SPECIFICATION DES BESOINS DE PROGRAMME

pour

Dominion Logs Datamining

Réalisé par
VER VALEM Willian
BAYOL Elmer
LOPEZ-FARFAN M. Liliana
TAGNAOUTI Khaoula

March 30, 2016

Contents

1 Introduction

1.1 Objectif

Ce document a pour but de clarifier et de décrire aussi précisément que possible les besoins pour le développement du programme *Dominion DataMining*.
Dominion DataMining est un programme qui se compose de 3 parties:

- **Modélisation des donnée:** responsable de la lecture et du remodelage des données à stocker.
- **Stockage de données:** stocke les données analysées.
- **Analyse des données:** effectue les requètes vers la base de données et traite les données stockées.

1.2 Public visé et suggestions de lecture

Ce document est destiné à tout utilisateur, développeur, testeur, chef de projet ou de la documentation écrivain qui a besoin de comprendre l'architecture de base du système et de ses spécifications.
Voici les utilisations potentielles pour chacun des types de lecteurs:

- **Développeur:** le développeur qui veut lire, changer, modifier ou ajouter de nouveaux besoins dans le programme existant, doit tout d'abord consulter ce document et mettre à jour les besoins en manière appropriée afin de ne pas détruire leur signification réelle et de transmettre les informations correctement aux prochaines phases du processus de développement.

- **Utilisateur:** l'utilisateur de ce programme en revue les schémas et les spécifications présentées dans ce document et détermine si le logiciel a toutes les besoins appropriés et si le développeur du logiciel a mis en œuvre tous.

- **Testeur:** le testeur a besoin de ce document pour vérifier que les besoins initiaux de ce programme correspondent actuellement au programme exécutable correctement.

1.3 Contexte du projet

Le jeu de cartes*Dominion* a été hébergé sur un serveur à partir d'Octobre 2010 à Mars 2013. Les parties jouées sur ce serveur ont été enregistrés dans des *logs* qui gardent la trace de toutes les actions des joueurs;ces *logs* ont été mis à la disposition du public. Mais certaines informations par rapport aux décisions prises par les joueurs n'ont pas été enregistrées.
Un wiki sur le jeu a été créé, il offre des conseils d'experts dans la prise de décision au cours du jeu: il offre principalement des conseils au niveau de la stratégie. Le but de ce projet, proposé par Yvan Le Borgne, chercheur au Labri, est de traiter des *logs* (plus de 12 millions de parties) afin de répondre à plusieurs questions. Il consiste principalement à comparer les données recueillies à l'information du wiki afin de valider l'utilité des idées et des informations données par des experts. Nous allons au moins créer des outils de validation offrant suffisamment efficaces.

Afin d'Atteindre cet objectif, le but du projet est de créer une base de données contenant tous les jeux joués et de produire une analyse basée sur ces informations. Pour ce faire, notre programme devra d'abord extraire les données des *log*, car ils sont trop volumineux et trop compliqués à utiliser directement (structure non-standard). Nous aurons à mettre en place des structures de données qui nous permettent d'avoir un meilleur stockage de données. Un moyen simple pour représenter ces données sera offert à l'utilisateur.

Certaines informations sont également manquantes dans les *logs*(par exemple: les cartes piochées au cours d'un tour). Nous allons essayer de trouver une méthode pour reconstruire cette information et l'ajouter à l'information existante. Nous allons également essayer d'optimiser les opérations effectuées par l'analyse de données en trouvant un équilibre entre la mémorisation des demandes et le recalcul.

Enfin, nous devrons décider quelle stratégie est utilisée avec chaque jeu, ou même reconnaître les changements de stratégie lors d'un match. Nous pourrions aussi reconnaître qui a appliqué une première stratégie (qui l'a *inventé*). Si nous pouvons détecter assez stratégie liée aux données, nous pouvons également détecter le processus d'apprentissage d'habitude des joueurs.

