La Programmation Orientée Agent

Présentation et Elements de modélisation

Emmanuel ADAM

Université Polytechnique des Hauts-De-France

UPHF/INSA HdF

Pourquoi utiliser des agents

Cas d'application

- Un agent est une IA légère et distribuée
  - → L'utilisation d'un agent unique est inutile
  - → Utilisation d'une communauté d'agents
- Utilisation d'une IA si besoin d'adaptation
  - → utilisation d'agents en cas de besoin d'adaptation dans un système réparti (gestion de flotte de véhicules, gestion intelligente d'un vaste SI ...)
  - → utilisation d'agents pour la simulation d'entités adaptatives (trafic routier, ...)

Élément de définition d'un agent

Définition générale

- Un agent est une entité adaptative, rationnelle, autonome, capable de communication et d'action.

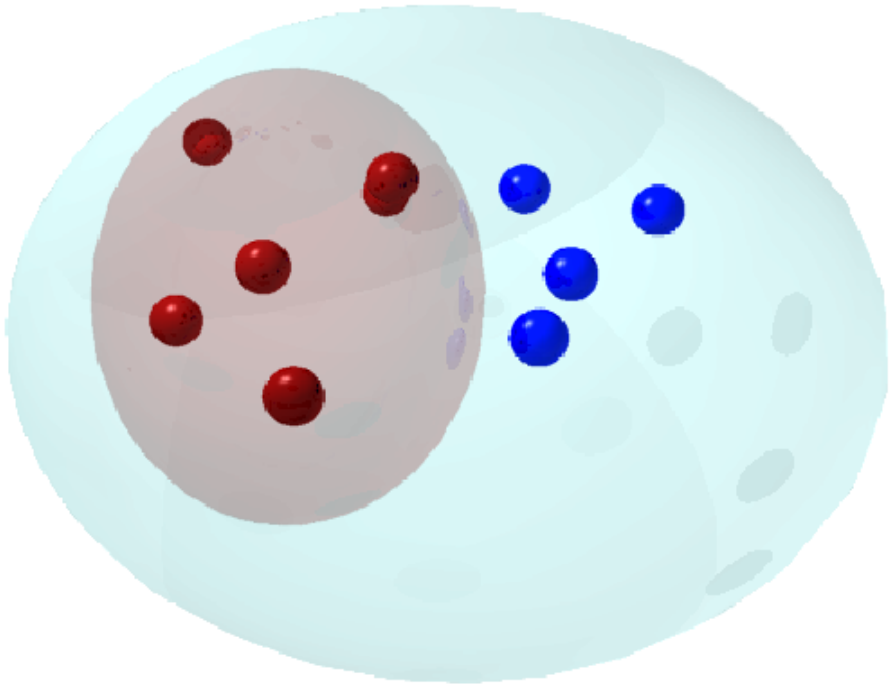
  Selon [Gasser 89], [Demazeau 95], [Ferber 95], [Nwana 96], [Chaib-Draa 96], ...

- Un agent est :
  - autonome
    - planifier ses actions pour atteindre l'objectif
  - capable de perception
  - adaptatif
    - apprendre
    - modifier le comportement
  - communiquant
    - coopérer
    - négocier

Système Multi-Agent

SMA, Monde et Environnement

- Un *système multi-agent (SMA)* est constitué d'agents.
- Les entités en interaction avec le SMA forment l'*Environnement*.
- L'ensemble SMA + Environnement est appelé *Monde*



Structure d'un agent

Grille d'analyse fonctionnelle [Ferber 95]

Les agents possèdent des fonctions décrites selon différents axes :

...	*Personnel*	*Environnemental*	*Social*	*Relationnel*
Représentationnel	**Connaissances**			
Organisationnel	**Planification**			
Conatif	**But, désirs, contraintes**			
Interactionnel	**Perception, coopération**			
Productif	**Action**			
Conservatif	**Conservation, protection**			

Programmer un agent

Dimension Représentationnelle

- Environnement dynamique, voisins mobiles $\rightarrow$ un agent possède des **croyances** $\equiv$ des connaissances dont la valeur, la véracité peut évoluer dans le temps (ou non)
- Nécessité de coder les :
  - croyances sur soi (son nom, adresse, son rôle, ...)
  - croyances sur son environnement (état des routes, des documents, ...)
  - croyances sur les autres
  - croyances sur son mode d'interaction avec les autres selon le rôle

