Analyse des besoins et tests pour le projet
Réalisation d'un jeu d'Othello

Morgane Badré
Benjamin Letourneau,
Vincent Wilmet,
Nicolas Yvon

Ce document présente une version préliminaire de l'architecture de notre projet.

1 Architecture du projet

Le projet à développer est conséquent, certains modules à réaliser sont plus compliqués que d'autres donc nous avons pensé segmenter en différentes briques. Le principe de la segmentation (voir figure 1) en module est qu'ils sont tous indépendants et peuvent être testés indépendamment. Les blocs IA et logiciel sont les plus importants alors que la gestion de fichiers, l'interface et l'éditeur de plateau sont les moins importants. Le second intérêt de cette architecture est que chaque module peut être remplacé par un autre plus ou moins équivalent sans compromettre l'intégrité et le bon fonctionnement du logiciel. Par exemple, pour le bloc de l'IA, si une personne de la communauté souhaite l'améliorer, il peut tout simplement coder le module IA et l'intégrer au logiciel pour le tester.

Développant en JAVA, les différents modules sont liés par le biais d'interfaces. Ainsi, chaque module présentera une interface définie et dûment commentée qui représentera le comportement du module.

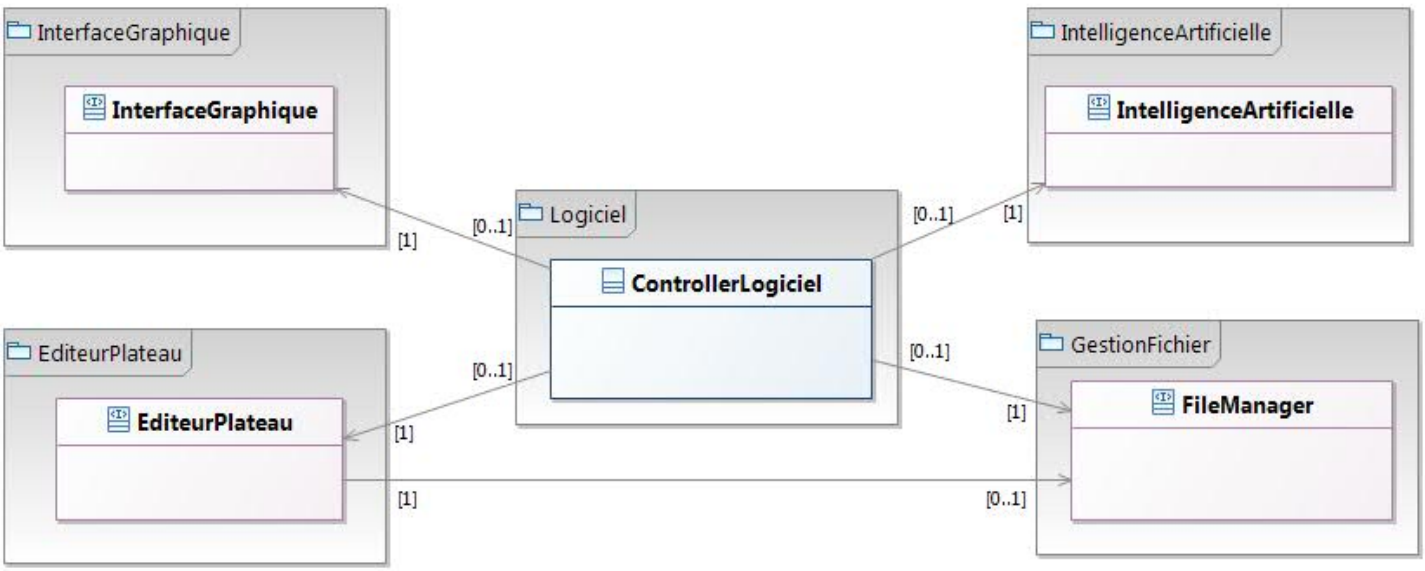

FIGURE 1 – Diagramme de classe du projet global

1.1 Module Logiciel

Il reprend trois design pattern : State, Factory, MVC.

— State sera utilisé pour coder les différents types de pion sur le plateau
— Factory sera utilisé pour la génération d'objets
— Et l'architecture complète du logiciel sera en MVC pour la maintenabilité et la facilité de lecture/modification du code.

1.2 Module d'Intelligences Artifcielles

Ce module (voir figure 2), le plus compliqué et plus important à réaliser dans notre projet, reprend le design pattern strategy. En effet, on représentera les différents types d'IA en strategie : naïve, force brute, évolutive, ...

