Agents logiciels : diversité et typologie

Bernard ESPINASSE
Aix-Marseille Université (AMU)
LSIS UMR CNRS 7296
2012

- **Diversité, définition des agents logiciels**
- **Une typologie des agents logiciels (Nwana)**
- **Agents collaboratifs**
- **Agents interface**
- **Agents mobiles**
- **Agents réactifs**
- **Agents hybrides**
- **Systèmes multi-agents hétérogènes**

Références bibliographiques (1)

Cours :

- **Drogoul A.** Cours "Intelligence collective, systèmes multi-agents et IAD", Université Paris 6 - LIP6.
- **Gleize M.P.**, Cours "Intelligence collective", Université de Toulouse, IRIT.
- **Nwama H.,** "Sofware Agents", Knowledge Engineering Review, vol. 11, N°3, pp. 1-40, sept 96, Cambridge University Press.
- **Esfandiari B.**, "Sofware Agents" Course, University of Carleton, Canada.
- **Chaïb-Draa B.**, Cours "Systèmes multi-agents", Université Laval, Québec, Canada.
- **Finin T. & Labrou Y.**, Tutorial "Agents Communication Languages", ASA/MA.
- ...

Articles :

- ...

Références bibliographiques (2)

Livres :

- **Weiss G. - editor (00), Multiagent Systems**, MIT Press.
- **Ferber J. (95), Les systèmes multi-agents**, InterEditions.
- **Singh M. (94), Multiagent Systems,** Springer Verlag.
- **Conte R., Castelfranchi C. (1995), Cognitive and Social Action**, UCL Press.
- **Haddadi A. (95), Communication and Coopération in Agent Systems**, Springer Verlag.
- **O'Hare G.M.P. & Jennings N.R. - editors (96), Foundations of Distributed Artificial Intelligence**, Wiley-Interscience.
- **Bradsham M. - editor (97), Software Agents**, AAAI Press - The MIT Press.
- **Huhns M.N. & Singh M.P. - editors (97), Readings in Agents**, Morgan-Kaufmann.
- ...

Comme toute production de connaissance, ce cours s'inspire de nombreuses contributions existantes. Que leurs auteurs, malheureusement pas toujours cités, en soient sincèrement remerciés.

Plan

- **Diversité, définition des agents logiciels**
- **Une typologie des agents logiciels (Nwana) :**
  - **Agents collaboratifs**
  - **Agents interface**
  - **Agents mobiles**
  - **Agents réactifs**
  - **Agents hybrides**
  - **Systèmes multi-agents hétérogènes**

Agents Logiciels: Historique

Grands projets historiques :

- **MACE** (Gasser *et al.*, 1987)
- **DVMT** (Lesser & Corkill, 1981)
- **MICE** (Durfee & Montgomery, 1989)
- **MCS** (Doran *et al.*, 1990)
- the **contract network** coordination approach (Smith, 1980; Davis & Smith, 1983)
- **MAS/DAI** planning and game theories (Rosenschein & Zlotkin, 1985, 1989, 1994)
- **DRESUN** (Carver *et al.*, 1991; Carver & Lesser, 1995)
- **ARCHON** (Wittig, 1992; Jennings *et al.*, 1995), ...

Depuis, nombreux travaux sur des systèmes d'agents :

- **interaction** et **communication** entre agents
- la **décomposition** et la **distribution** de tâches
- **coordination** et **cooperation**
- **résolution de conflits** par **négociation**, ...

Diversité des agents logiciels

Plusieurs appellations liées soit à une **apparence**, soit à un **rôle** spécifique :

- **knowbots** (i.e. knowledge-based robots) : ont une connaissance experte dans un domaine particulier (personal assistants, ...)
- **softbots** (software robot)
- **taskbots** (task-based robots) : réalisent des tâches spécifiques
- **userbots**
- **robots** : habitent le monde physique
- **personal agents, personal assistants** : agents assistant un utilisateur spécifique
- **...**

Mais aussi :

search agents, report agents, presentation agents, navigation agents, role-playing agents, management agents, search and retrieval agents, domain-specific agents, development agents, analysis and design agents, testing agents, packaging agents and help agents

"in 10 years time most new IT development will be affected, and many consumer products will contain embedded agent-based systems" [Guilfoyle 95]

Qu'est ce qu'un agent logiciel (sofware agent)?

DéfinitionS :

- un système informatique, **situé** dans un environnement, et qui **agît** d'une façon **autonome** et **flexible** pour atteindre des **objectifs** pour lequel il a été conçu [Sycara & Wooldridge]
- une entité computationnelle **active**, avec une **identité persistente**, qui peut **percevoir**, **raisonner** et **agir** dans son **environnement** et qui peut **communiquer** notamment avec d'autres agents [Huhns]
- un composant **logiciel** ou **matériel** qui est capable **d'agir** afin d'accomplir des **tâches** au nom de son **utilisateur** [Nwana]

situé : capable d'agir sur son **environnement** à partir d'**entrées sensorielles** qu'il en reçoit

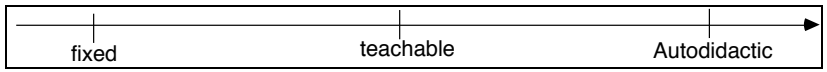
autonome : capable **d'agir** sans l'intervention d'un tiers (humain ou agent) et de **contrôler** ses propres **actions** ainsi que **son état interne** (autonomie limitée à son comportement dans une société d'agents)

flexibilité :

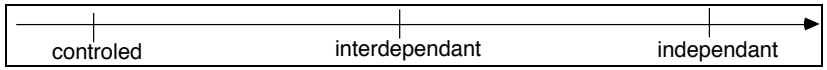
- **réactif** : capable de percevoir son environnement et de réagir à temps
- **proactif** : capable de prendre l'initiative et être oppurtuniste au bon moment
- **social** : capable d'interagir avec les autres agents quand la situation l'exige (pour réaliser ses tâches ou coopérer avec eux)

Dimensions d'un agent [Huhns]

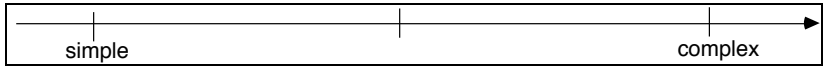
- **Dynamisme de l'aptitude d'un agents à apprendre:**



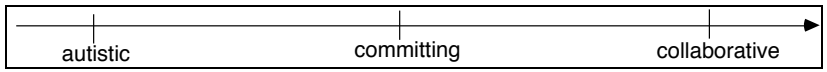
- **Autonomie:**



- **Interactions:**



- **Sociabilité (conscience):**



Systèmes d'agents logiciels

Caractéristiques générales des systèmes multi-agents (SMA):

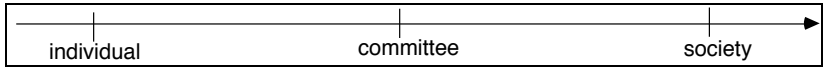
- **modularité** : pour réduire la complexité
- **vitesse**: due au parallélisme
- **fiabilité**: due à la redondance
- **flexibilité**: nouvelles tâches peuvent être gérées plus facilement grâce à la modularité

Génie logiciel "orienté multi-agents" conduit à concevoir et développer des **modules** [Huhns]:

- **actifs**
- **spécifiés de façon déclarative** (en terme de "quoi" et non de "comment")
- **contiennent des croyances sur le monde**, plus particulièrement sur eux-mêmes et les autres
- **volontaires**

