Méthodes de développement de systèmes multi-agents

Marie-Pierre Gleizes, Carole Bernon, Frédéric Migeon, Gauthier Picard

Méthodes de développement de systèmes multi-agents

MARIE-PIERRE GLEIZES, CAROLE BERNON, FRÉDÉRIC MIGEON ET GAUTHIER PICARD

Résumé : Les systèmes multi-agents (SMA) ont montré leur pertinence pour la conception d'applications distribuées (logiquement ou physiquement), complexes et robustes. Le concept d'agent est aujourd'hui plus qu'une technologie efficace, il représente un nouveau paradigme pour le développement de logiciels dans lesquels l'agent est un logiciel autonome qui possède un objectif, évolue dans un environnement dynamique et interagit avec d'autres agents au moyen de langages et de protocoles. Souvent, l'agent est considéré comme un objet « intelligent » ou comme un niveau d'abstraction au-dessus des objets et des composants. Les méthodes de développement orientées objet – au vu des différences entre les objets et les agents – ne sont pas directement applicables au développement de SMA. Il est alors devenu nécessaire d'étendre ou de développer de nouveaux modèles, de nouvelles méthodes et de nouveaux outils adaptés au développement de systèmes multi-agents. L'objectif de cet article est d'établir la spécificité du paradigme multi-agent, de donner un aperçu du processus de développement d'un SMA au travers de la méthode ADELFE et de donner les caractéristiques des principales méthodes de conception de SMA en donnant les caractéristiques essentielles de chacune.

Mots clés : Systèmes multi-agents, méthodes de développement, processus de développement, méta-modèles

1. INTRODUCTION

L'augmentation de la puissance des machines, les performances des réseaux de communication, l'avènement du Web sont à l'origine de demandes d'applications toujours plus complexes dans lesquelles la distribution est inhérente. Les données et le contrôle sont logiquement ou physiquement distribués mais la distribution peut aussi être sémantique nécessitant ainsi de définir des ontologies pour permettre une meilleure communication. Le système à concevoir est souvent ouvert et permet ainsi à des éléments le constituant d'intégrer ou de partir du système. Il doit évoluer dans des environnements dynamiques.

La technique des systèmes multi-agents permet de répondre aux demandes provenant de telles applications. Un système multi-agent est défini comme un macro-système constitué d'agents autonomes qui interagissent dans un environnement commun pour réaliser une activité collective cohérente [12]. Un agent est une entité physique ou virtuelle autonome, située dans un environnement et capable de réaliser des actions flexibles et autonomes dans cet environnement dans le but d'atteindre ses objectifs [32]. Il est en général réactif c'est-à-dire qu'il maintient des interactions avec son environnement et répond aux changements de cet environnement pour satisfaire ses buts. Il est proactif ; c'est-à-dire qu'il peut générer et atteindre des buts. Il n'est pas dirigé seulement par les événements mais prend aussi des initiatives pour satisfaire ses buts. Finalement, il est social c'est-à-dire qu'il a l'aptitude à interagir avec les autres agents via des langages de communication afin de satisfaire ses buts [21].

Le paradigme multi-agent permet, d'une part, de faciliter la compréhension et la modélisation des systèmes complexes. Le paradigme agent est un nouveau niveau d'abstraction qui permet d'exprimer une application en termes d'agents autonomes qui jouent des rôles et rendent des services dans une organisation. Par exemple, un système de

simulation du système ferroviaire [11] peut être facilement modélisé avec des agents trains, des agents régulateurs, des agents gares… Il permet, d'autre part, de fournir de nouveaux paradigmes de résolution de problèmes complexes où le concepteur n'a plus à programmer le processus de résolution (comme, par exemple, l'écriture de l'algorithme) mais il doit concevoir les agents et leur donner les moyens d'interagir entre eux mais aussi avec l'environnement du système et le processus de résolution émerge de l'auto-organisation des agents.

