USTHB
FEI, Département d'informatique
Master S2I, S2

Bab-Ezzouar le 27 Mai 2015

EMD Technologie des agents

Exercice1 (Préparation d'une fête) (10 pts)

Préparer une fête demande des efforts importants et la qualité des résultats recherchés dépendent des organisateurs de l'évènement et des paramètres liés à l'évènement comme le nombre des invités, le budget à mettre pour l'événement, la logistique, la restauration et l'endroit de la rencontre. Habituellement, toute une équipe de personnes s'attelle à partager les tâches pour réussir l'événement. Une façon de répondre aux exigences de la qualité est de développer un système mono-agent virtuel à cet effet. Le cahier de charge adressé à l'agent est :

- Type d'événements (mariage, conférence, …), date (fixe) et endroit (optionnel)
- Liste des invités et leur provenance.
- Budget global.

L'agent doit fournir :

- Le lieu de la tenue de la fête.
- Le plan de la logistique (transport des invités, …)
- Restauration (menu du repas préparé en concertation avec l'utilisateur)
- Programme culturel (lieux touristiques à visiter)

1) Il est clair que le type d'agent à développer est un agent cognitif. Préciser l'environnement dans lequel il est situé.
   L'environnement de l'agent se compose d'Internet où il peut chercher les agences de voyages, les traiteurs et les agences de service comme le transport et de l'utilisateur.
2) Quel est le type d'architecture approprié de l'agent ? Expliquer chacun de ses composants.
   Il s'agit d'une architecture BDI : Beliefs, Desires and Intentions
3) Donner les croyances de l'agent, son but ainsi que ses plans.
   Croyances : cahier de charge
   But : tenue de l'événement
   Plans : plan pour la recherche du lieu s'il n'est pas imposé par l'utilisateur
   Un plan pour la restauration : consulter les traiteurs avoisinants et constituer le menu en concertation avec l'utilisateur. Prendre le moins disant et le déduire du montant global.
   Un plan pour la logistique : consulter les agences de transport avoisinantes et déterminer les plans de transports pour chacun des invités. Prendre le moins disant et le déduire du montant restant.
   Un plan pour le programme culturel : consulter les agences de voyages et demander un programme touristique. Prendre le moins disant et le déduire du montant restant.
4) En déduire le pseudocode de l'agent en Jason.
   organisationEvenements.mas2
   MAS organisationEvenements {

```
        infrastructure: Centralised
        environment: Environnement
        agents: agentOrganisateur;
}
```

Agent organisateur

agentOrganisateur.asl

```
        /* Initial beliefs*/
        typeEvennement(fête/conférence/mariage…).
        date(jj/mm/aaaa).
        Endroit(salle).
        budget(100 000).
        listeInvités(…..).

        /* Initial goals */
        !organiser(typeEvenement).
        !start.

        /* Plans */
        +!organiser(typeEvennement) : start & endroit(X) & listeInvités(Y)
          <- !restauration(Y) ;
             !logistique(Y) ;
             !programmeCulturel(Y).
        +!organiser(typeEvennement) : start & listeInvités(Y)
          <- !lieu(Y) ;
             !restauration(Y) ;
             !logistique(Y).
             !programmeCulturel(Y).
        +!lieu(Y) : budget(Z)
          <- chercherEndroit(Y,Z) ;
             budget(Z) ;
        +!restauration(Y) : budget(Z)
          <- chercherRestaurant(Y,Z) ;
             budget(Z) ;
        +! logistique (Y) : budget(Z)
          <- chercherTransport(Y,Z) ;
             budget(Z) ;
        +! programmeCulturel(Y) : budget(Z)
          <- preparerProgramme(Y,Z) ;
             budget(Z) ;
```