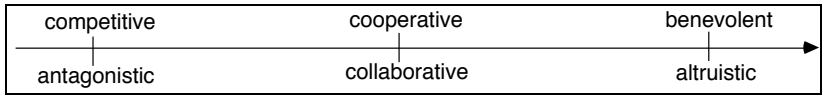
Dimensions d'un système multi-agents [Huhns]

- **Echelle dans le nombre d'agents :**

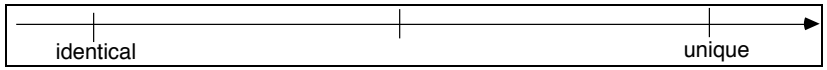


- **Interactions:**

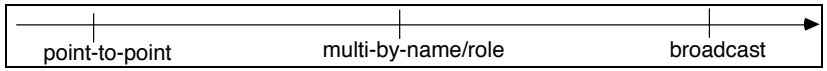
- **Coordination (propre intérêt):**



- **Hétérogeneïté d'agents:**



- **Paradigme de communication:**



Typologie d'agents [Nwana 99]

Critères de classification

Agents mobiles / statiques :

- agent **mobiles** : agents capables de se **déplacer sur un réseau**
- agents **statiques** : agents **pas capables de se déplacer sur un réseau**

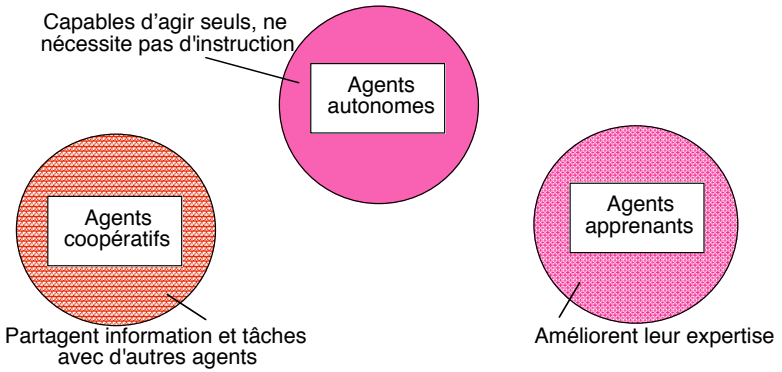
Agents collaboratifs / réactifs :

- agents **collaboratifs** :
  - possèdant un **modèle de raisonnement symbolique interne**,
  - se **coordonnent** entre eux par **planification** ou **négociation**
- agents **réactifs** :
  - ne possèdent **pas de modèle de raisonnement symbolique interne**,
  - agissent par **stimulis - réponse**
  - ont un **comportement répondant à l'état courant de l'environnement** dans lequel ils sont immergés

Typologie d'agents [Nwana 99]

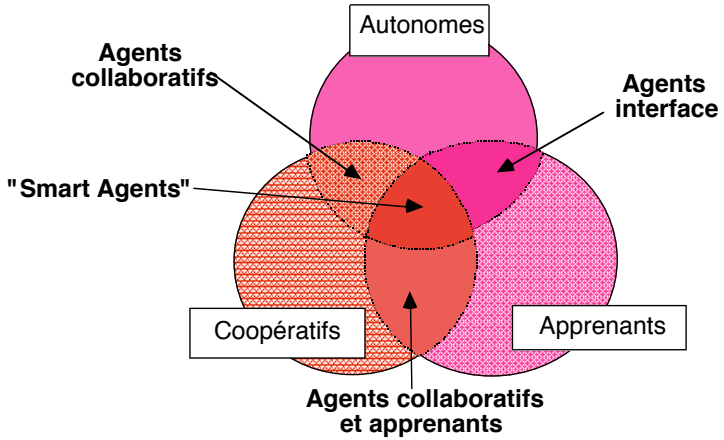
Attributs majeurs des agents :

- **autonomie** : agents **autonomes**
- **coopération** : agents **coopératifs**
- **apprentissage** : agents **apprenants**



Typologie d'agents [Nwana 99]

Combinaison des 3 attributs majeurs : 6 types d'agents



Attributs secondaires d'un agent

- la **véracité** : il ne communique pas de mauvaises informations sans le savoir
- le **bénévolat** : il n'a pas de buts incompatibles et essaiera de faire ce qu'on attend de lui
- la **rationnalité** : il agira de sorte à atteindre ses objectifs, au moins dans la limite de ses convictions
- la **versatilité** : il poursuit plusieurs buts et il s'engage dans de plusieurs tâches
- la **persistence** : il est temporellement continu ou il se dégrade au fil du temps
- les **attitudes émotionnelles** : en a-t-il assez d'être sollicité ? quel rôle a l'émotion dans l'élaboration d'agents (Bates, 1994)?
- les **attitudes mentales** : a-t-il des attitudes mentales comme des croyances, des convictions, des désirs et des intentions (BDI agents - Rao & Georgeff 95)

Agents collaboratifs

- **autonomes** et **coopérent** entre eux
- **négocient** avec les autres agents en vue d'atteindre des ententes lors de la résolution distribuée de problèmes
- possèdent des **capacités d'apprentissage limitées**

Finalités :

- résoudre des **problèmes :**
  - **"naturellement" distribués** (réseaux de capteurs distribués (DVMT - Durfee et al., 1987), control aérien, ...)
  - **"trop grands"** ou **"trop complexes"** pour un seul agent
  - associés à des **sources d'information** ou **d'expertise distribuées**
- permettre **interconnectivité** et **interopérabilité** de **systèmes existants** (legacy systems)
- accroître :
  - la **modularité** (et ainsi réduire la complexité),
  - la **vitesse** (par parallélisme),
  - la **fiabilité** (par redondance),
  - la **flexibilité** (à partir d'une organisation plus modulaire) et
  - la **réutilisabilité** au niveau de la connaissance (donc partageabilité des ressources)

Agents collaboratifs :systèmes, théories et modèles

Quelques systèmes connus :

- **PLEIADE** : [Sycara et al. 1995]
- **DVMT** : [Durfee et al. 1987]
- **MACE** : [Gasser et al. 1987]
- **ADEPT** : [O'Brien et al. 1996]
- **GRATE/GRATE*** : [Jennings 1993]
- **ARCHON** project [Wittig, 1992; Jennings *et. al.*, 1993], …

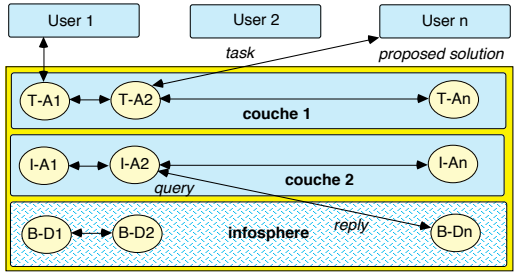
Théories et modèles :

- **Théorie raisonnable de l'intention de Cohen & Levesque** (90)
- **Architecture BDI** (Rao & Georgeff 92) : caractérisation d'un agent rationnel en terme d'attitudes mentales de Croyances, Désirs et Intentions (Beliefs, Desires and Intentions) - usage de **logiques épistémiques**
- **AOP (Agent Oriented programming - Shoham 93)** : un état mental d'un agent est décrit par ses croyances, décisions, capacités et obligations - langage basé sur des opérateurs épistémiques et modaux.
- **Autres contributions** sur les attitudes mentales : IRMA [Bratman et al. 88], GRATE/GRATE* [Jennings 93], ...