Programmer un agent

Dimension Organisationnelle

- Un agent possède plusieurs actions, plusieurs stratégies (suite d'actions) définies a priori ou construites
- l'organisation consiste à choisir la **bonne** stratégie
- Nécessité de coder les :
  - fonctions d'utilité des stratégies (leurs valuation par rapport au contexte stocké dans les croyances)
  - la méthode de choix de stratégies à appliquer
- les stratégies d'actions portent sur : l'agent lui-même, son environnement, son groupe, ses relations avec le groupe

Programmer un agent

Dimension Conative

- Un agent possède des **désirs** : les objectifs à atteindre par l'agent
  - des **buts** : un but est une étape d'un objectif, c'est le prochain point à atteindre
- des **contraintes** : les limites pouvant être de capacité (mémoire, nombre d'actions), sociale (nombre de partenaires), relationnelle (nombre d'interactions)

Dimension Interactionnelle

- Un agent **interagit**
  - avec l'environnement,
  - avec d'autres,
  - selon des protocoles pré-définis, selon les rôles

Programmer un agent

Dimension Productive

- Un agent possède des actions permettant d'**agir**
  - sur lui même,
  - sur l'environnement,
  - sur le groupe,
  - sur ses relations avec les autres
- Les comportements similaires à plusieurs agents sont codés dans des rôles/services

Dimension Conservative

- Un agent surveille et protège
  - son état,
  - son l'environnement,
  - son le groupe,
  - ses relations avec les autres

