Master Informatique
Cours Systèmes Multi-Agents

Aspects Logiciels des Systèmes Multi-Agents

Copie transparents (à venir) en :
`http://www-poleia.lip6.fr/~briot/cours/sma-logiciel04-05.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

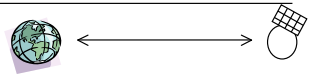
- Pourquoi les Agents et les Systèmes Multi-Agens ? Motivations
- Problèmes Scientifiques
- Classification (Types d'Agents)
- Code et Agents Mobiles
- Principes des Systèmes Multi-Agents
- Positionnement des Agents dans l'Evolution de la Programmation
- Langages et Architectures d'Agents
- Architectures Réflexives comme Techniques d'Adaptation
- 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 (apprennent 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) »

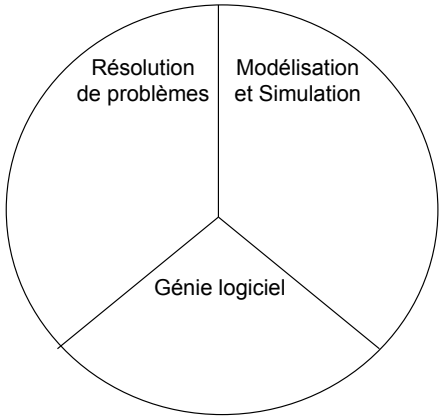
  - De agens « celui qui fait, qui s 'occupe de »
    - « Personne chargée des affaires et des intérêts d'un individu, groupe ou pays, pour le compte desquels elle agit »
    - « Appellation de très nombreux employés de services publics ou d'entreprises privées, généralement appelés à servir d'intermédiaires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

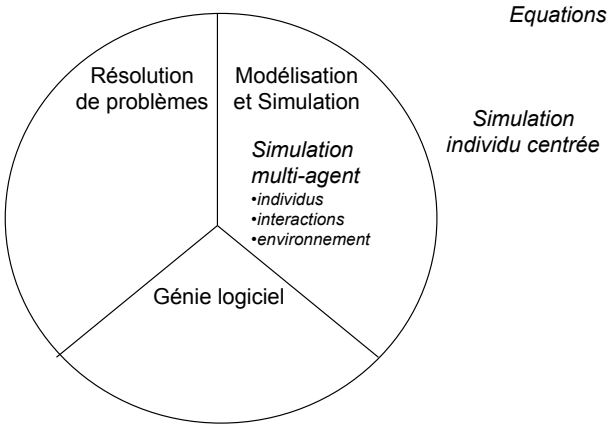
Qu'est-ce qu'un agent ? (2)

- [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.

Les Systèmes Multi-Agents (SMAs) : pour quelles classes de problèmes scientifiques ?



Modélisation et simulation



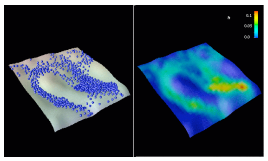
Equations

Simulation individu centrée

Simulation multi-agent

Exemples :

- Éthologie (colonies de fourmis)



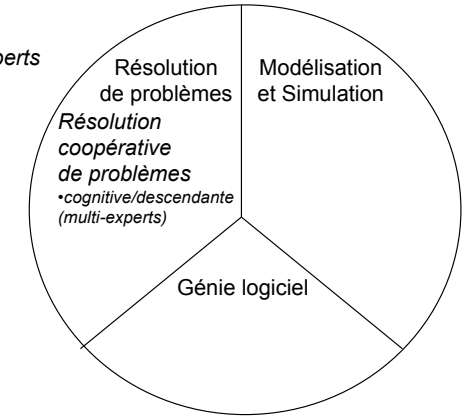
- Hydrologie
- Transports

- Phénomènes sociaux
- …

Résolution de problèmes

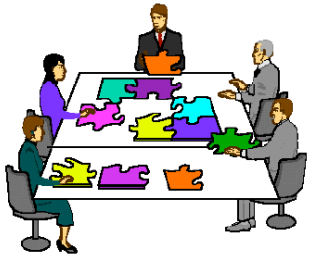
Systèmes experts

Recherche Opérationnelle
•approches exactes
•méta-heuristiques

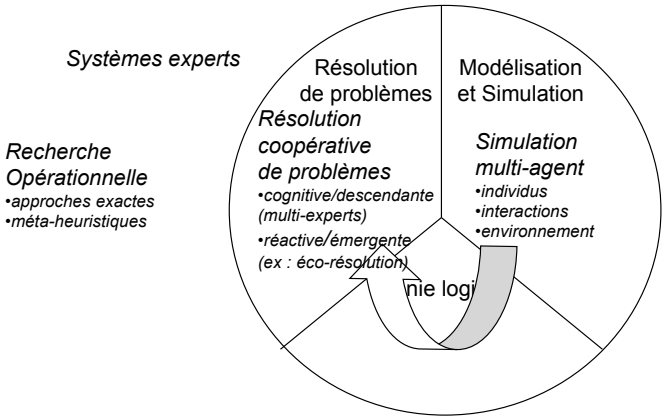


Résolution de problèmes multi-agent

- Agents cognitifs
  - agents = experts solveurs
  - multi-expert
  - décomposition fonctionnelle
  - approche descendante
  - coordination explicite

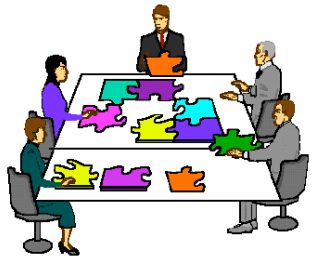


Les SMA : pour quelles classes de problèmes scientifiques ?

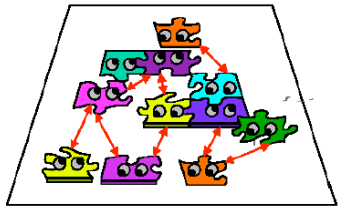


Résolution de problèmes multi-agent

- Agents cognitifs
  - agents = experts solveurs
  - multi-expert
  - décomposition fonctionnelle
  - approche descendante
  - coordination explicite

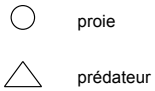


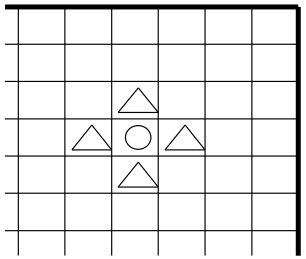
- Agents réactifs
  - agents = parties du problème lui-même
  - décomposition structurelle
  - approche ascendante
  - comportement collectif émergent



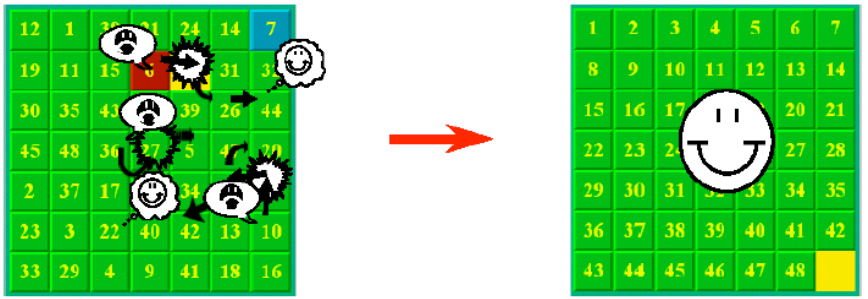
Exemple des proies-prédateurs

- 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 et répulsion (faible) entre prédateurs



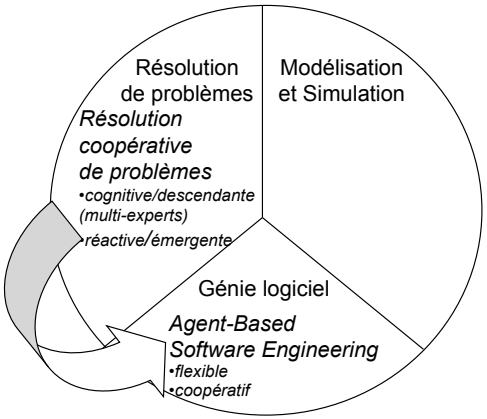


Eco-résolution [Ferber 89, Drogoul 91]



[tiré de Drogoul 98]

Génie logiciel

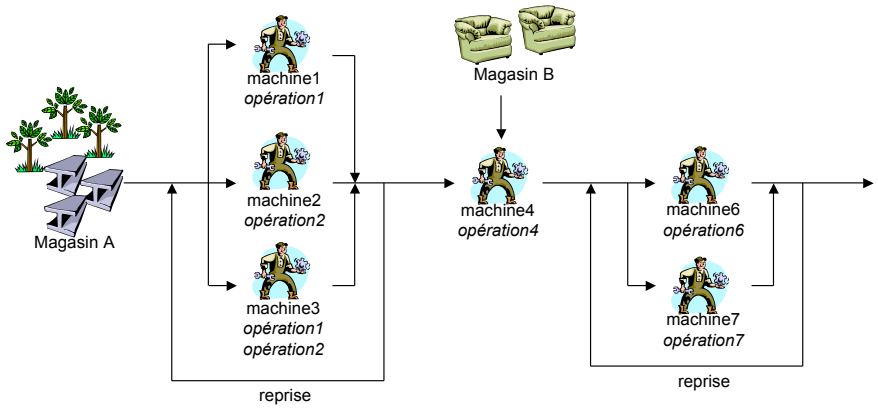


Méthodes Objet *Systèmes répartis*

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

- Problème clé du logiciel : évolution, adaptation
  - autres logiciels, 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]
  - Possibilité d'objectifs en compétition, avec moyens de réconciliation
  - *à l'inverse (et complémentaire) de la programmation « sécuritaire » classique*
  - 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)
  - Les composants logiciels peuvent en partie eux-mêmes organiser leur assemblage (appareillement, appel d'offre, négociation…)
  - Vers un plus grand découplage, ou plutôt un couplage plus sémantique et plus flexible : objets -> composants -> agents (et ensuite ?!)
  - A rapprocher du "Ever late binding" (C -> C++ -> Java -> ...)