Exemple : le système Pleiades [Sycara 1995]

Objet du système : agents collaboratifs participant à une prise de décision

- le système est utilisé pour développer le "CMU's *Visitor Hosting System* "
- agents communiquent (KQML) et coopèrent pour créer et gérer l'horaire d'un visiteur

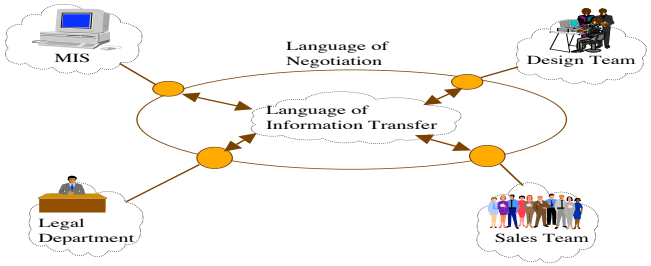


- couche 1: **task-specific collaborative agents** (T-A) : réalisent des tâches pour un utilisateur, e.g. arranger un rendez-vous, en collaboration avec d'autres T-A (coordination et plannification), ils ont recours à des I-A
- couche 2 : **information-specific collaborative agents** (I-A) : assistants informationnels qui collaborent pour fournir des réponses aux questions posées par des T-A en interrogeant l'"Infosphere" (collection de ressources hétérogènes internet)

Le projet ADEPT de BT (British Telecom)

Consortium : BT Laboratories, ICI Engineering Technology, Loughborough University of Technology, Queen Mary & Westfield College (3 ans - 33 années hommes)

Objectif : aide à la communication entre les services de l'entreprise dans l'échange d'information lors d'un processus d'affaire



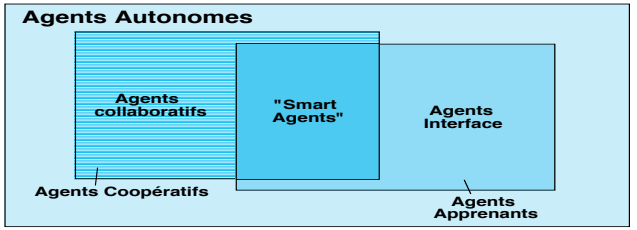
- **processus d'affaire** perçu comme un **ensemble d'agents** offrant des services et négociants,
- **négociation** :
  - associée à un **coût**, un **délai** et un **degré de qualité**
  - conduit à un **engagement des agents**

Les agents collaboratifs : principaux défits [Nwana]

- **disposer d'une véritable ingénierie** :
  nécessité de méthodes et d'outils de conception et d'implémentation de systèmes à base d'agents collaboratifs [Smith 96b]
- **importance de la coordination entre agents** :
  nécessité de disposer de véritables théorie de la coordination évitant anarchie et blocage (travaux formels et experimentaux encore indispensables)
- importance de la **stabilité**, la "**scalability**" et de la **performance** :
  demande encore beaucoup de travaux de recherche
- **systèmes existants** (legacy systems):
  besoin de définir des techniques et methodologies pour intégrer agents et systèmes existants
- **apprentissage** :
  comment ces systèmes apprennent ? l'apprentissage ne conduit-il pas à l'instabilité ? quelles sont les architectures appropriées pour les différents types de problèmes? est-on sûr que les agents ne passent pas plus de temps à apprendre qu'à réaliser ce pour quoi ils sont fait?
- **évaluation des systèmes d'agents collaboratifs** :
  pas encore abordé, comment vérifier et valider leurs spécifications fonctionnelles ?

Agents interface

- accent sur **l'autonomie**, **l'apprentissage** dans l'exécution de tâches pour des utilisateurs
- fournissent une assistance **proactive** à l'utilisateur pour une application précise
- leurs interactions avec les autres agents sont en général limitées



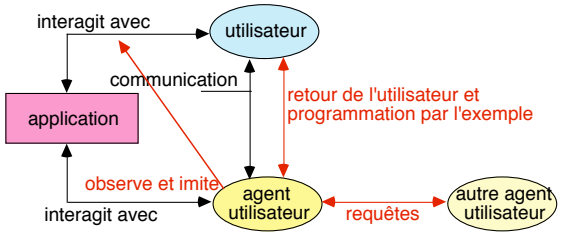
Agents interfaces = assistant personnel collaborant avec **l'utilisateur**

Finalités :

- ils apportent une **assistance à un utilisateur** : par ex. dans l'apprentissange de l'usage d'une application particulière (un tableur, un OS (Unix), ...)
- ils **déchargent l'utilisateur** et le développeur de tâches laborieuses
- ils **peuvent s'adapter** aux habitudes de l'utilisateur et à ses préférences
- ils peuvent aider plusieurs **utilisateurs à se partager leur savoir-faire**

Fonctionnement des agents interfaces [Maes 94]

- il **observe** et **surveille** les actions de l'utilisateur dans l'interface (regarde par dessus son épaule), apprend de nouveaux raccourcis et suggère de meilleurs façon de réaliser une tâche
- il agit comme un assistant personnel autonome qui **coopére** avec l'utilisateur en accomplisant certaines tâches dans l'application



- il **apprend** a mieux assister un utilisateur de différentes façons :
  - en **observant** et en **imitant** l'utilisateur
  - en recevant un **retour** positif ou négatif de l'utilisateur
  - en recevant des **instructions explicites** de l'utilisateur
  - en **demandant des conseils** à d'autres agents

Applications des agents interfaces [Maes 94]

- **assistants** [Kozierok & Maes 93] :
  - l'agent **Calendar** assiste son utilisateur dans la planification de réunions impliquant acceptations, rejets, ordonnancement, négociation et réordonnacement des horaires
  - il apprend ses préférences et ses engagements (elle n'aime pas les réunions le mercredi, il les préfère le matin, ...)
- **guides** [Liebermann 95] :
  - l'agent **Letizia** apporte une assistance dans la recherche sur le Web en conduisant, en coopération avec l'utilisateur, une recherche en largeur d'abord concurente pour des sites pouvant intéresser celui-ci
  - il découvre ses intentions à partir de son comportement
- **aide-mémoire** [Rhodes & Starner 96] :
  - l'agent **Remembrance**, associé à l'éditeur Emacs,
  - assiste un utilisateur dans la rédaction de documents (e-mail, articles, rapports, ...) en pouvant lui ramener sur un mot clé donné, les documents les plus pertinents que l'utilisateur a déjà dans la mémoire de son ordinateur

Applications des agents interfaces (suite) [Maes 94]

- **filtrage** [Sheth & Maes 93]:
  - l'agent de filtrage de nouvelles **NewT** aide l'utilisateur à filtrer et sélectionner des nouvelles dans un flux continu de nouvelles
  - il s'agit de créer des agents spécialisés (sports, finance, ...) et de les entrainer par renforcement en exploitant le contenu, les mots clés, l'auteur et la source (algo. génétiques)
- **assistant d'achat/vente** [Chavez & Maes 96] :
  - **Kasbah** represente une place de marché (un site web) où des agents agissent pour leur propriétaires,
  - peuvent filtrer la annonces et retenir celles qui intéressent leur propriétaires et négicier, acheter et vendre des articles
- **...**

Les agents interface : défis majeurs [Nwana]

- **Efficacité** :

  démontrer que la connaissance apprise avec des agents interface peu vraiment réduire la charge de travail d'utilisateur
- **Techniques d'apprentissage** :

  déterminer **quelles** sont les techniques d'apprentissage qui sont préférables, pour **quels** domaines, et **pourquoi**
- **Garantir la «vie privée»** des utilisateurs
- **Capacité des agents** :

