

Techniques de développement de Logiciel [TDLOG]

Rapport de Projet – Beer Games

Élèves : Daniel OKANE, Egor SEDOV, Luis Augusto YOKOTA RIZZO

Paris, 23 janvier 2019.

Table des matières

[Objectifs initiaux 3](#_Toc1136594)

[Architecture du code 5](#_Toc1136595)

[Éléments réalisés 6](#_Toc1136596)

[Difficultés rencontrées 6](#_Toc1136597)

[Organisation du projet 6](#_Toc1136598)

[Formation sur les plusieurs bibliothèques utilisées 7](#_Toc1136599)

[Corrections des bugs entre plusieurs ordinateurs 8](#_Toc1136600)

[Faire le TDD avec l’interface Web/Serveur Flask 9](#_Toc1136601)

Table des figures

[Figure 1 – Site d’OpenAnalytics pour la simulation d’un Beer Game 3](#_Toc1136604)

[Figure 2 – Le site de Tensorflow pour la visualiation des résultats d’une intelligence artificiel 4](#_Toc1136605)

[Figure 3 – Le site de r2d3 4](#_Toc1136606)

[Figure 4 – Les perspectives de chaque acteur dans un projet 7](#_Toc1136607)

# Objectifs initiaux

L’objectif au début du projet était de faire un site de simulation d’un jeu de Beer Game. L’idée étant de montrer la performance des agents qu’utilisent le Machine Learning (Reinforced Learning). L’agent IA était développé par l’équipe Digital d’Argon et l’entreprise veut pouvoir démontrer à ses clients les compétences de ces agents. L’objectif de l’entreprise étant de montrer que l’IA peut surperformer les humains et les formules classiques du monde de la Supply Chain. Donc on avait aussi besoin de faire une page explicative sur les concepts du jeu et la base théorique du Machine Learning d’une façon simple.

* Site d’explications sur les concepts théoriques
  + Le Beer Game
  + Le Machine Learning
  + Le Reinforced Learning
* Site pour faire tourner le jeu
  + Le joueur se positionne dans la chaîne
  + Il jeu les 40 tournées
  + Il voit ses résultats vs la performance de l’agent IA

On a décidé de travailler avec la méthodologie Scrum et nous avons utilisé Trello. Le lien pour accéder au Trello utilisé est <https://trello.com/b/INZeehmY>. Après le *« kickoff meeting »* les objectifs étaient un peu modifiés en fonction du besoin d’Argon. Les clients n’ont pas l’intérêt de jouer une simulation mais simplement de voir les résultats d’une façon explicative. Donc l’idée de la deuxième page a été modifiée.