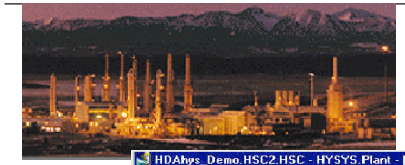
Exemple : Atelier de production

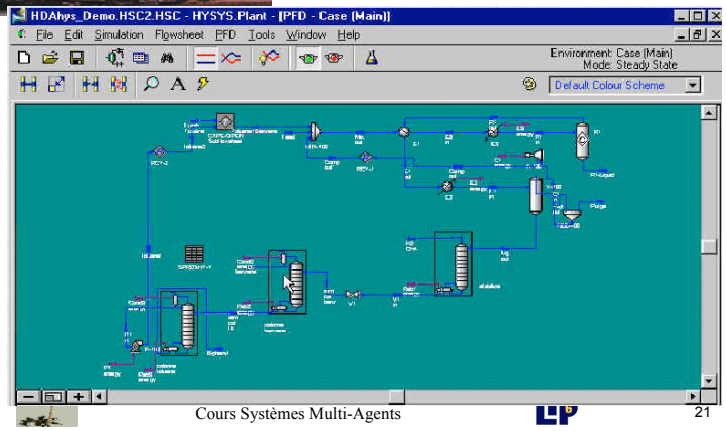


Exemple : Atelier de production (Jennings 1999)

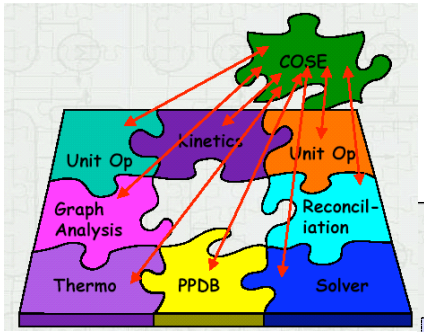
- Vision classique : descendante, centralisée et planifiée
  - conception explicite et globale par le concepteur
  - ordonnancement global des tâches
  - Pros : planification, vérification
  - Cons : gestion des retards et pannes (inévitables), par réordonnancement ou traitement d'exceptions
- Vision multi-agent : émergente
  - Machines et pièces sont des agents
  - Décisions locales de chacun
  - Négociation entre machines et pièces pour le traitement
  - Coordination pour l'assemblage et création d'agents composites
  - Pros : adaptabilité, robustesse, règles locales
  - Cons : moindre prédictabilité

Exemple : Procédés pétrochimiques

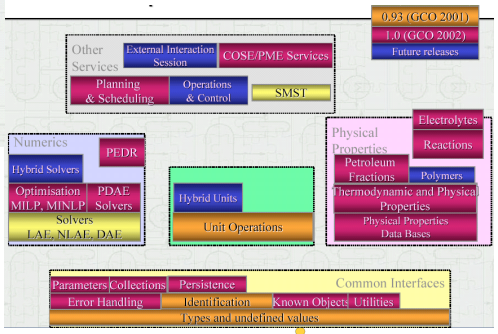




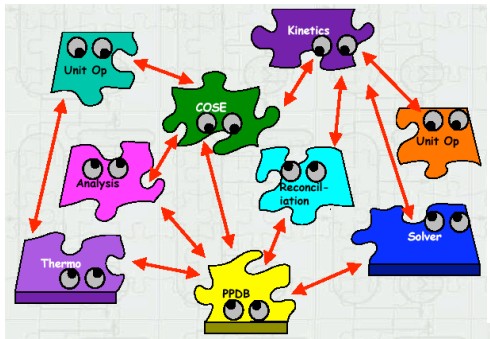
Procédés pétrochimiques
« Componentification » : Projet CAPE-OPEN [Braunschweig et al.]



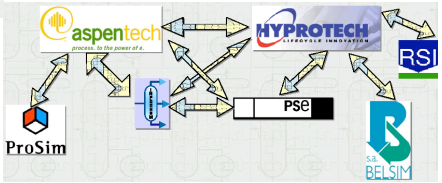
- Interopérabilité
- Standard d'interfaces
- Composants sur étagères



Procédés pétrochimiques
« Agentification » : Projet COGENTS [Braunschweig et al.]

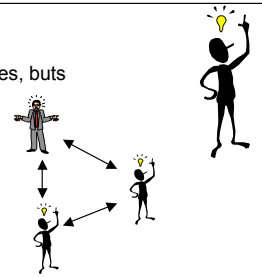


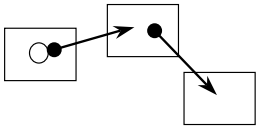
- Assistance à l'assemblage
- Appareillement (match-making)



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





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

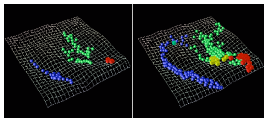
typologie (Babel agents) 2/3

- 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)
  - modèles de comportement arbitrairement complexes
  - interactions arbitrairement complexes (ex : sociales : irrigation parcelles)
  - niveaux hiérarchiques (ex : bancs de poissons)
  - espaces et échelles de temps hétérogènes
  - couplage de processus



- agents de loisir
  - virtuels (ex : jeux vidéo)
  - virtuels-physiques (ex : Tamagotchi)
  - physiques (ex : Furby, robot-chien Aibo de Sony)

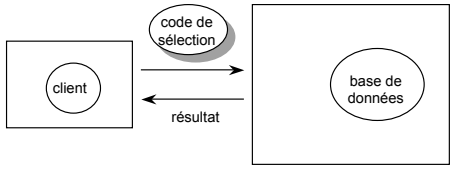
- agents artistiques
  - art interactif
  - éco-système
  - créatures virtuelles

Agents robotiques

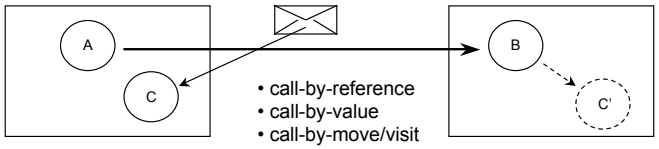
- jouet
- équipe (RoboCup)
- aspirateur
- tondeuse-à-gazon
- démineur
- ...

Code/objets/agents mobiles

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

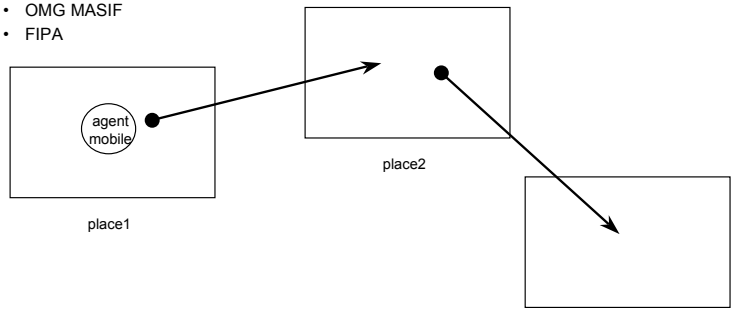


- Objet mobile
  - PostScript (code + données constantes)
  - Emerald [Black et al. IEEE TSE 87]



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

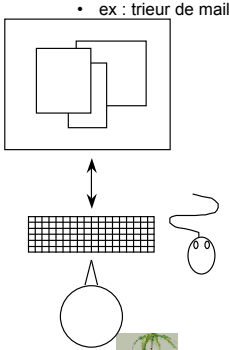


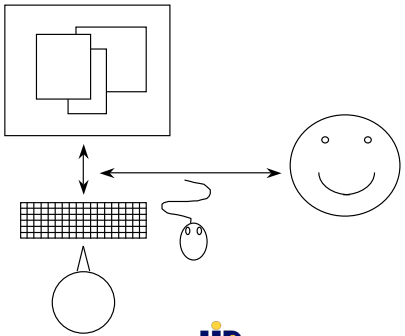
Agents mobiles (2)

- 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 et Multi-Agents pour l'Art Digital

- Evolution de l'art
  - De l'objet artistique créé seulement par l'artiste...
  - ...au processus artistique interactif (avec participation du public)

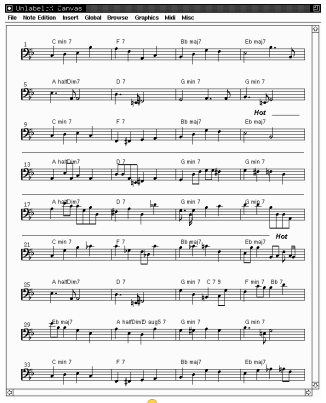
- Utilisation de techniques digitales
  - De l'art auto-référentiel et opposé à la science...
  - ...vers une nouvelle convergence avec la science ?
    - » (ex : visualisation scientifique, vie artificielle, cognisciences)
  - Mais pas (encore) de méthodes systématiques (au sens science/technologie)

- Quel rôle ont ou peuvent avoir les concepts d'agent et de système multi-agent ?
  - Participation du public... et de créatures virtuelles (médiateurs entre artistes et public ?)
  - Des agents rationnels automates...
  - ...aux agents interactifs et collaborateurs
  - Simulation d'éco-systèmes interactifs
  - Ex. en : Musique (improvisation)
  - Animation - créatures et mondes virtuels

Agents Internes et Agents Médiateurs

Agents et Musique

- Agent improvisateur (lignes de basse walking/bebop)
- bassiste de Jazz [Ramalho 1997]
- idée simuler le comportement d'un bassiste de Jazz (bebop)
- agent rationnel
- modèle "cognitif" d'un bassiste
- résultat de nombreuses interviews
- techniques sophistiquées
- notion d'interaction abstraite
  - (scénario/contexte)



Agent bassiste

- analyse harmonique de la grille d'accord
  - par découpage en segments (cadences harmoniques, ex : II-V-I)
  - règles de production
- règles d'improvisation
  - ex : jouer un arpège sur cet accord,
  - jouer la dernière phrase en la transposant,
  - jouer chromatique...
- mécanisme de sélection et de composition de ces règles
- contraintes temporelles / temps réel :
  - si pas assez de temps, recherche de clichés "plans" dans une mémoire musicale indexée (**raisonnement à partir de cas**)
  - mémoire musicale constituée de lignes de basse (ex : Ron Carter) sur un certain nombre de standards
- notion de scénario (découvert en temps réel)
  - ex : le pianiste joue en mode dorien,
  - le soliste joue beaucoup de notes,
  - la police arrive
- évaluation - test de Turing musical

Ornithology

Agent continuateur

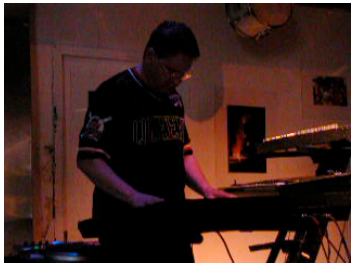
- Interaction homme - agent

- Continuator project [Pachet 2001]
- idée : ne pas remplacer
- mais "étendre" ("continuer") le jeu d'un musicien
- apprend le style

- différents modes d'utilisation :
  - étendre (ex : jouer plus vite qu'Al di Meola !)
  - jouer avec son double
  - jouer avec son musicien préféré
  - collaboratif, hybride

Agent continuateur

- apprentissage incrémental (on-line non supervisé)
- chaînes de Markov améliorées
- pas de connaissance musicale a-priori (à la différence du bassiste virtuel)
- collaborateur plutôt qu'automate (Vaucanson) remplaçant l'homme
- sensibilisation/pédagogie (expérimentations dans une école)



Agents et Animation

- Créatures virtuelles

- De l'automate réactif
  - Microsoft agents

- A l'automate plus cognitif
  - Agents de formation
    - » (Hayes-Roth, Johnson...)

