fonctions min () et max (), qui sont recurcivess entre eux, le min appelle le max et le max appelle le min car pour pouvoir évaluer l'état en allant en profondeur de l'arbre.

Niveau du joueur

Le niveau de l'agent est défini par la profondeur de de recherche dans l'arbre.

TableAgent

Intelligence :

Les règles d'échecs sont implémentées dans la classe Board, dont il y a les calculateurs des mouvements légales (class Move, Move Transition) dans chaque Player c'est automatiquement gérer pour éviter les mauvais déplacements.

Comportement :

Celui-là comporte deux comportements dont :

(01) CyclicBehaviour : pour écouter les demandes.
(02) CyclicBehaviour : pour gérer et maintenir le jeu.

Interface de la table d'échecs

Voici l'interface du jeu d'échec rien ne se passe avant que deux agents se mettent d'accord :

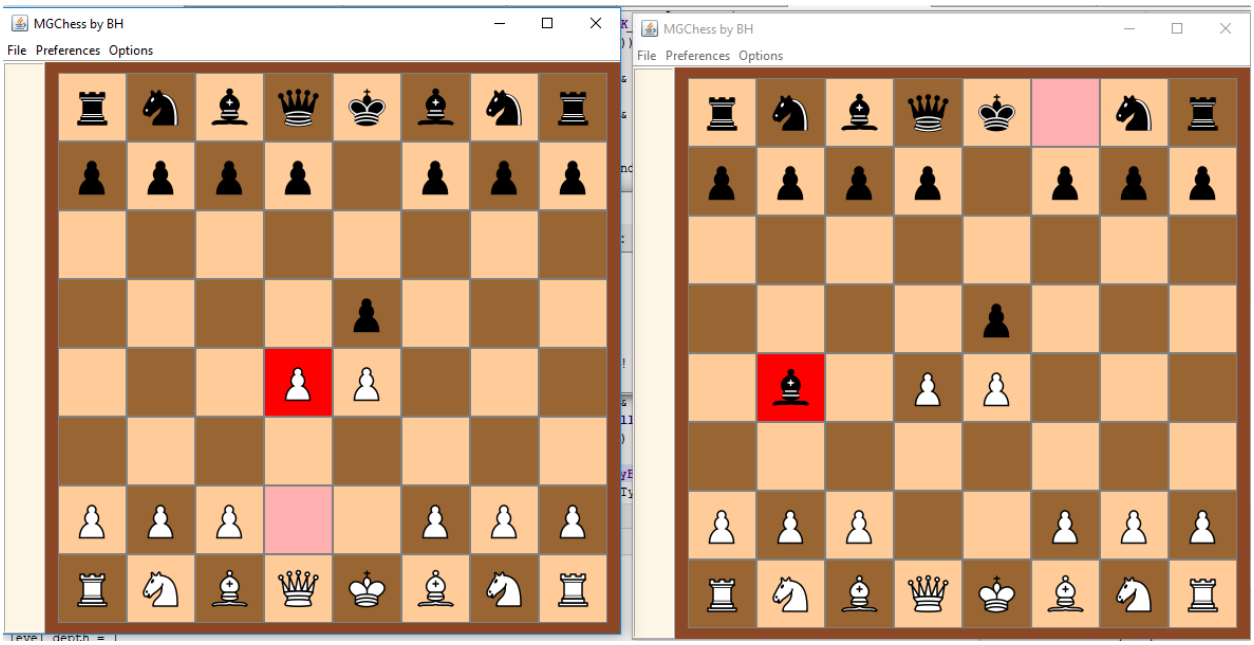

Figure 16 : Interface du jeu

Interface des agents

Voici l'interface des agents pour voir les messages entrants et sortant échangé pas les agents et gestion des agents.