La conception multi-agent requiert donc des méthodes associées. Les premières méthodes ont vu le jour dans les années 1995 mais, c'est depuis les années 2000 que les recherches dans ce domaine ont été très nombreuses [1, 2, 18, 22]. On distingue trois grandes familles de méthodes : les plus nombreuses issues de l'ingénierie du logiciel et qui s'inspirent des méthodes orientées objet (figure 1), celles qui sont issues de l'ingénierie des connaissances et les approches issues de plates-formes (figure 2).

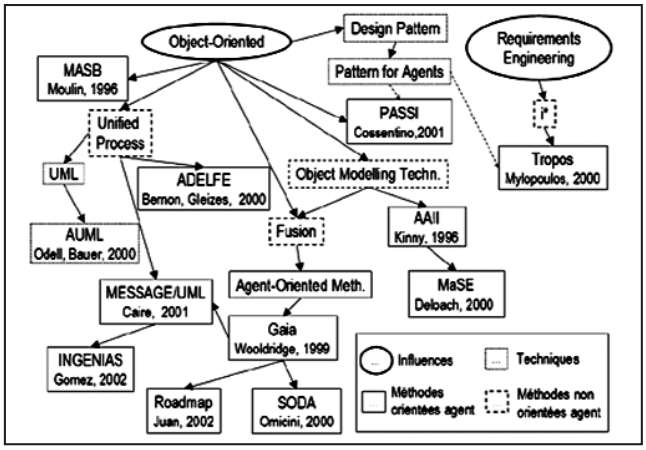

Figure 1 – Méthodes issues des méthodes orientées objet

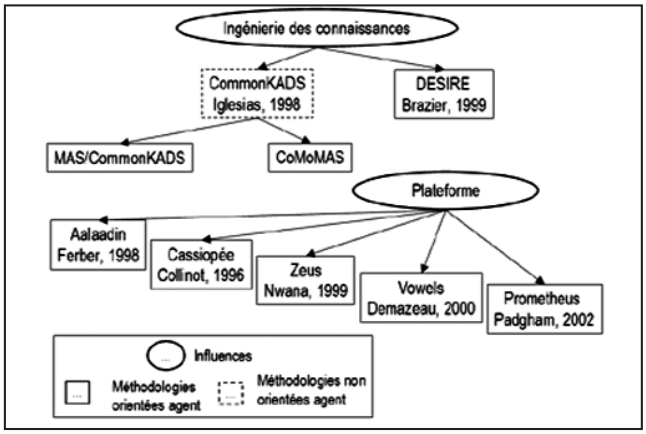

Figure 2 – Méthodes issues de l'ingénierie des connaissances ou de plates-formes

Dans la suite de l'article nous présentons les spécificités et activités caractéristiques des principales méthodes de développement multi-agent. Ces activités sont conduites lors de processus spécifiques à chaque méthode, présentés dans la section 2, et reposent, pour la plupart, sur des méta-modèles propres, présentés dans la section 3. Les activités d'analyse sont décrites dans la section 4, la conception dans la section 5 et enfin l'implémentation dans la section 6. Ces activités représentent le cœur des processus de développement multi-agent à l'heure actuelle. La section 7 conclut sur les enjeux de ce domaine.

2. PROCESSUS DE DÉVELOPPEMENT

La plupart des méthodes de développement de systèmes multi-agents s'inspirent de l'ingénierie orientée objet, et il est donc naturel que les cycles de vie et processus utilisés dans la conception de tels systèmes soient de types classiquement utilisés en conception modulaire : cascades, itératifs, en V, en spirales [5]. De plus, les méthodes multi-agents intègrent, pour la grande majorité, les grandes phases de développement que sont l'analyse des besoins, la conception de l'architecture (ou analyse), la conception détaillée, le développement (ou implémentation) et le déploiement, même si elles ne sont pas clairement identifiées en tant que telles.

