Les systémions: un modèle d'agents logiciels ontiques

Eric Sanchis

Laboratoire Gestion et Cognition. IUT Ponsan - Université Paul Sabatier.
115, route de Narbonne. 31077 - TOULOUSE - Cedex. FRANCE
sanchis@iut-rodez.fr

Résumé

De nombreux domaines de l'informatique utilisent aujourd'hui la notion d'agent logiciel. Alors que la plupart de ces derniers désignent des entités à la durée de vie éphémère et unitâches, c'est à dire capable d'exécuter une seule tâche, un nouveau modèle d'agent est proposé, introduisant *finalité naturelle* et *plurifonctionnalité*. Une architecture à deux couches est construite au moyen d'une analyse fine des propriétés communément associées à un agent, distinguant *qualités* et *attributs*. Le modèle est ensuite utilisé pour étudier différentes familles d'agents.

Mots clé : agent commun, agent systémique, finalité naturelle, propriétés, architecture d'agent

La dynamique qui anime le développement de nouvelles applications informatiques ne serait vraisemblablement pas ce qu'elle est si l'individualisation des modules logiciels qui les composent n'était pas aussi prononcée. La maîtrise de cette anthropomorphisation croissante nécessite que les concepteurs d'applications développent sans cesse de nouveaux modèles conceptuels et architecturaux. Dans cet objectif, nous proposons une nouvelle architecture des entités de base présentes au sein de ces applications. Cette architecture, essentiellement basée sur le *paradigme agent*, propose une encapsulation et une articulation des propriétés affectées à un agent.

L'article se décompose en trois sections. La première section introduit dans un premier temps la *métaphore agent* sous deux aspects complémentaires : l'agent comme concept - c'est à dire comme entité munie de propriétés de complexité variable - et l'agent comme outil - c'est à dire comme moyen structurel de construction d'une application logicielle -. Ensuite, une définition systémique de la notion d'agent sera proposée. La section 2 présentera deux propriétés supplémentaires qu'inclut un agent systémique, la propriété de transformation et la finalité naturelle. Cette dernière sera artificialisée pour conduire à une nouvelle architecture d'agent logiciel : les *agents ontiques*. La dernière section précisera les composantes de cette architecture générique et son opérationnalité sera illustrée par l'analyse de plusieurs classes d'agents.

1. Agents communs et agents systémiques

Le terme *agent* constitue par excellence, ce que les anthropologues ont appelé un terme «polythétique», c'est à dire "qui se réfère à différentes notions qui se ressemblent mais ne relèvent pas pour autant d'une seule et même définition" [Needham (1975)], [Sperber (1996)]. En effet, ne serait-ce qu'en informatique, de nombreux champs de recherche utilisent la notion d'agent logiciel soit comme *concept*, soit comme *outil*, ou bien des deux à la fois : intelligence artificielle, vie artificielle, systèmes et applications répartis, applications liées à la nouvelle économie (e-applications), ingénierie concourante, etc. Pour illustrer cette diversité sémantique nous ferons appel à deux disciplines très différentes : les *systèmes multi-agent* et les *applications réparties orientées agent*.

1.1 *L'agent dans les systèmes multi-agent*

La discipline reine où l'agent est considéré comme *concept* est le domaine des «systèmes multi-agent» (SMA), branche très active de l'intelligence artificielle. Celle-ci considère un agent comme une entité possédant un certain nombre de *propriétés*, au demeurant très différentes les unes des autres telles que l'intelligence, l'autonomie, la perception ou la réplication. On peut affirmer que les SMA ont fourni l'étude la plus vaste et la plus riche sur ce que pouvait être un agent [Ferber (1995)], [Jennings (1998A)], [Jennings (1998B)].