Figure 1 – Site d’OpenAnalytics pour la simulation d’un Beer Game

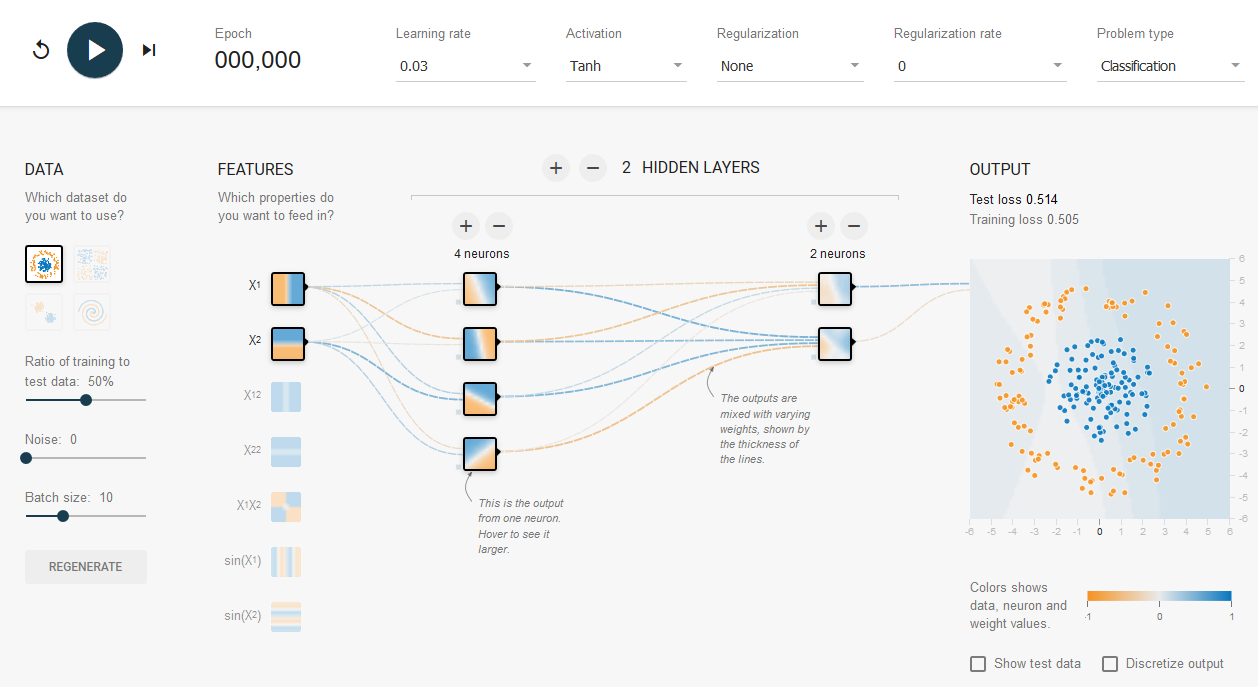


Figure 2 – Le site de Tensorflow pour la visualiation des résultats d’une intelligence artificiel

Le site qui avait inspiré la première conception du projet est <https://beergame.opexanalytics.com/#/> . La version après le rendez-vous initial était plus inspirée dans <http://playground.tensorflow.org>.

Pour la première page on a aussi s’inspiré dans la possibilité de faire des visualisations graphiques plus intéressants qu’utilisent des bibliothèques comme D3.js. Un site que nous a inspiré pour cette page est le site r2d3 (disponible sur <http://www.r2d3.us/>). Avoir le scroll comme moyen d’interaction entre les utilisateurs et le site est devenu une des idées principales, vu sa praticité et simplicité d’utilisation.



Figure 3 – Le site de r2d3

Une autre caractéristique de cette première page est de pouvoir s’adresser à deux publiques distincts. Le premier est la gamme des utilisateurs plus centrés business qui ne veulent pas tous les détails techniques et veulent seulement comprendre les impacts que la technologie peut avoir dans leur business. Le deuxième public est celle des data scientistes qui sont aussi intéressés par les détails techniques. On n’a pas voulu faire deux pages séparés mais surtout cacher les détails techniques dans le code HTML et le rendre disponible à partir des cliques faites par ces utilisateurs.

La première page va réussir si le *storytelling* présenté donne envie aux utilisateurs de voir les résultats de simulations dans la deuxième page. Pour vérifier l’efficacité de cette page on va partager le contenu avec des personnes externes pour obtenir du feedback et améliorer la page d’accueil.

# Architecture du code

Tous les contenus du fichier UX étaient développés par le groupe dans le contexte du projet. Les autres fichiers étaient surtout résultat des autres projets faites par Argon et pour un projet Mélusine 3A du département de Génie Industriel de l’École des Ponts. L’architecture base de notre projet est la suivante :

* Dossier avec les fichiers pour faire tourner la simulation
  + Fichiers peu utilisés dans le contexte de ce projet
* Dossier avec les fichiers pour faire tourner le serveur Web/Flask
  + Fichiers Python pour faire tourner le serveur
  + Templates
    - Pages HTML
      * Structure du code HTML
      * JavaScript pour faire les interactions
  + Statique
    - Images
    - CSS
    - JS
  + Documentation
    - Le rapport

L’interaction entre les deux dossiers principaux se fait à partir du fichier Simulation.py. Le fichier va recevoir du serveur Flask les attributs de la simulation choisi par l’utilisateur dans le site/chez le client. Ensuite le programme doit chercher entre les agents IA pré-entrainés (des fichiers générés par TensorFlow) l’agent correspondant. Avec l’agent récupéré, le programme exécute la simulation avec l’agent intelligent et une autre avec les agents de benchmark. Les agents de benchmark sont les agents qu’utilisent des formules de *supply chain* classiques et qui seront comparés aux agents IA pour montrer aux clients la puissance de l’IA.

Une fois que la simulation est faite, le serveur rendre des résultats sous le format de dictionnaire sur python. Ces dictionnaires sont après interprétés et envoyés au client par l’utilisation du JSON. Les JSONs sont ensuite utilisés par le code Java Script pour rendre les graphiques dynamiques.

Les deux pages HTML sont indépendants sauf pour les fichiers de CSS. Les deux utilisent le CSS de Bootstrap customisé par le groupe. La page 1 d’introduction n’a pas de communication avec le serveur, mais la page 2 a plusieurs interactions avec le serveur pour faire tourner la simulation.