Par exemple, ADELFE [3, 27] et INGENIAS [16] reposent sur le processus unifié (Unified Process, UP [20]) et intègrent à ce processus de nouvelles étapes et définitions de travaux spécifiques au domaine multi-agent, alors que PASSI [7] et ASPECS [15] reposent sur un processus itérable original. Certaines méthodes ne couvrent qu'une partie du processus, comme GAIA [33, 35] ou SODA [25] qui se concentrent exclusivement sur l'analyse des besoins et la conception, ou Prometheus [26] et MaSE [31] qui n'abordent pas l'analyse des besoins. Généralement, les méthodes ne proposent pas de processus allant jusqu'au déploiement, s'appuyant sur le principe que les modèles multi-agents sont plus du niveau de l'analyse (notamment pour Tropos [6], issue de l'ingénierie des besoins) et de la conception, exception faite de PASSI ou INGENIAS qui couvrent tout le cycle de développement.

Outre les phases couvertes par les processus, il est aussi intéressant d'identifier ce que les processus proposent en termes de livrables, de gestion de la qualité et de directives de conduite. Les méthodes reposant sur le processus unifié, comme ADELFE ou INGENIAS, héritent de sa gestion des risques et de la qualité de la production. Certaines méthodes ont pris un soin particulier à la définition des processus et des productions, en utilisant notamment des formalismes dédiés comme le SPEM. Ainsi ADELFE et PASSI ont été les premières méthodes à proposer une formalisation SPEM de leur processus, augmentant ainsi la lisibilité de leur suivi et livrables.

3. MÉTA-MODÈLE

Quelles que soient les phases couvertes par ces méthodes, l'ingénieur y exprime ses réflexions en utilisant les concepts définis dans le langage de

modélisation associé. Ainsi, pour des systèmes multi-agents, est-il important de définir la nature des agents et du système associés à la méthode. Même si, comme nous l'avons fait en introduction, une définition consensuelle peut être donnée, elle n'est pas suffisamment précise pour être appliquée pendant les phases de modélisation. Il est alors nécessaire de disposer de moyens de définir des langages de modélisation spécifiques.

Depuis la fin des années 1990, l'Ingénierie Dirigées par les Modèles (IDM) permet de recentrer les processus de développement sur les modèles, et non plus sur le code [10]. Les outils issus de ce domaine permettent de concevoir des langages de modélisation dédiés à un domaine d'expertise ; ils peuvent être associés à des éditeurs graphiques générés automatiquement et permettent d'obtenir des produits dérivés de ce modèle (comme du code) par transformation [28]. Cette pratique nécessite une modélisation (un méta-modèle) des concepts du domaine source et une autre pour le domaine cible (par exemple, la plate-forme de développement).

Cette approche a déclenché la définition de plusieurs méta-modèles dédiés aux domaines particuliers des SMA et de leur implémentation. On trouve, par exemple, AMAS-ML qui s'intègre dans ADELFE pour la définition d'agent coopératif. Le méta-modèle CRIO [15], associé au processus ASPECS, s'inspire de PASSI pour la définition de systèmes holoniques et en reprend trois domaines sémantiques : le domaine pour la plate-forme cible ; le domaine définissant le type particulier de SMA, holoniques pour CRIO ; et enfin, un domaine permettant de capturer les exigences que devra remplir le système et se focalisant sur la notion de rôle. Cette notion de rôle forme également le fondement du méta-modèle de GAIA, qui, en outre, met l'accent sur la notion de service que fournit un agent au sein d'une organisation ainsi que sur les responsabilités et activités liées à un rôle.

Face à la diversité des approches et au nombre grandissant de méta-modèles engendrés, des essais de factorisation ont déjà été étudiés [4]. Cependant, les possibilités offertes par l'IDM orientent actuellement les travaux plutôt vers une définition des méthodologies par parties composables et réutilisables [8, 19].