  étendre les agents interface afin qu'ils soient capables de **négocier** avec d'autres agents
- **Domaines d'application** :

  étendre si possible le champ d'application à d'autres domaines innovants comme les loisirs et la maison

Agents Mobiles - Agents pour Internet

l'agent autonome possédant une **identité** capable de **voyager sur un réseau**

agent mobile = unité logicielle capable de :

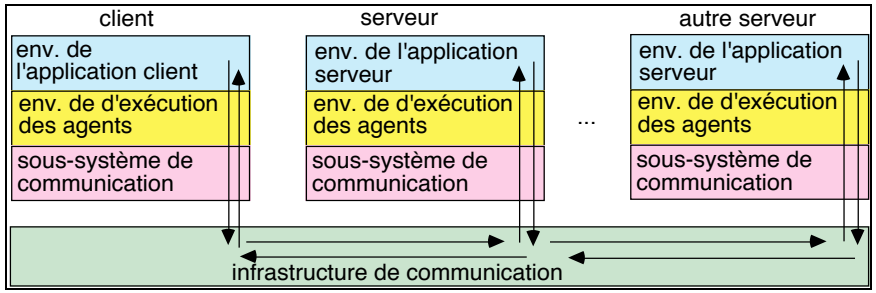
- **errer dans des réseaux** (locaux ou globaux comme le WWW)
- **migrer** d'une machine à une autre
- **interagir** avec différentes machines
- **collecter de l'information** pour son propriétaire
- **revenir "à la maison"** en ayant fait le travail fixé par son propriétaire

Finalités :

- **réduction de la charge du réseau :**
  - ils consomment moins de ressources du réseau en déplaçant le calcul vers les données au lieu de l'inverse
  - ils ne nécessitent pas une connexion continue entre machines
- interaction asynchrone : ils peuvent remplacer des communications (ex : documents mobiles)
- paradigme de programmation intéressant : ils cachent les cannaux de communication mais pas la localisation de l'exécution et facilitent ainsi le déploiement d'application réparties

Fonctionnement des agents mobiles

revient à une migration de process déplacant l'agent d'un client qui émet une requête (pour information, transaction ou mail) vers un serveur capable de satisfaire la requête :

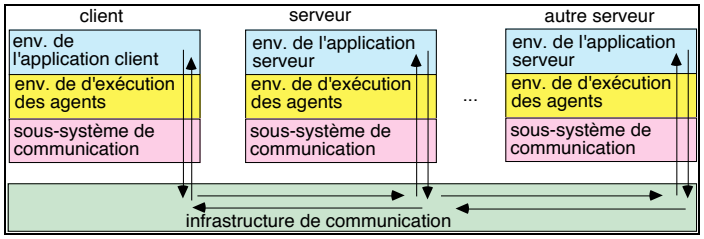


1- l'application passe des informations à l'environnement d'exécution des **agents** via **l'API**

2 - initialisation d'une instance d'agent (process, thread)

3 - l'agent **exécute une instruction créant un nouveau process fils** s'exécutant dans un message exprimé sous une forme indépendante de la machine

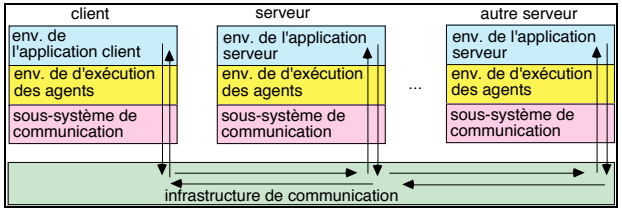
Fonctionnement des agents mobiles (suite)



3 - l'agent **exécute une instruction créant un nouveau process fils** s'exécutant dans un message exprimé sous une forme indépendante de la machine :

- le **message envoyé** à sa destination finale ou à des destinations intermédiaires prenant en charge le transfert selon son contenu ("Semantic Routing")
- le **message arrive au serveur** destinataire où il est livré à l'environnement d'exécution de l'agent par le sous-système de messagerie du serveur
- dans l'environnement d'exécution de l'agent, le message reçu est **reconstitué en un exécutable** (process ou thread)
- l'**exécution du programme agent client se poursuit** avec l'instruction suivante

Fonctionnement des agents mobiles (fin)



4 - pendant son exécution sur le serveur, l'agent peut :

- **terminer** son exécution
- **passer l'information** qu'il a reçu de l'application client à l'application serveur
- effectuer une **action de récupération de données** sur le serveur
- **déterminer** s'il doit **rendre visite à un autre serveur** :
  - si le service nécessaire n'est pas accessible ou mal adapté
  - en fonction les données qu'il a reçues du serveur sur lequel il se trouve
- **répéter le processus de migration** en lançant un nouveau processus fils pouvant ramener l'agent à son client d'origine ou l'envoyer à un autre serveur client
- se **suspendre** sur le serveur en attendant qu'un événement approprié ne le reveille : **agent "résident"** sur le serveur (permanent si l'utilisateur souhaite un service répété)

Caractéristiques des agents mobiles

Mobilité :

- **Forte mobilité**: **code** et **données** sont transférés sur le serveur (host) - langages de script en plus qu'un simple transfert d'objet
- **Faible mobilité**: seules les **données** sont transférées Mobility - transfert de code Java sur une machine virtuelle

Remarque :

- un **code mobile** (comme les applets) voyage généralement juste d'un point A à un point B
- les **agents mobiles** ont un **itinéraire** et peuvent voyager dans plusieurs sites de façon séquentielle.

Asynchrone :

- un agent envoyé, **pas nécessaire d'attendre les résultats** : programmer l'agent pour qu'il attende aussi longtemps que nécessaire
- **pas nécessaire de rester connecté au réseau** : un agent peut attendre que l'on soit reconnecté au réseau avant de faire sont rapport

Sécurité, 3 types de menaces :

- un **agent** peut être **malveillant** sur le host
- le **host** peut être **malveillant** pour l'agent
- **l'agent** peut causer des **dégâts au réseau**

=> authentication (en utilisant le certificat X509) et **encryptage** (avec SSL)

Application des agents mobiles

Caractéristiques générales des applications:

- **équilibrage** de **charge dynamique**
- **déploiement dynamique** de services
- **systèmes connectés** de façon **intermitente**

Applications spécifiques :

- **Collecte de données à partir de sources multiples :**

  ils peuvent collecter des informations réparties sur un grand nombre d'ordinateurs reliés à un réseau :

  Ex : archivage réseau : un agent mobile visite le réseau, collecte l'information sur l'état d'archivage de chaque disque et retourner à son point d'origine pour faire un rapport

- **Recherche et filtrage :**

  à la place de l'utilisateur, ils peuvent visiter de nombreux sites du Web, identifier l'information accessible à chaque site, et construire un index de liens vers des éléments d'information qui répondent à certains critères.

Application des agents mobiles (suite)

- **Surveillance :**

  ils peuvent être envoyés pour attendre que certains types d'informations soient disponibles :

  Ex : un agent peut aller sur une place boursière, attendre qu'une action particulière atteigne un certain prix puis en acheter à la place de l'utilisateur

- **Distribution ciblée d'informations :**

  les agents peuvent distribuer des nouvelles interactives ou de la publicité à des destinataires ciblés

- **Négociation :**

  ils peuvent obtenir des informations en interaction avec d'autres agents.

