DEA *IARFA*

1/2 Module Tronc Commun

Introduction aux Agents

Copie transparents en :

`http://www-poleia.lip6.fr/~briot/cours/intro-agents-iarfa01-02.pdf`

Jean-Pierre Briot

Thème OASIS

(Objets et Agents pour Systèmes d'Information et Simulation)

Laboratoire d'Informatique de Paris 6

Université Paris 6 - CNRS

`Jean-Pierre.Briot@lip6.fr`

Plan

- Pourquoi les Agents ?
- Positionnement Historique (Evolution de l'IA et de la Programmation)
- Classification (incluant Agents Mobiles et Agents Assistants)
- Principes
- Limites
- Différences entre Objets et Agents
- Architectures Logicielles et Architectures d'Agents
- Frameworks, Composants et Patterns
- Conclusion
- Références et Pointeurs

Motivations - pourquoi les agents?

- Complexité croissante des applications informatiques, plus ouvertes, plus hétérogènes, plus dynamiques
  - exemple : le Web et toutes les couches et services qui le supportent
  - comment décomposer, recomposer, interopérer, gérer l'évolution, adaptation (aux autres modules logiciels, à l'environnement, aux utilisateurs...), contrôle, négocier (partage ressources, prise de RdV),...
  - limitations des approches informatiques classiques : statiques, homogènes, interfaces rigides, objets/composants sans initiative propre, client serveur
  - difficile à maîtriser par des humains

Exemples

- Contrôle de sonde/vaisseau spatiale
  - Distance avec le contrôle au sol -> temps de réaction
  - -> Nécessité d 'un contrôle local : autonomie
  - capacités de prises de décision en cas de situations non prévues : initiative
- Recherche d'information sur Internet
  - Processus long et difficilement prédictible (attente, découverte, pannes…)
  - -> Délégation du cahier des charges : guidé par les objectifs
  - ex : recherche multilingue - coopération de différents agents
    - » (personnalisation, ontologie, dérivations, traduction, etc.) [Projet SAFIR]
- Prise de RdV
  - fastidieux, attentes (indisponibilité ou déconnexions)
  - -> PDAs assistants (apprend habitudes utilisateur et initiative) et coopératifs
- etc, ex : Surveillance de réseaux
  - détection, intervention, réparation

Idées

- Agents logiciels
  - autonomie
  - mission
  - initiative
  - niveau connaissance
  - adaptation
  - inter-opérabilité

  - De plus, ils peuvent être coopératifs (avec autres agents)
    - ex : prise de RdV distribuée

  - On parle alors de :
- Systèmes multi-agents
  (issus du domaine « résolution distribuée de problèmes »)
  - protocoles de communication
  - protocoles de coordination
  - organisations

Qu'est-ce qu'un agent ?

- Petit Robert :
  - De agere « agir, faire »
    - « Celui qui agit (opposé au patient qui subit l'action) »
    - « Ce qui agit, opère (force, corps, substance intervenant dans la production de certains phénomènes) »

  - De agens « celui qui fait, qui s 'occupe de »
    - « Personne chargée des affaires et des intérêts d'un individu, groupe ou pays, pour le compte desquels elle agit »
    - « Appellation de très nombreux employés de services publics ou d'entreprises privées, généralement appelés à servir d'intermédiaires entre la direction et les usagers »

- American Heritage Dictionary :
  - « one that acts or has the power or authority to act... or represent another »
  - « the means by which something is done or caused; instrument »

Qu'est-ce qu'un agent ? (2)

- [Ferber 95]
  - on appelle agent une entité physique ou virtuelle
    - qui est capable d'agir dans un environnement,
    - qui peut communiquer directement avec d'autres agents,
    - qui est mue par un ensemble de tendances (sous la forme d'objectifs individuels ou d'une fonction de satisfaction, voire de survie, qu'elle cherche à optimiser),
    - qui possède des ressources propres,
    - qui est capable de percevoir (mais de manière limitée) son environnement,
    - qui ne dispose que d'une représentation partielle de cet environnement (et éventuellement aucune),
    - qui possède des compétences et offre des services,
    - qui peut éventuellement se reproduire,
    - dont le comportement tend à satisfaire ses objectifs, en tenant compte des ressources et des compétences dont elle dispose, et en fonction de sa perception, de ses représentations et des communications qu'elle reçoit.

Rappel historique (vis à vis de l'IA)

- Concept d'agent rationnel à la base de l'intelligence artificielle (IA)
  - système informatique autonome
    - connaissances, buts, pouvoirs, perceptions, raisonnement/délibération (résolution, planification, déduction, etc.), actions
    - système expert
- Limitation : Autarcie !!
  - autarcie logicielle : difficile à faire collaborer avec d'autres logiciels
  - autarcie sociale : censé remplacer l'homme, pas de collaboration (expert humain en dehors de la « boucle »)
- Réponses
  - agents coopératifs
    - systèmes multi-agents
    - issus de la résolution distribuée de problèmes
      - distributed artificial intelligence (DAI versus GOFAI)
  - agents assistants

Rappel historique (vis à vis de la programmation)

- Interview Les Gasser, IEEE Concurrency 6(4):74-81, oct-déc 98

- langage machine
- assembleur
- programmation structurée
- programmation par objets
- programmation par agents !

- concept d'action persistante
- programme qui tente de manière répétée (persistante) d'accomplir quelque chose

- *mission et initiatives pour l'accomplir*

action persistante

- programme qui tente de manière répétée (persistante) d'accomplir quelque chose
  - pas la peine de contrôler explicitement succès, échec, répétition, alternatives...
- description de :
  - (quand) but == succès
  - méthodes alternatives
  - * apprentissage (de nouvelles méthodes)
- ressources :
  - processus
  - itération (tant que)
  - options/solutions (situation -> action)
  - capacité de choix (on line - sélection d'action)
  - recherche (search) -- en cas de nouvelles situations
  - feedback sur le choix

Une vision différente du logiciel (vers un couplage sémantique et adaptatif)

- Problème clé du logiciel : évolution, adaptation
  - profil utilisateur, programmeur, environnement, contraintes - ex : QoS, ...
  - Pour un système (logiciel) complexe, impossible de prédire au moment de la conception toutes les interactions potentielles
  - Ceci est rendu encore plus difficile si l'on considère l'évolutivité du logiciel ainsi que celle de son environnement (autres logiciels)
- Vers des composants logiciels « adaptables »
  - Les interactions non prévues deviennent la norme et non plus l'exception [Jennings 1999]
  - Le couplage entre composants est abordé au niveau des connaissances et non plus au niveau des types de données (ce qui est sûr mais rigide)
  - Vers un plus grand découplage : objets -> composants -> agents (et ensuite ?!)
  - A rapprocher du "Ever late binding" (C -> C++ -> Java -> ...)

typologie (Babel agents) 1/3

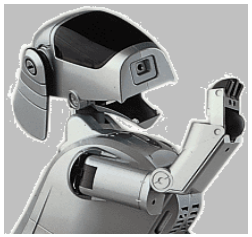
- agents rationnels
  - IA, comportement délibératif, perceptions, croyances, buts
  - ex : systèmes experts
- systèmes multi-agents
  - résolution distribuée (décentralisée) de problèmes
  - coordination, organisation
  - ex : robotique collective
- agents logiciels
  - ex : démons Unix, virus informatiques, robots Web
- agents mobiles
  - code mobile -> objet mobile -> agent mobile (processus)
  - motivations : minimisation communications distantes, informatique nomade
  - technologie en avance sur les besoins
  - problèmes de sécurité, coquilles vides

typologie (Babel agents) 2/3

- agents assistants
  - secrétaire virtuelle (trie le mail, gère les RdVs...)
  - < logiciel utilisateur + assistant >
  - filtrage collaboratif
    - ex : recommandation achats CDs par recherche de similarité des profils puis transitivité
  - computer-supported cooperative work -> communityware (pour citoyens)
  - agents « émotionnels »
- agents robotiques
  - architectures de contrôle de robots
  - sélection de l'action
  - robotique collective (ex : RoboCup, déminage...)
- vie artificielle
  - alternative à l'IA classique
  - modélisation/simulation des propriétés fondamentales de la vie (adaptation, reproduction, auto-organisation...)
  - importation de métaphores biologiques, éthologiques...
  - ex : algorithmes à base de fourmis (agents) pour routage de réseaux

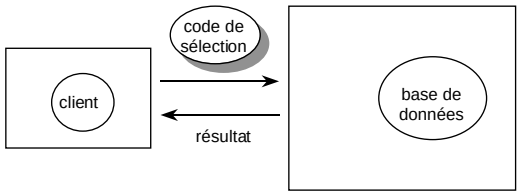
typologie (Babel agents) 3/3

- simulation multi-agent
  - simulation centrée individu vs modèle global (ex : équations différentielles)
  - modèle de comportement arbitrairement complexe
  - interactions arbitrairement complexes (ex : sociales : irrigation parcelles)
  - niveaux hiérarchiques (ex : bancs de poissons)
  - espaces et échelles de temps hétérogènes
  - collaboration informaticien - spécialiste
- agents de loisir
  - virtuels (ex : jeux vidéo)
  - virtuels-physiques (ex : Tamagotchi)
  - physiques (ex : Furby, robot-chien Aibo de Sony)

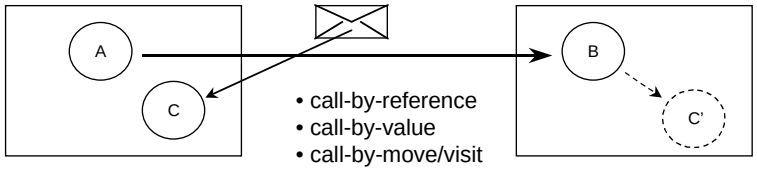


Code/objets/agents mobiles

- Code mobile
  - rapprocher (code) traitement des données
  - ex : SQL

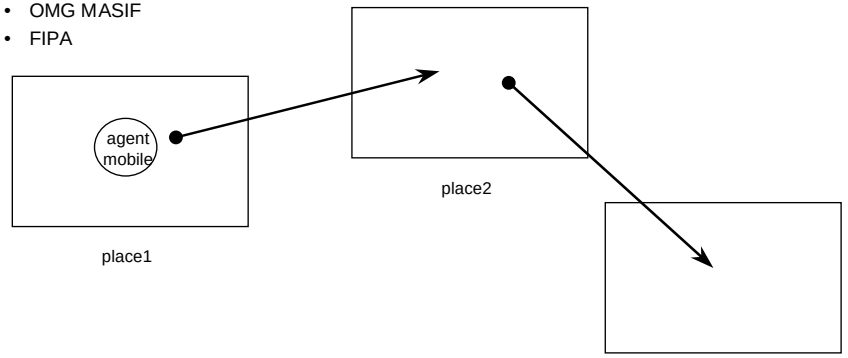


- Objet mobile
  - PostScript (code + données constantes)
  - Emerald [Black et al. IEEE TSE 87]



