

# Dominion Datamining

Groupe PdP Dominion Datamining

[2016-04-12 Mar]

# Sommaire

- 1 Introduction
- 2 Analyse du besoin
- 3 Architecture
- 4 Tests
- 5 Choix Techniques
- 6 Implémentation
- 7 Résultats

# Introduction

## Le contexte du projet

- Un serveur ouvert d'Octobre 2010 à Mars 2013
- 12 Millions de logs à traiter (dont la taille générale est 13Go)
- Un Wiki offre des conseils au niveau de la stratégie

# Introduction

## Le but du projet

Dominion DataMining est un programme qui se compose de trois parties principales :

- Modélisation des données
- Stockage des données
- Analyse des données

# Introduction

## Règles du jeu

### Aperçu d'une partie



# Introduction

## Datamining

Une démarche de datamining s'organise selon 5 grandes étapes :

- Définition du problème
- Collecte des données
- Construction du modèle d'analyse testé
- Etude des résultats
- Diffusion

# Analyse de l'existant

## Description des données

- En-tête
- Résumé du match
- Résumé du joueur
- Etapes du jeu

# Analyse de l'existant

## Incohérence dans les données

- Numéro de log
- Nom d'utilisateur
- Données manquantes
- Format du log

# Analyse du besoin

## Besoins fonctionnels

- Interface utilisateur
- Parser
- Stocker les données
- Analyser les données

# Besoins fonctionnels

## Interface utilisateur

- Connection avec la base de données
- Utilisation d'outils
  - proposés par le programme
  - développés par l'utilisateur

# Besoins fonctionnels

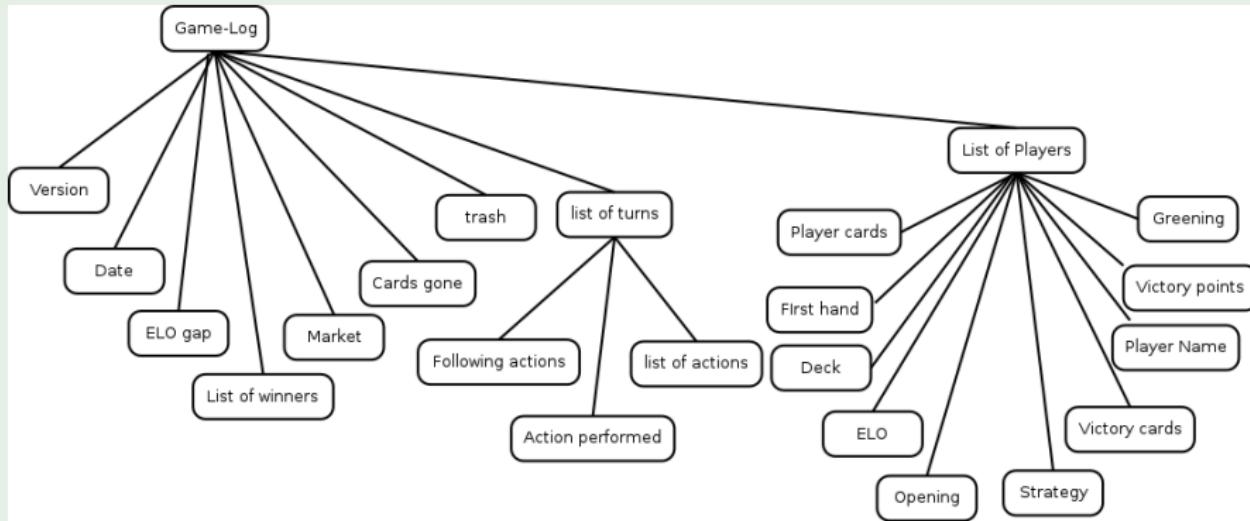
## Parser

- Préparer les données (décompression)
- Format du log

# Besoins fonctionnels

## Creation de l'objet game-log

### Représentation d'un game-log



# Besoins fonctionnels

## Stocker les données

- Création de la base de données
- Communiquer avec la base de données
- Compression des données