Les types d'agents

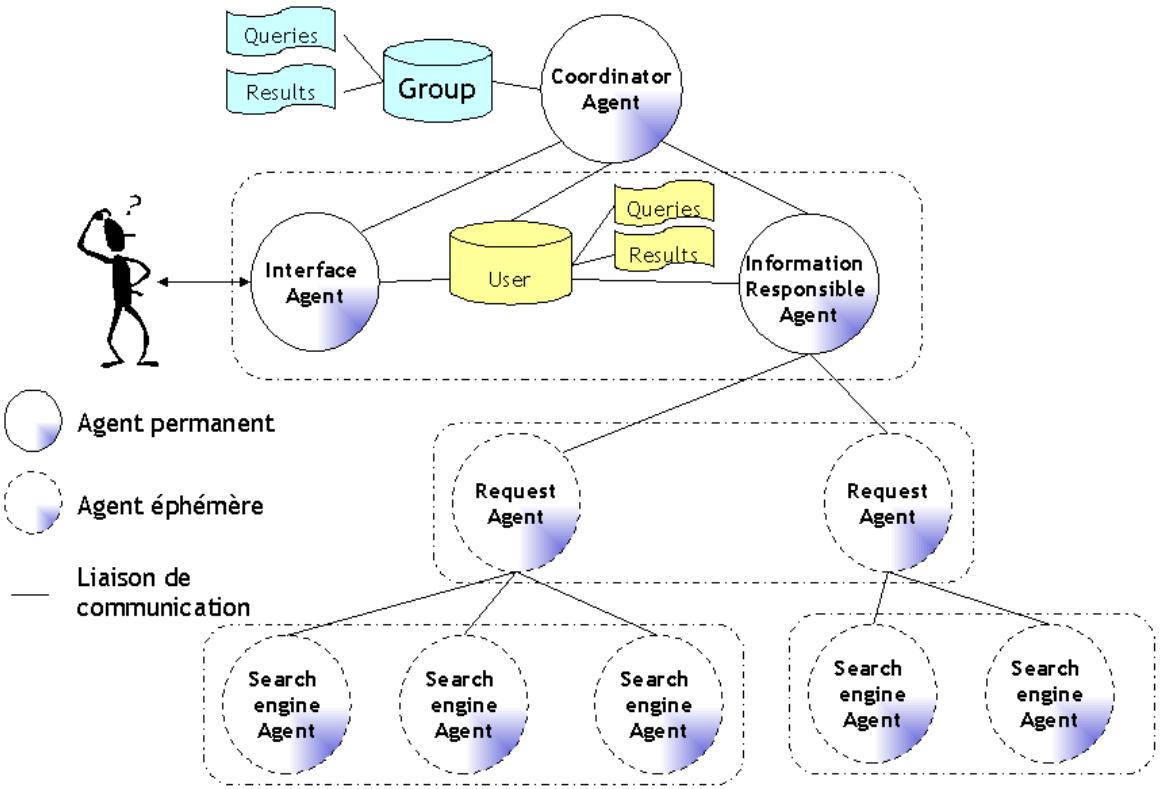
Agents cognitifs et réactifs

- Les *agents cognitifs* possèdent une représentions partielle du monde, communiquent avec les autres agents et élaborent de stratégies. → utilisés pour la gestion, le contrôle d'entités distribuées (robots, véhicules autonomes, machines outils, ...)
- Les *agents réactifs* ont une perception limitée et fonctionnent par action/réaction. → utilisés pour la simulation (de trafic, de mouvement de foule, de propagation d'incendie, ...)
- Certains simulateurs combinent les différents types d'agents pour différents **niveaux de simulation**
- Certaines applications combinent les différents types d'agents pour différents niveaux de gestion (stratégique, tactique & opérationnel)
  - Exemple : le niveau **stratégique** décide de recharger le véhicule autonome électrique ; le niveau **tactique** calcul le chemin le plus intéressant compte tenu des croyances ; le niveau **opérationnel** adapte le trajet par rapport aux perceptions (de la route, ...)

Couplage de types d'agents

Exemple : la recherche d'informations

Cet exemple dans la recherche d'information fait interagir des agents d'interfaces, des agents d'informations et des agents de collaboration.



Communication entre agents

communication directe / indirecte

- Les agents doivent communiquer, cela peut se faire
  - indirectement : par modification de l'environnement (soit sur un zone précise ('tableau noir'), soit sur des objets partagés)
  - directement par envoi de messages : nécessité d'utiliser un standard (ex. FIPA ACL)

FIPA-ACL 1/2

structure d'un message FIPA-ACL

Un message en FIPA ACL est constitué de champs obligatoires et facultatifs :

performative :	type de l'acte communicatif
sender :	emetteur du message
receiver :	destinataire(s) du message
reply-to :	destinataire de la réponse au message
content :	contenu du message
language :	type de langage utilisé
encoding :	type de codage du message
ontology :	ontologie sur laquelle est basée le message
protocol :	type de protocole utilisé
conversation-id :	identifiant de la conversation
reply-with :	type de réponse souhaitée
in-reply-to :	nom de la requête
reply-by :	type de réponse

FIPA-ACL 2/2

Liste de performatifs de FIPA ACL :

Les performatifs possibles sont :

- *Accept Proposal, Agree, Cancel, Call for Proposal, Confirm, Disconfirm, Failure, Inform, Inform If, Inform Ref, Not Understood, Propagate, Propose, Proxy, Query If, Query Ref, Refuse, Reject Proposal, Request, Request When, Request Whenever, Subscribe*

Protocoles

Protocoles de communications

Des normes de communications fixant les règles d'interactions peuvent être utilisées, par exemple :

AchieveRE : un Initiateur envoie un message, le receveur peut répondre par not-understood, refuse ou agree. Suite à l'accord (agree), le receveur retourne un message de type inform (réponse) ou failure.

FIPA - Contract NET : Protocole d'établissement d'un contrat (de vente par exemple).

FIPA - Propose : un Initiateur propose à Participant d'effectuer une action .

FIPA - Subscribe : un Initiateur envoie un message à un Participant de souscription de service.