Illustrons cette richesse par un exemple. Pour Jennings et Wooldridge [Jennings (1998A)], un agent est une entité informatique *située* dans un environnement, qui est capable *d'agir de manière autonome* et *flexible* de façon à remplir ses objectifs, c'est à dire ceux qui lui ont été assignés lors de sa conception.
L'aspect *situé* signifie que l'agent perçoit son environnement et qu'il est capable de le modifier.
L'autonomie, bien que difficile à caractériser de manière précise, signifie que l'agent est capable d'agir sans l'intervention directe d'êtres humains ou d'autres agents et qu'il est capable de contrôler ses propres actions et son état interne.
Enfin, c'est la *flexibilité* qui caractérise, pour les auteurs, un *agent intelligent*. Etre *flexible* signifie alors que l'agent est capable :

- de percevoir son environnement et répondre en temps voulu aux changements qui s'y produisent (*réactivité*),
- d'avoir un comportement dirigé par son but, d'être opportuniste et capable d'initiative quand c'est approprié (*proactivité*),
- d'interagir, quand c'est approprié, avec d'autres agents artificiels ou humains de manière à compléter leur résolution de problème et d'aider les autres par leurs activités (*sociabilité*).

Cette notion de flexibilité propose une vision *intégrée* de l'intelligence, c'est à dire un ensemble de facultés liées à celle-ci mais dont les traits particuliers de chacune d'elles disparaissent au profit d'une aptitude complexe; le point de vue *intégré* se distingue du point de vue *modulaire* qui consiste à séparer les facultés élémentaires liées à l'intelligence telles que le raisonnement, l'apprentissage et d'autres encore, afin de les traiter de manière indépendante.

D'un point de vue technique, un SMA est une application où les agents sont organisés en réseau(x) faiblement couplé(s) qui travaillent ensemble pour résoudre un problème commun, les possibilités de chaque entité prise séparément ne permettant pas de le résoudre.
Chaque agent possède donc des connaissances et savoir-faire limités, ce qui l'oblige à interagir avec d'autres pour mener à bien le projet commun.
Les SMA mettent en oeuvre des entités aux interactions complexes, comme la *coopération*, la *coordination* ou la *négociation*. Ces trois types d'interactions peuvent brièvement se définir de la manière suivante [Jennings (1998B)] :

- il y a *coopération* lorsque les agents travaillent ensemble vers un but commun,
- la *coordination* consiste à organiser les activités liées à la résolution d'un problème de manière à éviter les interactions néfastes et à exploiter les interactions bénéfiques,
- la *négociation* a pour objet la gestion des conflits entre agents, c'est à dire qu'elle consiste à aboutir à un accord acceptable par l'ensemble des agents impliqués.

Ces interactions sophistiquées ne sont possibles que par l'intégration au sein des agents de mécanismes et protocoles de communication de haut niveau (actes de langage).
Enfin, les notions de *société* (également appelée *collectivité*), d'*organisation* et les relations entre entités que celles-ci induisent, jouent un rôle crucial dans un SMA.

1.2 *L'agent dans les applications réparties*

Plus récent que les SMA, le domaine des *applications réparties orientées agent* s'en est fortement inspiré dans un premier temps, puis progressivement écarté pour n'en conserver que quelques aspects. Illustrons cette filiation par la caractérisation d'un agent énoncée par Etzioni et Weld [Etzioni (1995)] :

- il est *autonome* : un agent est capable de prendre des initiatives et d'exercer un degré de contrôle non trivial sur ses propres actions ; cela signifie qu'il est
  - . *orienté but*, c'est à dire qu'il accepte des requêtes de haut niveau indiquant ce que veut l'utilisateur et est responsable de leur traitement

. *participatif* : un agent n'obéit pas aveuglément aux commandes mais à la capacité de modifier les requêtes, de demander des clarifications ou même de refuser de satisfaire certaines requêtes

. *flexible* : les actions d'un agent ne sont pas « inscrites en dur »; il est capable de choisir dynamiquement les actions à effectuer et dans la séquence voulue, en réponse à l'état de son environnement externe

- il possède une *personnalité* bien définie et un état émotionnel
- il *s'adapte* aux préférences de son utilisateur, en se basant sur son expérience antérieure. Il s'adapte aussi automatiquement aux changements de son environnement
- il est *communicant* : l'agent est capable de soutenir des conversations complexes avec les autres agents, y compris avec des individus, de manière à obtenir leur aide
- il est *mobile* : un agent est capable de se transporter d'une machine à l'autre et à travers d'architectures de systèmes et de plateformes différentes
- il est *continuellement actif* : ce n'est pas un programme qui accepte une requête, produit une sortie puis se termine.

