[1. Présentation générale du problème 2](#_Toc414961900)

**CAHIER DES CHARGES FONCTIONNEL**

* **Structure émettrice**

Info1 - Groupe B

* **Emetteurs**

CLEMENT Ian [ian.clement@etu.univ-nantes.fr](mailto:ian.clement@etu.univ-nantes.fr)

DECAMP Grégoire [gregoire.decamp@etu.univ-nantes.fr](mailto:gregoire.decamp@etu.univ-nantes.fr)

DIAS Fabien [fabien.dias@etu.univ-nantes.fr](mailto:fabien.dias@etu.univ-nantes.fr)

GOUREAU Nicolas [nicolas.goureau@etu.univ-nantes.fr](mailto:nicolas.goureau@etu.univ-nantes.fr)

* **Destinataires**

TAMZALIT Dalila [dalila.tamzalit@uni-nantes.fr](mailto:dalila.tamzalit@uni-nantes.fr)

MOTTU Jean-Marie [jean-marie.mottu@univ-nantes.fr](mailto:jean-marie.mottu@univ-nantes.fr)

[1.1. Projet 2](#_Toc414961901)

[1.1.1. Finalités 2](#_Toc414961902)

[1.1.2. Espérance de retour sur investissement 3](#_Toc414961903)

[1.2. Contexte 3](#_Toc414961904)

[1.2.1. Situation du projet par rapport aux autres projets de l’entreprise 3](#_Toc414961905)

[1.2.2. Études déjà effectuées 3](#_Toc414961906)

[1.2.3. Études menées sur des sujets voisins 3](#_Toc414961907)

[1.2.4. Nature des prestations demandées 4](#_Toc414961908)

[1.2.5. Parties concernées par le déroulement du projet et ses résultats (demandeurs, utilisateurs) 4](#_Toc414961909)

[1.2.6. Caractère confidentiel s'il y a lieu 4](#_Toc414961910)

[1.3. Énoncé du besoin (finalités du produit pour le futur utilisateur tel que prévu par le demandeur) 4](#_Toc414961911)

[1.4. Environnement du produit recherché 4](#_Toc414961912)

[1.4.1. Listes exhaustives des éléments (personnes, équipements, matières…) et contraintes (environnement) 4](#_Toc414961913)

[1.4.2. Caractéristiques pour chaque élément de l’environnement 4](#_Toc414961914)

[2. Expression fonctionnelle du besoin Diagramme de cas d’utilisation 5](#_Toc414961915)

[**Fonctions Principales** 5](#_Toc414961916)

[**Fonctions Complémentaires** 6](#_Toc414961917)

[**Fonctions Techniques** 7](#_Toc414961918)

[3. Cadre de réponse 8](#_Toc414961919)

[3.1. Pour chaque fonction 8](#_Toc414961920)

[3.1.1. Solution proposée 8](#_Toc414961921)

[3.1.2. Niveau atteint pour chaque critère d’appréciation de cette fonction et modalités de contrôle 8](#_Toc414961922)

[3.1.3. Part du prix attribué à chaque fonction 8](#_Toc414961923)

[3.2. Pour l’ensemble du produit 8](#_Toc414961924)

[3.2.1. Prix de la réalisation de la version de base 8](#_Toc414961925)

[3.2.2. Options et variantes proposées non retenues au cahier des charges 8](#_Toc414961926)

[3.2.3. Mesures prises pour respecter les contraintes et leurs conséquences économiques 8](#_Toc414961927)

[3.2.4. Outils d’installation, de maintenance … à prévoir 8](#_Toc414961928)

[3.2.5. Décomposition en modules, sous-ensembles 8](#_Toc414961929)

[3.2.6. Prévisions de fiabilité 8](#_Toc414961930)

# Présentation générale du problème

## Projet

### *Finalité**s*

Notre projet a pour but de développer un mastermind en ligne qui sera accessible à tous. Pour cela nous essayerons de faire une interface graphique qui conviendra aux jeunes mais aussi aux personnes plus âgées. Celle-ci sera aussi simple d’utilisation et correspondra aux attentes que l’on peut avoir dans le développement d’un tel jeu. De plus, nous rajouterons de petits éléments qui rendront notre projet unique. Notre jeu comprendra une intelligence artificielle contre laquelle le joueur pourra jouer mais aussi un mode 1 contre 1 pour que deux joueurs distincts puissent s’affronter. L’intelligence artificielle sera aussi capable de compter les points, de générer des combinaisons aléatoires et aussi de préciser au joueur recherchant la combinaison, quels pions sont bien placés et lesquels ne le sont pas ou du moins ne sont pas à la bonne place, tout cela avec un code couleur spécifique.