Agents et Animation

- Vers une collectivité interactive
  - Narration interactive (Interactive story telling)
  - [Cavazza 2000]

Narration interactive

- Histoire avec plusieurs personnages (ex : Sitcom "Friends")
- Acteurs artificiels (Rachel, Ross...)
- Intervention/influences du spectateur
  - ex : dire (voix) une information à un des acteurs
  - déplacer un objet (cacher un objet, fermer une porte...)

- acteur artificiel
  - plans hiérarchiques
  - planification des actions
  - replanification si échec de l'exécution du plan

Simulation d'éco-systèmes - Mondes interactifs

- Ex : installations de Sommerer et Mignonneau
- souvent inspiration au départ naturelle
- (éco-systèmes, plantes, animaux dans aquarium virtuel...)
- entités virtuelles (plantes, animaux...)
- interaction avec le public (création, évolution...)
- influences de la vie artificielle (Artificial Life)
  - comportements
  - évolution
  - adaptation...

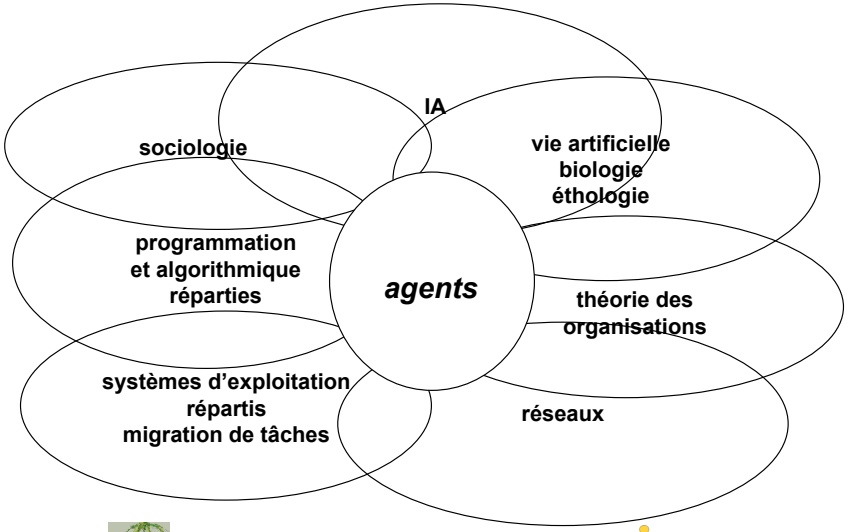
Mondes interactifs - Interactive Plant Growing

- Plantes réelles comme interfaces entre plantes virtuelles et public
  - (inversion des médiateurs - en général médiateur virtuel d'un phénomène réel)
  - interface : différence de potentiel électrique entre plante et public

Mondes interactifs - A-Volve

- Aquarium virtuel peuplé de créatures virtuelles
- Créatures conçues/créées par interface graphique
- Modèle bio-mécanique (dépend de la forme définie à la création)
- Interaction du public avec créatures virtuelles (ex : protection contre prédateurs)
- Évolution (prédation, reproduction), influencée par le public

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

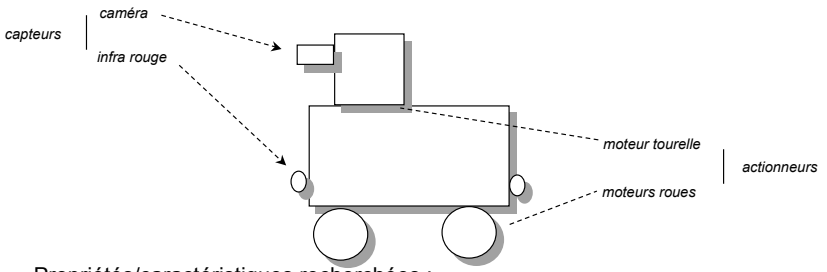
- agents cognitifs
  - représentation explicite
    - soi
      - connaissances (beliefs)
      - buts (intentions)
      - tâches
      - plans
      - engagements
    - environnement
    - autres agents
      - compétences
      - intentions
  - architectures complexes, souvent modèle logique (ex : 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 d'ordonnancement, générateurs de nombres pseudo-aléatoires

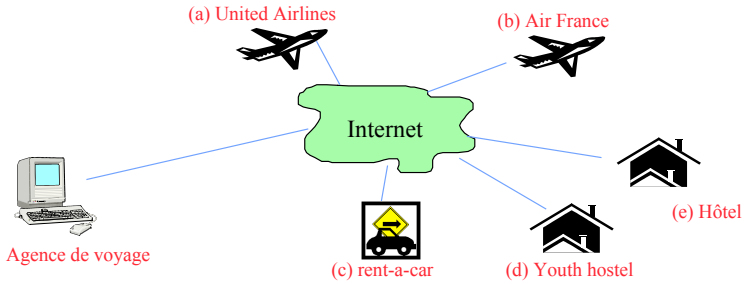
Agent, dans l'œil de l'observateur ??

- 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

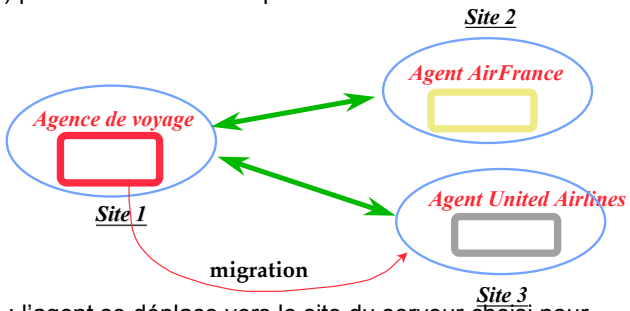
- décomposition des tâches, plans, sous-buts
- assignation aux agents
  - division du travail (spécialisation) vs totipotence
  - organisation, rôles
  - réseaux d'accointances
    - représentations des capacités des autres agents
  - appel d'offre
    - Contract Net protocol [Smith IEEE Transac. Computers 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)

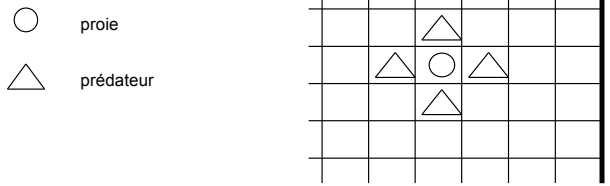
- agents cognitifs
  - organisations explicites
- agents réactifs
  - organisations semi-implicites
    - façonnement de l'environnement, ex : fourmilière
    - « auto-organisation », ex : stigmergie des colonies de fourmis
- Exemple : extraction de minerai par des robots [Ferber 95]
- spécialisation 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)

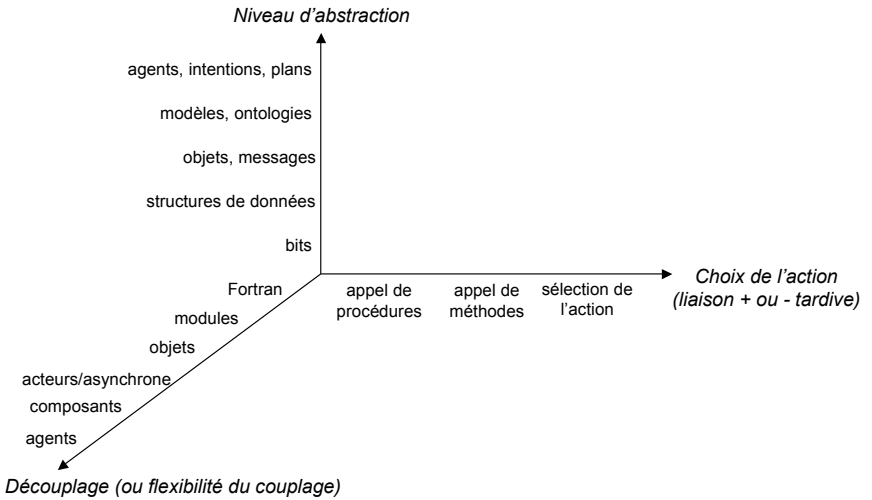
- Langages et protocoles de communication
- interoperabilité d'agents (CORBA des agents)
- KQML [Finin et Labrou 94] message
  - contenu
  - langage (d'expression du contenu)
    - ex : Java, Smalltalk, KIF, XML
  - ontologie
    - hiérarchie de concepts pour un domaine donné (ex : commerce e-, automobile...)
  - 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

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...

Evolution de la programmation



Axe 1 - Choix de l'action - Evolution de la programmation

[Odell 99]

	Programmation monolithique	*Programmation modulaire*	*Programmation par objets*	*Programmation par agents*
Comportement	non modulaire	modulaire	modulaire	modulaire
Etat	externe	externe	interne	interne
Invocation (et choix)	externe	externe (appel)	externe (message)	Interne (règles, buts)

Axe 1 - Choix de l'action - Evolution de la programmation

- Interview Les Gasser, IEEE Concurrency 1998

- langage machine
- assembleur
- programmation structurée
- programmation par objets
- (programmation par composants)
- programmation par agents !

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

- *mission et initiatives pour l'accomplir*

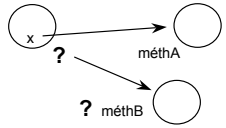
action persistante

- 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

Axe 2 - Découplage - Des objets aux composants

Limites des objets :

- granularité encore trop fine-moyenne
  - pas trop bien adapté à la programmation à grande échelle
- pas encore assez modulaires
  - références directes entre objets
  - donc connexion non reconfigurable sans changer l'intérieur de l'objet
    - » objet appelé
    - » nom de la méthode appelée

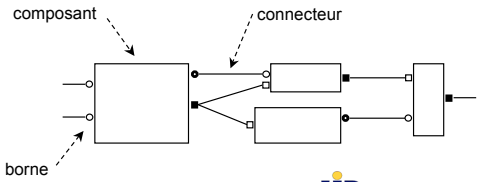


- mal déployables - peu prêts à l'emploi
  - orphelins sans leur classe
  - mal documentés (protocole d'utilisation)

Composants

Idées :

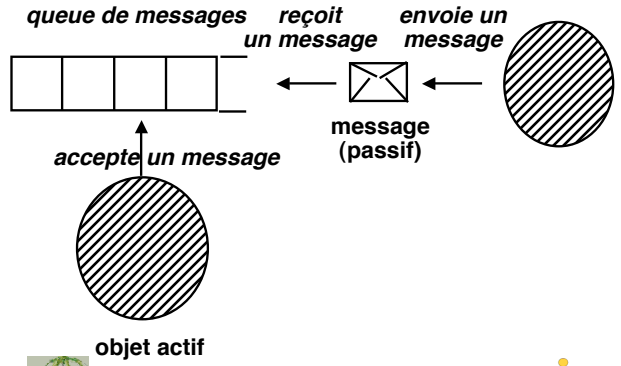
- composants
  - plus «gros»
  - plus autonomes et encapsulés
  - unités de déploiement
  - paramétrables
  - symétrie retrouvée : interfaces d'entrées mais aussi de sorties
  - plusieurs interfaces (de sortie et d'entrées) : notions de "bornes"
- réification des relations/connexions entre composants
  - références hors des objets -> couplage externe (mais reste explicite)
  - notion de connecteur