Cette définition "maximaliste" d'un agent a tout d'abord été reprise lors des premiers travaux de la FIPA (*Foundation for Intelligent Physical Agents*) puis a été fortement épurée, ne gardant que les trois dernières propriétés (la communication, la mobilité et la continuité temporelle). Il est important de préciser que cette organisation internationale avait et a toujours pour objet la standardisation des différentes plates-formes technologiques dédiées à la construction d'applications réparties à agents mobiles.

Pour de nombreux chercheurs qui travaillent dans le cadre des applications réparties, le terme agent désigne une granularité de l'entité active, une unité de structuration, plutôt qu'un ensemble de propriétés complexes. Une *application répartie à agents mobiles* (ARAM) peut être définie de la manière suivante :

- elle est mise en oeuvre par des agents, et non par des modules logiciels de nature quelconque
- les agents se meuvent et communiquent pour fournir un service,
- il n'y a pas de contrôle centralisé au sein de l'application,
- aucun agent ne connaît l'état global de l'application,
- l'exécution des différentes entités est asynchrone,
- les données traitées sont logiquement ou géographiquement réparties.

Le point de vue « instrumental », « technicien » de l'agent y est particulièrement prononcé. De ce fait, la méthodologie développée dans la construction d'une ARAM fournit un complément opératoire précieux pour implanter le concept d'agent. Mais est-ce suffisant ?

1.3 *L'agent systémique*

La double caractérisation d'un agent présentée précédemment pourrait laisser entendre que tout a été dit sur ce que peut recouvrir la notion d'agent et qu'il ne resterait plus qu'à affiner les travaux antérieurs : dans le cadre des SMA, un agent est un ensemble de propriétés, dans le domaine des applications réparties, un agent est une technique de structuration d'entités mobiles et communicantes.
Nous pensons qu'il n'en est rien et qu'au moins une dimension importante a été négligée : la *finalité*. Pour introduire celle-ci dans notre présentation, nous associerons étroitement le concept d'agent à celui de *système général* tel qu'il a été décrit par Le Moigne [Le Moigne (1977)]. En effet, cette mise en correspondance a pour avantages, non seulement d'intégrer de manière naturelle la propriété de finalité, mais également de plonger l'agent dans sa dimension temporelle. En paraphrasant la définition d'un système général, nous dirons qu'un *agent systémique* est une entité (a) plongée dans un environnement, (b) munie d'une structure, (c) qui possède une activité par rapport à quelque finalité, et (d) se transforme dans le temps.

Précisons rapidement chaque trait d'un *agent systémique* perçu comme entité logicielle :
(a) l'agent est plongé dans un environnement : cela signifie que l'agent est situé dans un univers où il y a action et réaction réciproque.
(b) l'agent est muni d'une structure, c'est à dire qu'il possède une architecture logicielle lui permettant de s'exécuter dans un système informatique

(c) il possède une activité par rapport à quelque finalité : il s'agit de l'aspect d'un agent que nous développerons principalement dans les sections suivantes
(d) l'agent se transforme dans le temps : la faculté de se transformer physiquement (et non seulement sous l'aspect informationnel) est une autre dimension qui n'a pas reçu, à notre avis, toute l'attention nécessaire.