4. Analyse

De manière classique, au cours d'un processus de développement, la phase d'analyse des besoins, ou spécification fonctionnelle, doit exprimer les fonctionnalités du système à concevoir du point de vue de l'utilisateur. Il s'agit dans un premier temps d'établir un cahier des charges consensuel entre clients, utilisateurs et concepteurs sur ce que le système doit faire, ses limites et ses contraintes. Ces besoins, fonctionnels ou non, sont ensuite organisés et gérés de manière à définir plus précisément le système et son environnement.

L'analyse exprime les besoins des utilisateurs en termes liés au domaine et à ce que le système doit faire. Dans le cas des systèmes multi-agents, lorsque la tâche du système est clairement identifiée, cette phase doit mener à identifier les agents qui y interviendront ainsi que leurs interactions.

La méthode de conception la plus dirigée par les besoins est Tropos dont le processus de développement s'appuie sur les concepts issus de l'ingénierie des besoins. Son analyse des besoins préliminaires et finals s'appuie sur le cadre de modélisation i* [34] dans lequel les intervenants sont représentés comme des acteurs qui dépendent les uns des autres pour atteindre des buts, remplir des tâches ou produire des ressources. Des diagrammes d'acteurs sont étendus de manière incrémentale en analysant chaque but afin d'exprimer les dépendances stratégiques entre acteurs. Les besoins finals étendent ce modèle conceptuel en incluant le système en tant qu'acteur ; les dépendances avec son environnement exprimant les besoins fonctionnels ou non. La conception architecturale définit l'architecture globale du système en l'affinant en termes de sous-systèmes connectés par des flots de données et de contrôle. Des styles architecturaux, s'appuyant sur des structures sociales et intentionnelles, sont proposés et doivent être évalués en fonction d'un certain nombre de critères (sécurité, adaptabilité, disponibilité…) identifiés lors de l'analyse des besoins.

Les méthodes de conception dont le cycle de développement repose sur le « processus unifié » mettent elles aussi l'accent sur les besoins utilisateurs au travers de techniques et de notations issues du domaine de l'objet (cas d'utilisation, diagrammes de séquences…) telles ADELFE, ASPECS, INGENIAS, MaSE ou PASSI. D'autres, comme GAIA ou Message, bien qu'optant pour un processus de développement moins formel adoptent aussi des concepts similaires.

Pour certaines méthodes, l'environnement du système est modélisé dès la phase d'analyse, soit en le qualifiant, comme dans ADELFE, soit en construisant une liste des ressources utilisables par les agents comme dans GAIA, ou bien en identifiant ce que les agents peuvent percevoir, comme dans INGENIAS, ainsi que les actions qu'ils peuvent y effectuer comme dans Prometheus.

Certaines méthodes effectuent une analyse fonctionnelle du système (ADELFE, PASSI, Prometheus), d'autres le font au travers de rôles comme GAIA, INGENIAS, PASSI ou SODA ;

MaSE ou Message identifient ces rôles dès l'analyse des buts du système. Enfin INGENIAS ou Tropos expriment les besoins sous la forme de buts.

Si la plupart des méthodes identifient des agents n'ayant pas de caractéristiques particulières ; d'autres, comme Tropos ou Prometheus, définissent plutôt des agents BDI [29] ou du moins ayant des buts pour INGENIAS ; ASPECS est plutôt orientée vers des systèmes holoniques et enfin, ADELFE est dédiée aux agents coopératifs. Ces agents sont identifiés à partir des rôles dans la majorité des cas, des buts dans Tropos et par les problèmes de coopération dans ADELFE.

Il est à noter qu'au niveau de la phase d'analyse, peu de méthodes s'interrogent sur l'utilité de la technologie agent ou sur le type d'agents visé lors de l'implantation en fonction du problème posé au concepteur. Elles supposent que ce dernier a déjà choisi ce mode d'implantation ; seule ADELFE remet en question l'approche de développement proposée en suggérant, au besoin, d'adopter une autre méthode de développement.