Objets actifs / acteurs - découplage temporel

Désynchronisation (découplage) entre :

- émission
- réception
- traitement

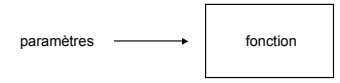


Axe 3 - Abstraction

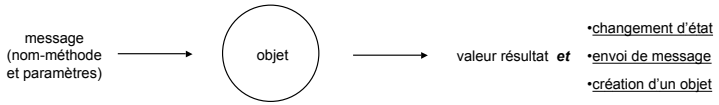
- au niveau de l'entité
  - agent non purement procédural
    - connaissances
      - ex : états mentaux, plans, règles d'inférence des agents cognitifs
  - pro-activité
    - pas uniquement purement réactif
- au niveau d'un ensemble d'agents
  - différents modes de communication
    - via l'environnement, ex : colonies de fourmis
    - messages typés, ex : KQML (inform, request, reply...), protocoles (ex : ACL)
  - 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 toujours explicitement dans l'implantation

Différences entre Objets et Agents

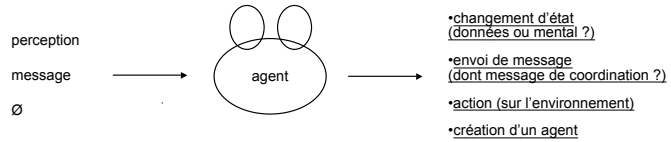
- fonction (pure)



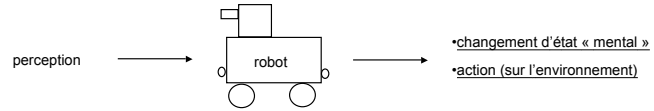
- objet



- agent

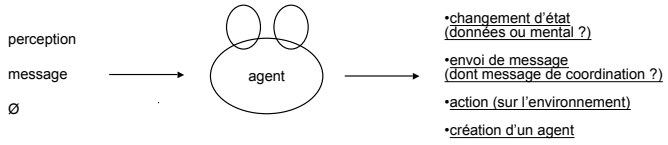


- robot



Construire des agents

- Problème essentiel 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)



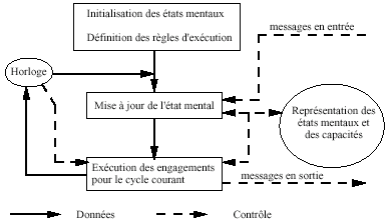
- On appelle communément architecture d'un agent la structure logicielle qui réalise cette sélection

Construction des agents

- Comment programmer cette architecture ?
  - dans un langage spécifique
    - » ex : Agent0, April, CLAIM
    - » 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-men



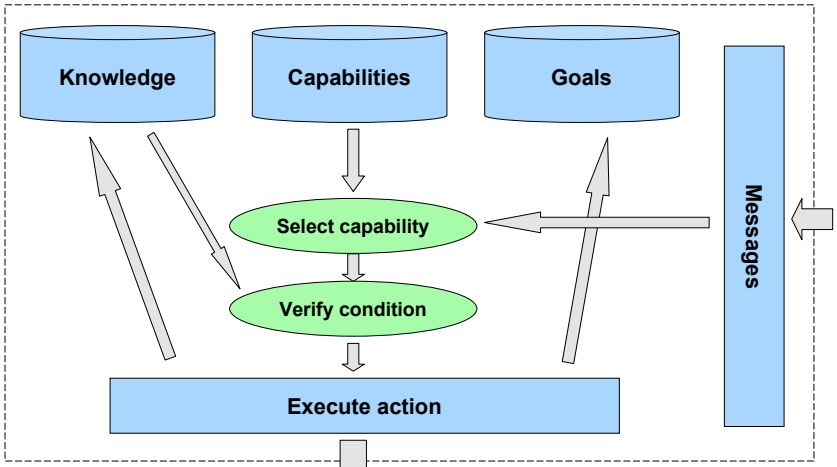
Langage de programmation multi-agent CLAIM

- CLAIM [El Fallah & Suna 03]

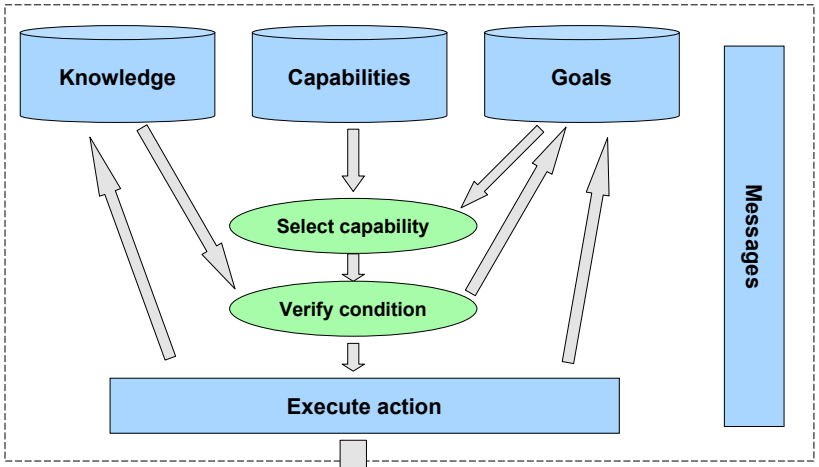
combine :

- Programmation
  - modèle de calcul concurrent
  - sémantique opérationnelle
  - mobilité
  - modèle de distribution/mobilité inspiré des Ambients [Cardelli]
- Comportement/Raisonnement
  - réactif - chaînage avant - orienté message
    - » capacités activées à la réception d'un message
  - proactif - chaînage arrière - orienté but
    - » tente de résoudre un but
  - Les deux comportements/modes sont activés simultanément
    - » un thread Java pour le comportement réactif
    - » un thread Java pour le comportement proactif
    - » un thread Java pour activer les processus CLAIM
- Plateforme d'exécution SymPa
  - opérationnelle
  - distribuée
  - compatible avec le standard OMG MASIF

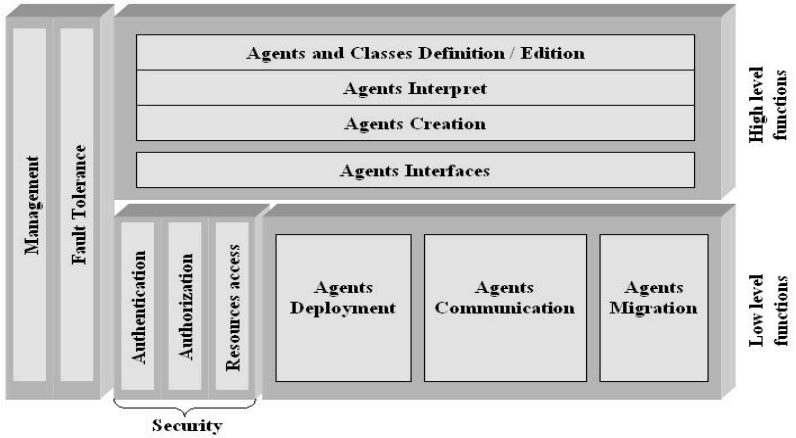
Comportement réactif



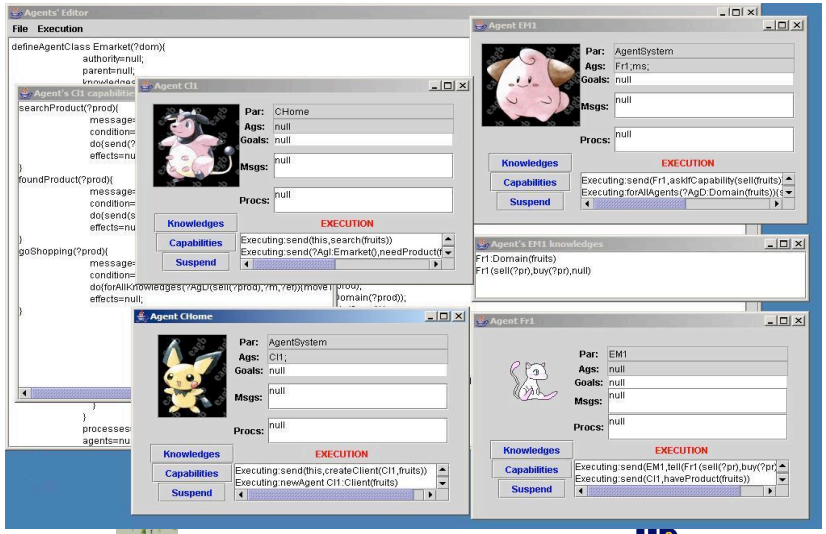
Comportement proactif



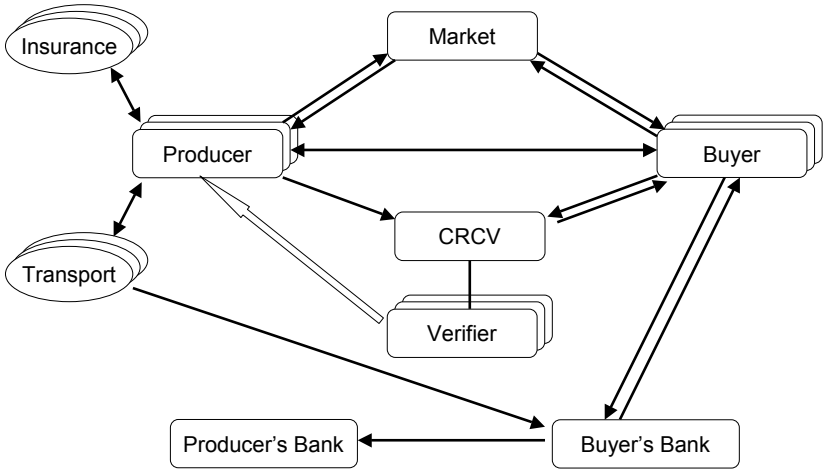
Mécanismes offerts par SyMPA

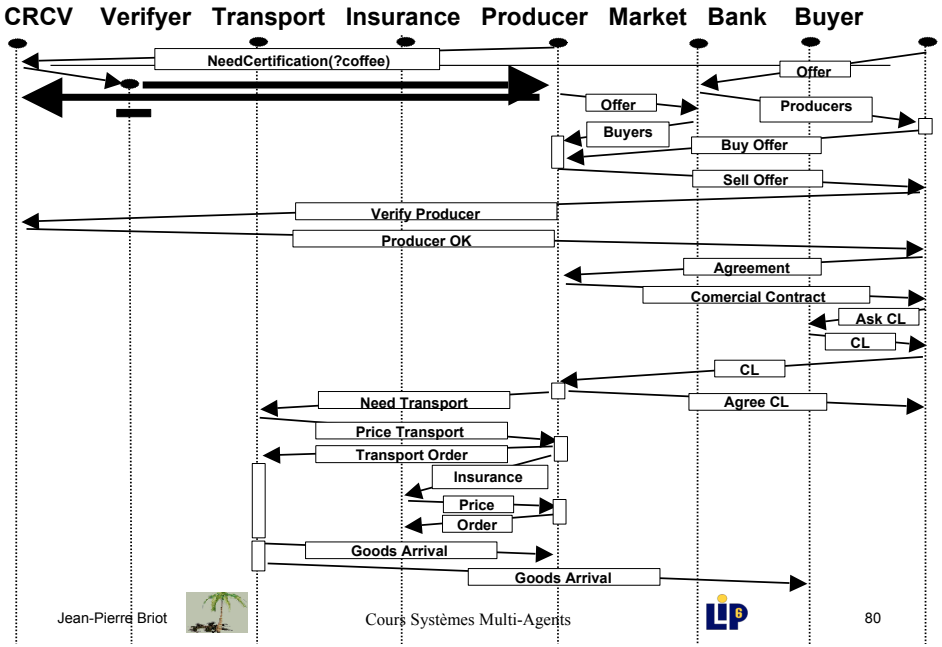


Environnement de développement



Exemple : Marché électronique du café à Veracruz (Projet LAFMI)