2. Finalité et agents ontiques

Il est trivial d'affirmer qu'un agent, comme tout artefact, est finalisé. Mais qu'entend-on généralement par cela ? Si nous nous limitons aux deux domaines précédemment abordés, il s'agit pour un SMA d'aboutir à la résolution du problème posé par ses concepteurs et pour une ARAM, de rendre le service spécifié. En termes plus généraux, un système finalisé a pour but d'accomplir la mission qui lui a été confiée. Il serait réducteur d'en rester là.

2.1 *la finalité naturelle*

Les sciences philosophiques nous ont enseigné que « la finalité objective de la nature » - comme l'a si joliment écrit Kant [1]- pouvait être une *finalité externe* ou bien une *finalité interne*. Ces deux aspects de la *finalité naturelle* peuvent être présentés de la manière suivante [Morfaux (1980)] :

- la *finalité externe* est "ce par quoi une chose dans la nature sert à une autre de moyen en vue d'une fin", et
- la *finalité interne* "a pour fin l'être même dont les parties sont considérées comme moyens les unes pour les autres ".

Nous identifions immédiatement cette finalité externe avec la finalité dont nous avons coutume de munir nos agents et applications logiciels.
Pour Morin, la finalité interne est le propre de ce qu'il appelle le *computo vivant* ou le *Calculer pour Soi* : "La computation vivante doit sans cesse résoudre les problèmes du vivre, qui sont ceux du survivre (...). Les seconds problèmes clés (...) sont ceux de la nourriture et de la défense au sein d'un environnement aléatoire. Aussi l'être vivant compute-t-il son environnement, en extrait-il des informations afin de reconnaître ce qui peut le nourrir ou le détruire. L'unicellulaire lui-même (...) reconnaît des formes et des substances assimilables ou non assimilables, il reconnaît certaines répétitions/régularités/constances et peut détecter des évènements ou perturbations; il peut, dans ces conditions, extraire des informations, en fonction desquelles il pourra déterminer son comportement (rapprochement, fuite). (...).
La computation vivante est vouée essentiellement à l'organisation de l'être et de sa reproduction. La différence cognitive entre computation artificielle et computation artificielle nous apparaît maintenant dans toute sa radicalité : les machines résolvent nos problèmes, non les leurs, disait von Foerster. La bactérie compute pour sa propre organisation, sa propre production et sa propre reproduction " [Morin (1986)].

Cette distinction entre ces deux facettes de la finalité, en apparence si tranchée, peut-elle être partiellement gommée ? Nous pensons que cela est possible moyennant quelques précautions.
Pour cela, nous poserons le problème que nous avons à résoudre de la manière suivante :

- comment construire un agent logiciel qui intègrerait les deux aspects de la finalité naturelle,
- comment organiquement articuler les composantes qui mettent en œuvre la finalité externe et celles qui réalisent la finalité interne de l'agent.

La solution que nous proposons sera de type *architectural*.

2.2 *Agents ontiques*

D'un point de vue technique, la finalité externe de l'agent est matérialisée sous la forme d'une tâche à réaliser ou d'un service à fournir : par exemple, un agent mobile doit chercher et récupérer des informations d'une certaine nature parmi un ensemble de sites interconnectés.

[1] Kant E. *Critique de la faculté de juger*. Traduction et introduction de A. Philonenko, bibliothèque des textes philosophiques, Editions Vrin, Paris 1993

D'autre part, la finalité interne a pour objet de maintenir l'intégrité interne de l'agent logiciel, soumis aux fluctuations éventuelles de sa finalité externe (après avoir trouvé l'information cherchée, la nouvelle mission affectée à l'agent est de supprimer cette information sur les différents sites dépositaires) et aux modifications de son environnement immédiat (la charge du système qui l'héberge continue de croître, dégradant le fonctionnement de l'agent).

On remarquera immédiatement qu'un conflit peut survenir entre les deux finalités : l'exécution de la tâche conduit l'agent à devoir aller sur un site matériellement ou logiciellement incompatible.
Le découplage entre les deux finalités pose un problème d'arbitrage interne auquel les agents ontiques apportent une réponse qui leur est propre.