Agents mobiles

- A la différence du code ou de l'objet mobile, c'est l'agent mobile qui a l'initiative de son déplacement
- Langages :
  - Telescript (initiateur)
  - (Java-based) Odissey, Aglets, Voyager, Grasshopper, D'Agents (ex-AgentTcl), etc.
  - Standardisation :
    - OMG MASIF
    - FIPA

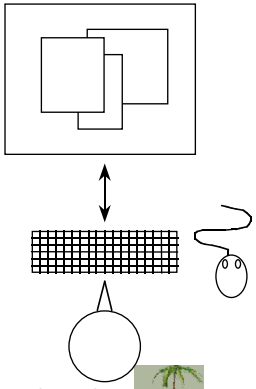


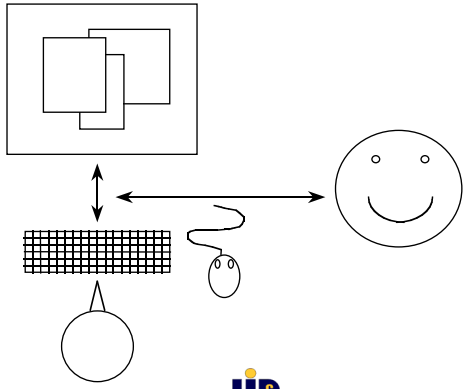
Agents mobiles (2)

- Avantages des (mis en avant par) les agents mobiles
  - Réduction du trafic (traitement local -> données échangées réduites)
    - agents mobiles vs RPC
  - Robustesse
    - Déconnexion du client mobile (informatique nomade : pause, tunnel, ombre…)
  - Confidentialité (traitement local)
    - (mais problèmes de sécurité)
  - Evolution logicielle
    - Off-line
      - Diffusion (versions) de logiciels (download)
    - On-line
      - Réseaux actifs
        - Données et Méta-données de contrôle (capsules)
- « Find the killer application ! »
  - Une nouvelle technique (parmi les) de programmation répartie
  - Combinaison (avec les autres : RPC, réplication, etc.) et non pas remplacement
- Recherches actuelles
  - Sécurité
  - Hétérogénéité
  - Collaboration (les agents mobiles actuels restent encore trop souvent solitaires)

Agents assistants

- Limitations des interfaces homme-machine classiques
  - à manipulation directe / explicite
  - rigidité, complexité, ne s'améliore pas à l'usage
- Agents assistants
  - adaptation au profil de l'utilisateur, automatisation de certaines tâches, rappel d'informations utiles, initiative
  - ex : trieur de mails, prise de RdVs

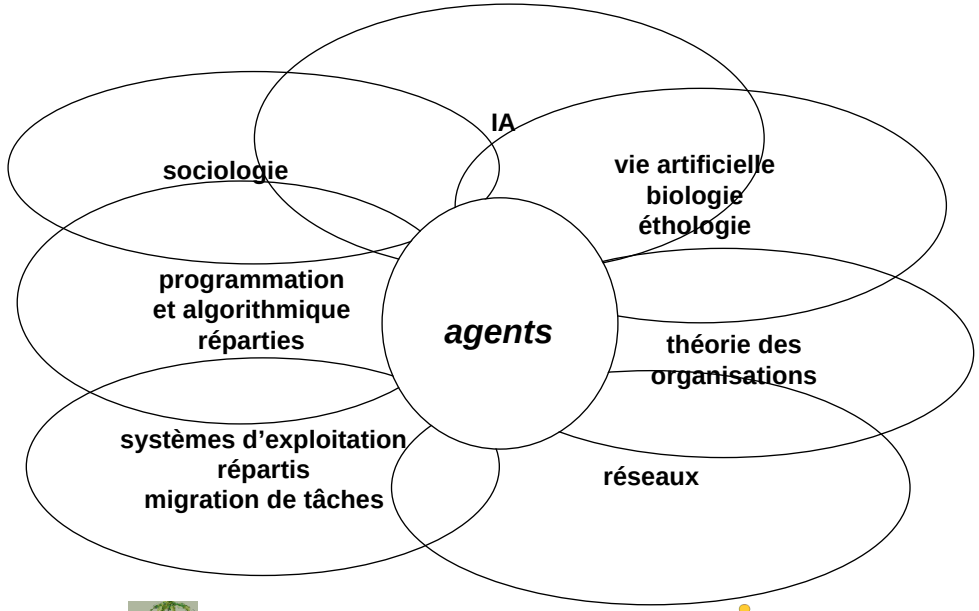




Agents assistants (2)

- Ex : Bargain Finder, Letizia, Firefly (MIT AI Lab)...
- « If you have somebody who knows you well and shares much of your information, that person can act on your behalf very effectively. If your secretary falls ill, it would make no difference if the temping agency could send you Albert Einstein. This issue is not about IQ. It is shared knowledge and the practice of using it in your best interests. » [Negroponte, Being Digital, 1995]
- Complémentarité (humain - agent)
  - Utilisateur : « lent » en calcul ; agent : « rapide »
  - Utilisateur : langage naturel et vision ; agent pas encore…
  - « Show what an agent what to do » vs « Tell an agent what to do »
  - Critique : agents of alienation [Lanier, 1995]
- Vers des agents assistants et coopératifs

agents



Différents niveaux d'agents (cf Les Gasser)

- ermites
  - représenter un humain
  - données+procédures (objet)+contrôle+ressources(processus) (acteur)
    - réactivité, autonomie
  - action persistante
    - pro-activité, mission
  - capacités entrées/sorties et communication
  - * mobilité
  - * apprentissage
- agents sociaux
  - langage de communication entre agents (KQML, ACL, XML...)
  - échange de données
  - tâches
  - modèle (représentations) des autres
- multi-agent
  - action collective
  - division du travail (spécialisation)
  - coordination/intégration (gestion des dépendances et de l'incertain)

agents cognitifs vs agents réactifs

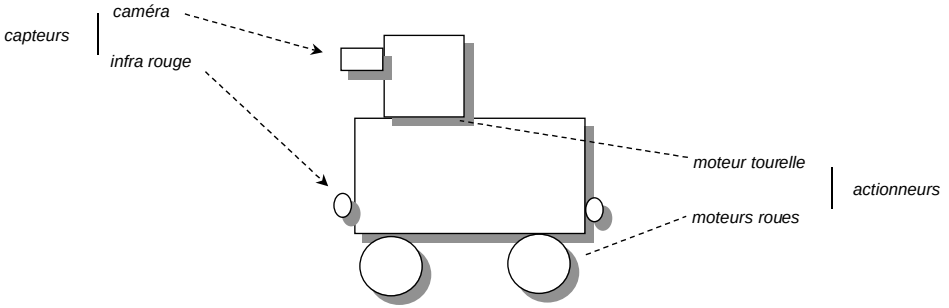
- agents cognitifs
  - représentation explicite
    - soi
      - connaissances (beliefs)
      - buts (intentions)
      - tâches
      - plans
      - engagements
    - environnement
    - autres agents
      - compétences
      - intentions
  - architectures complexes, souvent modèle logique (ex : BDI, Agent0)
  - organisation explicite
    - allocation et dépendances tâches
    - partage des ressources
    - protocoles de coordination/négociation
  - communication explicite, point à point, élaborée (ex : KQML)
  - petit/moyen nombre d'agents
  - top down, systématique
  - certaines validations formelles possibles

agents réactifs vs agents cognitifs

- agents réactifs
  - pas de représentation explicite
  - architectures simples
    - stimulus -> réponse
  - organisation implicite/induite
    - auto-organisation, ex : colonie de fourmis
  - communication via l'environnement
    - ex : perception/actions sur l'environnement, phéromones de fourmis
  - grand ou très grand nombre d'agents
    - redondance
  - robustesse
  - bottom up
  - validation expérimentale

Achitectures d'agents

- Architecture d'un agent = le "cœur" de l'agent, ce qui décide quoi faire
- Ex : architecture de contrôle d'un robot mobile autonome
  - problème clé : sélection de l'action (quoi faire ensuite ?)



- Propriétés/caractéristiques recherchées :
  - comportement à la fois délibératif et réactif
  - perception incertaine de l'environnement
  - robustesse (résistance aux pannes et aux dangers)
  - flexibilité de conception (boucle conception/évaluation)

Cela sera revu plus loin, à la lumière des architectures logicielles et des composants

Validation : la « grande » question

- Validations formelles
  - comportement individuel et collectif
    - modèles logiques, ex :
      - BDI [Georgeff et Rao] [Jennings et Wooldridge]...)
      - intentions jointes [Cohen]
    - coordination
      - réseaux de Petri, ex : [ElFallah]
    - négociation
      - théorie des jeux [Rosenschein...]
    - Mais en général contraint les modèles (certaines hypothèses de staticité, etc.)
- Validations semi-formelles
  - tests, couvertures de tests, invariants...
- Validations expérimentales
  - protocoles expérimentaux
  - reproductibilité des comportements et résultats observés
  - analyses de sensibilité (ex : aux conditions initiales)
  - attention aux influences des conditions d'exécution
    - ex : algorithme de séquencement, générateurs de nombres pseudo-aléatoires

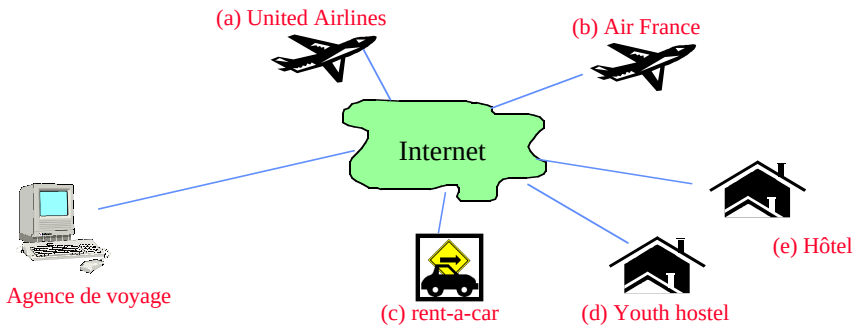
Agent, dans l'œil de l'observateur ??