2 Description globale

2.1 Fonctions de produit

Voici une liste des fonctions qui devraient être disponibles sur le logiciel:

- **Modélisation des données:**
  - Décompression de log
  - Parsing
  - Creation de game-log
  - Envoi de game-log data
- **Stockage de données:**
  - Créer une base de données orientée document
  - Se connecter à la base de données orientée document
  - Compression
  - Enregistrer le game-log
- **Analyse des données:**
  - Recueillir les données
  - Restorer le game-log
  - Plotting
  - Calculer ***ELO***
  - Reconnaître les stratégies
  - Reconnaître Greening

Une description plus précise de chaque fonctionnalité sera disponible dans la section 4 de ce document.

2.2 Classes de l'utilisateur et caractéristiques

Acteurs physiques:

- **Utilisateur:** l'utilisateur lance l'analyse des logs et des demandes des analyses à faire sur les données stockées qui seront ensuite représentées sous forme de graphes. Les actions de l'utilisateur peuvent être décomposées en deux parties:

- Étape de pré-calcul:
- Les données de l'analyse de l'utilisateur: une librairie en python sera offerte à l'utilisateur, pour qu'il puisse demander l'affichage des statistiques standard ou créer analyse complexe en utilisant son propre programme python assoscié à une bibliothèque proposant diverses fonctions.

Acteurs de Système:

- **Parser :** le parser est le système qui a accès en écriture à la base de données et est responsable du stockage des données analysées sur la base de données.
- **Analyzer :** l'analyseur est le système qui va interroger la base de données pour recueillir des données pour l'analyse, et sera en mesure de créer des requêtes persistantes.
- **Base de données:** La base de données stocke les données extraites des logs et réponds aux requêtes des autres parties du programme.

Le logiciel à développer est destiné à un usage personnel, il n'y a donc pas d'accès privilégié, l'utilisateur aura accès à l'intégralité du code et des données.

2.3 Conception et mise en œuvre Contraintes

Cette section présente un aperçu des problèmes possibles qui peuvent limiter le développement du logiciel:

- Le logiciel sera développé pour le système d'exploitation Linux et aucune autre plate-forme n'est nécessaire.
- Le logiciel doit fonctionner sur le matériel trouvé sur de simples ordinateurs personnels.
- Les données à analyser sont considérablement grandes (400Gb) et le client ne sera pas en mesure de fournir un serveur afin de travailler avec les données non compressées ce qui peut avoir un impact sur les performances du logiciel.
- Les *Logs* fournis ne sont pas a un format standard ce qui peut retarder le temps de développement l'analyseur les problèmes qu'il peut soulever au cours du développement ne sont pas connus pour le moment.

3 Les logs

3.0.1 Description des données

Les données qui ont été fournies par le client sont compressées en format tar.bz2, un fichier pour chaque jour de log, la taille totale des données compressée est 13 Go.
Les données décompressées font un total de 400 Go.
Chaque *Log* est au format *HTML*.
un *Log* doit contenir:

- **en-tête:** contient le numéro de jeu et les noms des gagnants.
- **résumé de match:** contient les cartes utilisées sur le match et comment le match a fini.
- **résumé de joueur:** contient les cartes de victoire de chaque joueur,le deck et les points.
- **game log:** cette partie contient tous les actions détaillées effectuées par les joueurs pendant le match.

Incohérence des données

Les *Logs* présentent des incohérences de données qui peuvent être problématiques lors du développement de l'analyseur. Voici quelques exemples de problèmes rencontrés:

- **Numéro de log**
  La numérotation des logs n'est pas unique ce qui implique qu'elle ne peut pas être utilisée pour identifier un log.
- **Noms d'utilisateur**
  Les logs montrent qu'aucune restriction n'a été faite sur les noms d'utilisateur du jeu et quelques noms d'utilisateur ont des mots clés et des caractères spéciaux qui peuvent entrer en conflit avec l'analyseur.
- **Données manquantes**
  Certains des logs ont une partie des données manquantes (comme: l'en-tête, la reprise des utilisateurs ...).
- **Format de log**
  La syntaxe utilisée lors de l'écriture des logs peut varier.