  Ex: en planification de réunion, un agent mobile consulte les agents représentatifs de chacune de ces personnes concernées et négocie une heure de la réunion

- **Commerce électronique :**
  - Parallélisme : les agents mobiles peuvent se déplacer de noeud en noeud et peuvent se reproduire en sous-agents
  - Loisirs : les agents représentent les joueurs qui rivalisent les uns avec les autres, à la place des joueurs
- **Data-mining :**
- ...

Plateformes de développement d'agents mobiles

Languages de développement :

- **Java class libraries** : tire avantage de la machine virtuelle universelle Java (Aglet, concordia, Voyager, Grasshopper, D'Agents)
- **langages de script** : tire avantage de langages interprétés (D'Agents, ARA, Tacoma)

Plate-forme Aglet (IBM) pour le developpement d'agents mobiles :

- programmé en **Java**, il peut **stoper** son exécution, se **déplacer** dans un réseau et **poursuivre** son exécution sur un autre serveur hôte (ses données et son code sont mobiles)
- **autonomes** : lorsqu'on le lance, il décide seul où il ira et ce qu'il fera, il décide individuellement de répondre à une requête ou non, ...
- à priori **ni intelligent** (agents intelligents) **ni représentatif de l'utilisateur** (agents logiciels interface) mais peut le devenir

Autres plate-formes :

- **D'Agents** (Dartmouth College)
- **Concordia** (Mitsubishi Electric ITA)
- **Voyager** (ObjectSpace?)
- **Grasshopper** (IKV++)
- **ARA** (Agents for Remote Access - Univ. of Kaiserslautern)

Les agents mobiles : quelques défits majeurs [Wayner 95 & Nwana]

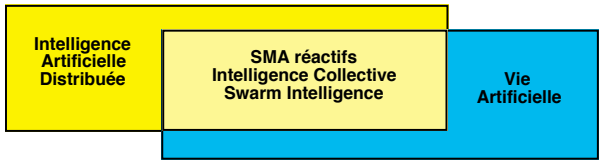
- **Transport**: comment un agent fait sa valise et se déplace de place en place ?
- **Authentication**: comment être sûr que l'agent est bien celui qu'il dit être ? comment savoir s'il a navigué sur divers réseaux sans être infecté par un virus?
- **Secrêt**: comment être sûr que votre agent préserve votre intimité ? que personne ne manipule pour ses intérêts votre propre agent? comment être sûr que votre agent n'est pas mort et que son contenu 'core-dumped'?
- **Sécurité**: comment vous protéger contre les virus ? comment éviter qu'un agent extérieur ne vienne chez vous consommer toute votre CPU dans une boucle sans fin?
- **Paiement**: comment l'agent paiera des services ? comment être sûr qu'il ne sera pas pris d'une folie furieuse qui conduira à une facture outrageuse?
- **Performance** : quel sera l'effet d'avoir des centaines, milliers et millions d'agents sur un réseau?
- **Services d'interopérabilité, de communication et de courtier**: comment se fournir les adresses des divers services offerts? comment exécuter un agent écrit dans un certain langage sur un moteur écrit dans un autre langage ? comment publier ou souscrire à des services donnés?

Agents réactifs

Années 80 : beaucoup de recherche en IA centrées sur les **problèmes de planification** : élaborer une suite d'actions qui part d'un état initial et qui atteint un but donné

Problèmes très difficile à résoudre : la taille de l'espace de recherche des algorithmes utilisés croît exponentiellement avec la complexité du problème

=> Brooks 91: alternative à l'approche symbolique = l'approche réactive



Courants majeurs de l'approche réactive:

- **SMA Réactifs : Agents réactifs de Brooks** [Brooks 91]
- **Intelligence collective ou "swarm" intelligence** (souvent inspirée de l'éthologie): résolution de problèmes et optimisation combinatoire : **EPS** (Eco Problem Solving) ou **éco-résolution** [Ferber], **ANT-ACS**
- **la vie artificielle** [Droghoul], ...

Agents réactifs

Caractéristiques :

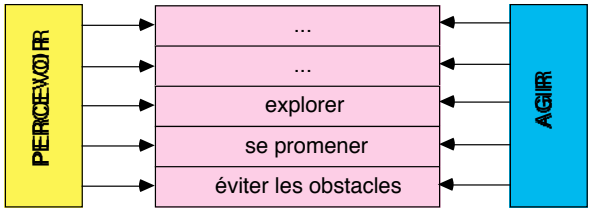
- ils n'ont **pas de représentation de connaissance symbolique** (ie: pas d'état, pas de représentation de l'environnement, pas de représentation des autres agents, ...)
- leur **comportement est simplement défini par un ensemble de règles perception-action**

Propriétés :

- **robustesse** et **tolérance aux fautes**
- **flexibilité**
- **adaptabilité**
- performance en **temps de réponses**

Agents réactifs de Brooks (MIT) [1991]

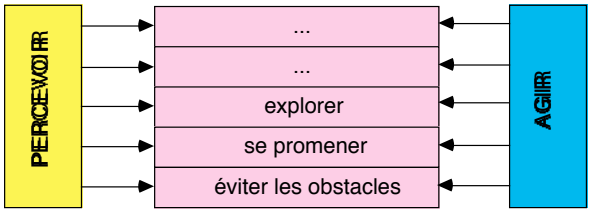
- **alternative aux approches symboliques pour résoudre des pbs de planification**
- un **comportement intelligent émerge** de l'interaction entre divers comportements plus simples
- **architecture dite de "subsumption" :**



- un **agent est associé à un ensemble de comportements** accomplissant une tâche donnée
- ces **comportements = règles de perception-action ordonnées en couche** de subsumption

Agents réactifs de Brooks (MIT) [1991]

Exemple [Steels] : robots collectant des pierres



Règles de comportement :

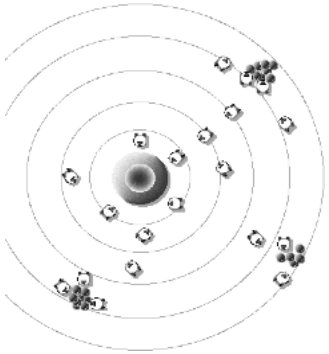
règle 1 : if detect an obstacle then change direction
règle 2 : if carrying samples and at the base then drop samples
règle 3 : if carrying samples and not at the base then travel up gradient
règle 4 : if detect a sample then pick sample up
règle 5 : if true then move randomly

Autres règles associées à un comportement avec communication :

règle 6 : if carrying samples and not at the base then 2 (radioactive) crumbs and travel up gradient
règle 7 : if sense crumbs then pick up 1 crumb and travel down gradient

Exemple du fourragement Collectif [Drogoul]

- Exploration et exploitation collective d'un environnement inconnu et dynamique, par des robots ou agents simulés : application très populaire en **Vie Artificielle**
- Bonne illustration de l'**auto-organisation**
- Beaucoup de techniques envisageables (inspirées la plupart du temps des **fourmis**)



Exemple du fourragement Collectif [Drogoul]

un ensemble simple de règles individuelles peut générer un comportement collectif **adaptatif** :

Règle Explorer
si je suis vide
si je ne perçois ni minerai ni marque
je circule aléatoirement