- bilame d'un chauffe-eau
- test de Turing
- est-ce qu'un objet/processus (distribué ?) pourrait faire la même chose ??
- rationalité
- intentionnalité
  - comportement individuel
  - comportement collectif
- Canon de Morgan (1894) - psychologie comparative - éthologie
  - « En aucun cas, nous ne pouvons interpréter une action comme la conséquence d'un exercice ou d'une faculté psychique plus haute, si elle peut être interprétée comme l'aboutissement d'une faculté qui est située plus bas dans l'échelle psychologique »
  - -> behaviorism (explication causale) vs intentionnel (explication fonctionnelle)
- mesures quantitatives « objectives » ?
  - ex : ajout d'un agent -> pas de dégradation des performances (éventuellement amélioration) [Ferber 95]

Décomposition parmi les agents

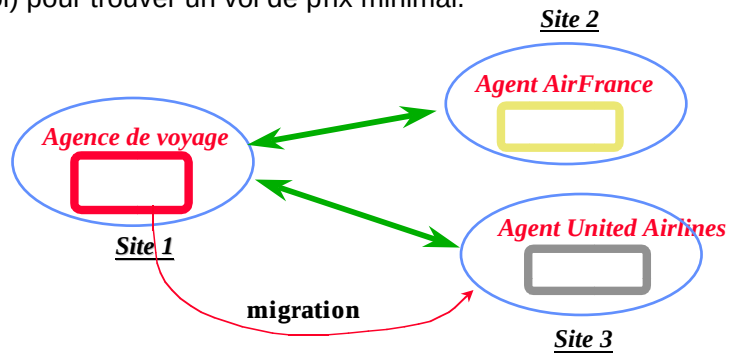
- décomposition des tâches, plans, sous-buts
- assignation aux agents
  - division du travail (spécialisation) vs totipotence
  - organisation, rôles
  - réseaux d'accointances
    - représentations des capacités des autres agents
  - appel d'offre
    - Contract Net protocol [Smith IEEE Transac. Computers 80]
  - market-based algorithms
    - mise aux enchères (protocoles : à la bougie, anglaise, hollandaise...)
  - formation de coalitions
    - (composition d'agents pour résoudre des tâches non faisables individuellement)

Ex : Scénario d'agence de voyage électronique [FIPA et projet CARISMA - thèse M.-J. Yoo, octobre 1999]



Exemple de protocole de coopération entre agents : choix du meilleur billet d'avion

- Deux agents serveurs de voyage, un agent agence de voyage
- Coopérer suivant un protocole d'appel d'offre (Contract net protocol) pour trouver un vol de prix minimal.



- Mobilité : l'agent se déplace vers le site du serveur choisi pour continuer la conversation (et optimiser les communications)

Organisations

- théorie des organisations - 3 points de vue [Scott 81] :
  - organisations rationnelles
    - collectivités à finalités spécifiques
    - objectifs, rôles, relations (dépendances...), règles
  - organisations naturelles (végétatives)
    - objectif en lui-même : survie (perpétuer l'organisation)
    - stabilité, adaptativité
  - systèmes ouverts
    - inter-relations/dépendances avec d'autres organisations, environnement(s)...
    - échanges, coalitions
- organisations d'agents (artificiels)
  - notion de rôle :
    - ex : client, producteur, médiateur ; attaquant, défenseur, gardien de but...
  - spécialisation des agents (simplicité vs flexibilité)
  - redondance des agents (efficacité vs robustesse)
  - relations
    - dépendances, hiérarchie, subordination, délégation
  - protocoles d'interaction/coordination
  - gestion des ressources partagées

Organisations (2)

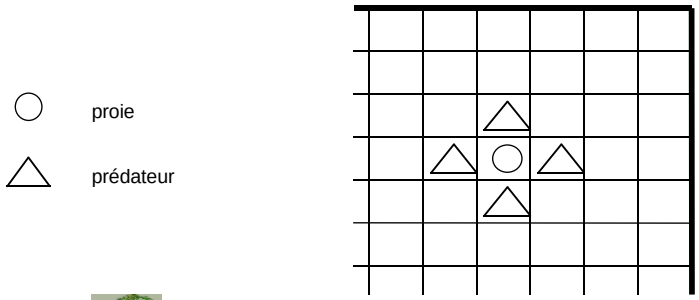
- agents cognitifs
  - organisations explicites
- agents réactifs
  - organisations semi-implicites
    - façonnement de l'environnement, ex : fourmilière
    - « auto-organisation », ex : stigmergie des colonies de fourmis
- Exemple : extraction de minerai par des robots [Ferber 95]
- spécialisation ou pas des agents
  - totipotents (un agent sait jouer tous les rôles = sait tout faire)
  - rôles prédéfinis : robots détecteur, foreur, transporteur
- organisations du travail :
  - équipes (partenaires affectés statiquement)
    - ex : 1détecteur, 3 foreurs, 2 transporteurs
  - appel d'offre (partenaires affectés dynamiquement)
  - « émergentiste »
  - évolutives
    - feedback environnement, apprentissage, algorithmes génétiques...

Coordination

- Motivations :
  - capacités individuelles insuffisantes (ex : charges trop lourdes à transporter)
  - cohérence (réguler les conflits sémantiques : buts contradictoires, accès aux ressources...)
  - efficacité (parallélisation de l'exécution des tâches)
  - robustesse, traitement de l'incertain
  - recomposition des résultats - solutions partielles
- Techniques
  - planification centralisée, semi-centralisée (synchronisation de plans individuels), distribuée, ex : Partial Global Plans [Durfee et Lesser IJCAI'87]
  - synchronisation d'accès aux ressources
    - algorithmique répartie
  - règles sociales
  - spécialisation (spatiale, objectifs…)
  - négociation
    - numérique, symbolique (agrégation, argumentation), démocratique (vote, arbitrage)
  - utilitarisme (théorie des jeux)
  - sans communication explicite
    - (environnement, reconnaissance d'intentions, de plans...)

Exemple des proies-prédateurs

- sur un environnement quadrillé, 4 prédateurs tentent d'encercler une proie
  - problème de coordination des mouvements des prédateurs
  - qualités : simplicité, généricité, efficacité, robustesse, propriétés formelles...
- approche cognitive
  - échange de plans (déplacements prévus), coordination
- approche réactive
  - attirance forte vers les proies, répulsion (faible) entre prédateurs



Communication

- environnement
  - perception, action (ex : consommation ressources)
  - traces (ex : phéromones)
- symbolique (messages)
  - medium (réseau, voix, vision...)
  - participants :
    - individuel - point à point
    - partagé - multicast
    - global - broadcast
    - publish/subscribe (événements)
    - par le contenu, Tuple-space, ex : Linda [Gelerntner 88]
- actes de langage - « dire c'est faire » [Searle 79]
  - composante locutoire
    - message, encodage
  - composante illocutoire
    - réalisation de l'acte de langage
    - performatifs : affirmer, questionner, annoncer, répondre...
  - composante perlocutoire
    - effets sur croyances des autres

Communication (2)

- Langages et protocoles de communication
- interoperabilité d'agents (CORBA des agents)
- KQML [Finin et Labrou 94] message
  - contenu
  - langage (d'expression du contenu)
    - ex : Java, Smalltalk, KIF, XML
  - ontologie
    - hiérarchie de concepts pour un domaine donné (ex : commerce e-, automobile...)
  - performatif (intention de la communication, lié à un type d 'interaction)
    - ex : ask, deny, register, recruit, request...

Note : beaucoup de choses implicites dans le monde objet deviennent explicites ici

- FIPA ACL (Agent Communication Language)
  - comme KQML, avec en plus :
  - sémantique formelle
  - protocole explicité
    - ex : FIPA-Contract-Net, FIPA-Iterated-Contract-Net

Limites (1/2) [Jennings 1999]

- No magic !
  - Un système développé avec des agents aurait probablement pu être développé avec des technologies plus conventionnelles
  - L'approche agent peut simplifier la conception pour certaines classes de problèmes
  - Mais elle ne rend pas l'impossible possible !
- Les agents sont des logiciels (presque comme les autres)
  - Principalement expérimental
  - Pas encore de techniques (é)prouvées
  - Ne pas oublier les aspects génie logiciel (analyse de besoins, spécification, conception, vérification, tests...)
  - Ne pas oublier les aspects concurrence/répartition
    - Problèmes (synchronisation…)
    - Mais également avantages (souvent encore peu exploités)
  - Réutiliser les technologies conventionnelles
    - Objets, CORBA, bases de données, noyaux de systèmes experts...
  - Utiliser les architectures agent existantes
    - Sinon vous passerez la majeure partie du temps dans la partie infrastructure et pas dans les spécificités des agents...

Limites (2/2)

- Trouver la bonne granularité
  - Equilibre à trouver entre : « un nombre est un agent » et « un seul agent dans le système »
  - *dans le monde objet : programmer une seule classe avec 1000 variables...*
  - Complexité vs modularité
- Importance de la structure (organisations, protocoles, connaissances…)
  - Il ne suffit pas de « jeter » ensemble des agents pour que cela fonctionne !
- Besoins en méthodologies
  - Cassiopée [Collinot & Drogoul 96]
  - Aalaadin/AGR [Ferber & Gutknecht 97]
  - Gaia [Jennings 99]
- Modélisation
  - Tentatives actuelles d'extension d'UML vers les agents (ex : AUML)
  - *Attention !* : UML est un ensemble de notations standardisée, et n'est *pas* une méthodologie

Vers des Méthodologies (analyse et conception) adaptées

- Find the agents !
  - Trop souvent, les agents sont (ou plutôt SEMBLENT) déjà donnés avant même l'étape d'analyse
    - ex : robots footballeurs
  - Mais, cela n'est pas toujours le cas
  - De plus, une identification (des agents) trop directe/intuitive ne sera pas forcément bénéfique dans la suite, car l'identification des agents :
    - quels concepts seront réifiés en agents
    - et lesquels ne seront pas !
    - quelle granularité...