Nous appelons *agents logiciels ontiques* (ou *agents ontiques*[2]), un agent logiciel muni d'une finalité interne qui lui confère une propriété d'*existence* indépendante de sa finalité externe. Lorsque l'expression des deux finalités conduit à un conflit interne, c'est la propriété d'existence qui est privilégiée : dans un agent ontique, l'existence prédomine sur la fonction.

La distinction Existence / Tâche contribue à dissocier clairement *ce que doit effectuer* l'agent de *ce qui matérialise son « essence propre »*.

Le principe d'une architecture duale posé, nous devons maintenant préciser son contenu.

3. Une architecture pour les agents logiciels ontiques

Lorsque les SMA et les ARAM ont été présentés dans la première section, le lecteur aura pu constater
- que les agents sont généralement décrits en termes de *domaines d'applications* et de *propriétés*
- qu'une même propriété (par exemple l'**autonomie**) est décrite de manière très différente suivant la caractérisation choisie (Jennings ou Etzioni)
- que toutes les propriétés ne sont pas de même nature ; il suffirait pour s'en convaincre de considérer l'**intelligence** et la **mobilité**.

Cela nous a conduits à distinguer deux grandes classes de propriétés : les *qualités* et les *attributs*.

3.1 *Qualités et attributs*

Une *qualité* peut être définie simplement comme une manière d'être, un aspect non mesurable d'un objet ou d'une entité. Des exemples de qualités sont **l'autonomie**, **l'intelligence**, la **sociabilité**, la **proactivité**.
Les qualités sont difficilement « mesurables » car une qualité est multidimensionnelle. Cela a pour conséquence qu'il existe de multiples modèles complémentaires pour une qualité : il existe, par exemple, plusieurs modèles de l'**autonomie**.

Contrairement à une qualité, un agent possède ou ne possède pas un *attribut*. Un attribut se réduit à un mécanisme plus ou moins complexe. Des propriétés que l'on peut qualifier d'attributs sont la **perception**, la **mobilité**, la **réplication**, **l'exécution continue**.

Il est important de noter que si nous accordons autant d'importance à distinguer mécanismes et qualités, c'est parce que nous pensons que ce sont ces dernières qui permettent de différencier les différentes classes d'agents. En effet, tous les chercheurs n'accordent pas le même intérêt aux différentes qualités incluses dans un agent. Ainsi, certains chercheurs considèrent une qualité particulière comme étant fondatrice de la

[2] ontique: adjectif construit par opposition à l'adjectif *ontologique*.
L'*ontologie* est la science ou l'étude de l'Etre et de ses attributs essentiels tel l'Existence.
Pour Heidegger, l'ontologie est la science qui porte sur *l'être de l'étant*.
Est ontique ce qui a trait aux êtres particuliers. Pour Heidegger, doit être qualifié d'ontique ce qui appartient à l'*étant* [Lercher (1985)], [NPD (1990)].

notion d'agent, tandis que d'autres considèrent que c'est la combinaison de plusieurs qualités qui distinguent les agents des programmes classiques. Donnons trois exemples significatifs. Franklin et Graesser [Franklin (1996)] fondent la notion d'agent sur l'**autonomie**. En introduisant la notion **d'action autonome flexible**, Jennings et Wooldridge [Jennings (1998A)] combinent **intelligence** et **autonomie**. Enfin, Huhns et Sing [Huhns (1997)] appuient leur définition d'un agent sur la **sociabilité**.

3.2 *Les systémions*

Les **systémions** (contraction de *systemic daemon*) [Sanchis (1999)] sont des agents logiciels qui intègrent dans leur architecture, les concepts de *finalité interne* et *finalité externe*, de *qualités* et d'*attributs*. Cette architecture se décompose en deux sous-systèmes :

- un *sous-système comportemental* : celui-ci contient les propriétés (qualités et attributs) propres à l'agent, c'est à dire les propriétés qui sont indépendantes de la tâche qu'il aura à réaliser. Cette couche logicielle implante la *finalité interne*
- un *sous-système fonctionnel* qui matérialise la *finalité externe*, c'est à dire les propriétés (qualités et attributs) relatives à l'accomplissement de la tâche éventuellement assignée à l'agent (Figure 1).