5. Conception

La conception s'attache à décrire le fonctionnement des agents identifiés lors de l'analyse (par le choix d'une architecture d'agent spécifique), l'architecture générale du système multi-agent (organisation) et les différents concepts spécifiques à la méthode utilisée. Ceci se traduit en règle générale, par la définition des classes d'agents et de leurs propriétés, compte tenu de l'héritage fort entre les concepts objets et les concepts agents.

Dans GAIA, par exemple, la conception architecturale aboutit au raffinage des modèles de rôle et d'interaction par l'analyse des structures organisationnelles. Lors de la conception détaillée, les modèles d'agent (détermination des types et instances d'agents) et de services sont spécifiés. Dans INGENIAS, la conception correspond à la définition de la société d'agents composant le système multi-agent, en décrivant les rôles et les protocoles de communication entre agents. Dans Prometheus ou Tropos, la conception concerne l'attribution de responsabilités, identifiées en analysant les fonctionnalités et les flots de données, à des agents qui seront principalement conçus en utilisant une architecture BDI. ADELFE, quant à elle, propose une architecture d'agent coopératif originale, offrant les ressources internes nécessaires à l'obtention de comportements coopératifs et auto-organisateurs.

Les principaux modèles manipulés lors de la conception multi-agent sont les mêmes que dans une conception objet : modèle de classes (parfois décliné comme modèle d'agents), modèle d'interaction et modèle de comportements. La plupart des méthodes commencent par une conception générale de l'architecture du système pour ensuite détailler les composants du système : agents et ressources. C'est lors de cette phase que la plupart des concepts multi-agents sont manipulés (rôle, groupe, organisation, tâche, etc.) et nécessitent parfois des notations spécifiques. Les interactions entre agents sont fréquemment spécifiées en utilisant les langages AUML [23], KQML [14] ou des directives FIPA-ACL (http://www.fipa.org).

La phase de conception résulte ainsi sur une définition claire de l'architecture du système multi-agent, décomposée en agents. Les comportements, les interactions entre les agents ainsi que les ressources nécessaires à l'atteinte de leurs buts sont également spécifiés de manière plus ou moins formelle en fonction des méthodes.

6. Implémentation

À partir de l'architecture définie à la phase de conception, la phase d'implémentation aboutit à la production du code testé. Actuellement, les concepteurs sont aidés dans cette tâche par les techniques d'IDM. Seules quelques méthodes orientées agent ont intégré des outils liés à l'IDM, il s'agit d'INGENIAS, MetaDIMA [17] et Tropos.

Par exemple, dans la méthode ADELFE, à partir de la théorie des AMAS [13], nous avons défini un méta-modèle dans l'optique de fournir un langage de modélisation dédié à un concepteur AMAS. Ce langage, AMAS-ML, doit servir de base à un environnement complet de développement dont il constitue une brique primordiale, celle qui autorise la spécification abstraite du système et des agents qui le composent. À partir de cette spécification et grâce aux outils que nous fournit l'IDM, nous pouvons dériver par transformations successives une API (Application Programming Interface) dédiée à chaque modèle AMAS-ML. La figure 3 présente les différentes étapes du processus de génération d'une API orientée agent spécifique, à partir d'un modèle AMAS-ML. Dans cette figure, chaque flèche représente une phase de transformation (ou de génération). La première de ces flèches correspond à la transformation d'un agent coopératif AMAS-ML en un modèle de microarchitecture µADL. L'objectif de cette transformation est de permettre la génération d'une API dédiée aux agents tels qu'ils sont exprimés dans le modèle AMAS-ML. Il s'agit donc de définir une transformation entre le méta-modèle AMAS-ML et celui de la micro-architecture µADL. Puis, à partir de cette architecture, un squelette de code peut être généré. L'outil MAY (Make Agents by Yourself) permet de construire les types d'agents adaptés au SMA qui doit être développé et de générer une interface de programmation (API) qui leur est dédiée.