  ...dépend beaucoup de l'objectif de la modélisation, des propriétés attendues...
- Cassiopée [Collinot et Drogoul 1996]
  - Objectif : Faire de la notion d'organisation l'objet véritable de l'analyse, qui peut être manipulée par le concepteur lors de la phase de conception, et/ou par les agents lors de l'exécution
  - Identifier les dépendances fonctionnelles entre les rôles (regroupement de comportements, mis en œuvre par des agents) qui sont inhérentes à l'accomplissement collectif de la tâche considérée.
  - Organisation : gestion (décentralisée et dynamique) des dépendances (entre rôles)

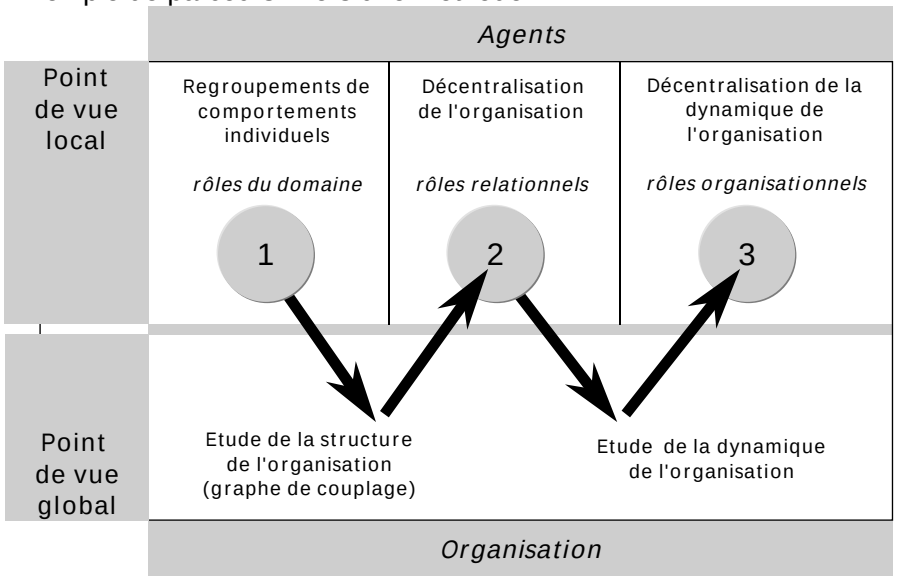
Cassiopée 1/2

- Un agent est composé d'un ensemble de rôles (3 différents niveaux)

	Rôles	*Typologie*	*Comportements*	*Signes échangés*
Agent	*dépendants du domaine*	*dépendant de l'application*	*dépendant de l'application*	–
	relationnels	agent influent	produit les signes d'influence en fonction du rôle du domaine	signes d'influence
		agent influencé	interprète les signes d'influence pour contrôler les rôles du domaine	
	organisationnels	initiateur	comportement de formation de groupe	signes d'engagement
			comportement de dissolution de groupe	
		participant	comportement d'engagement	signes de dissolution

Cassiopée 2/2

- Exemple de parcours : vers une méthode



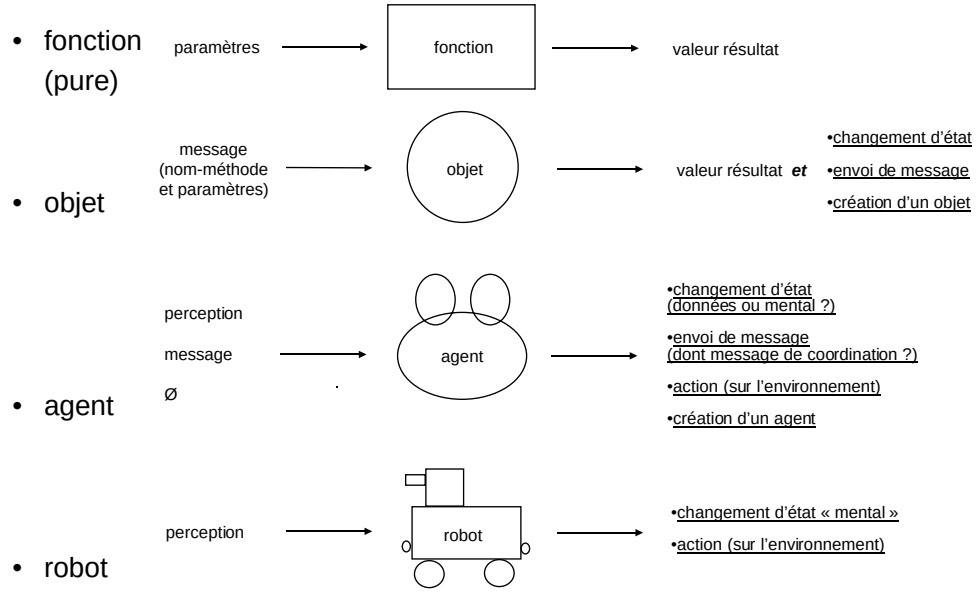
Agents et objets (concurrents, distribués)

- OK, donc les agents semblent avoir des caractéristiques différentes ou supplémentaires des objets
  - au niveau des entités (pro-actives vs réactives, déclaratives vs procédurales...)
  - au niveau des organisations (adaptatives vs statiques et déterministes...)
- Regardons cela de plus près...

Différences entre Objets et Agents (1ère passe)

- au niveau de l'entité
  - agent non purement procédural
    - connaissances
      - ex : états mentaux, plans, règles d'inférence des agents cognitifs
  - pro-activité
    - pas uniquement purement réactif
- au niveau d'un ensemble d'agents
  - différents modes de communication
    - via l'environnement, ex : colonies de fourmis
    - messages typés, ex : KQML (inform, request, reply...)
  - coordination
    - interactions arbitrairement complexes, pas juste client/serveur
- au niveau de la conception (vs implantation)
  - organisation
    - structuration forte/explicite, souvent dynamique, conditionnant les interactions, la division du travail, les accès aux ressources partagées... : les rôles et leur coordination
  - une conception sous forme d'agents peut ensuite être réalisée sous forme d'objets ou d'acteurs, le niveau agent n'apparaissant plus explicitement dans l'implantation

Différences entre Objets et Agents (2ème passe)



De la Simulation Objet à la Simulation Multi-Agent

- au niveau de l'entité
  - comportement non nécessairement purement déterministe
    - mémoire, connaissances, désirs, interactions
- au niveau d'un ensemble d'agents
  - différents modes de communication
    - via l'environnement, ex : colonies de fourmis
  - coordination
    - interactions arbitrairement complexes
  - simulation multi-niveau
    - un ensemble d'agents peut être aussi considéré (émerger) comme un agent avec son comportement propre
      - ex: émergence d'un banc de poisson, d'une rivière
- au niveau de la conception (vs implantation)

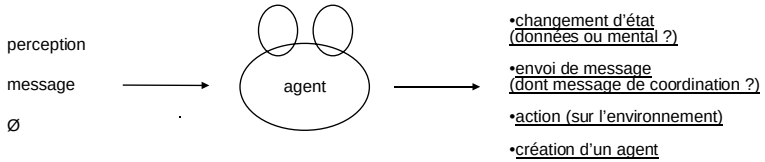
Bilan

- Le domaine des agents (agents logiciels, systèmes multi-agents...) est de fait encore relativement récent. Mais il aborde maintenant une nouvelle phase, des méthodes, des plates-formes de niveau pré-industriels sont maintenant proposées

- Quelque soit le type d'agent que nous envisagions, comment les construire ?
  - en ne réinventant pas la « roue » à chaque système
  - avec méthode et outils

Construire des agents

- Aspect essentiel du problème de la « sélection de l'action »
- Le calcul de cette sélection est a priori plus complexe que dans le cas des objets :
  - pas seulement procédural (ex : délibération)
  - nombreuses entrées (perception environnement, communication, coordination...)
  - « pro-activité » (et non plus juste « réactivité »), donc besoin d'arbitrage
  - mémoire complexe (ex : apprentissage)



- On appelle communément architecture d'un agent la structure logicielle qui réalise cette sélection

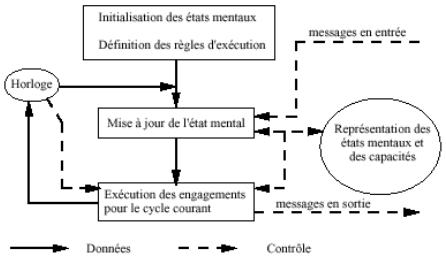
- savoir si on inclut dans l'architecture ou pas les modules d'actions, ex : de communication n'est pas essentiel ici.

Construction des agents

- Comment programmer cette architecture ?
  - dans un langage spécifique
    - » ex : Agent0, April
    - » avantages :
      - (censé être) spécialisé
      - de plus haut niveau
    - » inconvénients :
      - incompatibilité avec les standards (Java, etc.)
      - un seul langage est-il de toute manière adapté ?
        - ex : langages de communication (ACLs)
  - dans un langage généraliste
    - » Java, Smalltalk, C++, Lisp...
    - » et c'est donc l 'architecture qui concrétise la structure
  - Note : on peut utiliser des langages spécifiques pour les différents modules
    - » ex : KQML/ACL pour la communication
    - » ex : AgentTalk, SCD pour la coordination

Agent languages

- April [McCabe et Clark 95]
  - basé sur Prolog concurrent (Parlog)
  - utilisé par Fujitsu (McCabe)
  - assez bas niveau, manque de structure
  - langage d'acteur mais avec des restes d'habits Prolog :)
- Agent0 [Shoham 93]
  - basé sur la notion d'états mentaux (croyances et engagements)
  - unification du cycle de raisonnement et de traitement des messages



  - implémentation en Lisp
  - (do <temps> <action>)
  - (inform <temps> <autre-agent> <fait>)
  - (commit <condition-message> <condition-état-mental> <action1> ... <actionN>)

Architectures

- Nous appelons architecture d'un agent, la structure logicielle (ou matérielle) qui, à partir d'un certain ensemble d'entrées, produit un ensemble d'actions sur l'environnement ou sur les autres agents. Sa description est constituée des composants (correspondant aux fonctions) de l'agent et des interactions entre ceux-ci (flux de contrôle) [Boissier 2001]