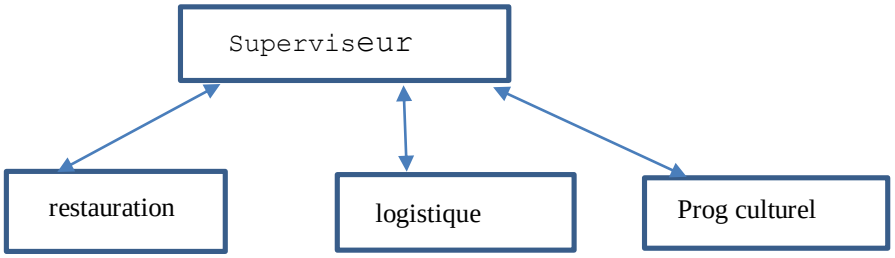
Environnement.Java

```
import jason.asSyntax.*;
import jason.environment.*;
import java.util.logging.*;

public class RoadEnv extends Environment{

@Override
public boolean executeAction(String ag, Structure action){
        if(action.getFunctor().equals("chercherEndroit")){
                int nbInvités = (int)((NumberTerm)action.getTerm(0)).solve();
```

```
                int budget = (int)((NumberTerm)action.getTerm(1)).solve();
                chercherEndroit(nbInvités, budget);
                }
        else if (action.getFunctor().equals("chercherRestaurant")){
                int nbInvités = (int)((NumberTerm)action.getTerm(0)).solve();
                int budget = (int)((NumberTerm)action.getTerm(1)).solve();
                chercherRestaurant(nbInvités, budget) ;
                }
        else if (action.getFunctor().equals("chercherTransport")){
                int nbInvités = (int)((NumberTerm)action.getTerm(0)).solve();
                int budget = (int)((NumberTerm)action.getTerm(1)).solve();
                chercherTransport(nbInvités, budget) ;
                }
        else if (action.getFunctor().equals("preparerProgramme")){
                int nbInvités = (int)((NumberTerm)action.getTerm(0)).solve();
                int budget = (int)((NumberTerm)action.getTerm(1)).solve();
                preparerProgramme(nbInvités, budget) ;
                }
        return true;
        }
void chercherEndroit(nbInvités, budget){
        List endroits = new ArrayList() ;
        endroits = lancer une recherche en ligne ;
        choisir l'endroit le moins cher avec un prix < budget ;
        budget= budget – prix ;
        Literal nouveauBudget = Literal.parseLiteral("chercherEndroit(X," +
budget+)");
    addPercept(nouveauBudget);
}
void chercherRestaurant(nbInvités, budget){
        List restaurants = new ArrayList() ;
        restaurants = lancer une recherche en ligne ;
        choisir le restaurants le moins cher avec un prix < budget ;
        budget= budget – prix ;
        Literal nouveauBudget = Literal.parseLiteral("chercherEndroit(X," +
budget+)");
    addPercept(nouveauBudget);
}
void chercherTransport(nbInvités, budget){
        List transport = new ArrayList() ;
        transports = lancer une recherche en ligne ;
        choisir les transports les moins chers avec un prix < budget ;
        budget= budget – prix ;
        Literal nouveauBudget = Literal.parseLiteral("chercherEndroit(X," +
budget+)");
    addPercept(nouveauBudget);
}
void preparerProgramme(nbInvités, budget){
        List programmes = new ArrayList() ;
        programmes = lancer une recherche en ligne ;
        choisir le programme le moins cher avec un prix < budget ;
        budget= budget – prix ;
```

```
        Literal nouveauBudget = Literal.parseLiteral("chercherEndroit(X," +
budget+)");
      addPercept(nouveauBudget);
}
}
```

5) Proposer une architecture MultiAgents basée sur le modèle d'interaction le plus approprié.
L'approche multi-agents est plus appropriée car le système comprend plusieurs tâches et il serait judicieux d'affecter chaque tâche à un agent. Le modèle d'interaction dans ce cas est le protocole 'contract net'. Un agent superviseur soumettra un appel d'offres avec un cahier de charge pour chacune des tâches de restauration, de logistique et de programme culturel. Pour chaque tâche, un agent est créé et adjugera un contrat avec le superviseur. Ce dernier communiquera avec le superviseur avec envoi de messages jusqu'à la finalisation de la tâche.



Exercice2 (Détection de communautés virtuelles) (10 pts)
Nous nous intéressons à la détection de communautés virtuelles dans un grand réseau d'interaction comme Internet. Le réseau est décrit à l'aide d'un graphe G = (N, A) où N est l'ensemble des nœuds et A l'ensemble des arêtes. Les nœuds représentent les membres faisant partie du réseau et les arêtes la présence d'interaction entre les membres. Une communauté est un ensemble de membres ayant un fort taux d'interactions. La plus petite communauté contient au moins trois membres formant un circuit. Une façon de détecter ces communautés est de développer un système multi-agents (SMA).

1) Quel est le comportement de chaque agent ? Est-il le même pour tous les agents ?
Un agent doit se déplacer à travers le réseau pour détecter des circuits. Plusieurs agents peuvent être lancés à cet effet. Un autre agent peut être introduit pour fusionner les circuits qui ont des nœuds en commun pour former de grandes communautés.
2) Spécifier les croyances, le but et le plan de chaque agent.
Agent circuit :
Croyances : le graphe G.
But : déterminer un circuit
Plan : se déplacer d'un nœud à un autre, mémoriser le chemin emprunté jusqu'à la rencontre d'un nœud déjà visité.
Agent Fusion :
croyances : tableau noir
but : fusionner des communautés pour en faire de grandes
plan : chercher les nœuds partagés par des communautés et les fusionner
3) Quel est le modèle d'interaction le plus approprié entre les agents du système ? Pourquoi ?
Comme les agents doivent coopérer à la recherche de la solution globale, le modèle d'interaction le plus approprié est le tableau noir. Un agent lorsqu'il détecte un circuit, il l'affiche sur le tableau s'il n'existe pas encore. L'agent fusion consulte le tableau à

la recherche de circuits partageant des nœuds communs. Dans l'affirmative, il élimine ces circuits et les remplace par leur fusion.

4) Proposer une architecture logicielle du système en implémentant le modèle d'interaction.
   Tous les agents sont implémentés à l'aide du langage asl. Le tableau noir sera implémenté à l'aide d'une fonction java qui sera appelée dans asl. Cette fonction utilise une structure de tableau et addpercept pour permettre aux agents de voir le contenu du tableau.
5) Donner le pseudocode du SMA en Jason.
   configuration.mas2
   Agent circuit
   Agent fusion

   Circuit.asl

   Fusion.asl

   Environnement.Java

   tableauNoir() ;
   array …