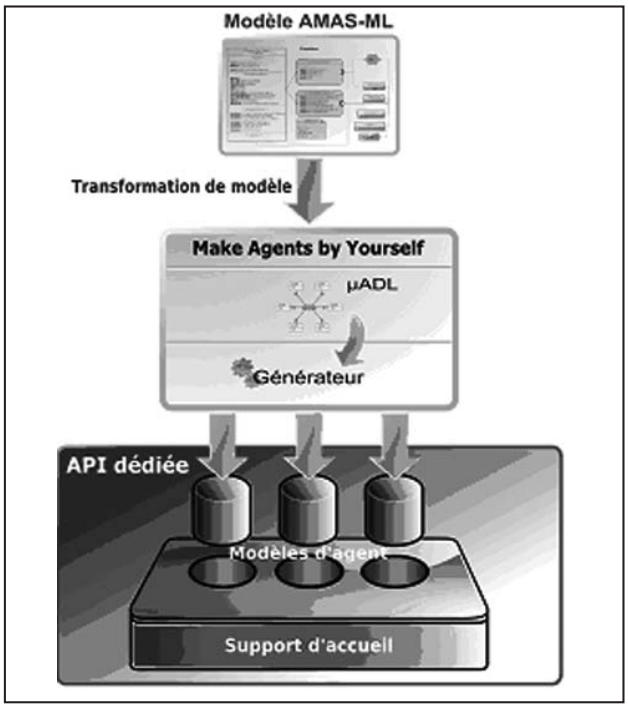

Figure 3. Génération d'API dédiée dans ADELFE

7. Conclusion

De par leur capacité à faciliter la conception d'applications complexes et distribuées, les systèmes multi-agents sont devenus en quelques années un mode de programmation incontournable. Permettre leur diffusion dans un cadre plus industriel qu'académique peut être facilité en fournissant des méthodes et des outils permettant à des ingénieurs de développer de tels systèmes. C'est pourquoi une multitude de méthodes de développement orientées agent a vu le jour ces dernières années. Cet article avait pour objectif de présenter les principales caractéristiques du cycle de développement associé à ces méthodes au travers de quelques exemples, une liste exhaustive ne pouvant être donnée. Toutes les méthodes ne couvrent pas le cycle de vie complet, de l'analyse des besoins au déploiement, la majorité se concentrant sur les étapes d'analyse et de conception, laissant le choix au concepteur d'appliquer des méthodes plus classiques pour implanter, tester et déployer le système multi-agent obtenu. L'argument dans ce cas étant que rien de spécifique ne distingue une application multi-agent d'une application objet classique. Toutefois, d'autres reconnaissent les spécificités des systèmes multi-agents et veulent guider le concepteur, parfois non spécialiste du domaine et confronté à des concepts agents, le plus longtemps possible. Ainsi, des méthodes formelles sont utilisée pas Formal Tropos pour valider les spécifications. ADELFE et PASSI étudient l'apport de la simulation afin d'assister le concepteur lors de la phase de validation du comportement des agents ou du système. La validation notamment en environnement dynamique constitue un des principaux défis du domaine. À la fin de la conception, MaSE, à l'aide de diagrammes spécifiant la structure finale du système, propose une aide au déploiement. Le déploiement est pris en compte de manière indirecte dans INGENIAS ou PASSI.

Toutes ces nombreuses méthodes de conception n'ont pas les mêmes objectifs, certaines sont plutôt généralistes, d'autres sont plutôt dédiées à un type d'agent (BDI, coopératif…) ou de système particulier (holonique, adaptatif…) et des critères de choix doivent alors être fournis à l'ingénieur afin de trouver la méthode idoine. Des comparaisons existent [9, 24, 30] qui peuvent toutefois se révéler, elles aussi, difficiles à utiliser. Donner alors la possibilité au concepteur de construire le processus de développement qui convient le mieux à son problème en combinant des fragments issus d'une base alimentée par les méthodes existantes devient alors une perspective. Et, dans un futur un peu plus éloigné, on pourrait aussi imaginer des processus qui se construisent d'eux-mêmes en fonction de l'environnement (domaine d'application, type d'agents utilisés, complexité du système cible…) dans lequel ils sont situés.