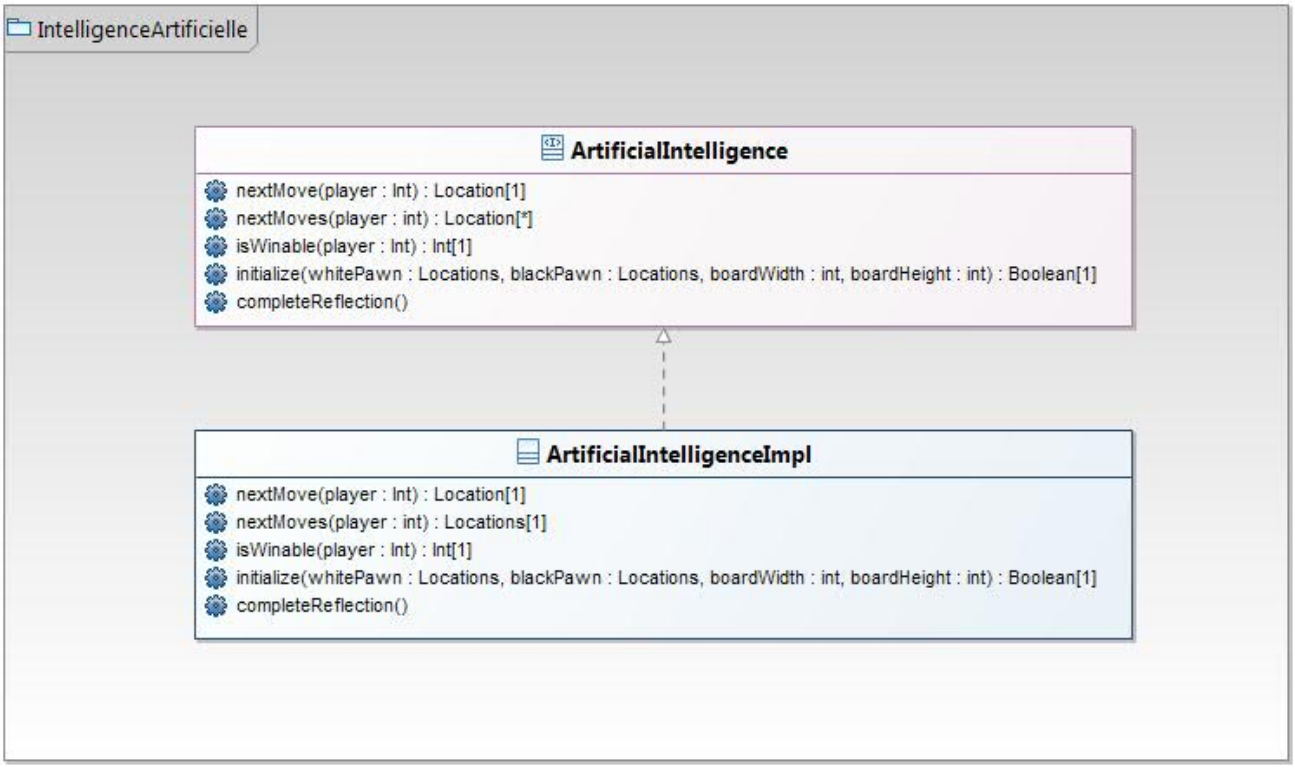

FIGURE 2 – Diagramme UML de classe du module d'intelligence artifcielle

1.3 Module de Gestion de Fichier

La gestion de fichiers (voir figure 3) ne contient pas de design pattern spécifique.