# Éléments réalisés

Grâce aux réunions fréquentes avec l’équipe d’Argon pour nous accompagner et orienter dans le projet, le résultat obtenu est assez proche des objectifs initiaux définis ensemble. Comme prévu initialement, le site contient :

* Première page avec des explications sur les concepts théoriques et des éléments visuels qui motivent le visiteur à jouer la simulation ;
* Deuxième page qui affiche chaque étape de la simulation et ses résultats. Dans cette page, le visiteur peut choisir des paramètres personnalisés selon ses besoins ou des scénarios standards qui correspondent à une industrie. Il est possible de changer les paramètres et tourner plusieurs simulations sans avoir besoin de relancer la page, pour permettre une meilleure expérience d’utilisateur.

Cependant, à la fin d’un projet il est normal d’avoir quelques éléments qui étaient initialement prévus et n’ont pas été faits. La fonctionnalité de sauvegarder un ensemble de paramètres, par exemple, n’a pas été ajouté car, pendant le développement du projet, nous avons observé que augmenter les numéros de paramètres irait faire la quantité des simulations croitre de manière exponentielle. Comme seulement les paramètres vraiment essentielles ont été gardés, il serait facile pour le visiteur de choisir l’ensemble de paramètres souhaité plusieurs fois.

Une autre différence qui ne concerne pas le résultat final, mais le développement du projet, est la méthodologie *Test Driven Development*. Même si au début du projet nous avons pensé en utiliser cette méthode en suivant les conseils des professeurs du cours, à cause des difficultés techniques (expliquées dans la prochaine partie du rapport) et du fait que l’entreprise client ne l’utilise pas, elle n’a pas été considérée comme une priorité.

# Difficultés rencontrées

On peut regrouper les problèmes rencontrés pendant le projet dans 4 catégories principales. On va expliquer avec plus de détails les problèmes rencontrés sur ces catégories dans les sections suivants.

## Organisation du projet

Avoir un « client » réel a beaucoup des avantages mais aussi quelques inconvénients. D’un point de vu d’organisation de projet, dans le contexte du cours, les soucis principales d’avoir un client sont les suivants :

* Concilier les agendas des tous les personnes pour faire les points hebdomadaires (fondamentales dans la méthodologie Scrum)

Avec les différents cours pris par les membres du groupe (des élèves en 3A et pas les 2A IMI qui ont les mêmes cours) et les disponibilités des collaborateurs d’Argon, on a eu des difficultés pour trouver des créneaux où tous les membres pouvaient participer de la réunion. Comme pour la méthodologie Scrum c’est important d’avoir tous les membres du projet, c’est compliqué de gérer ces demandes externes en plus des contraintes du groupe.

Sur certaines semaines on a dû faire la réunion sans un des membres du groupe. Mais après, pour pouvoir bien rattraper le contenu de la réunion et en plus la révision des points de chaque tâche, le temps dépensé était considérable.

* Perte de contrôle sur la décision final : changement des besoins et souhaites, nouvelles idées constamment

Pour établir les Sprints à chaque semaine et faire l’évaluation de l’avancement on fait toujours le point avec les clients. Parfois l’équipe est satisfaite avec l’état actuel du projet mais le client veut faire des modifications parce qu’il se rendre compte que l’idée précédente n’était pas la meilleure. En plus, comme à chaque Sprint on va créer une nouvelle liste de tâches parfois des nouvelles idées sont ajoutés au projet sans forcement prendre en compte leur difficulté ou les autres choses qui étaient prioritaires pour l’avancement. Ces itérations constantes avec le client donnent force à une vision plus comédique sur la gestion de projet :

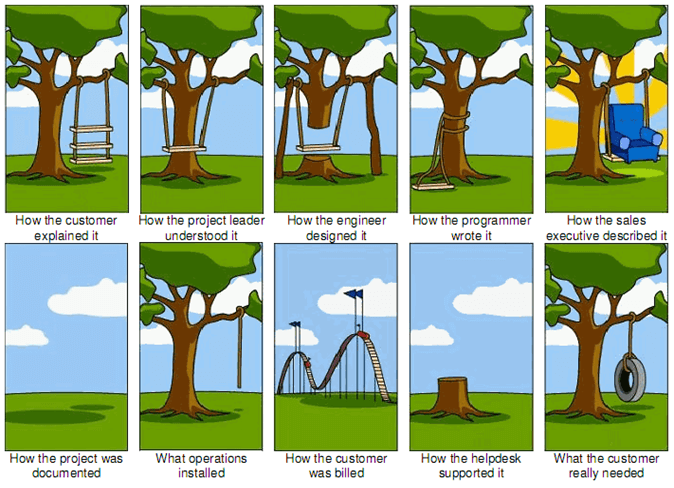


Figure 4 – Les perspectives de chaque acteur dans un projet

* Partage du Git avec les professeurs

Pour Argon la partage du code et le Git avait deux soucis principaux : la confidentialité du code développé en dehors le contexte du projet de l’interface graphique et la limite des utilisateurs dans le Git de l’entreprise. C’est une question mineure mais désagréable pour le suivi du développement.