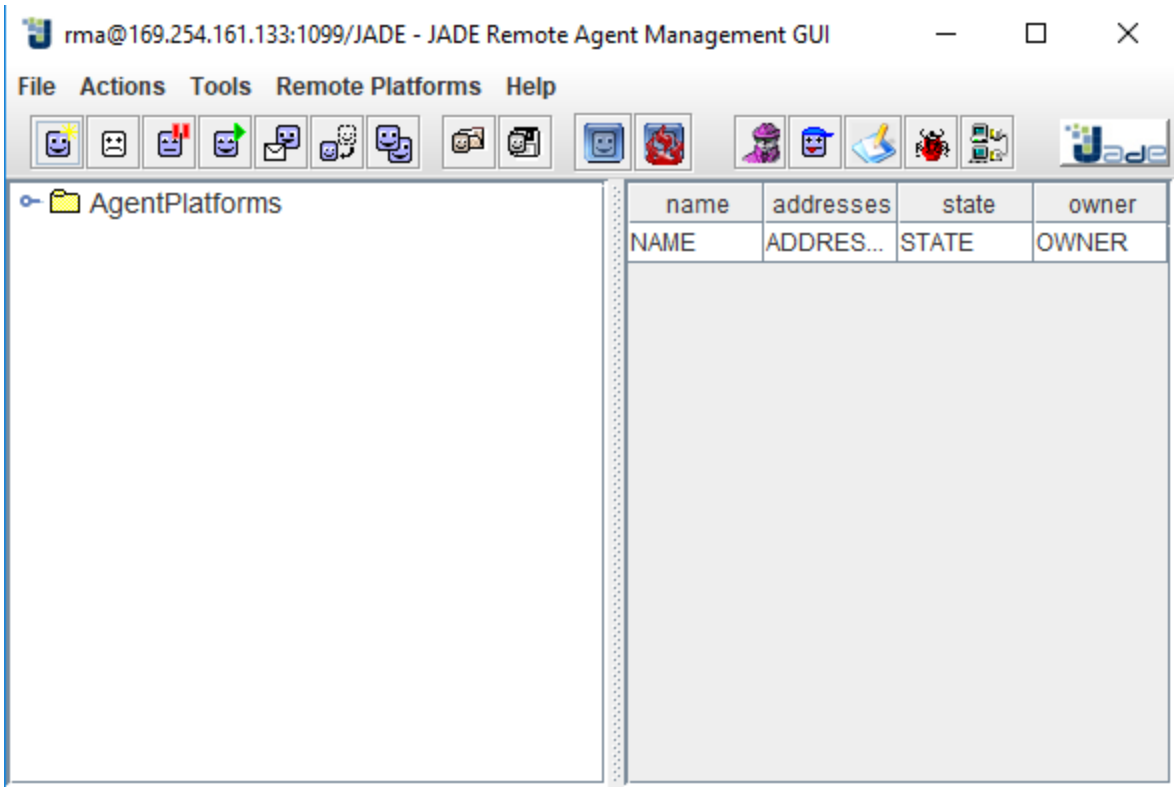

Figure 17 : Interface de JADE

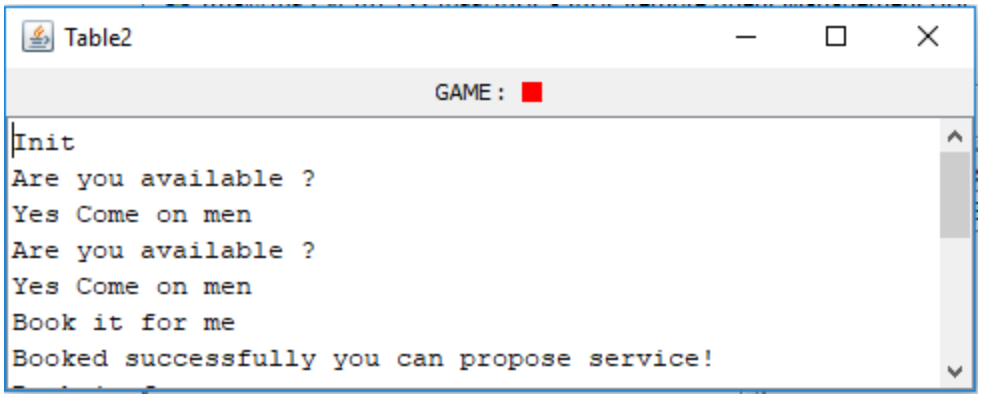

Figure 18 : Interface de la Table

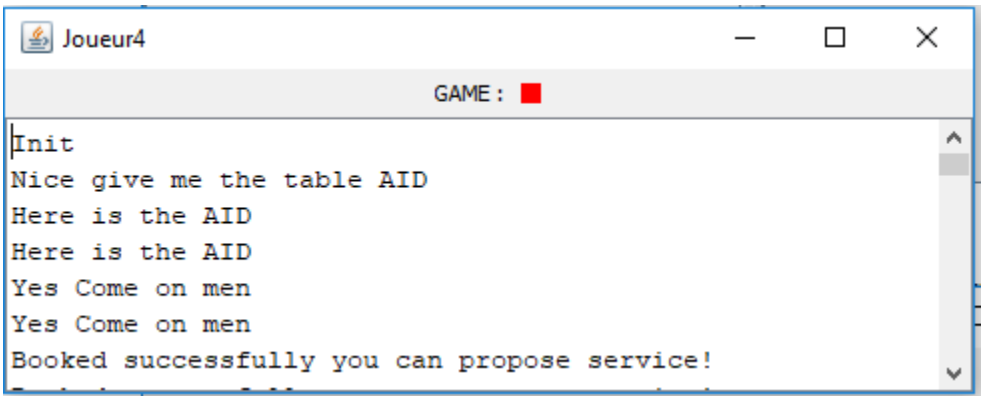

Figure 19 : Interface du joueur

4.5. Difficultés rencontrées

La limitation des tailles de paquets sur JADE nous a obligé de ne pas passer les déplacements par message.

Pour notre intelligence artificielle avec MiniMax on a eu des problèmes en voulant élargir la recherche au niveau 7.

V. CONCLUSION

Dans ce travail pratique, on a eu beaucoup de nouvelles connaissances surtout la mise en en pratique da la programmation orienté agent par approche GAIA. On a vu ses différents aspects mais les ressources disponibles sur Gaia sont assez limitées. De plus, Gaia nécessite de posséder une solide maîtrise de la logique temporelle, ce qui est aussi le cas de DESIRE. Ceci la rend plutôt difficile à adopter pour des ingénieurs. Comme précisé dans les concepts, Gaia est limitée aux applications à agents à forte granularité, peu nombreux, et avec une organisation statique, ce qui rend le passage à l'échelle difficile. Malgré quelques travaux sur une spécification de directives de passage à la plate-forme JADE, Gaia ne propose aucune étude d'implémentation, ce qui a, bien sûr, l'avantage de laisser le développeur libre d'utiliser n'importe quel langage de programmation. JADE est notre nouvel outil préféré après avoir réussi à faire marcher le système de jeu d'échec avec. Ce jeu qu'on a créé certes, il marche mais on peut ajouter des nouvelles fonctionnalités intéressantes comme la compétition, l'enregistrement des historiques, et doter à notre intelligence artificielle un système d'apprentissage.