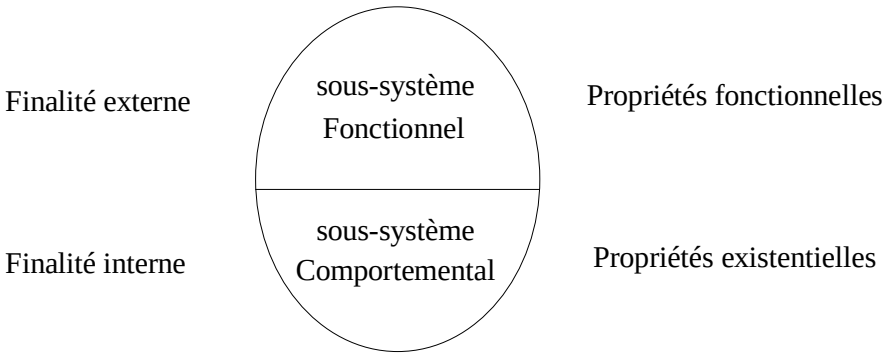

Figure 1 : Architecture d'un systémion

A l'aide de ce modèle d'architecture, nous allons décrire les différentes classes d'agents que celui-ci permet d'analyser. Commençons par les agents communs tels qu'ils ont été présentés dans la première section.

Les *agents communs* ne possèdent qu'un sous-système fonctionnel – leur sous-système comportemental est vide – Les propriétés qu'ils mettent en œuvre sont uniquement dédiées à l'exécution de la tâche qui leur a été assignée : les agents communs sont des agents *unitâche*.

Considérons maintenant, sous une forme simplifiée, les programmes informatiques de type ver et virus. Sans être exagérément réductionnistes, nous dirons qu'un ver a pour objet de se répliquer localement ainsi qu'à distance via un réseau de communication.
Suivant l'architecture systémion, un *ver* est constitué d'un sous-système comportemental qui intègre deux attributs : un mécanisme de perception (perception locale et perception de sites distants) et un mécanisme de réplication (réplication locale et réplication distante), attributs qui lui confèrent une certaine mobilité. D'autre part, ne réalisant aucune tâche particulière, un ver n'intègre pas de sous-système fonctionnel : ce dernier est vide et figé. En effet, son sous-système fonctionnel ne subira aucune modification opérationnelle pendant sa durée de vie. Nous dirons qu'un ver, sous sa forme simplifiée, est un agent ontique fonctionnellement déterminé vide.

Un *virus* possède les mêmes attributs comportementaux qu'un ver (même si leur mode de fonctionnement est différent), mais est muni d'un sous-système fonctionnel non vide : en effet, un virus a généralement pour tâche de détériorer les informations qu'il manipule. Comme pour un ver, le sous-système fonctionnel d'un virus est figé (mais non vide). Dans la terminologie systémion, nous dirons qu'un virus est un agent ontique fonctionnellement déterminé unitâche.

Les agents étudiés précédemment étaient tous fonctionnellement déterminés (vide ou unitâche), et au mieux, ontiques. L'architecture systémion a pour ambition de favoriser la conception et la réalisation d'agents *fonctionnellement non déterminés*. Un tel agent aura un sous-système fonctionnel qui variera au cours du temps, au grès des différentes tâches qu'il aura à accomplir.
Pour que cette modification dynamique de fonction puisse avoir lieu, l'agent intègrera dans son sous-système comportemental un mécanisme approprié (mis en œuvre par exemple à l'aide de techniques de modifications dynamiques de code), que nous appellerons faute de mieux attribut de *modification*. De manière à ne pas compromettre l'existence du systémion, le changement de tâche courante est déclenché par le sous-système comportemental mais les modifications sont effectuées dans le sous-système fonctionnel. Ce dernier constitue donc la partie la plus flexible du systémion.

