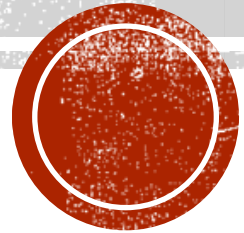
PROJET JAVA



XM - GUI

OBJECTIF DU PROJET

- **Problème à résoudre**

Dans le petit monde de la théorie de la décision, il existe un logiciel gratuit permettant d'envoyer des problèmes multi-critères d'aide à la décisions à des services web (ceux du Decision-Deck) pour résolution : Diviz.

Par exemple, en envoyant un ensemble d'alternatives (représentant différents appartements), chacune évaluée sur un plusieurs critères (la superficie, le prix, …), il est possible de demander à ces services web d'appliquer un algorithme pour résoudre ce probème(par exemple une somme pondérée)

Seulement, Diviz utilise en entrée différents fichiers XML compliqué à produire car ces documents doivent respecter les standard XMCDA pour être compris pas les web services du Decision-Deck.

On se propose de faciliter la vie de ces étudiants et chercheurs utilisateurs de Diviz….

OBJECTIF DU PROJET

- **Solution**

Une interface graphique très simple d'utilisation permettant de :

1. créer un problème multi-critères d'aide à la décision (à minima un ensemble d'alternatives, de critères et leur table d'évaluation)

2. Sérialiser ces information en documents XMCDA

3. Envoyer le MCP aux services web du Decision-Deck et récupérer le résultat de l'algorithme

PARTICULARITÉS DU PROJET

- **Réutilisation de code et standards existants :**
  - Documents XMCDA
  - Invocation des we bservices
- **Sérialisation et dé-sérialisation (XML) au cœur du projet**
  - Alternatives
  - Critères
  - Table de performance
  - Classement d'alternatives
- **Utilisation de web services particuliers**
  - API SOAP du Decision-Deck
  - Système de tickets

EQUIPE

Première paire de programmeurs

- **Ayoub**
  A déjà développé en Java, le plus à l'aise du groupe techniquement
- **Raphaël**
  N'a pas programmé en Java, ni programmé tout court sur un projet de cette ampleur

Deuxième paire de programmeurs

- **Louis** :
  N'a pas programmé en Java, mais connait d'autres langages (R, SAS)
- **Ambre** :
  N'a jamais développé en Java

FONCTIONNEMENT ADOPTÉ

- **Pair-programming**
  En modifiant les équipes lors des premières fonctions demandées.
  Puis en les fixant car plus pratique pour avancer efficacement :
  - Ayoub – Raphaël
  - Louis - Ambre

- **Régularité** :
  Davantage de commits sur la première moitié du projet, puis le rythme était moins intensif par la suite, mais régulier pour l'équipe 1.

- **Communication** :
  - Pour le projet : groupe Messenger et emails
  - Pour une équipe : appels téléphoniques, SMS, séances de code à deux sur la même machine

DIFFICULTÉS RENCONTRÉES (1/2)

- **Gestion de projet**
  - Du mal a travailler en groupe, les disponibilités et contributions de chacun pour travailler sur le projet étant très inégales
  - Blocages entre équipes dus à des fonctionnalités non terminées mais nécessaires pour avancer
  - Plus généralement, du mal à saisir la cohérence des fonctions demandées. Nous n'avons initialement pas réussi à identifier les fonctions nécessaires dans la construction d'un MVP et les autres « optionnelles ».

DIFFICULTÉS RENCONTRÉES (2/2)

- **Techniques**
  - Beaucoup de réutilisation de formats et codes déjà existants car ce logiciel doit communiquer avec les web services du Decision-Deck (standard XMCDA)
    - Sans doute proche de la réalité pour un programmeur
    - Mais de nombreuse heures nécessaires pour se familiariser avec cet existant
  - Plusieurs outils / concepts à découvrir et à utiliser en autonomie car nécessaires avant de les aborder en cours (voire non abordées)
    - Maven
    - Sérialisation
    - DOM
    - Web services
    - …

AXES D'AMÉLIORATION

- **Meilleure répartition du travail**
  - Compréhension globale du projet et des fonctions demandées serait améliorée si chacun participait à toutes les étapes
  - Eviterait les périodes de rush générant du code de moins bonne qualité
  - Eviterait les blocages entre les équipes

- Demander davantage de guidance, systématiquement
  - Validerait notre démarche (avant de se lancer)
  - Générerait moins d'heures « perdues » sur une fausse piste
  - Permettrait de s'appuyer sur des fichier exemples

UTILISATION DU LOGICIEL XM-GUI

- **Pour qui ?**
  - Pour les utilisateurs des algorithmes de résolution de problèmes multi-critères d'aide à la décision du Decision-Deck (étudiants, chercheurs,…)
  - Utilisateurs de Diviz souhaitant éviter le travail fastidieux de création manuelle des fichiers XML (documents XMCDA) représentant leur problème.

UTILISATION DU LOGICIEL XM-GUI

- **Comment ?**

Utilisation du logiciel XM-GUI pour :

1. Créer des alternatives, des critères, et leur table de performance via une simple interface graphique
2. Sérialiser ces informations en 3 documents XMCDA
3. Envoyer le problème multi-critères au service web de leur choix et récupérer le résultat de l'algorithme

DÉMONSTRATION

- **Avec le GUI basique**
  Création et sérialisation d'Alternatives et de Critères
  - Classes contract1 et contract2
  - Classes file1
  - Classes basicGui
- **Avec le GUI avancé**
  Création et sérialisation de la table de performances en plus
  - Classes evaluationsGui
- **Envoi à un service web basique**
  Récupération de la réponse de l'algorithme (classement des alternatives)
  - Classes WsCallRank et WSCall

PROCHAINE ÉTAPES

- **Rendre les appels aux web services génériques**
  Permettre d'appeler d'autres services web (donc d'utiliser d'autres algorithmes) du Decision-Deck
  - D'autres documents XMCDA à prendre en charge (via GUI)
  - D'autres formats de réponses à prendre en charge

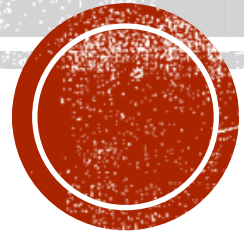
- **Améliorer le GUI**

- **Améliorer la gestion des tickets**
  Pour un job plus long, permettre de requêter facilement sur l'avancée d'un algorithme en cours

BILAN

- Projet challengeant, à la finalité motivante (utilisation par des étudiants et chercheurs)
- Nouvelles compétences en développement Java
- Mauvaise gestion de projet
- Beaucoup de travail en autonomie (découvrir et manipuler des concepts nouveaux, sous les contraintes du standard XMCDA à respecter)

PROJET JAVA



XM - GUI