Règle Trouver
si je suis dehors et vide
si je perçois du minerai
je le prends

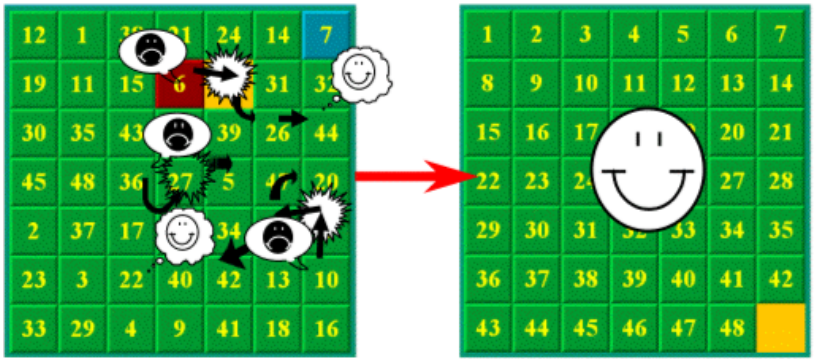
Règle Rapporter
si je suis dehors et plein
je pose deux marques
je reviens à la base

Règle Pister
si je suis dehors et vide
si je perçois une marque
je me dirige vers elle je la prends

Règle Déposer
si je suis à la base et plein
je dépose le minerai

Exemple de résolution de problèmes [Ferber 89][Drogoul 91]

- le problème est décomposé sur un ensemble d'agents
- sa résolution correspond à l'**état d'équilibre** de "**l'éco-système**" qu'ils forment :



Eco-Résolution - Eco Problem Solving (EPS)

un framework de SMA réactif dédié à la résolution de problèmes (existe en : Lisp, Smalltalk, C++, Java) qui s'appuie sur :

- une décomposition structurelle du problème (i.e. "objet")
- un modèle d'agent séquenceant trois comportements basiques (satisfaction, agression, fuite)
- une description explicite de l'état initial et de l'état final du problème.

modèle d'agent : chaque agent a :

- un but (un autre agent)
- des dépendances (d'autres agents)
- des accointances (")
- des gêneurs (")

modèle de comportement : un automate à états finis (différences entre les versions):

- satisfait: l'agent ne fait rien
- rechercheSatisfaction: l'agent cherche à se satisfaire
- rechercheFuite: l'agent a été agressé et tente de fuir
- fuite: l'agent fuit

Le principe d'Eco-Résolution (EPS)

Résolution du problème incrémentale et réactive

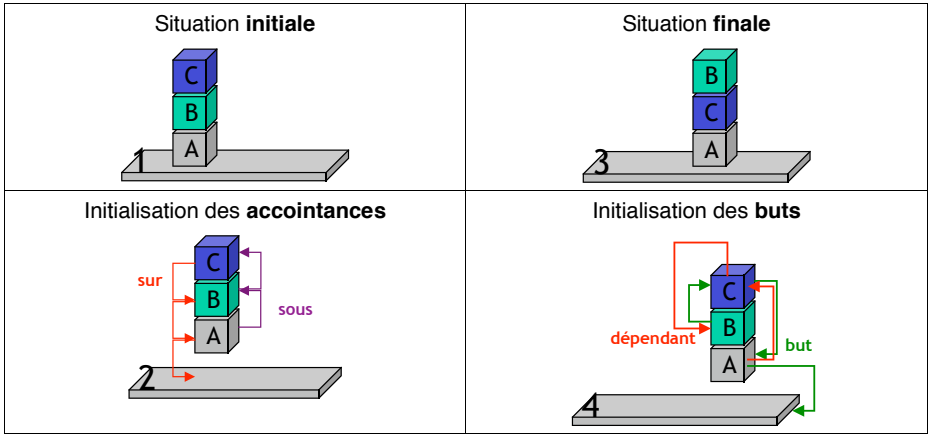
De façon informelle :

- **si le but d'un agent n'est pas satisfait** : l'agent demande à son but de se satisfaire et se place dans ses dépendances
- **quand un agent se satisfait**: il informe ses dépendances qu'elles peuvent se satisfaire
- **quand un agent ne peut se satisfaire**: il recherche les gêneurs parmi ses accointances et les agresse (leur demande de fuir)
- **quand un agent cherche à fuir**: il recherche les gêneurs parmi ses accointances et les agresse

Applications :

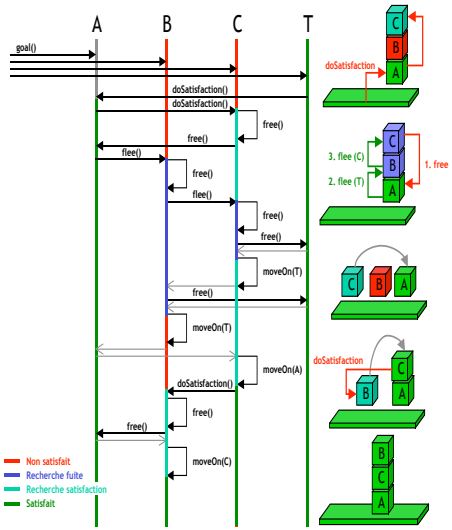
- **Productique** :
  - ordonnancement automatisé d'un atelier flexible - C. Sohier (ENS- Cachan) - chaque machine-outil, robot, outil, etc. est représenté par un éco-agent
  - ordonnancement d'atelier (K. Ghedira - Univ. Tunis III)
- **Satisfaction de contraintes**, recuit simulé distribué (K. Ghedira - Univ. Tunis III)
- **Conception de sous-systèmes électroniques embarqués** : projet Carosse (PSA - LIP6 / Drogoul & al. 98), ...

EPS: Exemple des cubes (1)



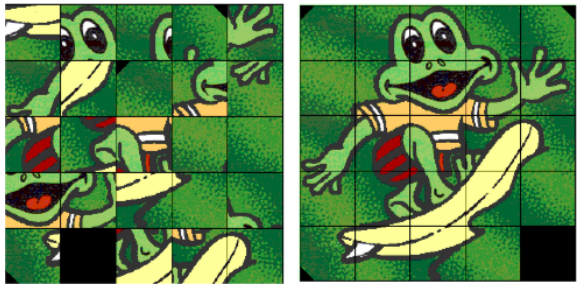
- **pas d'état du problème**, **pas de backward chaining**
- **résolution dynamique**, **solution non optimale** mais **convergence** [Drogoul, Jacopin 92]

EPS: Exemple des cubes (2)



EPS: Taquin (1) [Drogoul]

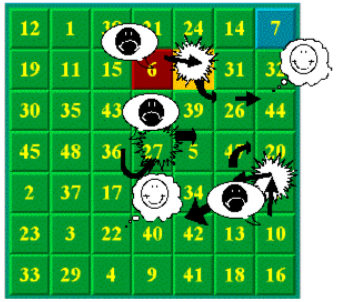
- **problème :**
  - très **populaire** en IA après les échecs,
  - fortement **combinatoire**



- **solution optimale pour le 5x5, impossible au-delà**
- **étudié par Korf: adaptations de A* (RTA*, LRTA*)**

EPS: Taquin (2)

- **2 types d'éco-agents :** les **cases** et les **palets**
- chaque **palet** :
  - a pour **but** une case
  - comme **accointances** les **cases adjacentes**
  - comme "**gêneurs**" les **palets posés dessus**



- la résolution est **séquentielle** dans le cas du taquin "normal" (un seul espace de liberté)

EPS: Taquin (3)

Ajout d'autres mécanismes :

- blocage temporaire des cases agressées et satisfaites
- blocage définitif des lignes satisfaites
- recherche de chemin (par propagation de signaux évitant les places bloquées)
- utilisation importante des contraintes (but de l'agent, etc.)