8. Références

[1] F. Arlabosse, M.-P. Gleizes, M. Occello : *Chapitre IV : Méthodes de conception* ; Observatoire Français des Techniques Avancées : Systèmes Multi-Agents, Série ARAGO 29, pp. 137-172, 2004.

[2] F. Bergenti, M.-P. Gleizes et F. Zambonelli (Editors) : *Methodologies and software engineering for agent systems* ; Klüwer, 2004.

[3] C. Bernon, V. Camps, M.-P. Gleizes et G. Picard : *Engineering adaptive multi-agent systems: the ADELFE Methodology* ; dans [18], pp. 172-202.

[4] C. Bernon, M. Cossentino, M.-P. Gleizes, P. Turci et F. Zambonelli : *A study of some multi-agent meta-models* ; Fifth International Workshop on Agent-Oriented Software Engineering (AOSE'04) at AAMAS'04, New York, États-Unis, juillet 2004, P. Giorgini, J. Mueller, J. Odell (Eds.), Springer Verlag, LNCS 3382, pp. 62-77, 2005.

[5] G. Booch : *Conception orientée objets et applications* ; Addison-Wesley, 1992.

[6] J. Castro, M. Kolp et J. Mylopoulos : *A requirements-driven development methodology ;* 13th International Conference on Advanced Information Systems Engineering (CAiSE'01), Stafford, GB, juin 2001.

[7] M. Cossentino : *From requirements to code with the PASSI methodology* ; dans [18], pp. 79-106.

[8] M. Cossentino, S. Gaglio, A. Garro et V. Seidita : *Method fragments for agent design methodologies: from standardisation to*

research ; International Journal of Agent-Oriented Software Engineering , 1(1), pp.91-121, avril 2007.

[9] K.H. Dam et M. Winikoff : *Comparing agent-oriented methodologies ;* Workshop on Agent-Oriented Information Systems (AOIS'03), Melbourne, Australie, 2003.

[10] J.-M. Favre, J. Estublier et M. Blay-Fornarino : *L'ingénierie dirigée par les modèles : au-delà du MDA* ; série Informatique et Systèmes d'Information, Hermès Sciences Lavoisier, janvier 2006.

[11] D. Feillée : *Simulation du système ferroviaire à base d'agents*, dans ce numéro, pp. 22-28.

[12] J. Ferber : *Les systèmes multi-agents* ; Inter-Editions, 1995.

[13] M.-P. Gleizes, V. Camps, J.-P. Georgé et P. Glize : *Conception de systèmes multi-agents à fonctionnalités émergentes*, dans ce numéro, pp. 8-13.

[14] T. Finin, R. Fritzson Don McKay et R. McEntire : *KQML as an agent communication language ;* Third International Conference on Information and Knowledge Management (CIKM'94), ACM Press, novembre 1994.

[15] N. Gaud : *Systèmes multi-agents holoniques : de l'analyse à l'implantation* ; Thèse de l'Université de Franche-Comté et de l'Université de Technologie de Belfort-Montbéliard, décembre 2007.

[16] J. Gomez Sanz et R. Fuentes : *Agent-oriented system engineering with INGENIAS* ; Fourth Iberoamerican Workshop on Multi-Agent Systems (Iberagents'02), 2002.

[17] Z. Guessoum. et T. Jarraya : *Meta-models & model-driven architectures ;* Contribution to the AOSE TFG AgentLink3 meeting, Ljubljana, 2005.