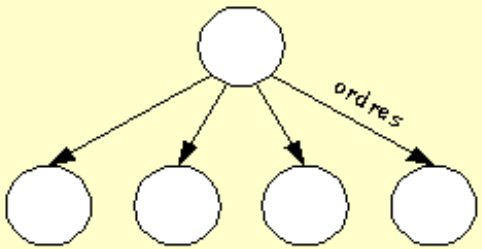
Structures de SMA

Organisations

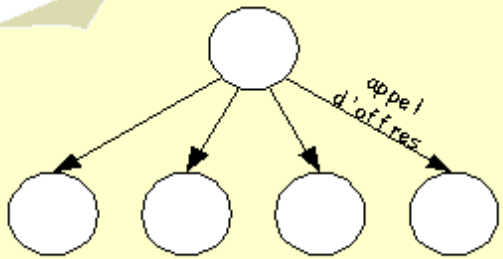
Les SMA peuvent être :

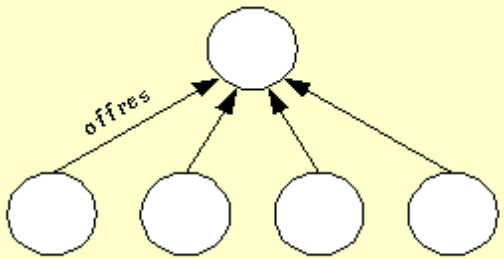
organisés a priori : en structure hiérarchique, de marché, de communauté, de société [Grislin95], [Mandiau99]

organisés par émergence : la structure de l'organisation apparaît suite aux interactions entre agents

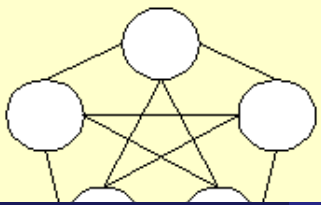


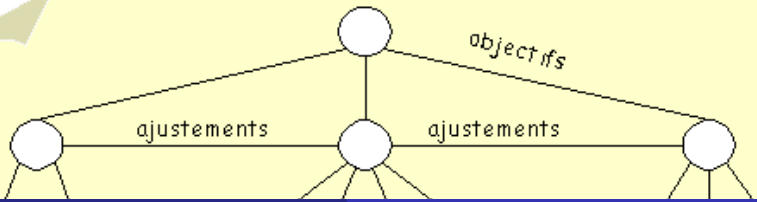
Structure hiérarchique
[Ito 98], [Odubiyi 97], ...





Structure de marché
[Bensaid 97], ...