### *Espérance de retour sur investissement*

Comme cité précédemment notre Mastermind sera destiné à tout public. Nous avons plus précisément pour but de combler toutes les personnes de 7 à 77 ans. En effet, le jeu basé sur un pattern MVC (Modèle vue contrôleur) va nous permettre de réaliser les souhaits de nos futurs utilisateurs avec simplicité. Ce pattern permet de combiner interface homme machine(IHM)/code grâce à un contrôleur qui fera la transition entre les deux. Nous voulons viser un public assez large car le mastermind est un jeu qui permet de développer sa réflexion, ce qui est bon pour les jeunes personnes mais aussi de conserver une bonne concentration et réflexion ce qui est aussi parfait pour les utilisateurs un peu plus âgés. En outre ce jeu permet aussi de rassembler des gens autour d’une partie tout cela dans une bonne ambiance. Notre jeu sera accessible gratuitement.

## Contexte

### *Études déjà effectuées*

Nos études se résument au fonctionnement du jeu en lui-même, et à la résolution de sa complexité algorithmique. En effet, il nous semble intéressant, en plus du mode joueur contre joueur, de développer un IA de plus ou moins haut niveau. Nous nous sommes intéressés aux codes de déchiffrement et de cassage du Mastermind, comme celui proposé par Knuth, qui permettent de mieux appréhender les enjeux et la complexité du jeu en lui-même.

### *Études menées sur des sujets voisins*

Nous avons étudié les moyens de rendre le jeu modulable tant dans son « gameplay » (difficulté) que dans sa construction (pattern). Ainsi, nous nous sommes informé sur le pattern Modèle-Vue-Contrôleur, qui nous semble une bonne option dans le développement d’un code objet modulable. Ensuite, pour les perspective d’IHM, une librairie proposé sur Internet, et compatible avec Java nous as interpellé.

### *Nature des prestations demandées*

Logiquement, notre projet doit se résumé au développement du Mastermind en langage objet, de manière a d’abord pouvoir jouer, joueur contre joueur. Il faut aussi que notre code sois modulable et efficace évidemment. Il nous faudra aussi ajouter une interface graphique à tout cela. Elle sera ajoutée plus tard, d’où l’importance de la modularité du code.

### *Caractère confidentiel s'il y a lie**u*

Notre code sera notre propriété, donc il devra rester plus ou moins confidentiel, d’où l’utilisation d’un dépôt Svn, qui permet de garder privé toutes les sources de notre projet. A part cela, notre projet n’as pas lui d’être entouré d’un secret trop épais, il ne représente pas une technologie ou une nouveauté qui puisse être convoitée par d’autre entreprise. Il n’empêche que des précautions seront à prendre quand a la diffusion des sources, surtout si le programme est en Java.

## Énoncé du besoin (finalités du produit pour le futur utilisateur tel que prévu par le demandeur)

## Environnement du produit recherché

### *Listes exhaustives des éléments (personnes, équipements, matières…) et contraintes (environnement)*

### *Caractéristiques pour chaque élément de l’environnement*

# Expression fonctionnelle du besoin Diagramme de cas d’utilisation

Ce tableau regroupe toutes les fonctions de notre projet. On y retrouve les **Fonctions Principales,** les **Fonctions Complémentaires,** ainsi que les **Fonctions Techniques.** C’est à partir de ce tableau que nous allons pouvoir concevoir les différents diagrammes de classe de chaque fonction.