4 Besoins fonctionnels

4.1 Modélisation des données

4.1.1 Décompression des logs

Description et priorité

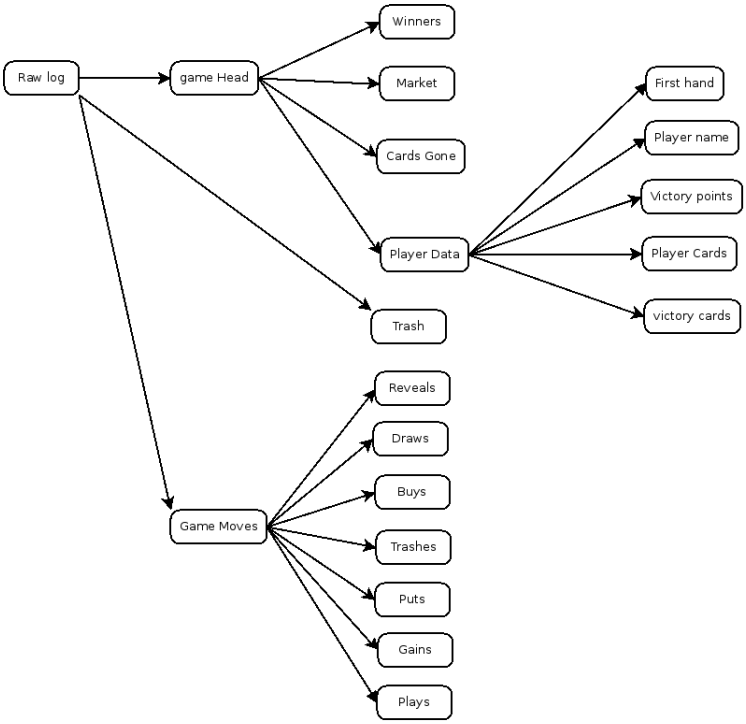
Niveau de priorité = haute

- Les fichiers tar.bz2 seront stockés dans un dossier.
- Le programme va décompresser un fichier spécifié à partir de ce dossier dans un dossier temporaire.
- Le programme va supprimer les logs décompressés à la demande du parser.

4.1.2 Parser

Description et priorité

Niveau de priorité = haute
L'analyseur va prendre les logs sous format *HTML* et va identifier les mots clés présents sur ces logs.
L'analyseur est chargé de lire les logs et collecter l'information importante sans perdre la structure du log.
Un aperçu des données pour être reconnu par l'analyseur est illustré dans ce diagramme:



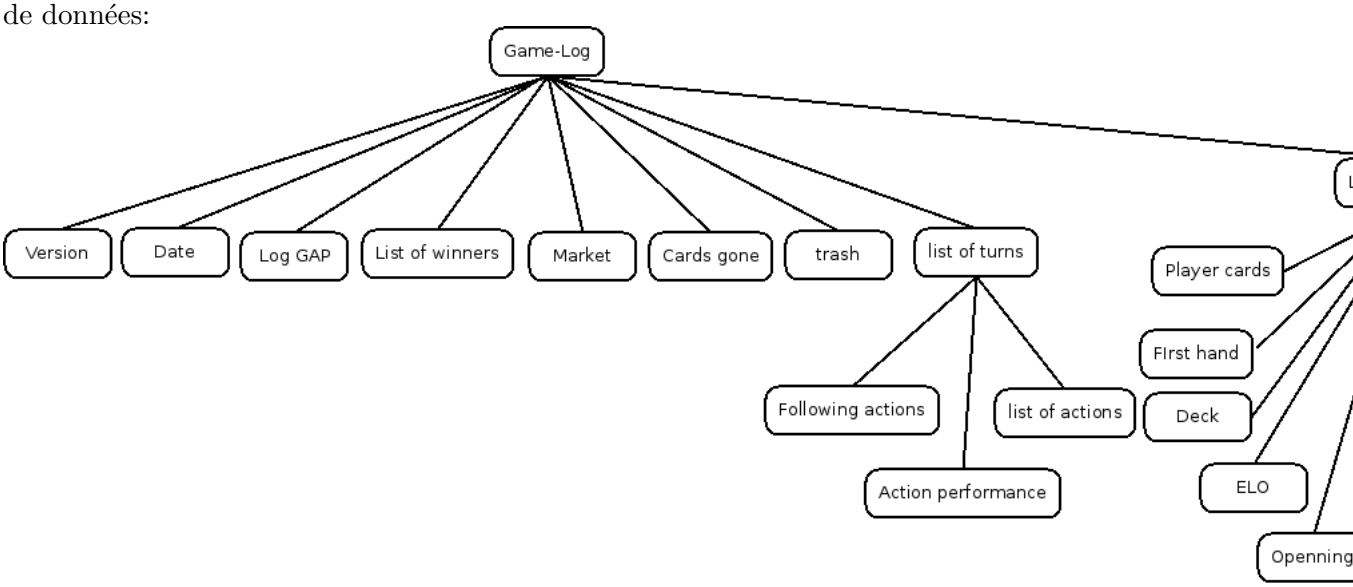
- Winners: liste des gagnants du jeu, il peut y avoir plusieurs gagnants en cas d'égalité.
- Market: liste des 10 cartes disponibles pour être acheté par les joueurs.
- Cards Gone: liste des cartes qui ont été entièrement acquises à la fin de la partie.
- Player Name: Nom du joueur.
- Victory points: nombre de points à la fin de la partie.
- Player cards: liste des cartes obtenues à la fin de jeu avec des noms et de quantités differents.
- Victory cards: liste des cartes de victoire le joueur avec le acheté.
- Trash: liste des cartes qui sont jetés à la fin de la partie.
- Game Moves: liste des actions effectuées pendant chaque tour d'un joueur.

