Published in *Colloque des Doctorants de l'Ecole Doctorale Information et Télécommunication (EDIT'03)*, pages 7-11, Université Paul Sabatier, Toulouse, France, Avril 2003.

ADELFE, une méthode de conception de systèmes multi-agents adaptatifs

ADELFE, a Method for Adaptive Multi-Agent System Designing

Gauthier Picard

Institut de Recherche en Informatique de Toulouse – UMR 5505
Université Paul Sabatier – 118, route de Narbonne, F-31062 Toulouse Cedex
picard@irit.fr

Résumé

Concevoir des logiciels capables de s'adapter à un environnement fortement dynamique impose une méthode de conception rigoureuse qui se distingue de l'approche globale descendante habituelle. L'objet de notre travail est donc de fournir une méthode (processus, notations et outils) pour la conception de systèmes multi-agents adaptatifs (ou AMAS). Ce travail s'inscrit dans le projet RNTL ADELFE qui doit fournir un atelier dont le but est de guider les développeurs au cours de la conception de tels systèmes.

Mots Clefs

Systèmes multi-agents, processus de conception, adaptation, émergence.

Abstract

Designing software able to adapt to dynamical environment induces to follow a rigorous design method that is distinguishable from classical top-down global approaches. Therefore, we aim at providing a method (process, notations and tools) to design adaptive multi-agent systems (or AMAS). This work is part of the ADELFE RNTL project that has to develop a toolkit which goal is to guide developers during the construction of adaptive self-organizing multi-agent systems.

Keywords

Multi-agent systems, design process, adaptation, emergence.

1 Introduction

Depuis le début des années 1990, le domaine des systèmes multi-agents ressent le besoin de produire des méthodes et des modèles pour concevoir des systèmes multi-agents. Ce phénomène n'est pas étonnant, étant donné le caractère prometteur de tels systèmes et l'intérêt de plus en plus grandissant que leur porte l'industrie logicielle. Seulement, avant de pouvoir faire le passage des laboratoires universitaires aux entreprises de développement de logiciels commerciaux, les industriels ont deux prérequis :

- Comprendre ces systèmes au moins de manière fonctionnelle, i.e. comment les construire ;
- S'assurer de leur bon fonctionnement, i.e. valider et tester les systèmes.

Ces deux contraintes impliquent la nécessité de définir des méthodes afin de concevoir des systèmes multi-agents.
Historiquement, le projet de recherche DESIRE [7] fut le premier à fournir une méthode pour la conception de systèmes multi-agents, i.e. un processus, des notations et des outils (DESTool), en manipulant les concepts compositionnels et fonctionnels. En France, la méthode CASSIOPÉE [5] repose sur des principes organisationnels entre agents en présentant la conception de système sur trois niveaux de comportement : élémentaire (comportement pour un agent), relationnel (dépendances entre comportements élémentaires) et organisationnel (la dynamique des comportements). Ceci implique une certaine rigidité des structures construites, et donc l'absence d'auto-organisation. Depuis, de nombreuses méthodes ont fait leur apparition.
Actuellement, TROPOS [4], VOYELLES [6] ou MESSAGE [3] en sont de bons exemples. Après étude de ces différentes méthodes et/ou formalismes, il s'avère que chacune se focalise, implicitement ou explicitement, sur un type d'application ou bien une architecture particulière. Malgré tout, jusqu'à présent aucune n'est dédiée aux problèmes d'adaptation ou d'émergence (*i*). De plus, la question de l'identification des agents, c'est-à-dire, de la détection, parmi un ensemble d'entités identifiées, d'entités particulières ayant des caractéristiques d'agents, est rarement abordée (*ii*). Enfin, bien qu'il soit implicite que chaque méthode possède son domaine d'application particulier, aucune ne se pose la question de la nécessité de son utilisation pour le développement d'un système donné (*iii*). Par conséquent, il est intéressant de répondre à ses problèmes par la définition d'une méthode, ADELFE, vérifiant ces trois critères (*i*, *ii* et *iii*) notamment en se fondant sur la théorie des Systèmes Multi-Agents Adaptatifs (AMAS).

Dans cet article, nous allons présenter la méthode ADELFE [1][2]. Premièrement, une brève présentation de la théorie des AMAS aura pour but de poser le paradigme agent sur lequel repose la méthode, dans la section 2. La section 3 décrit le processus ADELFE, autour des activités et étapes les plus spécifiques. Dans la section 4, les notations ainsi que les outils utilisés seront développés. Enfin, nous conclurons sur les perspectives de ce travail.