[18] B. Henderson-Sellers et P. Giorgini (Editors) : *Agent-oriented methodologies* ; Idea Group Pub, juin 2005.

[19] B. Henderson-Sellers : *Creating a comprehensive agent-oriented methodology: Using method engineering and the OPEN metamodel ;* dans [18], pp. 368-398, 2005,

[20] I. Jacobson, G. Booch et J. Rumbaugh : *The Unified Software Development Process* ; Addison-Wesley, 1999.

[21] N. R. Jennings : *Agent-oriented software engineering, multi-agent system engineering ;* 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW'99), Valence, Espagne, 1999.

[22] M. Luck, R. Ashri et M. D'Inverno : *Agent-based software development ;* Artech House Publishers, 2004.

[23] J. Odell, H.V. Parunak et B. Bauer : *Representing agent interaction protocols in UML ;* Workshop on Agent-Oriented Software En gineering (AOSE'01), P. Ciancarini, M. Wooldridge (Eds.), LNCS, Springer Verlag, Berlin, pp. 121-140, 2001.

[24] S.A. O'Malley et S. A. DeLoach : *Determining when to use an agent-oriented software engineering ;* Second International Workshop On Agent-Oriented Software Engineering (AOSE'01), pp. 188-205, Montréal, mai 2001.

[25] A. Omicini : *SODA: Societies and Infrastructures in the analysis and design of agent-based systems ;* Workshop on Agent-oriented Software Engineering (AOSE'00), Limerick, juin 2000, Springer, LNCS 1957, pp. 185-193, 2001.

[26] L. Padgham et M. Winikoff : Prometheus: *A pragmatic methodology for engineering intelligent agents* ; at OOPSLA'02, J. Debenham, B. Henderson-Sellers, N. Jennings, J. Odell (Eds.), pp. 97-108, novembre 2002, Seattle.

[27] G. Picard et M.-P. Gleizes : *The ADELFE Methodology – Designing Adaptive Cooperative Multi-Agent Systems*, pp. 157-176, dans [2].

[28] A. Perini et A. Susi : *Automating model transformations in agent-oriented modelling ;* 6th International Workshop on Agent-Oriented Software Engineering (AOSE'05), Utrecht, NL, 25-26 juillet, 2005.

[29] A. S. Rao et M. P. Georgeff : *BDI Agents: from theory to practice ;* First International Conference on Multi-Agent Systems, San Francisco, MIT Press, pp. 312-319, 1995.

[30] A. Sturm et O. Shehory : *A framework for evaluating agent-oriented methodologies* ; 5th Int. Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS'03), LNCS 3030, Springer-Verlag, pp. 94-109, 2004.

[31] M. Wood et S. A. DeLoach : *An overview of the multiagent systems engineering methodology* ; Workshop on Agent-Oriented Software Engineering (AOSE'01), P. Ciancarini, M. Wooldridge (Eds.), LNCS 1957, pp. 1-53, Springer Verlag, Berlin, janvier 2001.

[32] M. Wooldridge et N. R. Jennings (Guest Ed.) : *Special issue on intelligent agents and multi-agent systems* ; Applied Artificial Intelligence Journal, Francis & Taylor, 9(4), juillet/août 1995 et 10(1), janvier/février 1996.

[33] M. Wooldridge, N. R. Jennings et D. Kinny : *The GAIA methodology for agent-oriented analysis and design* ; Journal of Autonomous Agents and Multi-Agent Systems, 3(3), pp. 285-312, 2000.

[34] E. Yu : *Modelling strategic relationships for process reengineering* ; PhD Thesis, University of Toronto, Department of Computer Science, 1995.

[35] F. Zambonelli, N. R. Jennings et M. Wooldridge : *Organizational abstractions for the analysis and design of multi-agent systems* ; Workshop on Agent-Oriented Software Engineering (AOSE'00), P. Ciancarini and M. Wooldridge (Eds), LNCS, Springer-Verlag, 2000.