# Besoins fonctionnels

## Analyse

- Calcul de l'ELO
- Reconnaissance de stratégies
  - Greening
  - Big money
  - Autres stratégies

# Analyse du besoin

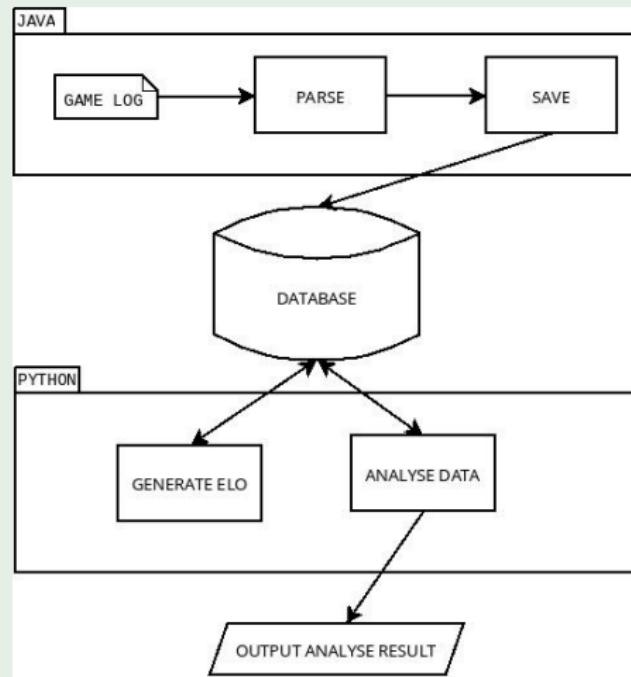
## Besoins non-fonctionnels

- Fiabilité
  - 5 à 10 % de logs ignorés
  - 100% des données d'un log récupérées
- Performance
- Qualités du logiciel

# Architecture

## Architecture globale

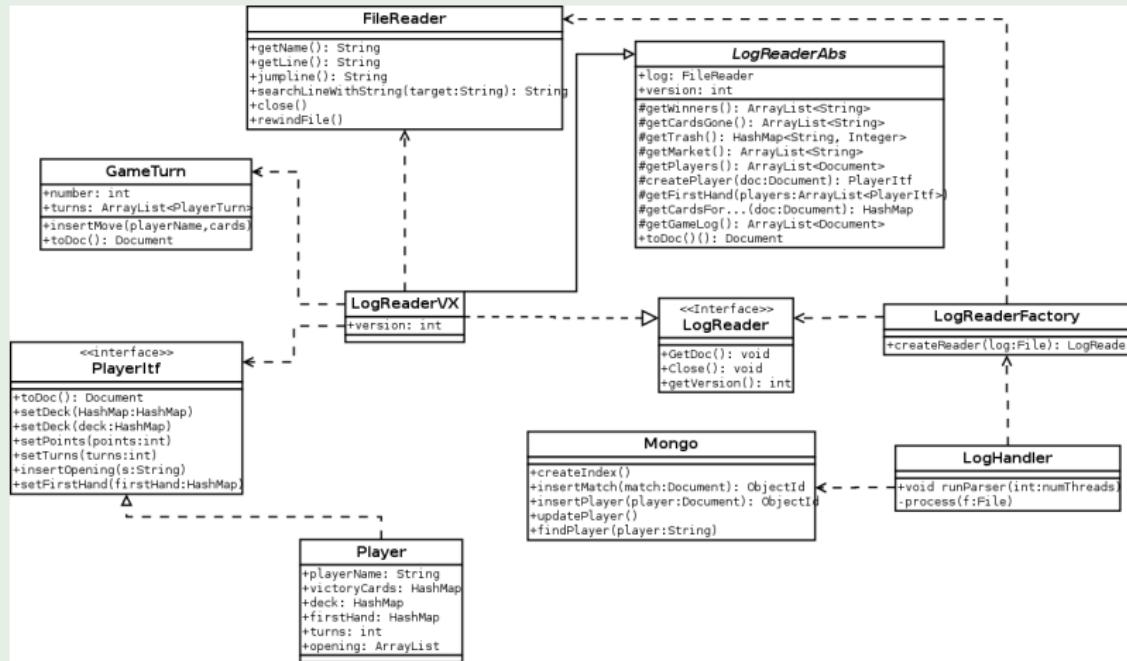
### Représentation globale de l'architecture