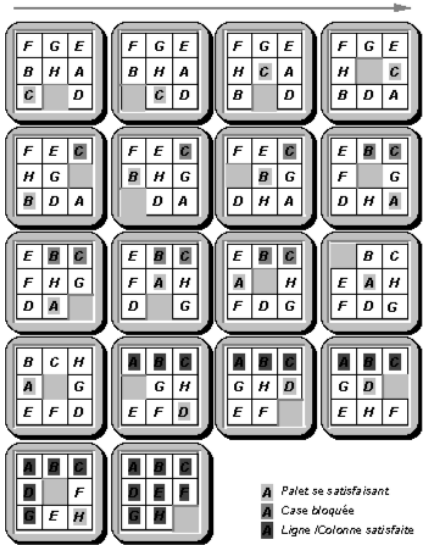
Heuristiques locales:

- **recherche de satisfaction**: se déplacer sur la case la plus proche du blanc parmi les cases les plus proches du but (ou sur le blanc s'il est adjacent)
- **recherche de fuite**: se déplacer sur la case la plus proche du but parmi les cases les plus proches du blanc (ou sur le but s'il est adjacent)
- **contraintes d'agression**: but de l'agent

Résultats:

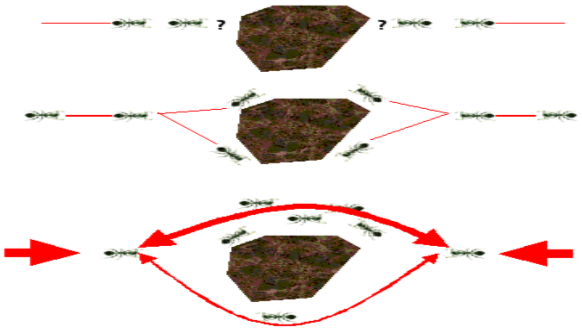
- **complétude et décidabilité démontrées** dans [Drogoul, Dubreuil 93]
- **résultats expérimentaux** obtenus jusqu'à des taquins de 1000x1000...
- **complexité** proche de la complexité approximée de la solution optimale
- **adaptation à des instances de la même classe de problèmes**: taquins à trous, irréguliers, etc.

EPS: Taquin (5)



ANT - ACS: Présentation

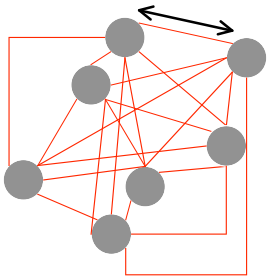
- systèmes de **résolution de problèmes combinatoires** basés sur une forte **analogie** avec les mécanismes mis en œuvre dans les **colonnes de fourmis** [Colorni - Dorigo 91] :



- les **fourmis** se déplacent en déposant des **phéromones** (qui **s'évaporent** au cours du temps) et sont **attirées** par elles
- **décision collective** du plus court chemin équivalente à un algorithme distribué d'optimisation

ANT: Voyageur de commerce ou TSP [Drogoul] (1)

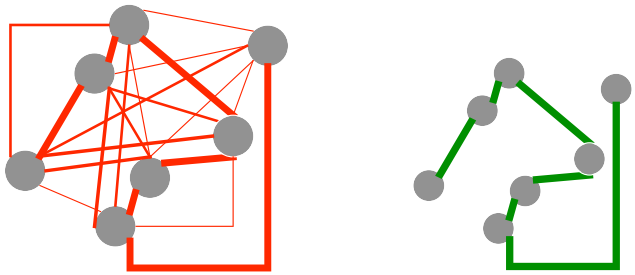
- **problème** du voyageur de commerce ou Travel Seller Problem (TSP):
  - soit un **ensemble N de villes** (noeuds) reliées entre elles par des **arcs valués (distance d)**
  - trouver le **chemin minimal** permettant de parcourir **l'ensemble des villes** en ne passant **qu'une fois** par **chaque ville** (circuit Hamiltonien)



- les TSP peuvent être **symétriques** ou **asymétriques** ($d_{ab} \neq d_{ba}$; a, b ∈ N)

ANT: Voyageur de commerce ou TSP (2)

- à chaque **arc** est associé : sa **distance** et un **taux de phéromones**
- les **fourmis font un tour complet** en choisissant les villes selon une **règle probabiliste** consistant à emprunter les arcs les plus courts et les plus riches en phéromones
- une **itération** consiste à faire parcourir toutes les villes par **toutes les fourmis**
- à **chaque itération** une règle fait que le **taux de phéromone**:
  - **s'évapore** en partie
  - **augmente** pour chaque arc visité en fonction de la longueur des parcours



ANT: TSP (3) : Description précise

- **N** ensemble de **n** noeuds, **Ni** l'ensemble des voisins du noeud **i** et **E** ensemble des arcs qui les connectent
- $\mathbf{d_{ij}}$ longueur de l'arc (i,j) ∈ E et $\boldsymbol{\eta_{ij}} = 1/d_{ij}$ sa valeur heuristique
- **F** ensemble de **m** fourmis
- **t** (compris entre **0** et **tmax**) identifie l'itération en cours
- $\boldsymbol{\tau_{ij}(t)}$ est le taux de phéromones associé à l'arc (i,j)
- **Mk** qui contient N est une **mémoire** associée à la fourmi k (qui lui permet de savoir quels noeuds elle a parcouru, et lesquels elle ne doit plus parcourir). Chaque noeud visité y est ajouté.
- **à chaque noeud i**, la **fourmi** construit une **table de décision A_i** contenant les éléments suivants :

$$a_{ij}(t) = \frac{\left[\tau_{ij}(t)\right]^{\alpha}\left[\eta_{ij}\right]^{\beta}}{\sum_{l \in N_i}\left[\tau_{il}(t)\right]^{\alpha}\left[\eta_{il}\right]^{\beta}} \forall j \in N_i$$

où $\boldsymbol{\alpha}$ et $\boldsymbol{\beta}$ représentent les poids relatifs de la trace de phéromones et de la valeur heuristique

ANT: TSP (4) : Description précise

- la probabilité avec laquelle elle choisit de se déplacer de **i** à **j** à l'itération **t** est :

$$p_{ij}(t) = \frac{a_{ij}(t)}{\sum_{l \in N_i} a_{il}(t)}$$

- quand toutes les fourmis ont réalisé leur tour, chacune dépose sur les arcs qu'elle a visité une quantité de phéromones :

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} 1/L^{k}(t) & si\ (i,j) \in T^{k}(t) \\ 0 & si\ (i,j) \notin T^{k}(t) \end{cases}$$

où $T^k(t)$ représente le chemin parcouru par la fourmi, et $L^k(t)$ sa longueur

- puis les phéromones sont décrémentés selon la règle :

$$\tau_{ij}(t) \leftarrow (1-\rho)\tau_{ij}(t)$$

où $\rho \in [0,1]$ représente le coefficient d'évaporation

ANT: TSP (5) : Description précise

Les paramètres qui pilotent l'algorithme sont donc :

- **$\tau_{ij}(0)$**: le taux initial de phéromones, qu'il faut initialiser
- **m**: le nombre de fourmis considérées
- **α** et **β**: les valeurs contrôlant les poids respectifs de l'heuristique et des phéromones
- **ρ** : le coefficient d'évaporation

En pratique, les meilleurs résultats ont été obtenus avec les valeurs suivantes :

- **$\tau_{ij}(0)$** = 0,1
- **m** = |N| (nombre de noeuds)
- **α** = 1et **β** = 5,
- **ρ** = 0,5

