Projet C++ : SMA Système Multi_Agents

December 7, 2011

1 Système MultiAgents

Un système multiagent (SMA) est un ensemble d'agents qui évoluent dans un environnement commun. Cet ensemble d'agents, pas nécessairement intelligents, constitue un système complexe d'où se dégage une intelligence collective. Cette intelligence provient de l'émergence d'un comportement global (Gasser, 1992)

2 Les Agents:

Un agent est une entité informatique réactive, proactive et dotée de capacités sociales, capable d'agir de manière autonome dans un environnement (Wooldridge, 1999). La réactivité fait référence au maintien d'un lien constant avec son environnement afin de répondre aux changements qui y surviennent. La pro-activité signifie que le système génère et satisfait des buts : son comportement n'est donc pas uniquement dirigé par des événements. Les capacités sociales indiquent que le système est capable d'interagir ou de coopérer avec d'autres systèmes. Les agents peuvent être décrits à différents niveaux d'abstraction : le modèle, l'architecture et l'implémentation (Wooldridge et al., 1995). Le modèle d'agent décrit l'agent, ses propriétés et comment on peut les représenter. L'architecture est un niveau intermédiaire entre le modèle et l'implémentation. Elle organise la construction de l'agent, de ses capacités et de son contrôle, conformément au modèle. L'implémentation assure la réalisation pratique de l'architecture des agents à l'aide de langages ou outils de programmation.

3 Modularité des SMA:

Les systèmes multiagents ouverts (Vercouter, 2000) partagent les caractéristiques des systèmes ouverts. Les entités élémentaires du système, que sont les agents, n'ont pas la possibilité d'avoir une représentation complète de l'environnement. Le système dans sa globalité, doit être modulaire et extensible. La modularité concerne le fait que le système multiagent est composé de plusieurs soussystèmes mis en relation. Ces soussystèmes ont

chacun leur propre mode de fonctionnement. L'extensibilité se traduit par le fait que le système multiagent supporte l'ajout et le retrait dynamique d'éléments. Ces éléments peuvent bien sûr être des agents, mais aussi des rôles, des connaissances ou des liens.

4 POO et SMA

les SMA définissent un paradigme de programmation qui étend la notion de programmation orientée-objet (POO) en y ajoutant les notions de processus et d'autonomie. Comme les objets, les agents sont caractérisés par un ensemble de données internes (encapsulées), éventuellement de données externes, et de capacités (ce qui correspond aux méthodes en POO).Mais parce qu'un agent est aussi un processus (thread) et parce que les agents interagissent (donc s'envoient des messages au sens POO du terme), les problèmes d'inter-blocage dans le système imposent de repenser le paradigme de programmation. En particulier, l'envoi de message entre agents ne peut plus être synchrone : dans un SMA, les agents s'envoient réellement des messages qu'ils stockent dans des boites aux lettres. De plus, parce que les agents sont autonomes (et donc décident localement des opérations à effectuer), les interactions ne peuvent plus être vues comme des appels de méthodes où un agent déclencherait un traitement chez un autre agent. Chaque agent peut en effet décider de ne pas traiter un message qu'il a reçu ou d'y répondre d'une manière nonconforme à ce que l'émetteur attend. C'est pourquoi le programmeur doit définir un ensemble de protocoles d'interaction qui permettent de spécifier les envois de messages possibles dans le SMA.

4.1 Différences entre POO et POA

Le tableau suivant montre les principales différences entre la POO et la POA:

	POO	POA
Unité de base	objet	agent
Paramètres définissant l'état de l'unité de base	pas de contraintes *	croyances, décisions, obligations, habiletés *
Processus de calcul et méthodes pour la réponse	envoi de messages *	envoi de messages *
Types de messages	pas de contraintes *	informer, demander, offrir, promettre, accepter, rejeter, ...
Contraintes sur les méthodes	pas de contraintes *	consistance, vérité, ... *

4.2 Méthodologies orientées agent basées sur l'orientation objet :La méthodologie AAII (Australian Artificial Intelligence Institute)

L'Institut australien d'intelligence artificielle a développé plusieurs systèmes multi-agents en utilisant leur technologie BDI-PRS (Belief-Desire-Intention - Procedural Reasoning System). La méthodologie AAII a été développée en partant de l'expérience accumulée

pendant la construction de ces systèmes. Dans cette méthodologie, on a un ensemble de modèles qui, dès qu'on les a élaborés entièrement, définissent les spécifications des agents. Il y a un modèle externe qui spécifie : (i) le point de vue système, notamment les agents et les relations entre ces agents ; (ii) le modèle d'agent, notamment les classes d'agents et les instances associées ; (iii) les rapports d'héritage entre les classes d'agents et les instanciations de ces classes au moment de l'exécution. Chaque agent a trois attributs : croyances, désirs et intentions, et on peut spécifier comment un agent hérite de ces attributs dans la hiérarchie des classes. Il y a aussi un modèle interne qui représente l'implémentation de l'agent. Les étapes pour concevoir un système multi-agents en utilisant la méthodologie AAII sont :

- Identifier les rôles pertinents pour le domaine d'application et développer une hiérarchie de classes d'agents ;
- Identifier les responsabilités associées à chaque rôle, les services exigés et fournis par le rôle et les buts associés à chaque service ;
- Pour chaque rôle, déterminer les plans qui peuvent être utilisés pour accomplir le rôle et les conditions du contexte pour lesquelles un plan est opportun ;
- Déterminer la structure de croyances du système, notamment l'information exigée par chaque plan et chaque but à accomplir.