- Allons voir du côté des architectures logicielles (et des composants), domaines explorés indépendamment des agents
- Les motivations sont différentes : concevoir des programmes à grande échelle (« programming in the large ») et pouvoir raisonner sur l'assemblage (connexion, compatibilité, propriétés) de composants logiciels
- Mais les principes sont proches et ces travaux éclairent :
  - les organisations d'agents (mais couplage encore trop fort par rapport aux agents)
  - et également surtout les architectures d'agents (au niveau d'un agent : « programming in the small »)

Architectures logicielles et organisations (d'agents)

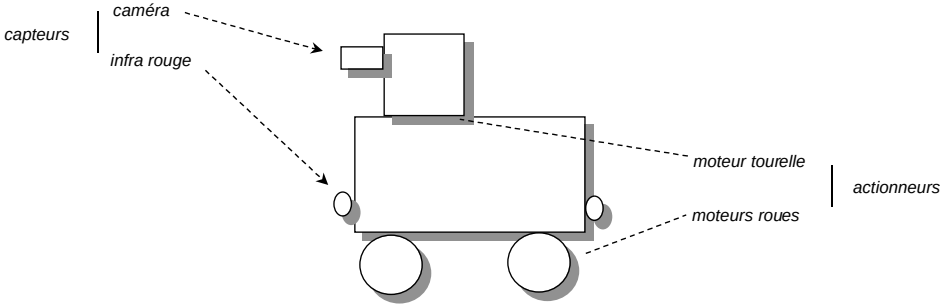
- Théorie (générale) des organisations - 3 points de vue [Scott 81] :
  - ***organisations rationnelles***
    - collectivités à finalités spécifiques
    - objectifs, ***rôles, relations (dépendances...), règles***
  - organisations naturelles (végétatives)
    - objectif en lui-même : survie (perpétuer l'organisation)
    - stabilité, adaptativité
  - systèmes ouverts
    - ***inter-relations/dépendances avec d'autres organisations, environnement(s)...***
    - échanges, coalitions

Architectures logicielles et organisations (d'agents) (2)

- Architectures logicielles
  - explicites
  - rationnelles
  - couplage explicite
    - au niveau des données (interfaces, typage)
    - et des modes d'interactions (connecteurs)
- Organisations d'agents (cognitifs)
  - explicites
  - rationnelles
  - couplage sémantique
  - réifiées
  - vers des organisations évolutives par elles-mêmes
- Organisations d'agents réactifs
  - bottom up émergentes (ex : société de fourmis)
  - conformantes top-down (cf. livre Alain Cardon, Conscience artificielle et systèmes adaptatifs, Eyrolles, 1999)

Achitectures d'agents - styles architecturaux (architectures logicielles)

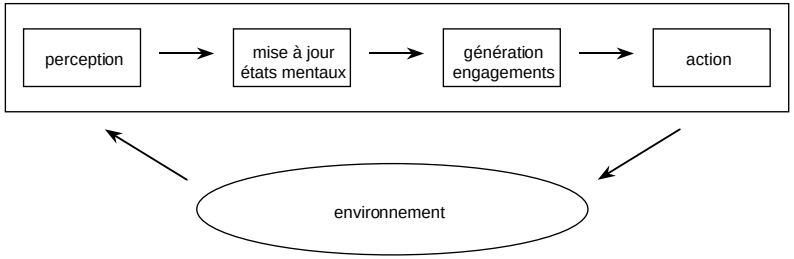
- Là, architecture = organisation individuelle / un agent (vision récursive)
- Exemple d 'application [Shaw et Garlan 96] :
  - (architecture de contrôle d'un) robot mobile autonome



- Propriétés/caractéristiques recherchées :
  - comportement à la fois délibératif et réactif
  - perception incertaine de l'environnement
  - robustesse (résistance aux pannes et aux dangers)
  - flexibilité de conception (boucle conception/évaluation)

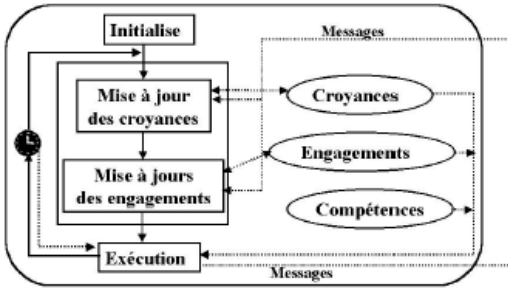
Architectures modulaires horizontales

- (une seule couche)
- cycle de calcul



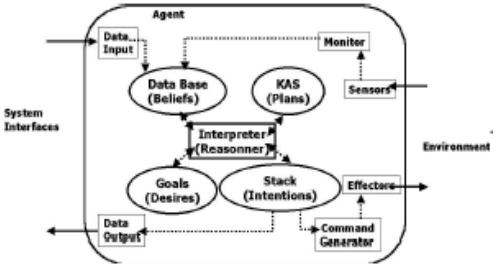
Architectures modulaires horizontales (2)

- **souvent basée sur notion d'états mentaux, engagements, intentions...**



AOP (Agent Oriented Programming) [Shoham 93]

plutôt dirigée par les états mentaux (data-driven)



PRS (Procedural Reasoning System) [Georgeff 87]

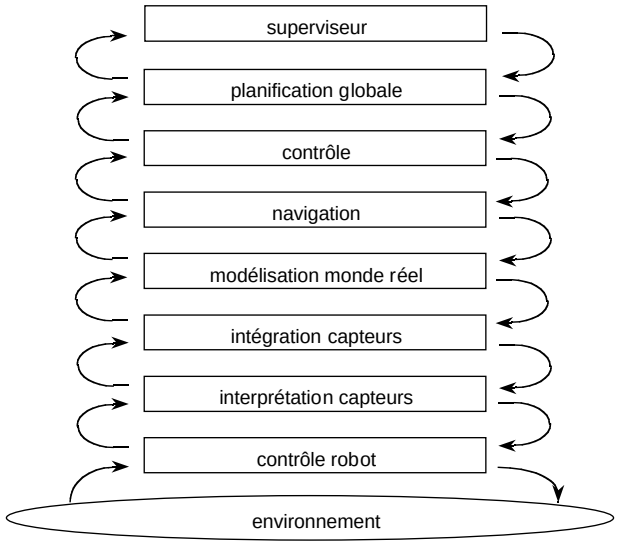
plutôt dirigée par les buts (goal-driven)

figures d'après [Boissier 2001]

Etats mentaux

- Etats mentaux
  - ex. d'architectures :
    - Agent0 [Shoham AI 93]
    - BDI [Rao et Georgeff 91]
  - formalisme logique (logique modale)
  - croyances
  - buts
    - comportements ou états désirés
  - plans
    - conditions de déclenchement
      - portant sur les croyances (data-driven) ou les buts (goal-driven)
    - actions ou sous-buts
  - intentions
    - intention = but persistant avec engagement d'accomplissement [Ferber@EcolIA'01]
    - intention = plan instancié (actif ou suspendu - en attente conditions)
    - intention = choix + engagement [Cohen et Levesque AI 90]
  - intentions jointes [Cohen et Levesque 95]
  - TeamWork [Tambe 99]

Architectures en couches (architectures verticales)



Architectures en couches (architectures verticales) (2)

- InteRRap [Müller 94]
- 3 couches activées en //
  - comportement - croyances sur état environnement
  - planification locale - croyances sur soi-même
  - planification coopérative - croyances sur et engagements avec les autres

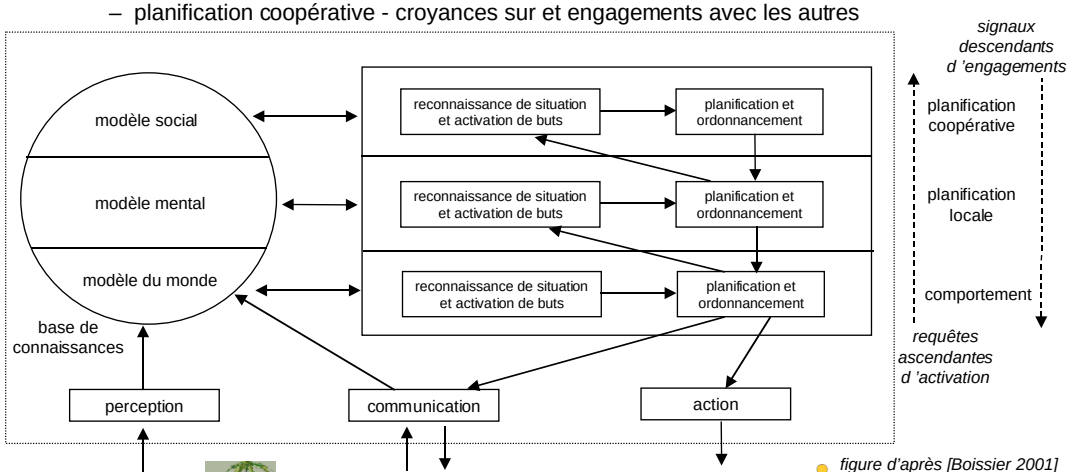


figure d'après [Boissier 2001]

Architectures en couches (architectures verticales) (3)

- TouringMachine [Ferguson 92]
- 3 couches activées en //
  - réaction
  - planification
  - modélisation (des entités y compris l'agent)
- contrôleur central - filtre perceptions et commandes (actions)
  - règles de contrôle (de censure et de suppression)

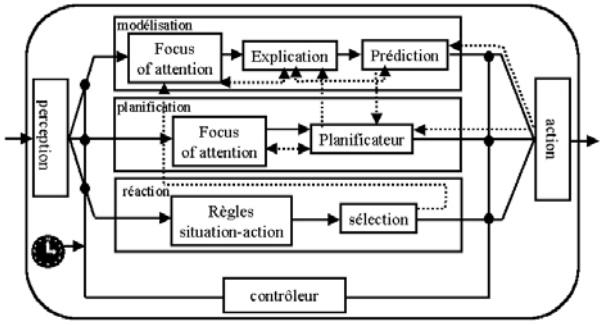
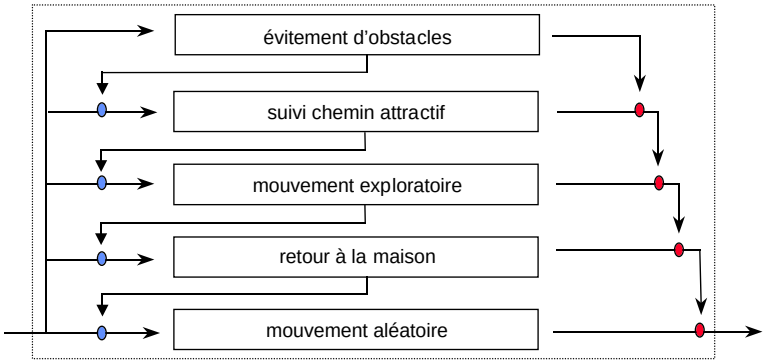


figure d'après [Boissier 2001]