Muni de ce mécanisme de transformation interne, le systémion pourra évoluer de différentes manières. Par exemple, à un instant t, l'agent pourra n'avoir aucune tâche intégrée dans son sous-système fonctionnel (ce dernier est alors temporairement vide); à l'instant t+1, il pourra intégrer une tâche T puis à l'instant t+2, intégrer une tâche T'. L'architecture systémion permet de construire des agents *pluritâches*.

Conclusion

L'introduction de la finalité naturelle dans le corps même de l'agent est une étape supplémentaire franchie dans l'individualisation du logiciel. L'artificialisation de cette propriété nous a conduits à élaborer une architecture d'agent qui dissocie clairement le *service à réaliser* (sous-système fonctionnel) et le *vecteur d'exécution* (sous-système comportemental). Cela offre la possibilité de concevoir des agents aptes à effectuer des tâches différenciées dans le temps. Dans cette optique, nous comptons approfondir nos recherches dans deux directions : l'étude de la dimension temporelle qui prend un sens nouveau dans notre construction logicielle et les différentes modalités de mise en œuvre et de gestion des interactions entre les deux sous-systèmes architecturaux.

Bibliographie

Etzioni O., **Weld** D. S. (1995). "Intelligent agents on the Internet: Fact, Fiction, and Forecast", *IEEE Expert* 10(4): pp 44-49, 1995.
Ferber J. (1995). "*Les systèmes multi-agents: vers une intelligence collective*", InterEditions, Paris, 1995.
Franklin S., **Graesser** A. (1996). "Is it an Agent, or just a Program ?: A Taxonomy for Autonomous Agents" in "*Intelligent Agents III – Proceedings of the Third International Workshop on Agent Theories, Architectures and Languages (ATAL 96)* , (Edited by Müller J. P., Wooldridge M. and Jennings N.), Lecture Notes in Artificial Intelligence, 1193, Springer Verlag, 1996.
Huhns M. N., **Singh** M. P. (1997). "The agent test", *IEEE internet computing*, sept.-oct. 1997, pp 78-79.
Jennings N., **Wooldridge** M. (1998A). "Applications of Intelligent Agents" in "*Agent Technology: Foundations, Applications, and Markets*" (Edited by N. R. Jennings and M. Wooldridge) Springer Computer Science, 1998.
Jennings N., **Sycara** K., **Wooldridge** M. (1998B). "A Roadmap of Agent Research and Development" in *Autonomous Agents and Multi-Agent Systems*, 1, pp 275-306 (1998), Kluwer Academic Publishers, Boston.
Le Moigne J. L. (1977). "*la théorie du système général – Théorie de la modélisation –*", Ed. PUF, Paris, 1977.
Lercher A. (1995). "*Les mots de la philosophie*". Ed. Belin, Paris, 1985.
Morfaux L. M. (1980). "*Vocabulaire de la philosophie et des sciences humaines*" Ed. Armand Colin, Paris 1980.
Morin E. (1986). "*La méthode III . La connaissance de la connaissance/1*". Ed. du Seuil, Paris, 1986.
Needham R. (1975). "Polithetic classification". *Man*, 10, pp349-369, 1975.
NPD (1990). "*Les Notions Philosophiques - dictionnaire*" Tome I, Vol. dirigé par Auroux S., Encyclopédie philosophique universelle, Ed. PUF, Paris, 1990.
Sanchis E. (1999). "Modular Autonomy for Simple Agents", *Third International Conference on Autonomous Agents, Workshop on Autonomy Control Software*, May 1-5 1999, Seattle (Washington).

Sperber D. (1996). "*La contagion des idées*", Ed. Odile Jacob. Paris, 1996.