Et en dehors les difficultés liées au fait que le groupe avait un client externe on a aussi rencontré des difficultés d’organisation interne.

## Formation sur les plusieurs bibliothèques utilisées

Liste des formations (langages de programmation et bibliothèques spécifiques utilisés) :

* Python
  + Lecture de Json
* Flask
  + Concepts de base
  + Interactions Flask-Javascript
* TensorFlow
  + Savoir récupérer les fichiers TF pour les mettre en utilisation
* HTML 5.0
  + Concepts de Base
  + CSS
* Bootstrap
  + Site disponible sur version mobile/en fonction de la taille de l’écran
* Java Script
  + Concepts de Base
  + Json
  + Interactions Flask-Javascript
* Chart JS
  + Création des graphiques interactives
  + Création des graphiques dynamiques à partir des fonctions JS
* JQuery
  + Modification des éléments dans la page de forme dynamique

La difficulté avec tous ces formations était surtout liée au fait qu’on ne savait pas jusqu’à où avancer dans la formation avant de commencer à concrétiser le code. Le Web development est un domaine vaste qui a des applications très diverses donc choisir bien la cible qu’on devait étudier c’était un peu compliqué vu l’inexpérience du groupe. L’orientation de Thierry et le suivi des objectifs avec Argon étaient utiles pour continuer l’avancement mais on a suivant pris des retards par apport aux objectifs établis dans les Sprints en fonction des décalages de formation et du besoin de retourner vers les formations plus basiques.

On a eu besoin de faire plusieurs allers et retours. Surtout dans la partie de communication entre serveur et client (navigateur), on a eu du mal à trouver des exemples qui on vraiment pu nous aider à bien avancer. C’est normal de rencontrer ce type de difficulté mais je pense que tous les années les élèves font des projets similaires, donc peut-être il serait possible de construire une liste des ressources et formations partagés entre tous les groupes. Par la liste de projets, on a pu constater qu’il y avait d’autres projets Web qui probablement ont utilisé des mêmes bibliothèques et sources que notre groupe.

## Corrections des bugs entre plusieurs ordinateurs

Dans le cadre de notre projet on a dû développer le code et le site à partir de trois ordinateurs différents. Un Linux et deux Windows. On a aussi mis l’application sur un serveur Linux (Pythonanywhere). Pythonanywhere est un service gratuit que nous avons trouvé pour héberger le site et disponibiliser le contenu pour l’utilisation mobile aussi. L’idée c’est qu’à la fin du projet on va passer les fichiers à l’équipe Argon qui va ensuite payer pour plus de « *bandwith* » dans le serveur pour pouvoir commencer à partager le site avec les clients.

Les problèmes que nous avons rencontrés étaient surtout liées à l’installation des certaines bibliothèques et l’exécution du fichier Flask pour faire tourner le serveur. Sur un des ordinateurs on n’a jamais réussi à faire tourner le serveur sans l’utilisation de Spyder (intégré dans Anaconda). Donc le code marchait bien sur deux ordinateurs mais pas pour le troisième. C’est un problème mineur mais qui doit être documenté. TensorFlow en spécial a causé de écarts entre l’exécutable du code dans les différents ordinateurs, mais on n’a jamais trouvé une solution parce que dans le serveur les fonctionnalités on bien fonctionné.

Les différences entre navigateurs sont aussi importantes à noter pour que dans les années prochaines les élèves puissent avancer plus rapidement. Sur une des bibliothèques on avait testé le code plusieurs fois sans succès. C’est après quand on a changé de Chrome vers Firefox que le code a commencé à bien marcher. Donc il faut se faire attention à la compatibilité de chaque morceau de code avec les différents navigateurs dans le développement Web.

## Faire le TDD avec l’interface Web/Serveur Flask

Dès le début nous avons eu des difficultés en appliquer la méthodologie de Test Driven Development. Déjà pour la partie python les fichiers faites par Argon étaient 99% prêts à utiliser mais ils n’étaient pas conçus avec le TDD comme base. Pour toutes les interactions entre serveur et recevoir les données la difficulté n’était pas si grand dans la coté serveur.

Cependant, pour faire les tests dans le client et navigateur on a eu plus de difficulté pour concevoir les tests. Thierry a proposé le Headless testing mais on n’avait pas compris comment l’utiliser en intégration avec la méthodologie TDD. Ces tests de fonctionnalité et réactivité de la page Web ne sont pas difficiles, mais comment changer la conception du code pour que les tests soient intégrés est moins évident.