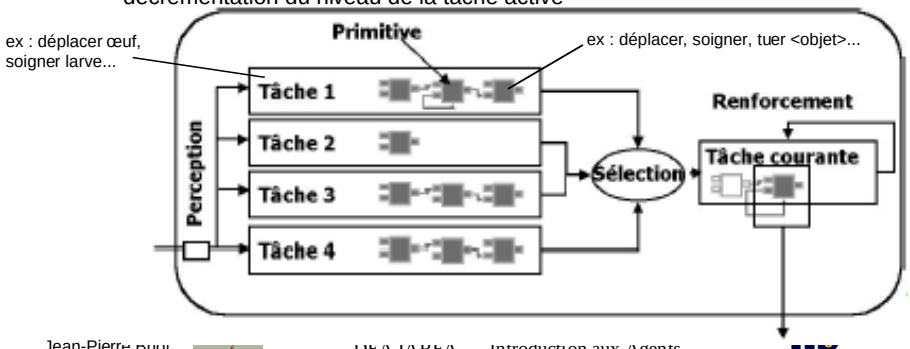
Architectures réactives en couches (verticales)

- « subsumption architecture » [Brooks 86]
- composants activés en parallèle
- compétition mais aussi hiérarchie
- priorités et inhibitions :
  - supplanter entrée composant inférieur
  - inhiber sortie composant inférieur



Architectures réactives en couches (2)

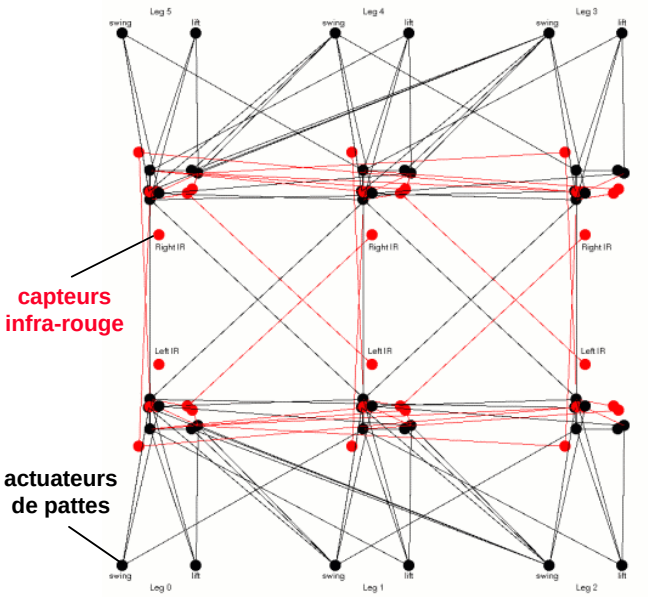
- MANTA [Drogoul 93]
- tâches indépendantes
  - poids (importance au niveau de l'agent)
  - niveau d'activation (calculé à partir du poids et de l'intensité des stimuli)
- sélection (par compétition) parmi les tâches
  - une (seule) tâche
    - nouvelle version (pour robots), primitives réflexes (ex : évitement d'obstacles) activables en //
  - niveau d'activation le plus élevé
  - décrémentation du niveau de la tâche active



Architectures d'agents

- hiérarchiques (cognitives/réactives)
  - ex : DIMA [Guessoum 96], InterRap [Müller 96]...
- componentielles
  - ex : Maleva [Lhuillier 98] [Meurisse 2000]
  - SCD [Yoo 98]
- composition d'actions
  - ex : Bene theory [Steels 94]
- connexionistes
- évolutionistes
  - algorithmes génétiques, morphogenèse

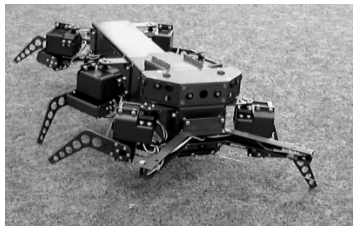
Evolution de l'architecture de contrôle d'un robot marcheur [Meyer et al. 98]



Evolution du programme de développement (instructions: DIVIDE, GROW, DRAW...)

Résultat :

Réseau noir : marche
Réseau rouge : évitement d'obstacle

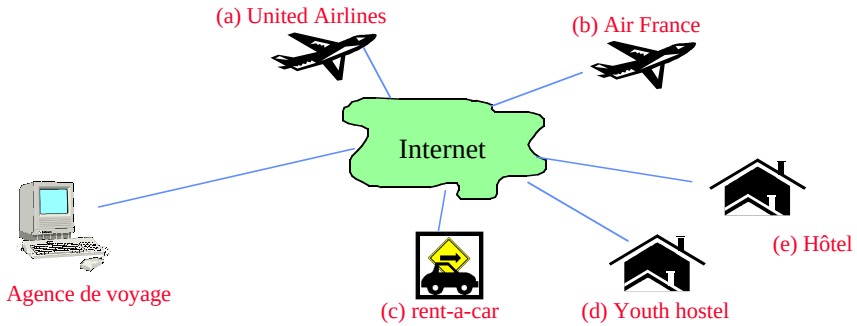


Plates-formes

- Architectures d'agents
- Bbiliothèques de
  - comportements
  - protocoles de coordination
- Outils
  - ex : noyaux de systèmes experts : JESS, JRules...
- Environnements de déploiement et d'exécution
- Environnements de visualisation et d'analyse des résultats
- Standard FIPA
- Plates-formes industrielles
  - Jack
  - AgentBuilder
  - Zeus, ...
- Plates-formes académiques
  - DIMA
  - MadKit
  - MASK, ...

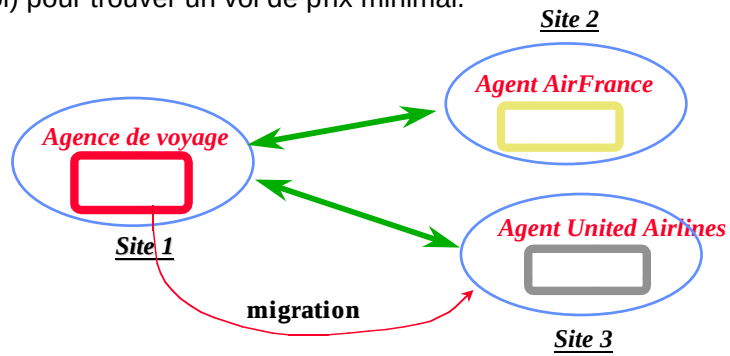
Ex : Framework et composants pour des agents mobiles coopératifs pour le commerce électronique [Yoo 1999]

Scénario de l'agence de voyage électronique (FIPA)



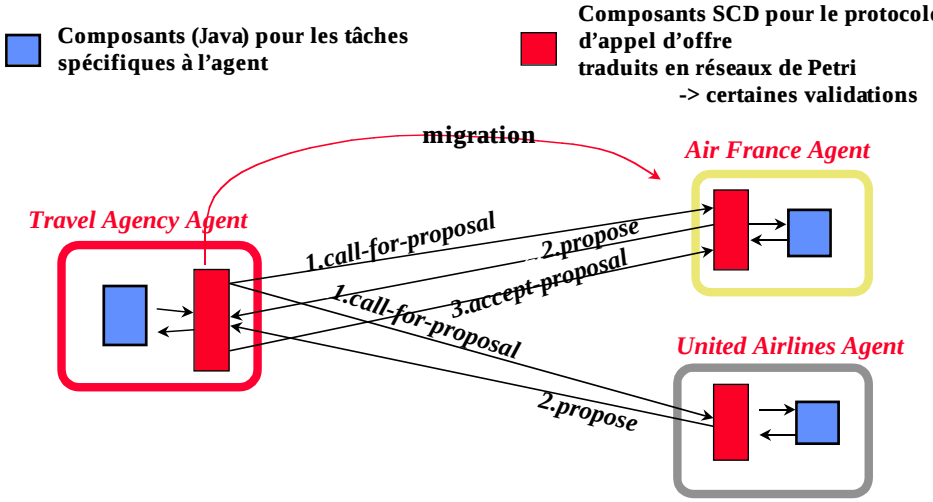
Exemple de protocole de coopération entre agents : choix du meilleur billet d'avion

- Deux agents serveurs de voyage, un agent agence de voyage
- Coopérer suivant un protocole d'appel d'offre (Contract net protocol) pour trouver un vol de prix minimal.