# Architecture

## Architecture parser

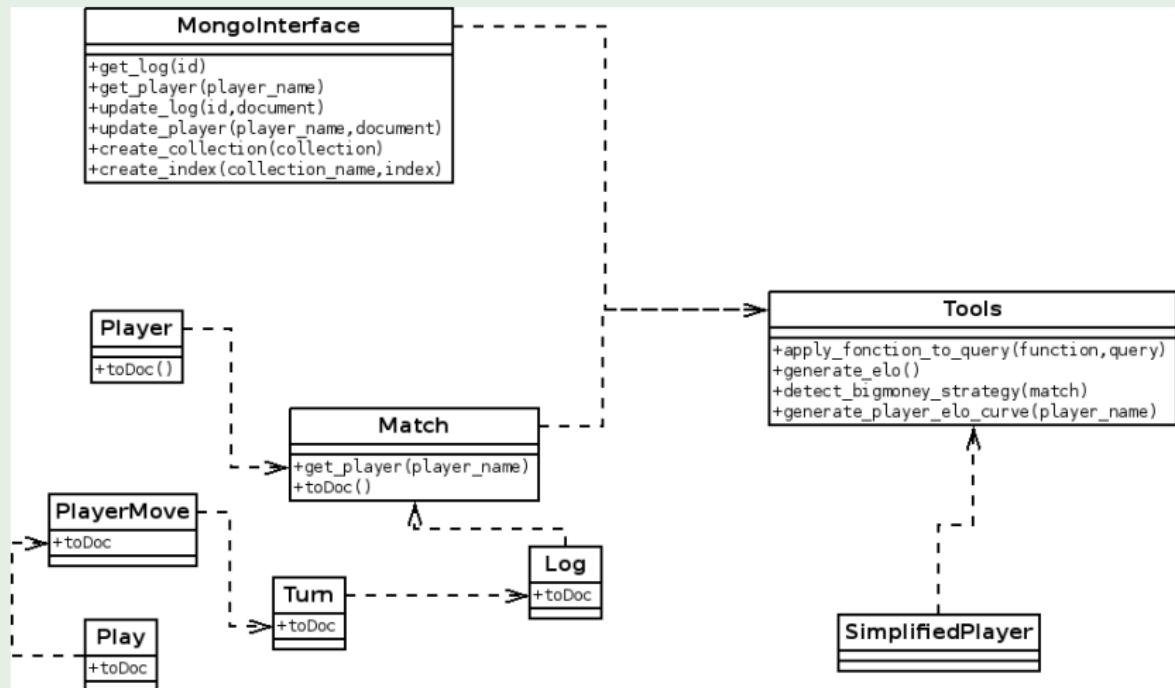
### Diagramme de classe du parser Java



# Architecture

## Architecture bibliothèque

### Diagramme de classe de l'architecture python



# Tests

## Tests parser

### Réultats tests de couverture du parser

Coverage Report - universite.bordeaux.app.GameDataStructure 74 %

Classes in this Package	Line Coverage
<a href="#">GameTurn</a>	83 %
<a href="#">GameTurn\$Play</a>	100 %
<a href="#">GameTurn\$PlayerTurn</a>	100 %
<a href="#">Player</a>	58 %
<a href="#">PlayerItf</a>	N/A

Coverage Report - universite.bordeaux.app.ReadersAndParser 31 %

Classes in this Package	Line Coverage
<a href="#">CheckCard</a>	90 %
<a href="#">FileReader</a>	67 %
<a href="#">LogHandler</a>	0 %
<a href="#">LogReaderFactory</a>	81 %