|  |  |  |  |
| --- | --- | --- | --- |
| **Fonction** | **Importance** | **Critères d’appréciation** | **Niveau d’appréciation** |
| **Fonctions Principales** | | | |
| **FP1** : Paramétrage partie. | 4 | - Choix de la couleur des pions.  - Nombres d’essais.  - Nombre de pions de la combinaison secrète.  - Possibilité de réutiliser une même couleur. | - Minimum 6 couleurs,  maximum 8 couleurs.  - Minimum 10 d’essais et 15 max.  - Minimum 4 pions dans la combinaison secrète, Max 6 couleurs.  - La combinaison secrète peut avoir ou non plusieurs fois la même couleur. |
| **FP2** : Création de combinaison secrète. | 5 | - L’utilisateur doit choisir le nombre de pions définis préalablement.  - L’utilisateur a le choix dans les couleurs disponibles. | - Le nombre de pions dans les combinaisons est égale au nombre de pions choisie préalablement.  - Seuls les couleurs disponibles sont utilisés. |
| **FP3** : Proposition et validation de la solution. | 5 | - L’utilisateur doit pouvoir construire une solution ayant le bon nombre de pions et utilisant les couleurs associés à la partie.  - Le joueur doit pouvoir soumettre sa combinaison pour la comparer avec la combinaison secrète. | - La solution doit être en format comparable à la combinaison secrète.  - Le joueur doit connaître les pions de bonnes couleurs bien placés.  - Le joueur doit connaître les pions de bonnes couleurs mal placés. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Fonctions Complémentaires** | | | |
| **FC1** : Fenêtre graphique de partie. | 4 | - Une fenêtre graphique s’affiche pour modéliser la partie en cours.  - La combinaison secrète doit pouvoir être occultée.  - La proposition doit être visible et modifiable à partir de cette fenêtre.  - Le joueur doit pouvoir savoir combien d’essai il a effectué. | - La fenêtre doit afficher les solutions.  - La combinaison secrète n’est pas visible.  - La composition de la proposition ce fait via l’interface graphique.  - Le nombre d’essai est affiché (explicitement ou implicitement). |
| **FC2** : Fenêtre graphique du menu. | 3 | - Une fenêtre graphique s’affiche pour pouvoir lancer une nouvelle partie ou voir les crédits. | - La fenêtre possède un bouton pour lancer le paramétrage de partie.  - La fenêtre possède un deuxième bouton pour lancer la fenêtre de crédit. |
| **FC3** : Fenêtre graphique de paramétrage de partie. | 3.5 | - Une fenêtre graphique doit s’afficher pour que l’utilisateur puisse modifier les paramètres de la partie qu’il va lancer. | - Un objet graphique permet le choix du nombre de couleurs dans les combinaisons.  - Un objet graphique permet le choix du nombre de pions dans les combinaisons.  - Un objet graphique permet le choix du nombre d’essais disponible aux joueurs.  - Un objet graphique permet l’autorisation du choix multiple de couleurs dans une combinaison. |
| **FC4** : Fenêtre graphique des crédits. | 2 | - Une fenêtre graphique s’affiche pour que l’utilisateur affiche les crédits du programme. | - Un texte affiche tous les développeurs et les programmes utilisés |

|  |  |  |  |
| --- | --- | --- | --- |
| **Fonctions Techniques** | | | |
| **FT1** : Comptage du score. | 3 | - Le score doit être calculé. | - Le score calculé doit être correcte à 100%.  - Le score est calculé à partir du nombre d’essais effectués. |
| **FT2** : Switch entre les joueurs. | 5 | - Le programme doit pouvoir gérer le tour des joueurs. | - Le programme ne commet aucune erreur sur le tour de jeux.  - Le 1er joueur choisit une combinaison secrète.  - Puis le 2ème joueur émet une proposition jusqu’à ce que le nombre essais soit épuisé ou que la proposition soit valide.  - Le 1er joueur compare à chaque proposition du 2ème joueur. (→ voir pour proposer comparaison automatique). |

# Cadre de réponse

Diagramme de classes métiers : à base de classes métiers (des concepts indépendants de la programmation, avec des attributs mais sans méthodes, ainsi que des relations entre les concepts sur les agissements des classes métiers les unes sur les autres)

Synthèse des jalons : tables synthétiques des jalons du projet.

## Pour chaque fonction

### *Solution proposée*

*Tâches à réaliser pour la solution -> WBS*

*Diagramme de séquence basée sur le diagramme de classe métier et sur les scénarios des cas d’utilisation (cf. section 2)*

*Jalon : date à laquelle la fonction doit être livrée.*

### *Niveau atteint pour chaque critère d’appréciation de cette fonction et modalités de contrôle*

### *Part du prix attribué à chaque fonction*

## Pour l’ensemble du produit

### *Prix de la réalisation de la version de base*

Nous ne prévoyons aucun prix pour la réalisation de base du fait que nous utilisons des logiciels gratuits.

### *Options et variantes proposées non retenues au cahier des charges*

### *Mesures prises pour respecter les contraintes et leurs conséquences économiques*

### *Outils d’installation, de maintenance … à prévoir*

### *Décomposition en modules, sous-ensembles*

### *Prévisions de fiabilité*