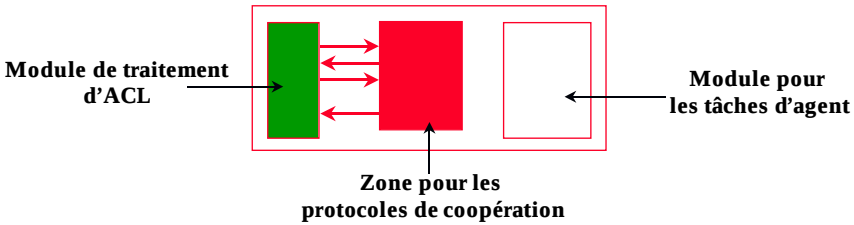
- Mobilité : l'agent se déplace vers le site du serveur choisi pour continuer la conversation (et optimiser les communications)

Coordination (Contract Net Protocol)



Framework d'agents

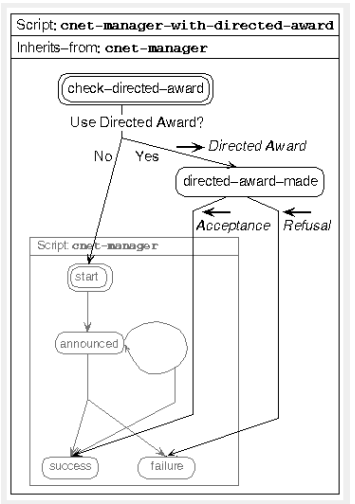
- Caractéristiques :
  - Dissocier le traitement du langage ACL du traitement du protocole de coopération
  - Le composant de traitement de l'ACL est proposé par le framework
  - Zone prédéfinie pour les protocoles de coopération
  - Connexions pré-établies entre le composant ACL et la zone pré-définie pour les protocoles de coopération



Facilite la modélisation

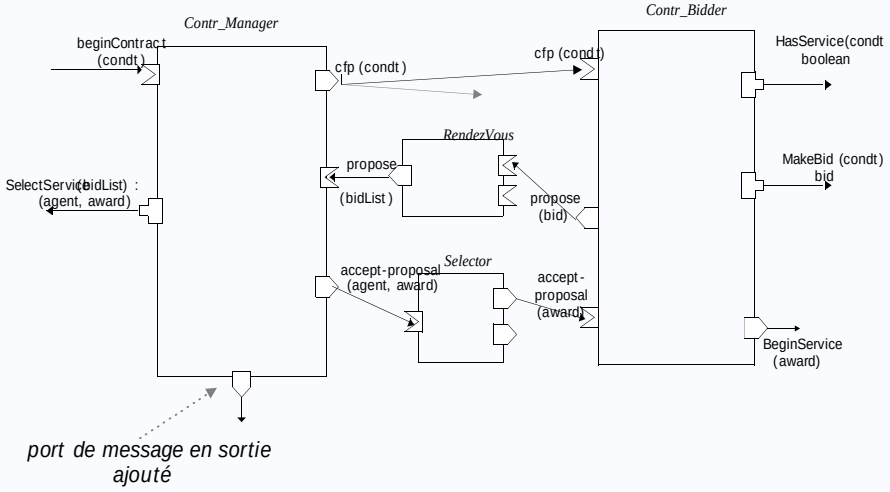
Réutilisabilité des composants (quelques résultats)

- Conception incrémentale de protocoles de coopération - à partir du Contract Net
  - par héritage
    - » ex : réalisation du protocole d'appel d'offre avec délai de temps : timeout Contract Net
  - par composition
    - » ex : extension en un FIPA-Iterated Contract Net
  - Expérimentations de réutilisation analogues avec AgentTalk (langage de coopération) par héritage [Kuwabara et al. 95]

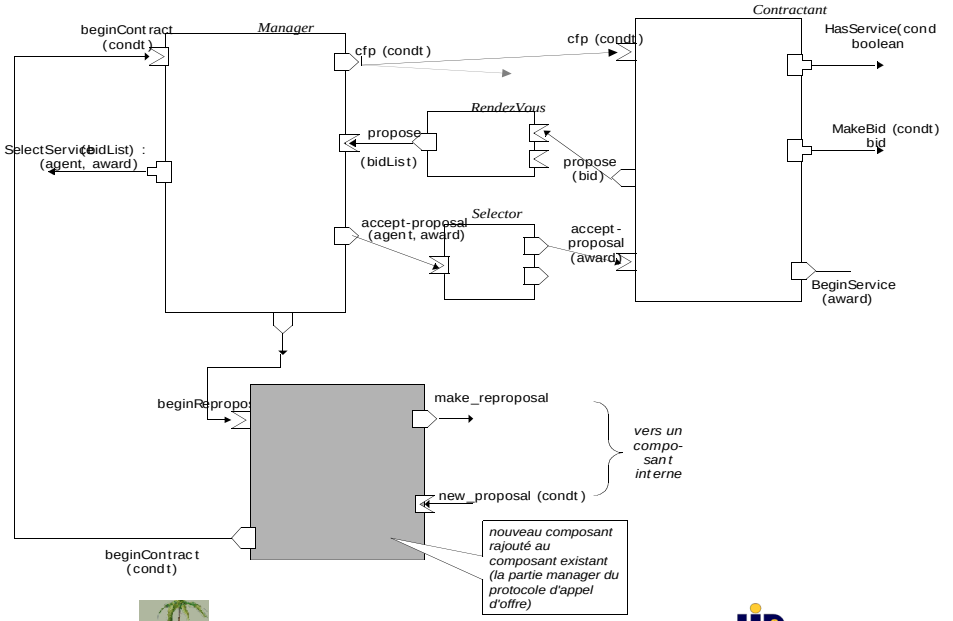


Raffinement de composants (par composition)

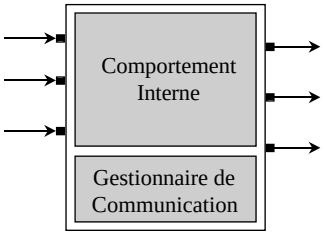
Iterated Contract Net Protocol

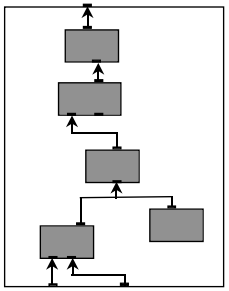


Raffinement de composants (par composition) (2)



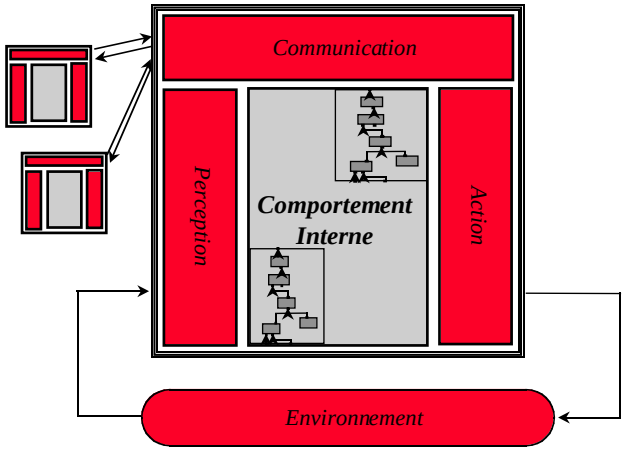
Composants d'agents Maleva [Lhuillier 98] [Meurisse 2000]





- Un Composant est défini par :
  - un Comportement Interne
  - des Bornes de Communication
- **Pas de référence explicite entre composants**
  - permet la modification du graphe des connexions indépendamment des composants.
  - entités *potentiellement* réutilisables car définition indépendante de l'environnement logiciel.

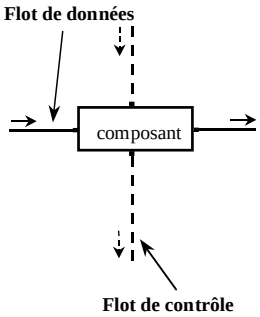
Maleva (2)



- Idée : modéliser le ***comportement interne*** des agents avec des composants

Maleva (3)

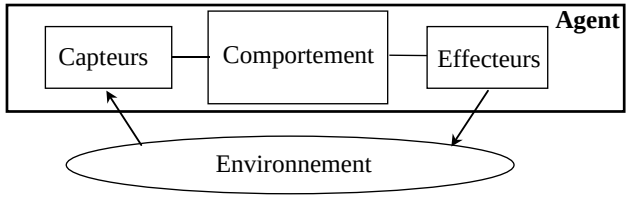
Principes :



- **Séparation explicite des flots de contrôle et de données**
  - Permet une plus grande généricité via l'expression de différents contextes de contrôle pour des mêmes composants
  - contrôle de haut niveau du séquencement (primordial pour limiter les biais de simulations)
- **2 types de bornes :**
  - ***Bornes de données***
    - Modification des ***variables d'instance*** du composant
  - ***Bornes de contrôle***
    - Un ***comportement encapsulé*** n'est enclenché que lors d'une activation via une borne de contrôle associée.

Exemples de Conception

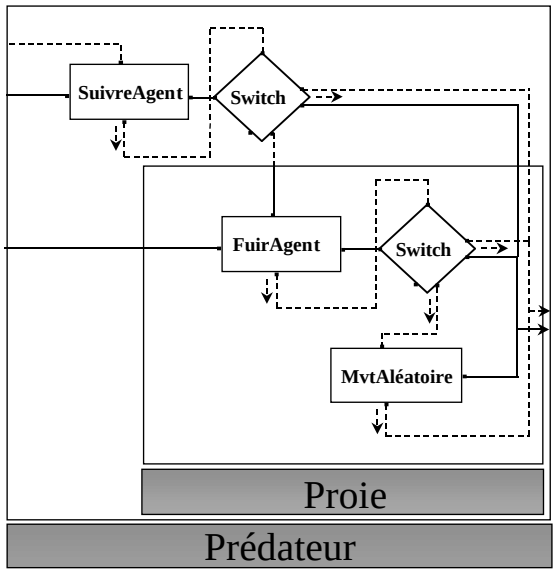
Agents situés dans un écosystème simulé



Exemple des Proies / Prédateurs

Les proies	Les prédateurs
•Fuient les prédateurs •Mouvement aléatoire	•Poursuivent les proies •Fuient les prédateurs •Mouvement aléatoire

Proies et prédateurs



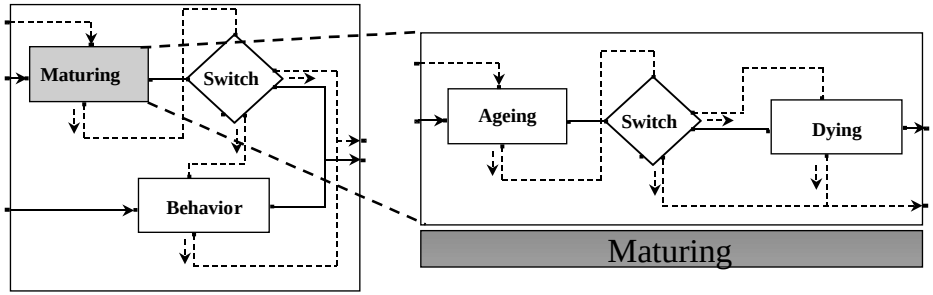
Exemples de Conception
Approche Descendante - *Les fourmis* [Guillemet et al. 98]

- Reformulation de MANTA [Drogoul 93] : simulation d'une colonie de fourmis
- Mise en évidence des notions de réutilisabilité et de dynamicité architecturale du modèle
- Différents types d'agents : reine, ouvrière, oeuf, larve
- Aspect dynamique des agents (et de leurs comportements) : passage de l'oeuf à la larve, de la larve à l'ouvrière
- Comportement commun de tout être vivant : agents évoluent en fonction du temps, naissent, vieillissent et meurent

-> ***pattern*** !