Coverage Report - universite.bordeaux.app.ReadersAndParser.Readers 74 %

Classes in this Package	Line Coverage
<a href="#">ActionTypes</a>	100 %
<a href="#">LogElements</a>	87 %
<a href="#">LogReader</a>	N/A
<a href="#">LogReaderAbs</a>	70 %
<a href="#">LogReaderAbs\$1</a>	100 %
<a href="#">LogReaderV1</a>	100 %
<a href="#">LogReaderV2</a>	100 %
<a href="#">LogReaderV3</a>	88 %

# Tests

## Tests bibliothèque

- Outils
  - Pytest
  - Mongomock
- Modules Testés
  - Match
  - Tools

# Tests

## Bugs rencontrés

- Cartes au pluriel
- Nom de joueurs à problèmes

# Choix techniques

- Java + Maven
- Python
- MongoDB

# Choix techniques

Java + Maven

## Pourquoi ?

- Langage commun entre les membres du groupe
- Bibliothèques utiles
- Une performance acceptable
- Facile à gérer

# Choix techniques

Python

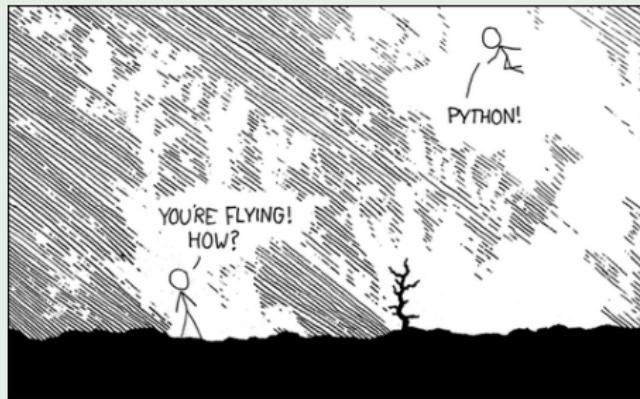
Pourquoi ?

- Demander à Yvan Le Borgne :-)

# Choix techniques

Pourquoi pas R ?

## Courbe d'apprentissage



I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!  
/ HELLO WORLD IS JUST  
print "Hello, world!"

I DUNNO...  
DYNAMIC TYPING?  
WHITESPACE?  
/ COME JOIN US!  
PROGRAMMING IS FUN AGAIN!  
IT'S A WHOLE NEW WORLD UP HERE!  
BUT HOW ARE YOU FLYING?

I JUST TYPED  
import antigravity  
/ THAT'S IT?  
/ ... I ALSO SAMPLED  
EVERYTHING IN THE  
MEDICINE CABINET  
FOR COMPARISON.  
/ BUT I THINK THIS  
IS THE PYTHON.

# Choix techniques

## langage C

### Mauvais choix ?

- C-Types
- C-Modules

# Choix techniques

MongoDB

## Pourquoi ?

- Vitesse
- Schema-less
- Conçu pour les grandes données
- Facile à travailler
- Je voulais apprendre

# Implémentation

- Parser
- Bibliothèque

### Regex + jsoup

- Regular Expressions
- Jsoup
- Data Structure
- Multithreading

# Implémentation

## Bibliothèque

### Python

- Data Structure
- Elo
- Greening
- Plotting
- MonogolInterface

# Résultats

## Phase de parsing

### Aperçu du fonctionnement du parser

```
ebayol@antonella:~/espaces/travail/master1/dominion_datamining$ ./allInOne
./allInOne: ligne 9: mongodb/mongodb-linux-x86_64-3.2.1/bin/mongo: Aucun fichier ou dossier de ce type
downloading mongodb
      % Total    % Received % Xferd  Average Speed   Time     Time   Current
          Dload  Upload Total   Spent    Spent   Left  Speed
100 53,8M  100 53,8M    0     0  9134k      0:00:06  0:00:06  --:--:-- 12,9M
starting database engine
about to fork child process, waiting until server is ready for connections.
forked process: 15612
child process started successfully, parent exiting
Compiling Dominion data mining
Running Dominion datamining

Generating index for logs on the field date
done

Generating index for logs on the field players.name
done

Generating index for logs on the field players.elo
done

Generating index for players on the field elo
done
Strating the Parser it can take a long time...
overall [==>] 7%  Parsing: [game-20101210-072317-d1082767.html]
```