Marché e- du café (1/4)

```
Agent CRCV {
   ...
   knowledge={
      Producer(Prod1,oro,7);
   }
   messages={Verifyer>tell(Producer(Prod2,oro,6));}
   ...
}
```

Message provenant d'un des vérificateurs

▼ *tell -> Connaissance ajoutée dans la base*

```
Agent CRCV {
   ...
   knowledge={
      Producer(Prod1,oro,7);
      Producer(Prod2,oro,6);
   }
   messages=null;
   ...
}
```


Marché e- du café (2/4)

```
Agent Prod1:CoffeeProducer { ...
   capabilities{
      haveCoffee(?type){
         message=haveCoffee(?type);
         condition=null;
         do{send(CRCV,needCertification(?type))}
         effects=null;
      } ... }
   processes={send(this,haveCoffee(oro));} ...
}
```

Premier message de démarrage

```
Agent CRCV { ...
   capabilities{
      certifyProducer(?type){
         message=needCertification(?type);
         condition=null;
         do{newAgent Ver:Verifyer().
            send(Ver,verify(sender,?type))}
         effects=null;
      } ... }
}
```

pré-condition (nulle)

partie conséquence de la règle

Marché e- du café (3/4)

```
AgentClass Verifyer() {
   ...
   capabilities{
      Verify(){
         message=verify(?AgP,?type);
         condition=null;
         do{moveTo(this,?AgP).
            ?q=Java(Verificator.verifyQuality(?type)).
            send(?AgP,quality(?type,?q)).
            moveTo(this,authority).
            send(authority,tell(Producer(?AgP,?type,?q))).
            acid
         }
         effects=null;
      }
   }
   ...
}
```

Mobilité sur le site du producteur

Retour au bureau

Absorption par l'autorité (CRCV)

Transmission de l'évaluation

Marché e- du café (4/4)

```
Agent Market() {
   ...
   capabilities{
     demandFromBuyer(){
       message=buyOffer(?type,?qual,?quant);
       condition=null;
       do{send(this,tell(Buyer(sender,?type,?qual,?quant))).
          forAllKnowledges(Producer(?prod,?type,?qual,?pr)){
            send(sender,tell(Producer(?prod,?type,?qual,?pr)))
          }.send(sender,demandAnswered(?type,?qual,?quant))
       }
       effects=null;
     }
     ...
   }
   ...
}
```

Pour intégrer cette connaissance

Recherche tous les producteurs correspondants
Les envoie un par un à l'acheteur
Envoi confirmation fin (des « tell ») à l'acheteur

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 :
  - 1) les organisations d'agents (mais couplage encore trop statique par rapport aux agents cf. dual : des composants aux agents)
  - 2) et également surtout les architectures d'agents (au niveau d'un agent : « programming in the small »)

1) Architectures logicielles et organisations (d'agents)

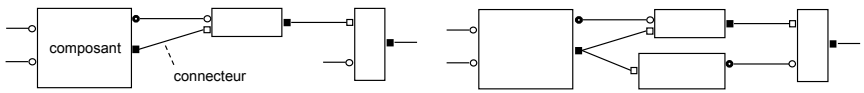
- Architectures logicielles
  - explicites
  - rationnelles
  - couplage explicite
    - au niveau des données (interfaces, typage)
    - et des modes d'interactions (connecteurs)
- Organisations d'agents (cognitifs)
  - explicites
  - rationnelles
  - couplage sémantique
  - réifiées
  - vers des organisations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

Architectures logicielles

- Programmation à grande échelle
- Configuration et reconfiguration d'applications modulaires/réparties

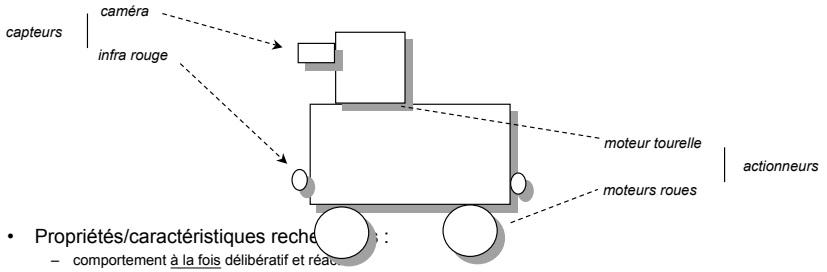
- Composants
  - pipes & filters, ex : Unix Shell `dvips | lpr`
  - couches, ex : Xinu, protocoles réseaux
  - événements (publish/subscribe), ex : Java Beans
  - frameworks, ex : Smalltalk MVC
  - repositories, ex : Linda, blackboards

- Connecteurs
  - appels de procédure, messages, diffusion d'événements, pipes&filters...



2) Architecture logicielle d'UN agent

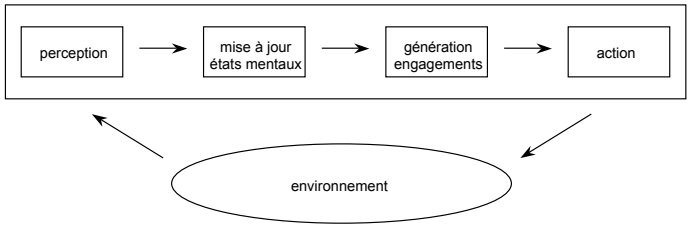
- Ici, architecture = organisation individuelle d'un agent (vision récursive)
- Exemple d'application [Shaw et Garlan 96] :
  - (architecture de contrôle d'un) robot mobile autonome



- 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)

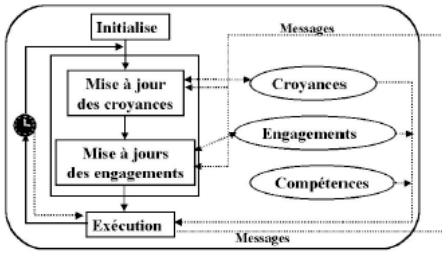
Architectures modulaires horizontales

- (une seule couche)
- cycle de calcul



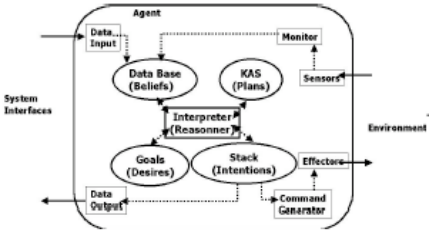
Architectures modulaires horizontales (2)

- **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]

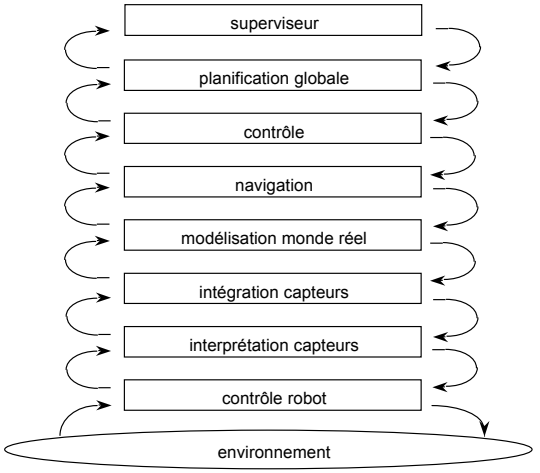
plutôt dirigée par les buts (goal-driven)

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

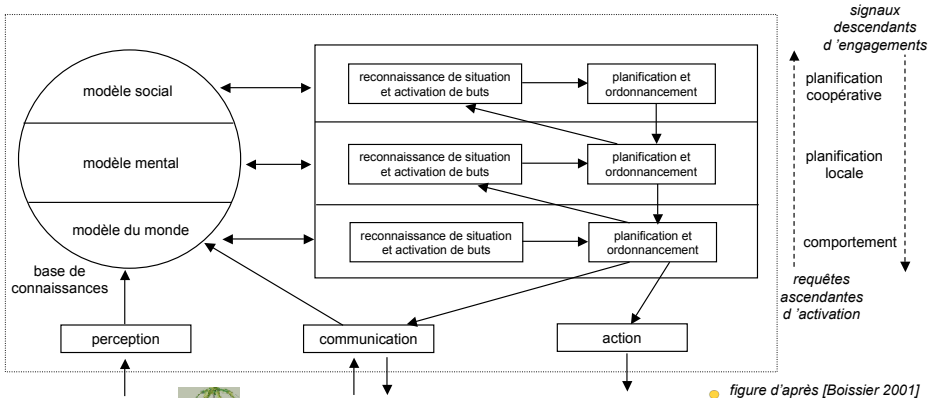


figure d'après [Boissier 2001]

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)

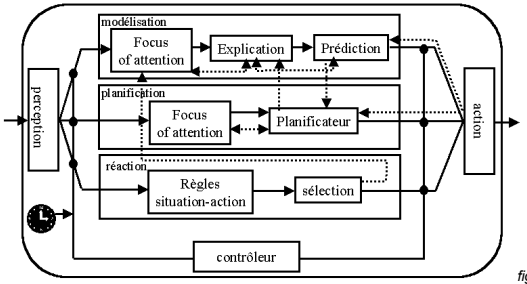
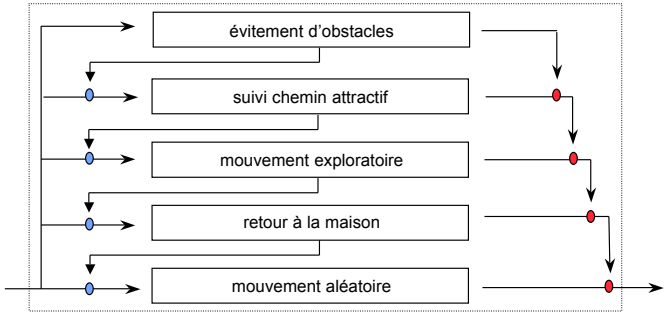


figure d'après [Boissier 2001]

