Compte rendu de réunion n°8

**Objet** : Présentation de l'avancé l'implémentation de la structure générique, présentation d'idées pour la structure de l'overlay.

Date : 27/04/2015 à 10h30.

Participants : Véronique DESLANDRES (**VDE**), Salima HASSAS (**SHA**), Kevin ESPENEL (**KES**).

Rédigé par : Kevin ESPENEL.

**Ordre du jour :**

1 – Présentation de l'implémentation de la structure générique des propositions

2 – Architecture de la couche overlay et intégration à la structure globale

3 – Organisation pour la suite du stage

4 – Remarques / Questions

**Déroulement :**

Pour garder une trace précise de ce qui a été dit dans l'ordre du jour, j'y ai reporté mes notes concernant tout ce que j'ai souhaité aborder durant la réunion. Les réponses et les échanges sont représentés en **vert**.

**1 – Présentation de l'implémentation de la structure générique des propositions**

Avancement du développement :

- Les parties Variable, Domaine et Condition sont implémentées.

- Il reste à faire les classes Contraintes et leur lien avec les Conditions ainsi que d'implémenter la classe Proposition qui utilise toutes les autres classes conçues avant.

- Terminer le plus rapidement l'implémentation générique en parallèle de l'implémentation de l'overlay pour pouvoir tester rapidement

Structure des propositions :

- La proposition est l'intermédiaire de communication entre l'agent et l'overlay.

- Une collection de méta-informations sert de **boite aux lettres des messages entre l'agent et l'overlay**. Structure très générique permettant de transmettre tous types d'information, pour permettre d'ajouter et d'enlever facilement des informations.

**OK pour cette approche qui permettra d'ajouter facilement n'importe quel type de méta-information.**

**2 – Architecture de la couche overlay et intégration à la structure globale**

- L'overlay a une structure dynamique dans laquelle il est facile d'inclure de nouveaux types de nœud

- Les nœuds de l'overlay peuvent également échanger des informations entre eux

- L'OverlayManager centralise la création des nœuds de l'overlay. Si cet objet n'est pas présent, le programme se comporte comme dans l'ancienne version => permet de faire des **comparatifs de performance avec/sans overlay**.

**Bien de pouvoir faire des comparatifs pour l'aspect recherche.**

- Scénario de création :

=> L'agent ajoute une nouvelle proposition

=> L'environnement applique l'algorithme de détection de conflits et lie les propositions en conflit.

=> L'overlay est notifié des conflits et relie les propositions concernées par un nouveau nœud de l'overlay

=> L'overlay peut alors influencer le comportement de l'agent en conflit en inscrivant des méta-informations sur sa proposition

**3 – Organisation pour la suite du stage**

- Tests unitaires

- Javadoc

- Modélisation des principales phases avec UML

- Récupération de jeux de tests

- Adaptation du processus de résolution à la nouvelle structure des propositions

- Enrichissement du comportement des agents (échange avec l'overlay)

**=>La modélisation UML n'est pas prioritaire, elle sera réalisée quand l'application sera stabilisée.**

**=> Il faut d'abord stabiliser l'architecture et faire des tests pour comparer avec l'ancienne version.**

**=> Faire ensuite des benchmarks avec des jeux de données existants pour comparer les performances de la résolution avec les autres solutions existantes**

**=> S'il reste du temps, étudier des frameworks graphiques pour représenter les performances sous forme de graphiques plutôt qu'avec des macro excel comme l'ancienne solution.**

**4 - Remarques / Questions**

**Prochaine réunion lundi 04/04 à 15h**

**Ok pour que je travaille de chez moi le mercredi 29/04 après-midi**

**Actions :**

KES

- Deux semaines pour intégrer la structure générique à l'ancienne solution et vérifier que les résultats sont bons.

VDE / SHA

- Eventuelles remarques ou idées pour améliorer les solutions proposées.

- Mise à disposition d'articles intéressants pour la biblio sur le wiki.