# Résultats

## Calcul d'ELO

### Elo : Code executé

```
In [12]: from DominionAnalyser.Analysers import Tools
from DominionAnalyser.Mongo import MongoInterface
import plotly.plotly as py
import plotly.graph_objs as go

In [2]: Tools.generate_player_table()
         0%                                         100%
[########################################] | ETA: 00:00:00 | Item ID: 5703e6522336eb5ea0c69
850
Total time elapsed: 00:05:51

In [3]: Tools.generate_elo()
         0%                                         100%
[########################################] | ETA: 00:00:00 | Item ID: 5703e5d82336eb5ea0c66
304
Total time elapsed: 00:15:16

In [5]: print(MongoInterface.logs_col.find_one().get("winners"))
['searsjs']

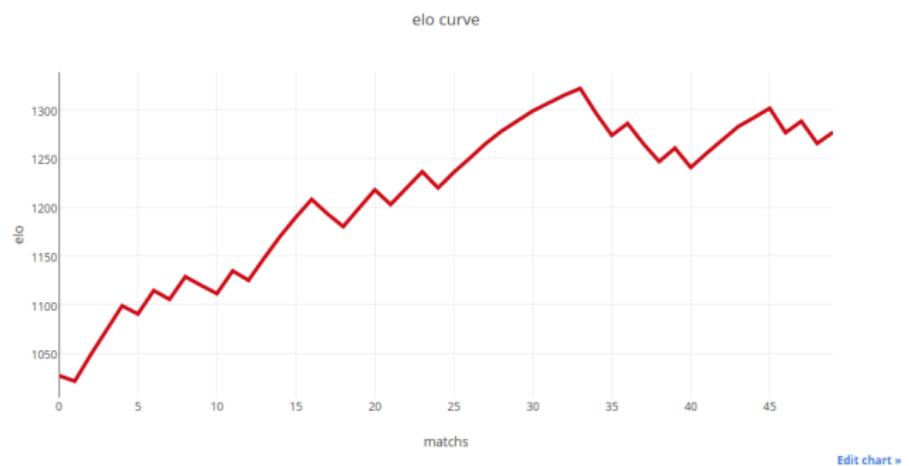
In [6]: playerCurve = Tools.generate_player_elo_curve("searsjs")
         0%                                         100%
[########################################] | ETA: 00:00:00 | Item ID: 5703e4ff2336eb5ea0c5dd9f
Total time elapsed: 00:00:00
```

# Résultats

## Calcul d'ELO

### Elo : résultat obtenu

Out[17]:



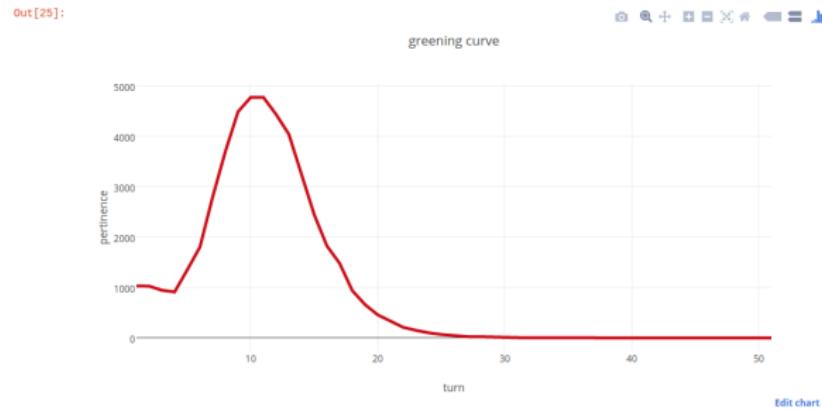
# Résultats

## Calcul du greening

### Greening : Code executé

```
In [18]: greening = Tools.generate_greening_list()
          0%
          #####
          859
          Total time elapsed: 00:02:26
```

### Greening : résultat obtenu



# Questions