Le parser doit lire chaque élément montré dans le graphique précédent en tant que partie des données lorsque l'utilisateur exécute le programme de parsing

4.1.3 Créer game-log

Description et priorité

Niveau de priorité = haute
Cette fonctionnalité est la création d'une structure de données où les données du log seront écrites. Voici une représentation graphique d'une vue d'ensemble de la structure de données:



4.1.4 Créer une base de données orientée document

Description et priorité

Niveau de priorité = haute
Ce module sera responsable pour créer la base de données orientée document, contenant des données au format JSON.

4.1.5 Se connecter à la base de données orientée document

Description et priorité

Niveau de priorité = haute

- La base de données orientée document sera en charge des échanges à travers un socket spécifique.
- La base de données orientée document recevra les logs de jeu au format JSON.
- La base de données orientée document recevra les demandes du programme relatif à chaque élément de données qu'il contient.

- La base de données orientée document enverra les résultats des demandes effectuées par le programme.

4.1.6 Compression

Description et priorité

Niveau de priorité = moyen

La base de données orientée document contiendra la plupart des données et un certain niveau de compression devra être appliquée. La base de données que nous allons utiliser est focalisé sur mongodb, et il offre 2 niveaux de compression *Zlib* et *Snappy*. Au cours du développement *Snappy* est utilisé car il offre une meilleure performance. Mais le programme final devrait donner à l'utilisateur la possibilité de choisir le niveau de compression lors du démarrage de l'analyse du log.

4.1.7 Sauvegarder game-log

Description et priorité

Niveau de priorité = haute

- La base de données orientée document permet de convertir la structure log de jeu dans un document au format JSON.
- Les données fournies au format JSON seront stockées dans la base de données.

4.2 Analyse des données

4.2.1 Recueillir des données

Description et priorité

Niveau de priorité = haute

- L'analyseur de données va convertir les résultats de la requête dans un format utilisable par le module de traçage.
- Si les données sont simples à lire (par exemple: un numéro, un nom, une phrase), l'analyseur de données sera tout simplement envoyer le résultat au format texte.
- La dernière requête sera mémorisée avec son résultat afin de gagner du temps si la prochaine requête faite par l'utilisateur est le même.
- Si l'utilisateur souhaite appliquer un script python de logs de jeu, le programme demandera les logs de jeu et les mettre dans un fichier texte, ce fichier contiendra chaque action faite par chaque joueur et le script sera appliqué à ce fichier.