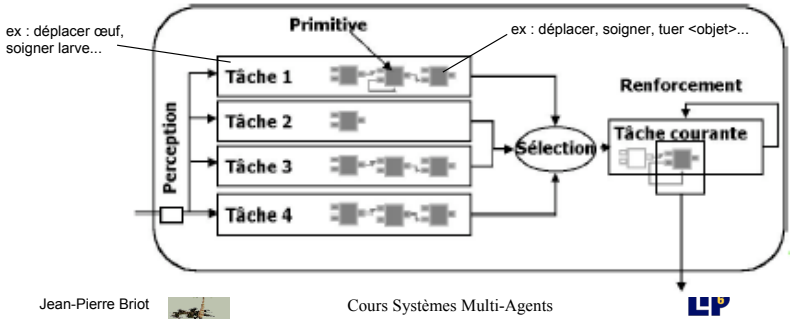
Architectures réactives en couches (verticales)

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



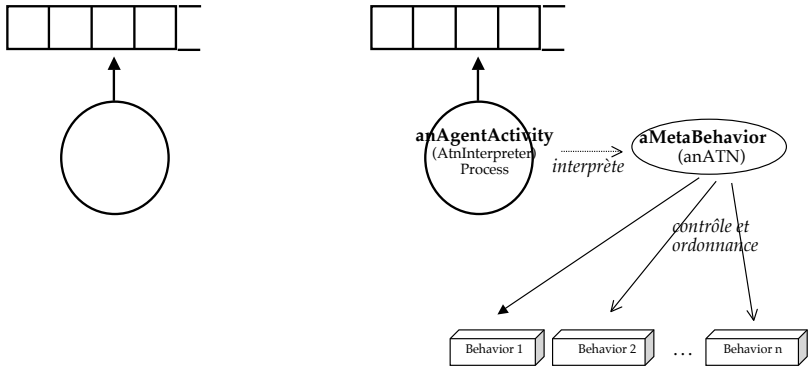
Architectures réactives en couches (2)

- 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
  - niveau d'activation le plus élevé
  - décrémentation du niveau de la tâche active



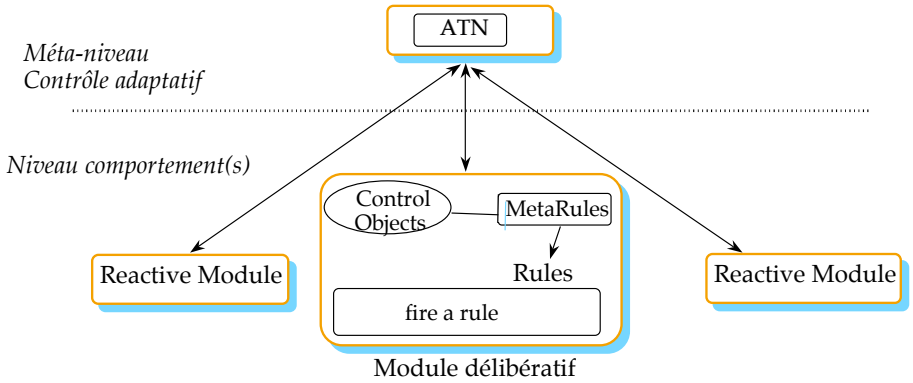
D'un acteur à un agent

- « From Actors to Agents » [Guessoum & Briot, IEEE Concurrency, 1999]
- du framework d'acteurs Actalk à l'architecture d'agent DIMA
- d'un acteur mono-comportement, vers un agent à méta-comportement ordonnanceur/contrôleur de plusieurs comportements



Architecture de DIMA

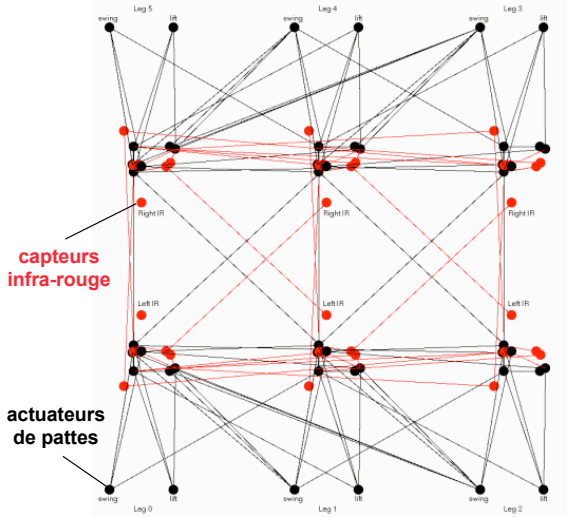
ATN = Augmented Transition Network (automate)



Architectures d'agents

- hiérarchiques (cognitives/réactives)
  - ex : DIMA [Guessoum 96], InterRap [Müller 96]...
- componentielles
  - ex : Maleva [Lhuillier 98] [Meurisse 2000]
  - SCD [Yoo 98]
  - Vulcano [Ricordel 2002]
  - MAST [Vercouter 2003]
- composition d'actions
  - ex : Bene theory [Steels 94]
- connexionistes
- évolutionistes
  - algorithmes génétiques, morphogenèse

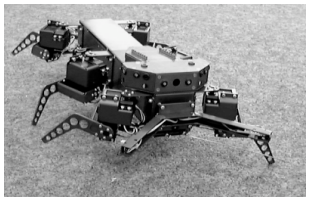
Evolution de l'architecture de contrôle d'un robot marcheur [Meyer et al. 98]



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

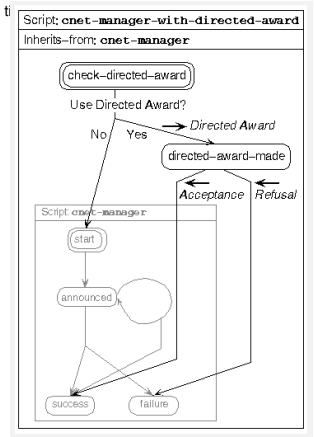
Résultat :

Réseau noir : marche
Réseau rouge : évitement d'obstacle

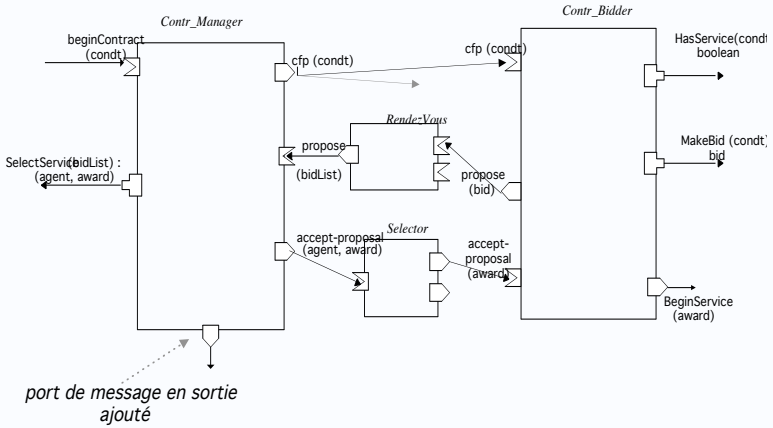


Réutilisabilité des composants (quelques résultats)

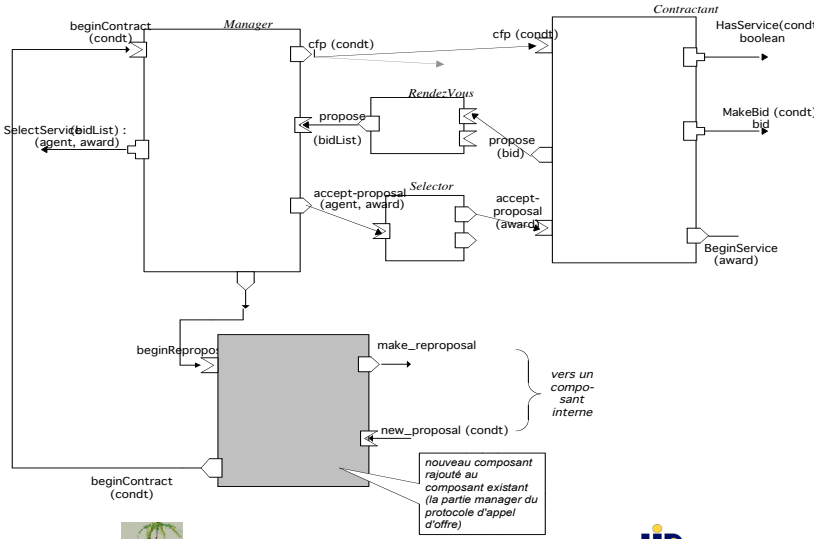
- Conception incrémentale de protocoles de coopération - à partir du Contract Net
- [Yoo 1998]
  - par héritage
    - » ex : réalisation du protocole d'appel d'offre avec délai de temps : ti
  - par composition
    - » ex : extension en un FIPA-Iterated Contract Net
  - Expérimentations de réutilisation analogues avec AgentTalk (langage de coopération) par héritage [Kuwabara et al. 95]



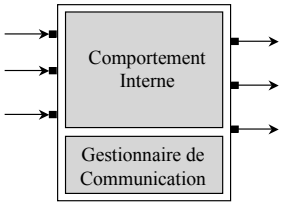
Ex : Iterated Contract Net Protocol (1/2)

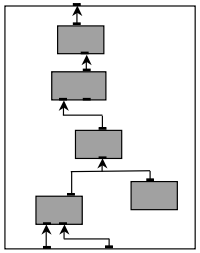


Iterated Contract Net Protocol (2/2)



Composants d'agents Maleva [Lhuillier 98] [Meurisse et Briot, TSI 2001]

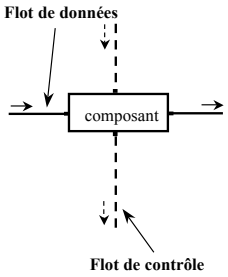




- Un Composant est défini par :
  - un Comportement Interne
  - des Bornes de Communication
- **Pas de référence explicite entre composants**
  - permet la modification du graphe des connexions indépendamment des composants.
  - entités *potentiellement* réutilisables car définition indépendante de l'environnement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

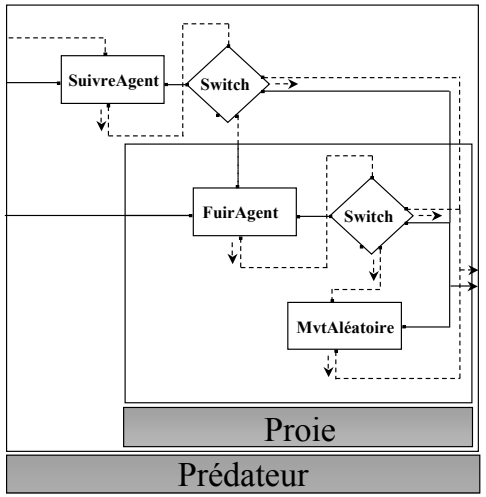
Maleva (2)

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.