Agents réactifs: intérêts et applications

Domaine d'application privilégié :

la simulation de systèmes complexes (biologiques, chimiques, physiques, sociaux, etc.) : **ABS** (Agents Based Simulation)

Intérêt de l'approche réactive permet aussi :

- d'obtenir de **nouvelles méthodes de résolution de problèmes** (cf. TSP)
- les **ANT-ACS** sont une classe d'algorithmes bien adaptés aux problèmes combinatoires qui peuvent s'exprimer sous forme de parcours
- de **représenter et de traiter des contraintes à différents niveaux**: fonctionnelles, structurelles, quantitatives, qualitatives, local, semi-global, global
- de **réifier les différents niveaux d'analyse** et leurs interactions
- d'obtenir des **solutions adaptées à des modifications dynamiques**

Agents réactifs: faiblesses et limites

- **pas encore de méthodologies claires** pour faciliter sa mise en oeuvre
- les agents doivent **posséder suffisamment d'informations pour agir** car ils ont généralement pas de modèle de leur environnement
- les **décisions** des agents basées que sur des **informations locales**
- **apprentissage difficile de prendre en compte** pour l'améliorer les performances des agents
- l'**émergence est trés difficile à construire**
- si agents à plusieurs couches, l'**interaction entre couches est complexe**
- **points critiques pour les ANT-ACS et l'EPS** :
  - espace de paramètres très important
  - convergence difficile à prouver
  - heuristiques locales difficilement généralisables

Agents hybrides

Caractéristiques :

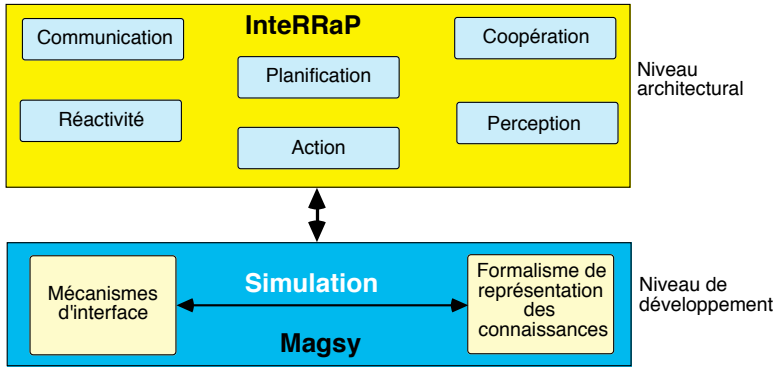
- jusqu'à présent, agents de type : collaboratif, interface, mobile et réactifs
- les agents hybrides combinent 2 ou plusieurs philosophies
- en général agent hybride combinent les philosophies collaborative et réactive :
  - le composant **réactif** se situe **en amont** du composant **délibératif**
  - le composant **réactif** apporte à l'agent **robustesse**, **rapidité** de réponse et **adaptabilité**
  - le composant **délibératif** traite des **problèmes à buts à long terme**
- architectures d'agents hybrides souvent **organisées en couches**

Architectures hybrides les plus connues :

- **InteRRaP** de Muller et al. (95) : combine agents collaboratifs et agents réactifs
- **Touring Machines** de Ferguson (92) : agents dynamiques, rationnels et mobiles
- les architectures de Hayes-Roth (91) : agents
- **SRK** : combine Skill, Rules et Knowledge, ...

Un agent hybride: InteRRaP [DFKI - Saarbrücken - Muller et al. 95]

- **Objet de InteRRap :** prédire, par **simulation**, le comportement d'un système complexe



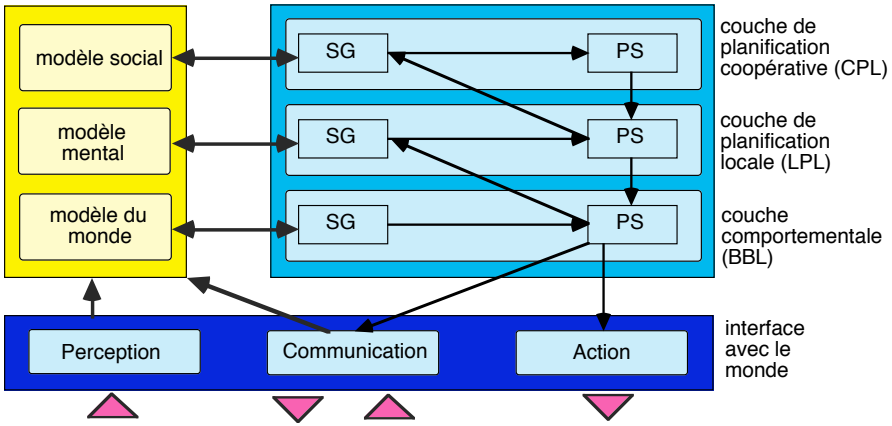
The InteRRaP architecture has been evaluated by constructing a FORKS application which simulates forklift robots working in an automated loading dock environment.

L'architecture InteRRaP [Muller et al. 95]

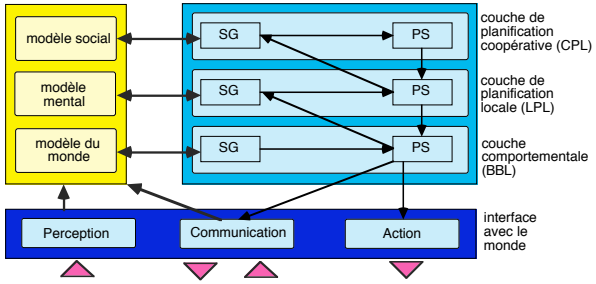
- **Niveau architectural : l'agent InteRRAP**
  - repose sur une base de connaissance et son unité de controle associée située au sommet du composant "perception-action" qui traite des communications de bas niveau
  - combine le **délibératif** et le **réactif** et est composée de **3 couches :**
    - couche BBL (**réactive** - behaviour-based layer)
    - couche LPL (**délibérative** - local planning layer)
    - couche CPL (**délibérative** - cooperative planning layer)

- **Niveau de développement : MAGSY**
  - Représentation **orientée objet** des connaissances
  - **Inférence** :
    - **chaînage avant** pour l'activation des conditions dans le BBL
    - **chaînage arrière** pour la planification (LPL et CPL)
  - **Simulation** : création d'agents, visualisation et contrôle, expérimentations et tests

Architecture d'un agent d'InteRRaP



Architecture d'un agent d'InteRRaP



- **couche BBL** (**reactive):** comporte un ensemble de patrons de comportement (PoBs - situation-action rules) décrivant les compétences réactives des agents assurant la reconnaissance rapide de situation en réaction à une situation critique
- **couches LPL et CPL** (délibératives): LPL implemente local goal-directed behaviour while the topmost CPL enables the agent to plan/cooperate with other agents in order to achieve multi-agent plans, as well as resolve conflicts

Ces 3 couches utilisent des modèles spécifiques dans la base de connaissance de l'agent (modèles du monde, mental et social) et mettent en oeuvre 2 types de processus (SG et PS) qui interagissent dans une même couche et avec les couches voisines de façon asynchrone.

Systèmes multiagents hétérogènes

Caractéristiques :

- intègrent des **agents de classes différentes**
- peut contenir un ou plusieurs **agents hybrides**

Finalités :

- les **systèmes ouverts** (ex: l'internet)
- **interopérabilité : … ACL**
- liens avec les **legacy systems**

Défis :

- **communication**
- **ontologies**
- **liens avec les « legacy systems »**