4.2.2 restorer game-log

Description et priorité

Niveau de priorité = moyen

Cette fonction est responsable pour restaurer l'information manquante sur les logs analysés, en utilisant la déduction basique sur la base des données obtenues par le log. En cas d'un en-tête du jeu manquant ou partie manquante:

- Gagnants / cartes de victoire / points de victoire manquant: le programme peut garder la trace des cartes détenues par les joueurs pendant le jeu afin de savoir combien de cartes victoire qu'ils ont à la fin de la partie.
- Market manquant: Le programme peut garder la trace des cartes achetées au cours du jeu afin de reconstituer partiellement ou totalement les cartes disponibles sur Market.
- Cartes Gone: Comme avec les données manquantes de Market, le programme permet de garder une trace de cartes achetées et compter la quantité achetée pour chaque carte, si le montant maximum de cartes est acheté, la carte a disparu à la fin de la partie.
- Nom du joueur manquant: le jeu se déplace partie d'un log assure également le suivi des noms des joueurs, le programme sera tout simplement de les restaurer dans l'en-tête.
- Cartes de joueur: le programme permet de garder une trace des actions effectuées au cours du jeu par rapport aux cartes de joueurs et les ajouter dans cette liste.

4.2.3 Plotting

Description et priorité

Niveau de priorité = haute

Le module de traçage va obtenir les données converties obtenues après le processus de recueillir des données et de les tracer dans une interface graphique. Ce module va fonctionner d'une manière similaire de *GnuPlot*.