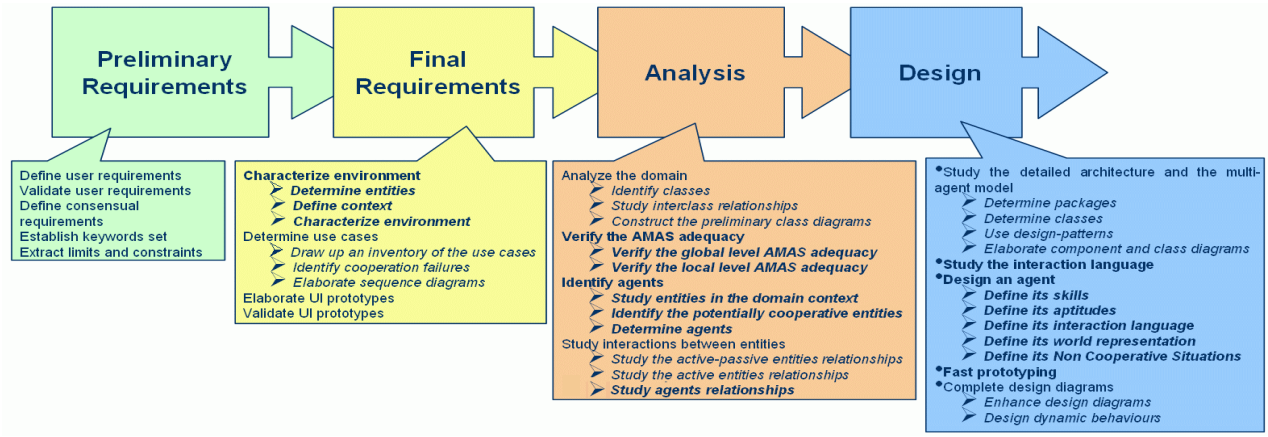

FIG. 1. Le processus de la méthode ADELFE. Les différentes activités et leurs étapes apparaissent sous les phases. Les activités et étapes soulignées par ADELFE sont en gras.

2 ADELFE et la théorie des AMAS

ADELFE est l'acronyme de **A**telier pour le **Dé**veloppement de **L**ogiciels à **F**onctionnalité **É**mergente. En créant ADELFE, notre but est, en effet, d'aider tout développeur – pas seulement les spécialistes des systèmes multi-agents adaptatifs – à concevoir des logiciels à fonctionnalité émergente. La méthode repose donc sur le paradigme agent et plus particulièrement sur la théorie des AMAS (pour Systèmes Multi-Agent Adaptatifs) [8].

Un système multi-agent adaptatif est un système multi-agent qui est capable de changer son comportement en cours de fonctionnement pour l'ajuster dans un environnement dynamique, soit pour réaliser la tâche pour laquelle il a été conçu, soit pour améliorer sa fonction ou ses performances.
La théorie des AMAS garantit qu'un système est fonctionnellement adéquat (il réalise la fonction souhaitée) si les échanges entre les agents qui le composent sont coopératifs, ce qui entraîne les conditions suivantes :

- un signal doit être interprété sans ambiguïté ;
- toute interprétation doit être informative (pas de redite,...) ;
- les conclusions doivent être utiles à autrui.

La particularité de la théorie des AMAS réside dans le fait que l'on ne code pas la fonction globale au sein d'un agent. Grâce à la capacité des agents à s'auto-organiser, le système est capable de s'adapter par lui-même et réalise une fonction qui n'est pas codée dans l'agent. L'objectif de l'auto-organisation est de permettre l'évolution de l'existant en fonction du contexte de façon à assurer la viabilité du système.
Tous les agents que nous considérons sont composés de cinq parties contribuant à leur comportement :

- Les *compétences* : ce qu'est capable de faire l'agent ou quels savoir-faire il peut apporter à la collectivité.
- Une *représentation* de soi, des autres ou de l'environnement : ce que l'agent connaît à propos de lui-même, des autres agents et de son environnement.
- Une *attitude sociale* : elle permet à l'agent de modifier ses interactions avec les autres. Cette attitude sociale se fonde sur ce que nous appelons "coopération" : si un agent détecte une Situation Non Coopérative (SNC), il agit pour revenir à un état dit coopératif.
- Un *langage d'interaction* : ce langage est nécessaire à l'agent pour communiquer de manière directe ou non.
- Les *aptitudes* : elles représentent la capacité qu'un agent possède pour raisonner sur ses représentations et sur ses connaissances ; par exemple, interpréter un signal ou une requête.

3 Le processus ADELFE

Le processus ADELFE s'inscrit dans le processus RUP (Rational Unified Process) [9] et en modifie des activités et/ou étapes en fonction des besoins propres à la technologie AMAS (figure 1). Tout le processus a été spécifié grâce au méta-modèle SPEM (Software Process Engineering Metamodel) qui est un dialecte UML (Unified Modeling Language) [12]. Dans cette section, uniquement les activités et étapes ajoutées ou modifiées du processus RUP sont détaillées.