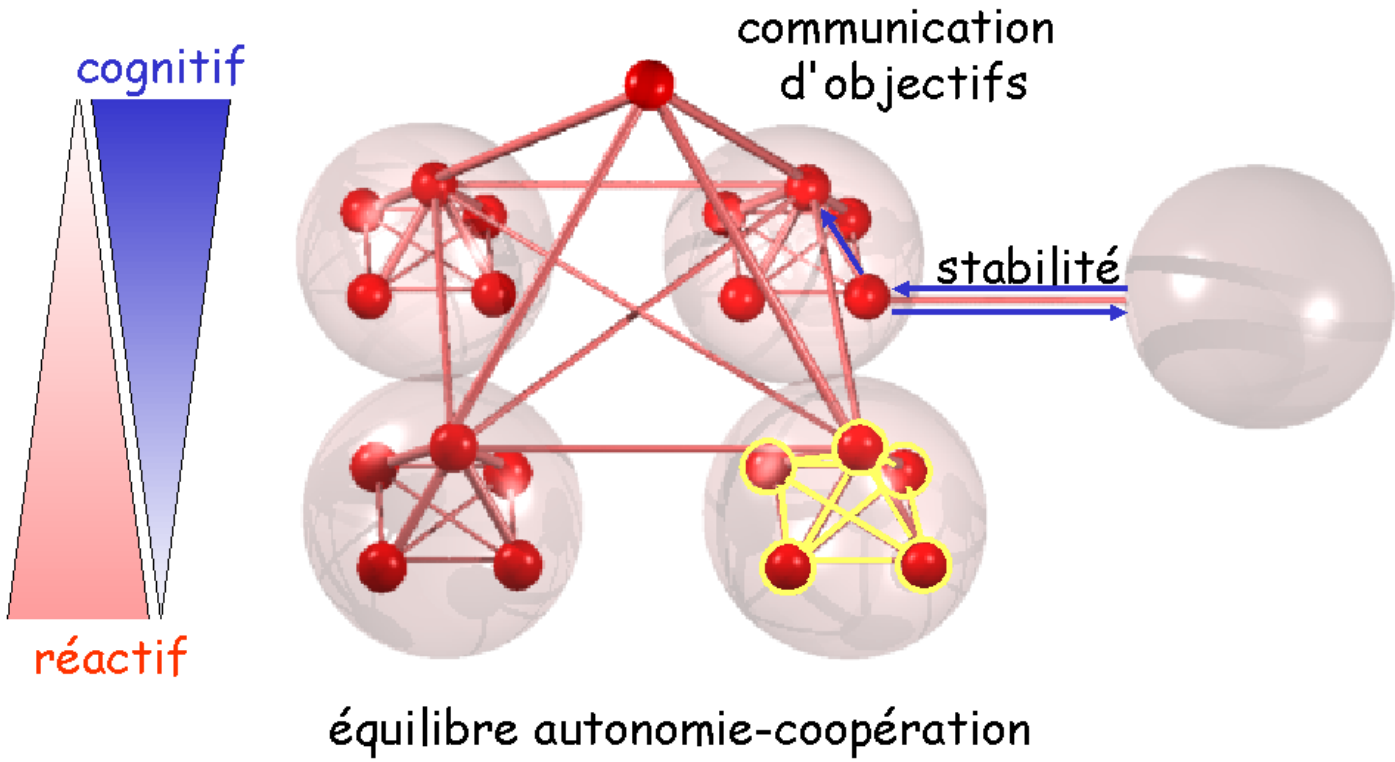




Système MultiAgent Holonique

SMA Holonique

Un sma holonique possède une structure récursive de société. Les agents (holons) sont stables, autonomes et coopérants.



Agentification : concevoir un SMA adapté au problème

Selon le Paradigme Voyelle (Y. Demazeau)

- Pour définir un SMA, il faut définir
  - A Les agents
  - E L'environnement
  - I Les interactions
  - O L'organisation
  - U Les utilisateurs
- à ceci il faut ajouter les **fonctions d'adaptations**
- ainsi que les **rôles joués par les agents**
- selon l'application visé, le cycle de développement diffère :
  - simulation : E, A, I, O, U
  - vehicules autonomes : U, E, A, I, O
  - gestion de SI : I, O, U, A, E ...

Agent vs Objet

Informatiquement, un agent est il un objet ?

- Un agent est un objet qui est adaptatif, rationnel, autonome, capable de communication et d'action.
  - *typiquement dans les plateformes de simulations agent, un agent est implémenté en objet*
  - *Un agent cognitif/délibératif est de préférence créé sous forme d'un objet ayant les caractéristiques d'un processus*
- Un objet $o1$ ne peut qu'appeler une méthode existante d'un objet $o2$ qui **doit** l'exécuter
- Un agent $a1$ peut demander l'exécution d'une méthode à l'agent $a2$: celui-ci peut décider de l'exécuter ou non (manque de capacité, signature de l'agent inconnue, ...)
- Un agent agit en fonction de son but et de ses contraintes.

Plateformes multi-agent de simulation

Quelques plateformes multi-agent pour la simulation

MadKit `http://www.madkit.net/madkit/`

MaDKit-5 is a lightweight Java library for designing and simulating Multi-Agent Systems (Java)

Gama `https://gama-platform.github.io/wiki/Home`

GAMA is a modeling and simulation development environment for building spatially explicit agent-based simulations (GAML)

RePast `https://repast.github.io`

The Repast Suite is a family of advanced, free, and open source agent-based modeling and simulation platforms (Java, C++)

NetLogo `http://ccl.northwestern.edu/netlogo/`

NetLogo is a multi-agent programmable modeling environment. (logo)

Plateformes multi-agent de gestion/contrôle

Quelques plateformes multi-agent pour la gestion, le contrôle

Jade `http://jade.tilab.com`

JADE (Java Agent DEvelopment Framework) is a software Framework fully implemented in the Java language. It simplifies the implementation of multi-agent systems through a middle-ware

Sarl `http://www.sarl.io`

SARL aims at providing the fundamental abstractions for dealing with concurrency, distribution, interaction, decentralization, reactivity, autonomy and dynamic reconfiguration

JaCaMo `http://jacamo.sourceforge.net`

JaCaMo is a framework for Multi-Agent Programming that combines three separate technologies : Jason - for programming autonomous agents ; Cartago - for programming environment artifacts ; Moise - for programming multi-agent organisations