Ex : Proies et prédateurs



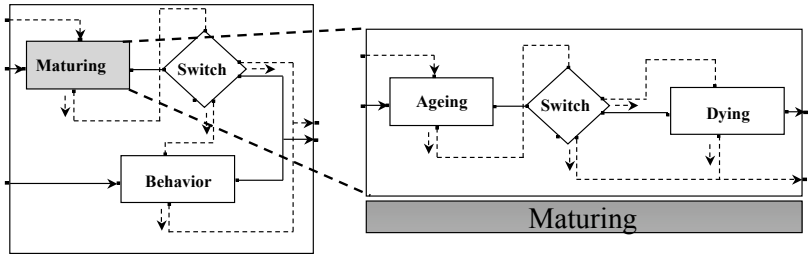
Exemples de Conception - Approche Descendante
Les fourmis [Guillemet et al. JFSMA'98]

- 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

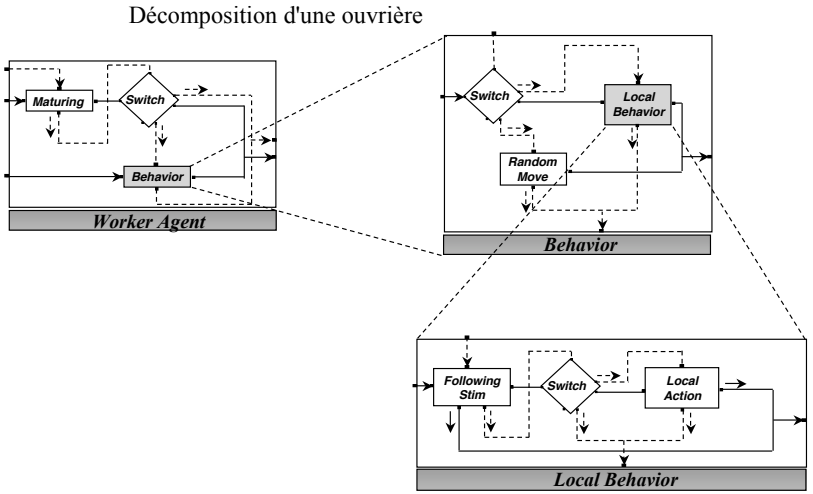
-> ***pattern*** !

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)*

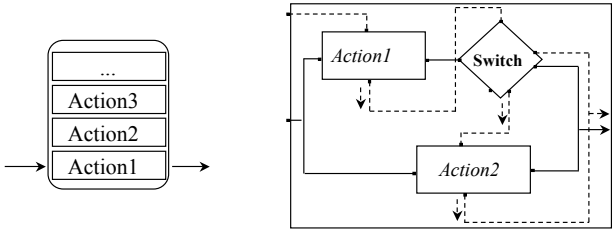


Maleva - Design Patterns

- 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)

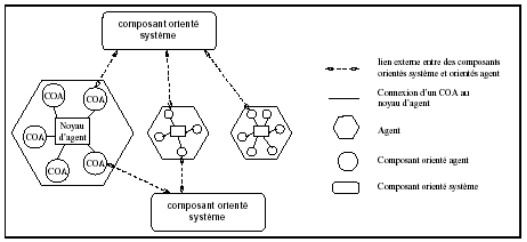
- décomposition du flot de contrôle des langages impératifs : *if...then...else*
- architecture de Subsomption de Brooks [Brooks91]



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

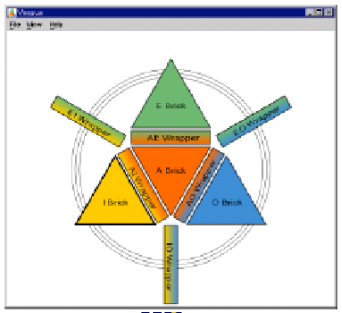
Autres modèles componentiels d'agents

- MAST [Vercouter 03]



- provided roles
- required roles
- sent events
- handled events
- priority

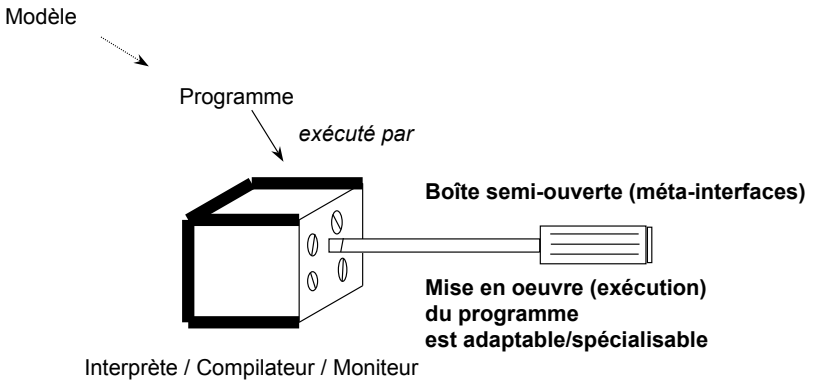
- Vulcano [Ricordel 02]
  - Décomposition selon les points de vue :
  - A(gent), E(nvironnement), I(nteraction), O(rganisation)
  - approche Voyelles [Demazeau 01]
- …



Expérience

- Réutilisabilité de composants pas très facile en pratique
- C'est l'architecture qui elle est bien réutilisée (stabilité)
- Cela rejoint les principes de Ralph Johnson :
  - Réutilisation du framework (l'architecture)
  - N'est réutilisable que ce qui a déjà été réutilisé
    - » (le framework est défini a posteriori, par factorisation de la partie stable
    - » d'un ensemble d'applications)
- Cf. bilan plus loin

Le problème du contrôle de l'exécution



Open Implementation [Kiczales 94]

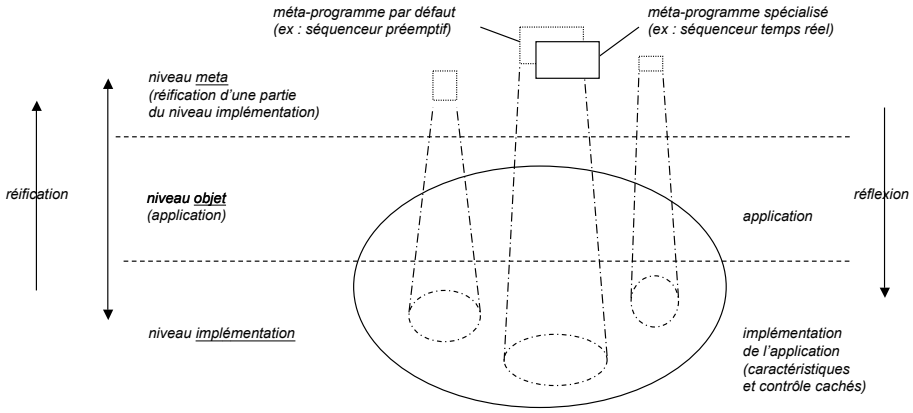
L'approche des architectures réflexives

- Diverses caractéristiques de représentation (statique) et d'exécution (dynamique) des programmes sont rendues concrètes (*réifiées*) sous la forme de *méta-programmes*
  - Habituellement elles sont invisibles et immuables (interprète, compilateur, moniteur d'exécution...)
- La spécialisation de ces méta-programmes permet de *particulariser* (éventuellement dynamiquement) l'exécution d'un programme
  - » représentation mémoire
  - » modèle de calcul
  - » contrôle de concurrence
  - » séquencement
  - » gestion des ressources
  - » protocoles (ex : résistance aux pannes)

  avec le minimum d'impact sur le programme lui-même

Réification/réflexion (2)

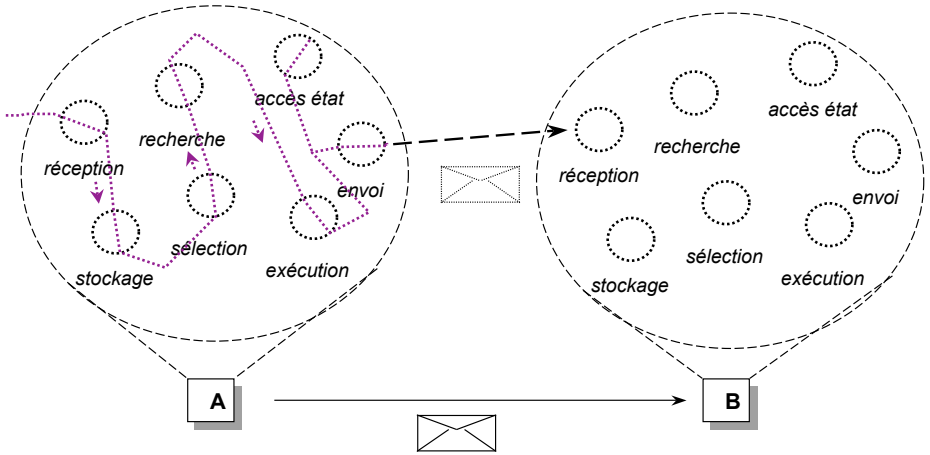
- Réification logicielle (méta-programmes)
  - Ex : algorithme de séquencement (scheduler)



Méta-composants

- *CodA* [McAffer ECOOP'95] est un exemple de modèle relativement général d'architecture réflexive
- Sept méta-objets/composants de base :
  - envoi de message
  - réception de messages
  - stockage des messages reçus
  - sélection du premier message à traiter
  - recherche de méthode correspondant au message
  - exécution de la méthode
  - accès à l'état de l'objet
- Les méta-composants sont :
  - spécialisables
  - (relativement) combinables

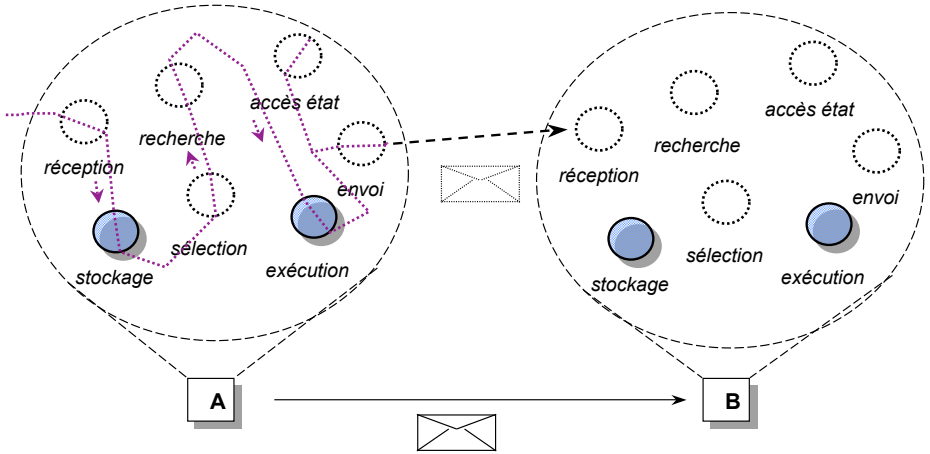
CodA



Ex : Exécution concurrente

- envoi de message
- réception de messages
- **stockage des messages reçus**
  - » file d'attente (FIFO)
- sélection du premier message à traiter
- recherche de méthode correspondant au message
- **exécution de la méthode**
  - » processus associé
  - » boucle infinie de sélection et traitement du premier message
- accès à l'état de l'objet