4.2.4 Calculater ELO

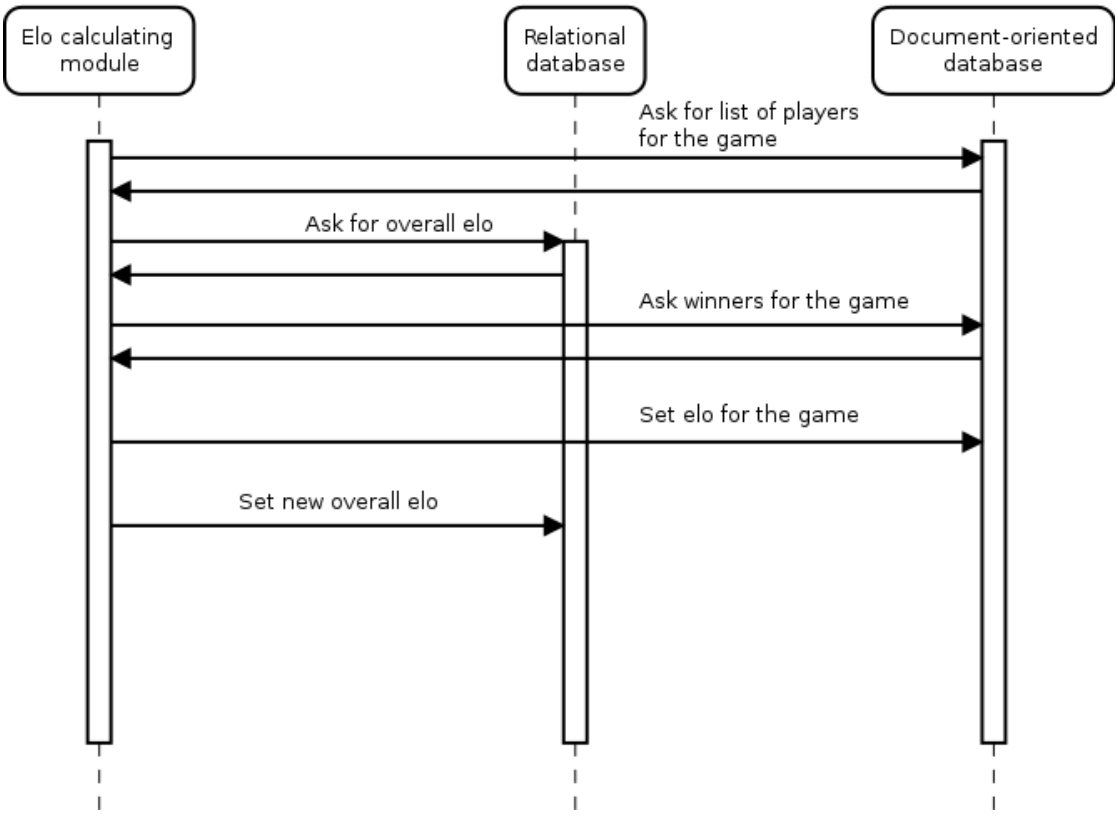
4.2.5 Plotting

Description et priorité

Niveau de priorité = haute

L'analyseur devra calculer la finale ELO pour chaque joueur et ce qui était un joueur ELO sur un jeu donné. Pour chaque jeu dans l'ordre chronologique, ce module va demander au elo initiale des joueurs de la base de données relationnelle et les gagnants du jeu, puis il va calculer la nouvelle elo de chaque joueur de l'issue du jeu. Le module sera ensuite définir le elo calculés dans la base de données orientée documents au jeu en cours d'analyse et mettra également à jour l'elo globale pour chacun des joueurs qui participent au jeu sur la base de données relationnelle.

Plus d'informations sur ELO peuvent être consultés sur https://en.wikipedia.org/wiki/Elo_rating_system
Le schéma suivant illustre le processus du calcul de elo:



4.2.6 Reconnaître les stratégies

Description et priorité

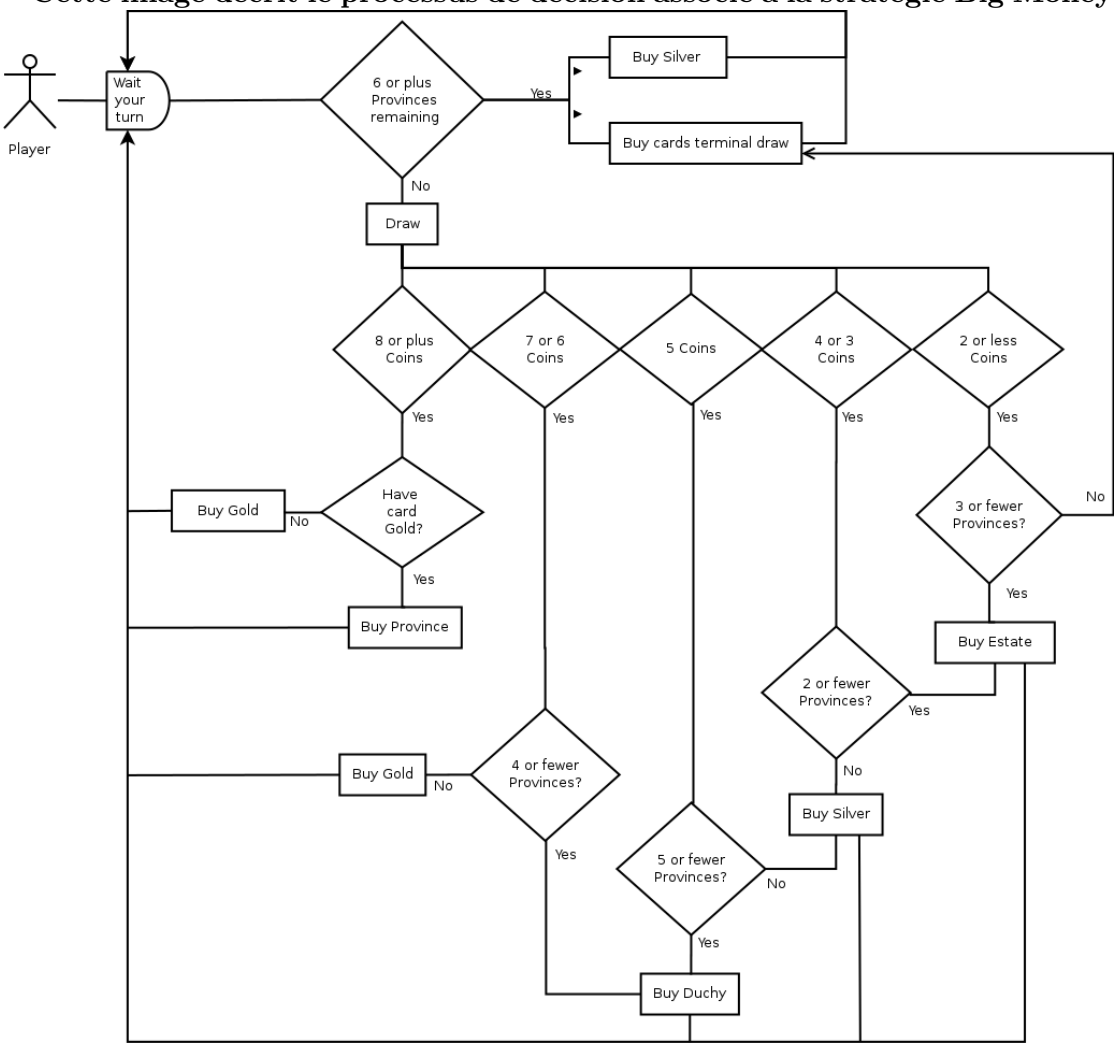
Niveau de priorité = moyen

Le wiki de dominion décrit quelques stratégies qui peuvent être utilisés dans le jeu.

Par exemple :
Big Money
Beyond Silver
Penultimate Province Rule
Pour plus de détails sur http://wiki.dominionstrategy.com/index.php/Strategy

Et l'analyseur devra être en mesure de reconnaître quelle stratégie a été utilisée sur un match donné (si une stratégie a été utilisée) et générer des statistiques.

Cette image décrit le processus de décision associé à la stratégie Big Money



The analyzer has to recognize when a player buys only money cards and specific cards related to the big money strategy. It also has to keep track of the remaining provinces to be bought.
Plus de détails http://wiki.dominionstrategy.com/index.php/Big_Money

Afin de reconnaître la stratégie inventé Beyond Silver, l'analyseur doit reconnaître lorsque certains types de cartes sont achetés, la liste de ces cartes peuvent être trouvés sur : http://wiki.dominionstrategy.com/index.php/Silver#Beyond_Silver

Afin de reconnaître que Penultimate Province Rule est respecté (comme expliqué à http://dominionstrategy.com/2011/03/28/the-penultimate-province-rule/), l'analyseur doit garder une trace de la quantité de Points de Victoire de chaque joueur à chaque tour.

4.2.7 Reconnaître Greening

Description et priorité

Niveau de priorité = haute

L'analyseur doit être capable de reconnaître le moment de ***greening*** sur chaque match. En savoir plus à propos de ***greening*** sur http://wiki.dominionstrategy.com/index.php/Greening.
Le programme va reconnaître quand greening arrive en détectant lorsque les cartes de victoire commencent à être achetées.

5 Besoins non fonctionnelles

5.1 Besoins de performance

Aucun besoin de performance spécifique n'a été faite par le client. Mais pour l'analyseur le temps d'exécution doit être inférieur à une heure.
L'utilisateur doit être capable de parcourir le processus d'analyse sur un ensemble de données précis sans devoir attendre que la même requête à refaire.

5.2 Fiabilité

Le client demande que le nombre maximum de log doivent être analysé, que les données fournies ont des incohérences et quelques logs ne seront pas possibles d'analyser.
L'analyseur devra analyser 100

5.3 Attributs de qualité de logiciels

Même si seulement une ligne de commande est prévue pour le programme final, elle devrait avoir une courbe facile à apprendre et être facile à utiliser.
Les données générées par l'analyseur n'ont pas besoin à être lisible par l'homme.
Les game log générés devront être analysés par l'utilisateur afin de permettre la plus grande souplesse possible dans le traitement des logs.

6 Besoins organisationnels

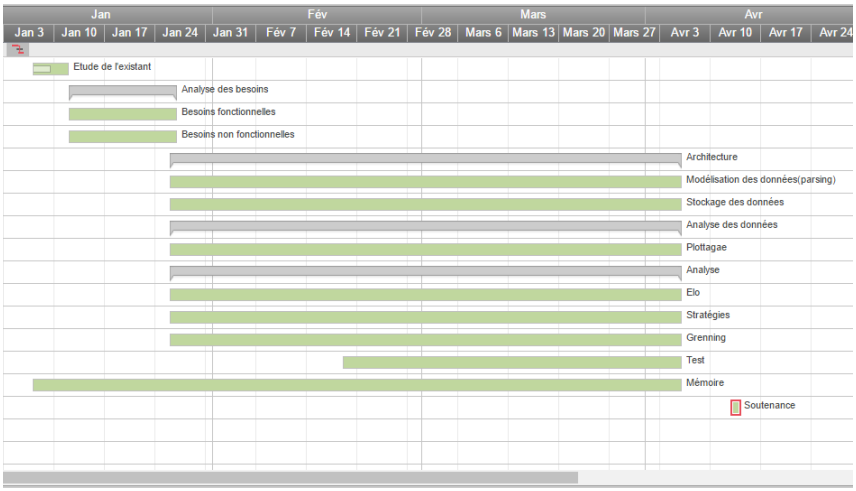
La répartition des tâches du projet et l'estimation de la durée de chacune d'elle sont présentées sur le planning suivant:

6.1 Planning prévisionnel

Tâches

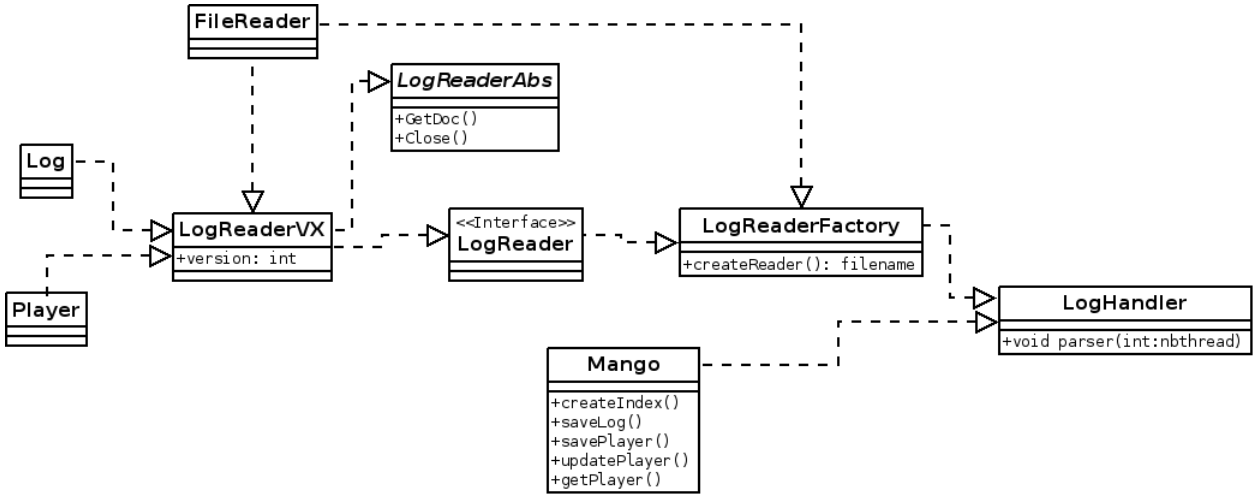
Nom	Date de début	Date de fin
Formulation de groupe	30/11/2015	30/11/2015
Réunion1 avec le Client	02/12/2015	02/12/2015
Réunion1 entre les membres du Groupe	04/12/2015	04/12/2015
Création de dépôt svn + GitHub	04/12/2015	04/12/2015
Réunion2 avec le Client	18/12/2015	18/12/2015
Réunion2 entre les membres du Groupe	06/01/2016	06/01/2016
Etude de l'existant	07/01/2016	11/01/2016
Réunion1 avec le chargé de TD	25/01/2016	25/01/2016
Réunion3 entre les membres du Groupe	29/01/2016	29/01/2016
Réunion3 avec le le Client	05/02/2016	05/02/2016
Réunion4 entre les membres du Groupe	18/02/2016	18/02/2016
Réunion2 avec le chargé de TD	04/03/2016	04/03/2016
Réunion5 entre les membres du Groupe	07/03/2016	07/03/2016
Réunion3 avec le chargé de TD	18/03/2016	18/03/2016
Réunion6entre les membres du Groupe	19/03/2016	19/03/2016
Réunion7 entre les membres du Groupe	21/03/2016	21/03/2016
Réunion4 avec le chargé de TD	31/03/2016	31/03/2016

6.2 Diagramme de Gantt



7 Architecture et description du logiciel

7.1 Structure des classes



8 Fonctionnement et tests

8.1 Fonctionnement de programme

8.2 Test

9 Bibliographie