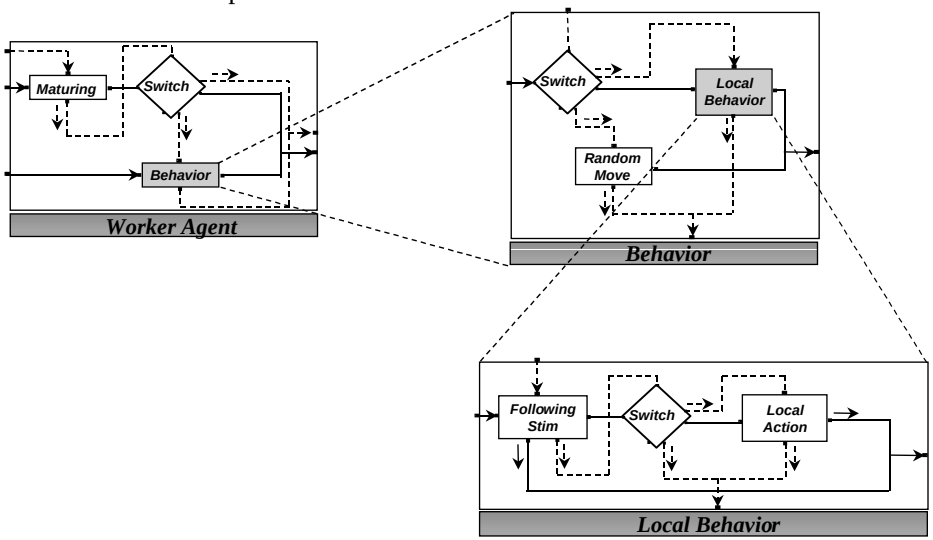
Maleva - les fourmis *(2)*

Comportement commun de tout être vivant : Le composant ***Maturing***
Agents évoluent en fonction du temps, naissent, vieillissent et meurent



Maleva - les fourmis *(3)*

Décomposition d'une ouvrière

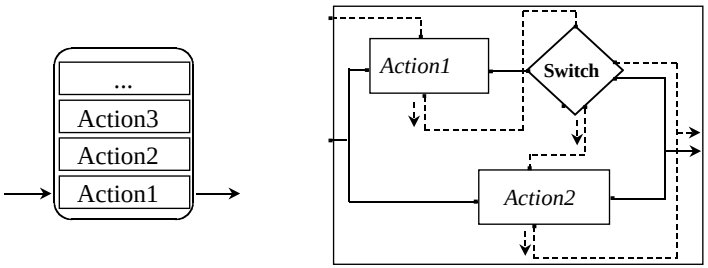


Maleva - Design Patterns

- Complexité potentielle des schémas de connexion de composants
- Idée : guider le concepteur d'architecture
  - Par contrainte : **typage de connexions**
  - Par aide à la conception : **design patterns**
- Recherche de *Design Patterns* à partir de schémas de composition récurrents

Maleva - Design Patterns (2)

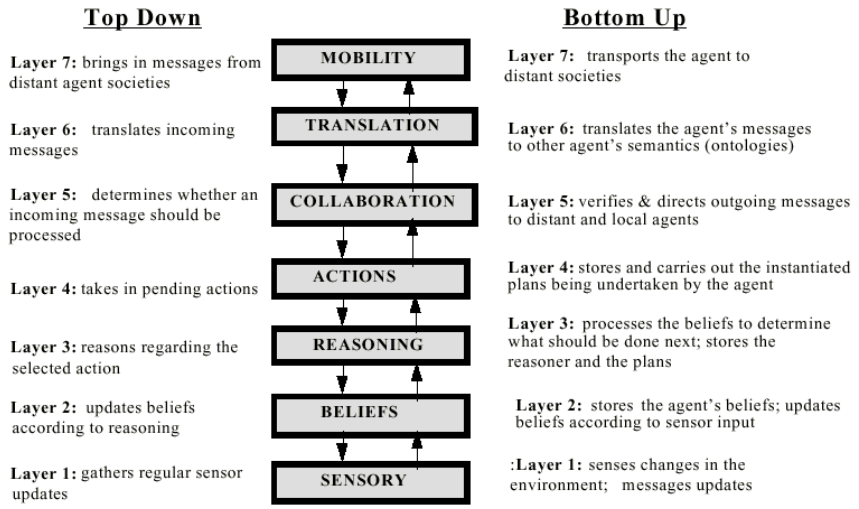
- décomposition du flot de contrôle des langages impératifs : *if...then...else*
- architecture de Subsomption de Brooks [Brooks91]



- capacité à vieillir (Composant ***Maturing***) dans des simulations d'écosystèmes

Autres Design patterns pour agents

- Marques [Sylvain Sauvage@EcolIA'01]
- Layered agent pattern [Kendall et al. 95]



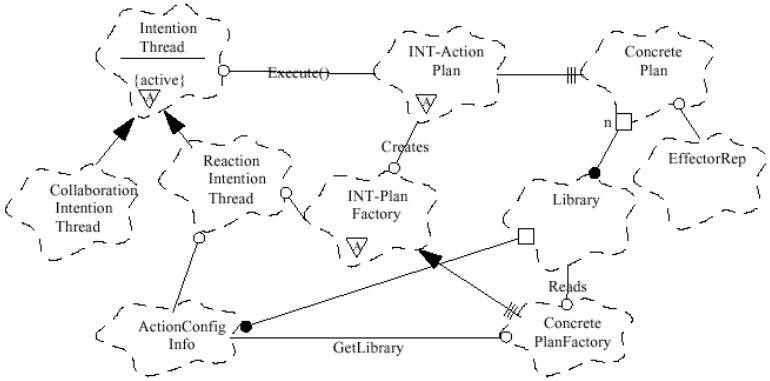
Plan as command [Kendall et al. 95]

Problem

How can a plan be encapsulated as an object ?

Forces

- Each Intention has a plan to execute. They have a wide range and are known only at run time.
- A plan specifies primitive actions, executed directly by the effectors or the Collaboration layer interface.
- There is a need to define a structure for plans that provides high level operations based on primitive ones.



Clone pattern [Kendall et al. 95]

Problem

How can an agent relocate itself and become resident in distant societies ?

Forces

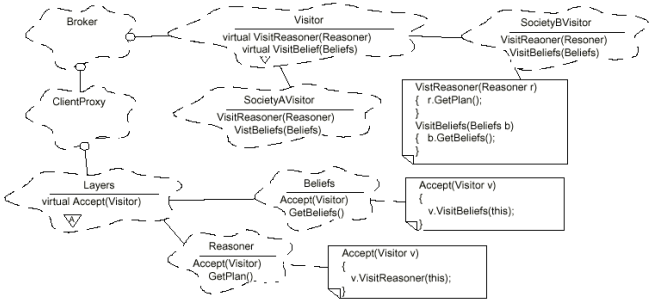
- An agent must be able to bring its capabilities, facilities, and state with it to a new society.
- The agent must be able to travel to a remote location and interact, negotiate, and exchange information in the new society.

Solution

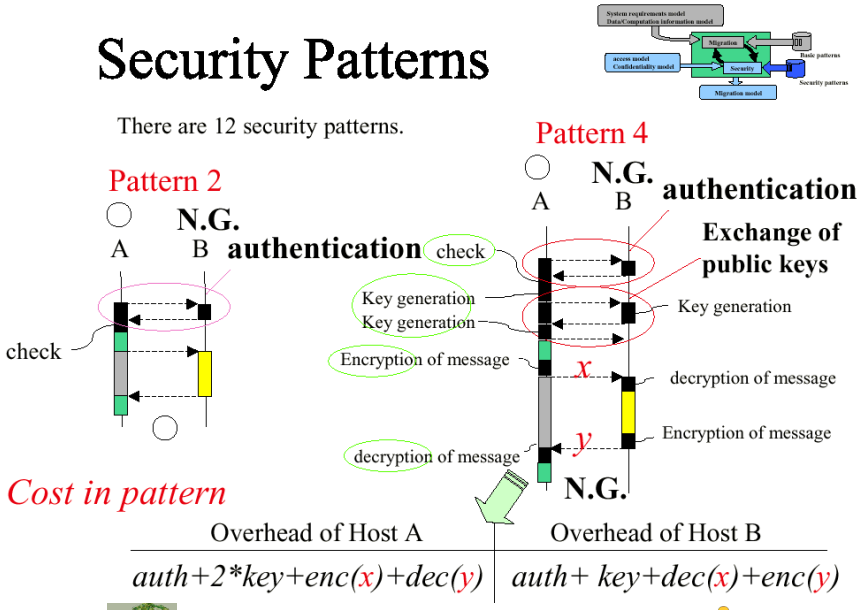
Make a copy or clone of the original agent, and place the new agent in the distant society. The clone must have all of the capabilities and facilities of the original agent, along with any state information.

Known Uses

The original use of agent self replication was cooperating mobile WAVE agents [3]. More recent approaches that utilize cloning include IBM Aglets [13], and the Agent Transfer Protocol (ATP) [14]. An aglet is a Java object that can move from one host on the Internet to another. When the aglet moves, it takes along its program code as well as its state (data). Bradshaw [5] refers to agent cloning as teleportation.



Security pattern for mobile agents [Honiden et al. 2000]



Plan : Conclusion

- Des objets aux agents
  - plus de sémantique
  - comportement et interactions arbitrairement complexes
  - couplage plus adaptatif entre entités
  - organisation explicite
- Convergence nécessaire entre :
  - systèmes multi-agents
  - génie logiciel
  - algorithmique répartie

Ouvrages et pointeurs

- Les Systèmes Multi-Agents, Jacques Ferber, Interéditions, 1995
- *Software Agents, édité par Jeff Bradshaw, AAI-Press - MIT-Press, 1997*
- *Multi-Agent Systems, édité par Gerhard Weiss, MIT-Press, 1999*
- Principes et Architecture des Systèmes Multi-Agents, édité par Jean-Pierre Briot et Yves Demazeau, Collection IC2, Hermès, 2001

- *Revue Autonomous Agents and Multi-Agent Systems, Kluwer*

- www.multiagent.com
- www.fipa.org
- www.agentlink.org