4.3 Architecture BDI

Une architecture BDI est conçue en partant du modèle "Croyance-Désir-Intention", en anglais "Belief-Desire-Intention", de la rationalité d'un agent intelligent.

5 Travail demandé

5.1 Modélisation

Elaborer le diagramme de classe ainsi que le diagramme de séquence en vous appuyant sur la figure 1. En partant de votre modèle de SMA , Modéliser une vente aux enchères : pour plus d'inspiration cf : `http://turing.cs.pub.ro/auf2/html/chapters/chapter5/chapter_5_3_2.html`

5.2 Implémentation

Implémenter votre modèle en c++.
Ecrire les jeux de test necessaires: lors de l'execution du programme on devrait voir les messages suivants:
Lancement des enchères :
Prix de lancement pour l'objet A
—Agent1
—Agent 2

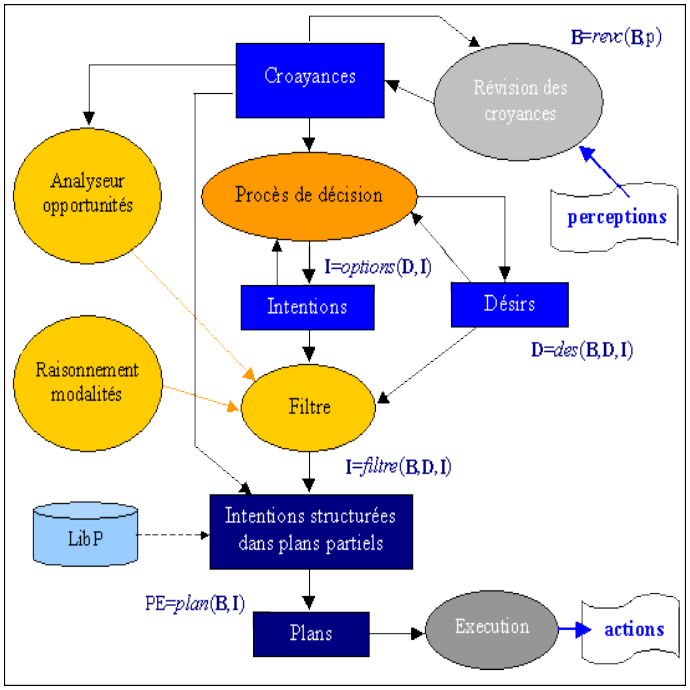

Figure 1: Figure 1 :Architecture BDI

....

Agent N emporte les enchères
Agent N est content (ou pas)
...
cloture des ventes

6 Bibliographie

`http://turing.cs.pub.ro/auf2/html/chapters/chapter6`
`www.irit.fr/GDR-I3/.../09-ModelisationEtSimulationMultiAgents.pdf`

7 Organisation

Il sera demandé le jour de la soutenance (encore à déterminer, mais courant Janvier 2012) :

- de fournir des explications raisonnées sur les choix opérés
- de produire des schémas UML obtenus à l'aide d'un logiciel quelconque de modélisation, pas à la mainde la structuration objet

- d'être capable de commenter l'implémentation réalisée
- d'être capable de faire une démonstration fonctionnelle du programme (on entend par démonstration fonctionnelle : une exécution qui ne plante pas et qui fournit un résultat raisonnable).

Le projet peut être réalisé au plus par trinôme et au moins par binôme, sans dérogation. Chaque intervenant devra rendre compte de sa contribution dans la production finale. Les équipes projet doivent être constituées à l'avance, déclarées au plus tard le 15 décembre 2012 auprès du responsable du cours (par mail faisant état des noms et prénoms des individus concernés) et, leur constitution ne pourra être modifiée sauf cas de force majeure.
Le(s) programme(s) devront impérativement être écrits en langage C++, et être exécutables sur une machine de l'UFR, ou sur son propre ordinateur portable.
Un rapport de quelques pages devra être envoyé aux chargés de TD . Ce rapport devra expliciter clairement l'ensemble des points les plus importants du projet. Il devra aussi contenir le code source du projet (ne pas hésiter à utiliser les commentaires pour expliquer vos déclarations et penser à utiliser des noms de variables assez parlants).
Tout choix d'implémentation devrait être justifié (les droits d'accès(public,protected ..), les structures de stockage , héritage ...)

La soutenance est obligatoire (sauf pour les dispenses officielles), toute absence conduira le jury à délivrer la note 0. Chaque personne devra intervenir (les silences seront jugés très négativement).