Exécution concurrente (2)

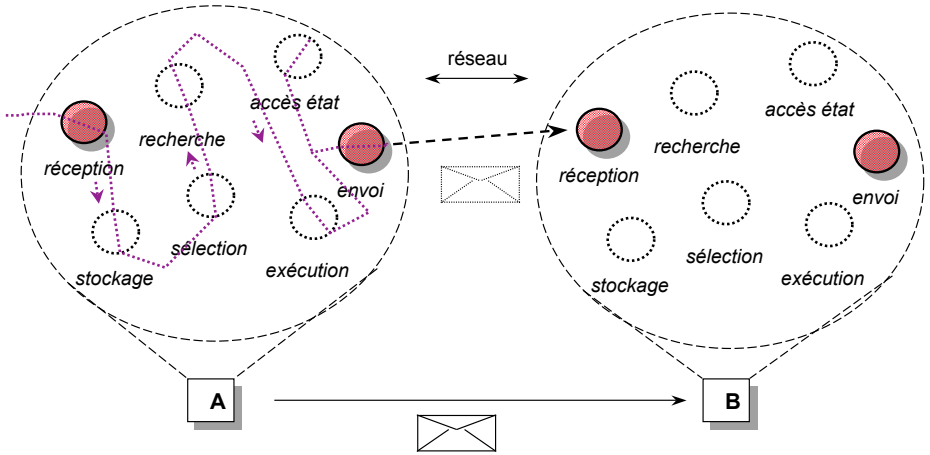


Ex : Exécution répartie

- **envoi de message**
  - » encodage des messages, envoi via le réseau
- **réception de messages**
  - » réception via le réseau, décodage des messages
- stockage des messages reçus
- sélection du premier message à traiter
- recherche de méthode correspondant au message
- exécution de la méthode
- accès à l'état de l'objet

- **encodage**
  - » discipline d'encodage (marshal/unmarshal)
- **référence distante**
- **espace mémoire**

Exécution répartie (2)



Application aux agents et multi-agents

- Contrôle du calcul
- Contrôle du déploiement
- Contrôle du comportement (ex : méta-comportement dans DIMA)
- Contrôle du raisonnement (ex : méta-règles)
- Contrôle de la réplication (ex : projet DarX)
- …

Robustesse (tolérance aux fautes) des SMAs

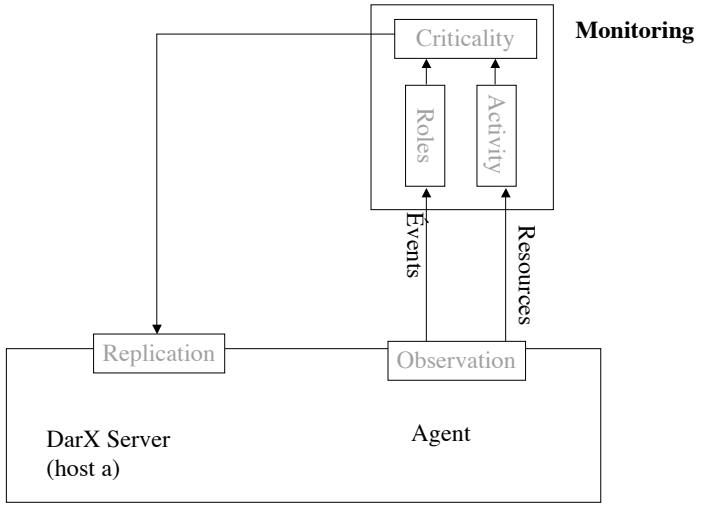
- Les systèmes multi-agent sont distribués par nature
  - mais encore rarement à large échelle
- La distribution (répartition) à large échelle implique la possibilité de pannes partielles
- « A distributed system is one on which I cannot get any work done, because a machine I have never heard of has crashed. » [Leslie Lamport]
- Ex :
  - Gestion de crises
  - Travail collaboratif
  - Supervision
  - …

Projet DarX [Briot, Guessoum, Sens et al. 02]

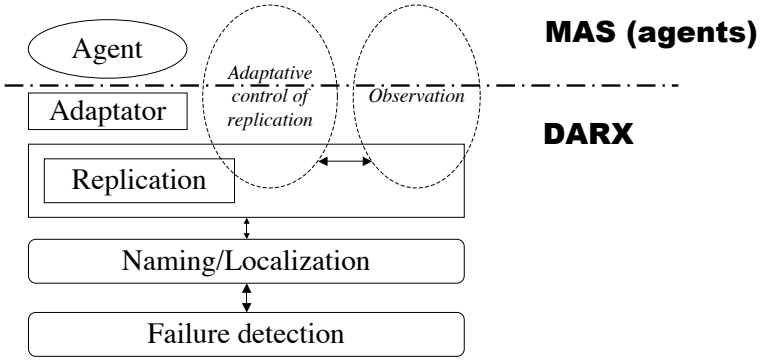
- Appliquer des protocoles de réplication
  - **dynamiquement**
  - et **automatiquement** : **où**, **quand** et **comment** c'est utile
    - où : à quels agents
    - quand : au bon moment (période de temps)
    - comment :
      - quelle politique (passif, actif…)
      - combien de répliques
      - où les créer
      - quel protocole de cohérence
      - …

- À partir de quelle information ?
  - système
    - » charge, latence, historique de pannes…
  - sémantique
    - » Rôles
    - » dépendances
    - » intentions
    - » …

Architecture d'observation/décision [Guessoum et al. 02]

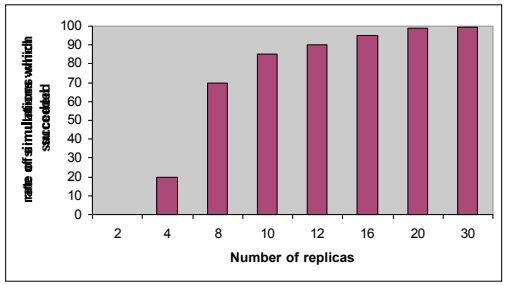


Architecture DARX



Performances : Exemple de la prise de rendez-vous par des agents assistants

- Agenda
  - 20 agents sur 8 machines
  - 100 pannes injectées (tuer aléatoirement un agent initiateur de réunions)



Limites des agents pour le génie logiciel (1/2) [Jennings 1999]

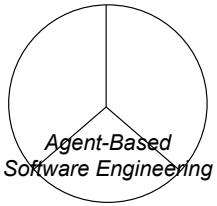
- « No magic ! »
  - Un système développé avec des agents aurait probablement pu être développé avec des technologies plus conventionnelles
  - L'approche agent peut simplifier la conception pour certaines classes de problèmes
  - Mais elle ne rend pas l'impossible possible !
- Les agents sont des logiciels (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], Tropos…
- Modélisation
  - Propositions d'extension d'UML vers les agents (ex : AUML, MAS-ML)
  - *Attention !* : UML est un ensemble de notations standardisée, et n'est *pas* une méthodologie

Génie logiciel pour les multi-agents

- Approche multi-agent pour le génie logiciel (Agent-Based Software Engineering) :
- Concevoir des applications sous forme de modules logiciels autonomes et coopératifs



Point de vue dual et complémentaire (les deux souvent réunis sous le terme Agent-Oriented Software Engineering)

- Les systèmes multi-agents nécessitent des méthodes et outils de génie logiciel adaptés (Software Engineering for Multi-Agent Systems)
- Ex : Série d'ouvrages Software Engineering for Large Scale Multi-Agent Systems (SELMAS), Springer-Verlag, 03-04

Génie logiciel pour les multi-agents

- Méthodologies (GAIA, TROPOS, Cassiopée…)
- Ingénierie des modèles (MDA)
  - MetaDima, TAO…
- Architectures
- Plates-formes
- Standards (FIPA)
- Design patterns ([Kendall 95], [Honiden 00], [Sauvage 04]…)
- Techniques avancées (Aspect-Oriented Programming [Garcia 04],…)

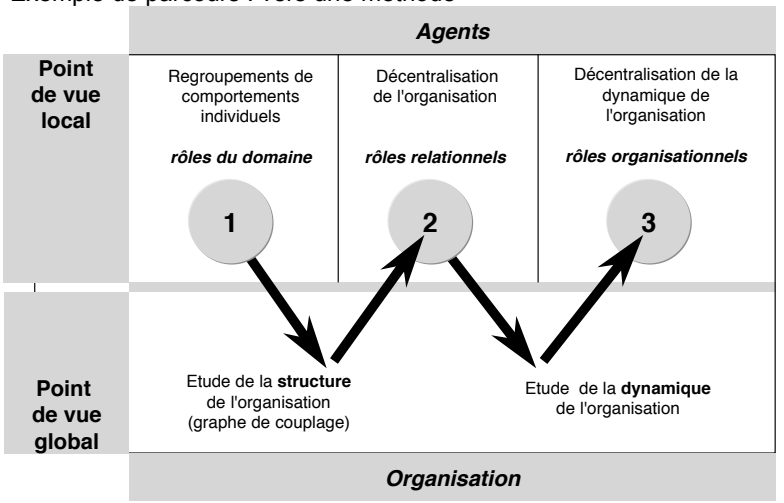
Cassiopée 1/2

- 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

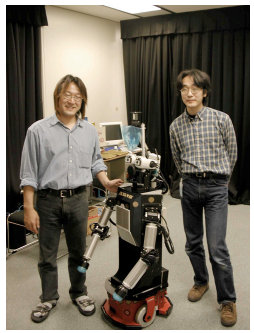


Plates-formes

- 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, ...

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
  - *ex : projet ARP (Agents Résistants aux Pannes)*
- Agents et humains
  - aborde la tension entre autonomie et collaboration
  - potentiel de médiation, mais efforts à faire vers une meilleure socialité
  - impact sur nos comportements à étudier
- Exemples d'enjeux industriels actuels
  - Services Web
  - Jeux vidéo



Ouvrages et pointeurs

- Principes et Architecture des Systèmes Multi-Agents, édité par Jean-Pierre Briot et Yves Demazeau, Collection IC2, Hermès, 2001

- 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*

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

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