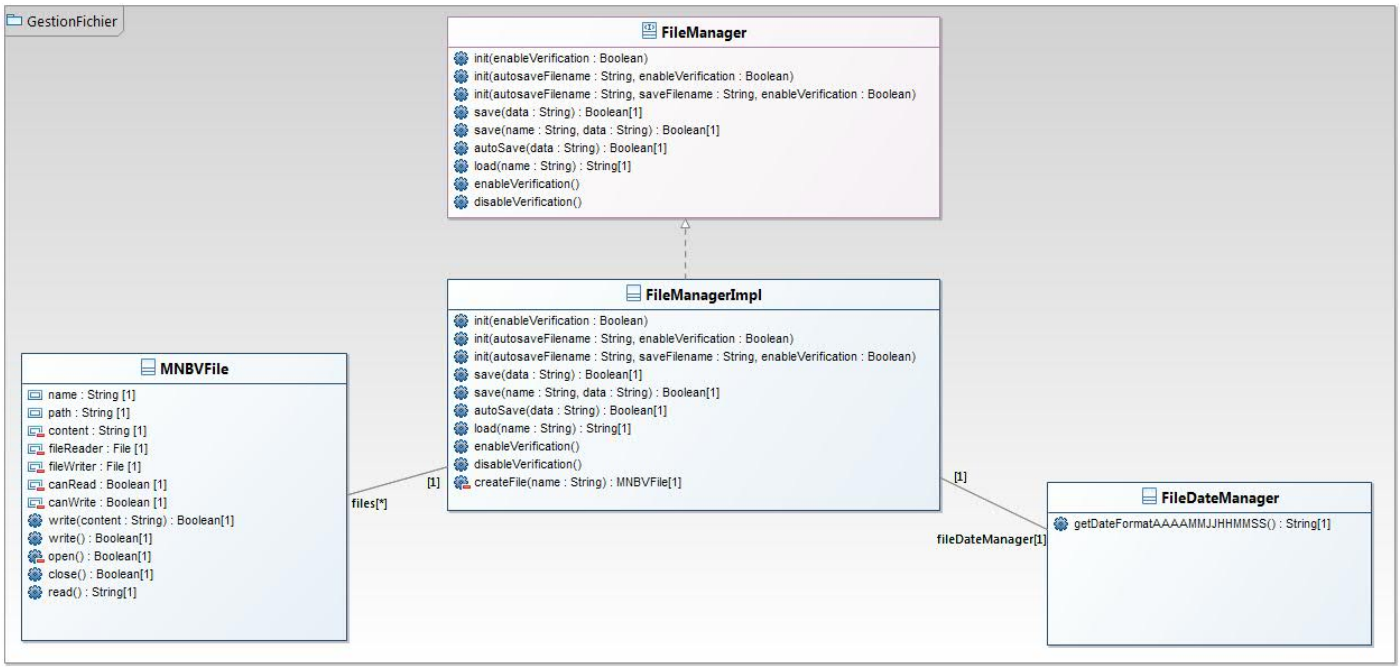

FIGURE 3 – Diagramme UML de classe du module de gestion de fichier

1.4 Module d'Interface Homme-Machine (IHM)

L'IHM ne contient pas de design pattern spécifique.

1.5 Module d'Editeur de Plateau

Ce module (voir figure 4) n'utlisera pas de design pattern car si cette fonctionnalité est implémentée, elle ne sera pas forcément graphique mais plus probablement utlisable par la console donc l'architecture sera simple.

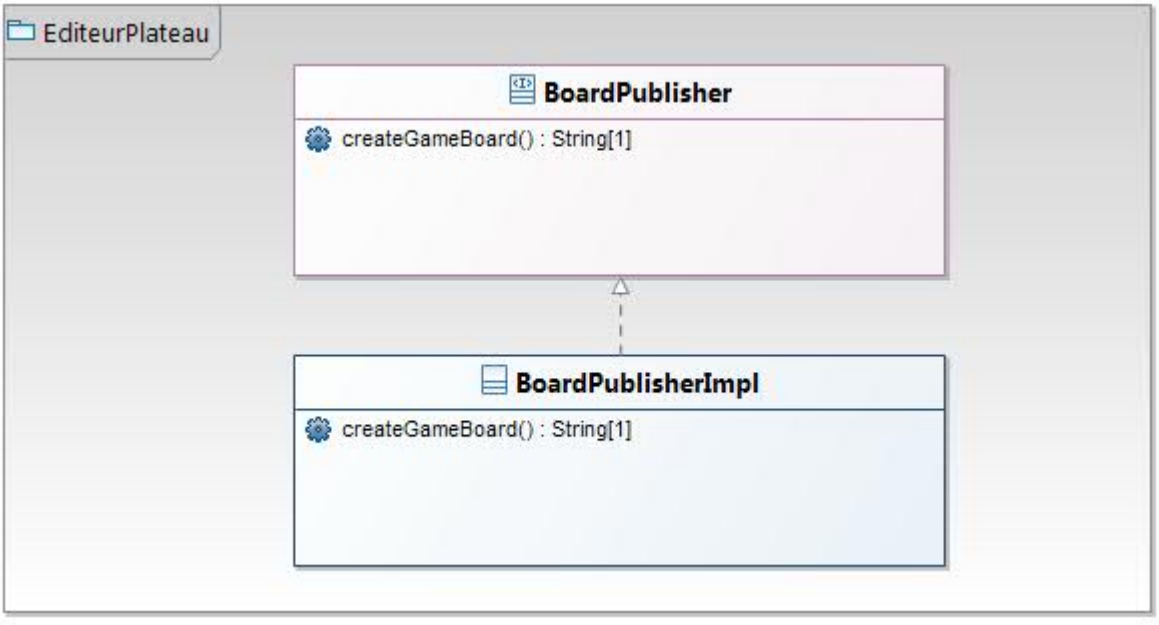

FIGURE 4 – Diagramme de classe du module éditeur de plateau