3.1 Les besoins préliminaires

Les besoins préliminaires forment un ensemble d'activités dédiées à l'expression des besoins par l'utilisateur. Il en résulte un cahier des charges consensuel entre développeur et client. Ces activités ne diffèrent en rien du RUP classique.

3.2 Les besoins finals

Les besoins finals correspondent à l'étude de l'environnement du système à étudier ainsi qu'à ces interactions avec les utilisateurs. Compte tenu que le couplage entre environnement et système dans la théorie des AMAS est une condition nécessaire à l'auto-organisation des agents, l'étape de **caractérisation de l'environnement** est primordiale pour la bonne suite de la conception. L'environnement peut être caractérisé comme étant :

- *accessible* si un agent peut obtenir une information complète, précise et à jour à propos de l'état de son environnement.
- *déterministe* si toute action a un effet unique

garanti – il n'y a aucune incertitude quant à l'état de l'environnement résultant de l'action de l'agent.

- *statique* s'il ne subit aucun changement de son état si ce n'est par les actions exécutées par les agents.
- *discret* s'il n'y a qu'un nombre fixé et fini d'actions et de perceptions possibles sur lui.

Cette caractérisation [13] permet de pointer du doigt les différents problèmes auxquels le système devra faire face dès les phases préliminaires de conception.

Une autre activité, l'**identification des échecs de coopération** entre le système et son environnement, a pour but d'aider le concepteur à détecter les problèmes liés au principe de coopération au sens de la théorie des AMAS. Cette identification s'affinera tout au long du processus et permettra, lors de l'analyse, d'identifier les agents.

3.3 L'analyse

A partir des spécifications obtenues lors des besoins finals, l'analyste peut effectuer une analyse du domaine et identifier les différentes entités en jeu et ainsi construire un diagramme de classe préliminaire.

À partir de ce moment, il devient raisonnable de se poser la question de la pertinence de l'utilisation de la technologie AMAS pour la conception du système à étudier. En effet, toutes les applications ne nécessitent pas son utilisation – si un algorithme efficace est déjà connu pour résoudre le problème par exemple [11]. L'activité **vérification de l'adéquation des AMAS** propose au concepteur un certain nombre de critères afin de l'aider à déterminer l'adéquation des AMAS à deux niveaux :

- Niveau global (système) : « Est-ce que la technologie AMAS est nécessaire pour implanter le système ? »
- Niveau local (composantes du système) : « Est-ce que certaines composantes du système ont besoin d'être modélisées comme des AMAS, i.e. est-ce qu'une décomposition récursive est nécessaire ? »

L'**identification des agents** est aussi une activité caractéristique d'ADELFE n'apparaissant dans aucune autre méthode orientée agent. Elle repose sur une analyse des dépendances fonctionnelles et des possibles échecs de co-opération à partir des cas d'utilisation jusqu'à leurs diagrammes de séquence. En traçant les problèmes de co-opération du niveau global au niveau des composantes ainsi qu'en étudiant leurs propriétés (autonomie, interaction, …), il est possible de déterminer quelles sont les entités susceptibles d'être des agents au sens de la théorie des AMAS.

Enfin, l'**étude des relations inter-agents** permet de modéliser les relations entre agents à partir de diagrammes de séquences pour les aspects événementiels ou bien à partir de diagrammes de classes pour les aspects taxonomiques ou organisationnels statiques.

3.4 La conception

La conception doit définir une architecture détaillée pour le système en termes de paquetages, de sous-systèmes, d'objets et d'agents. Ce sont des activités importantes d'un point de vue multi-agent du fait qu'une caractérisation récursive du système multi-agent est obtenue à ce stade. Cela implique différents processus de conception pour les différents niveaux du système identifiés.

Une architecture agent permet au concepteur de doter les agents de modèles de compétences, de représentations du monde (de lui, des autres et de son environnement), d'aptitudes, d'une attitude sociale et d'un langage d'interaction. À ce stade, une liste des situations non coopératives est établie. Le modèle de conception est composé de classes (dans des diagrammes de classes), de classes d'agents et d'interactions. Une conception détaillée résulte de diagrammes état-transition (pour modéliser le comportement des objets et des agents) et de diagrammes d'activités UML.

À ce stade, les agents sont identifiés. Une activité importante est alors d'**étudier les langages d'interaction** entre ces agents. Cela correspond à une spécification grâce à des diagrammes de protocoles AIP (voir section 4.2) des protocoles qu'utiliseront les agents afin de communiquer. Ces protocoles sont génériques dans le sens où ils ne sont pas rattachés à une classe, mais seulement à des rôles. Ce n'est qu'à l'activité suivante qu'ils seront instanciés à des classes d'agents.

La **conception des agents** correspond à la spécification des différentes composantes d'un agent (voir section 2) : les compétences, les aptitudes, le langage d'interaction, les représentations et la coopération. La **détermination du langage d'interaction** correspond à l'attribution d'un ou de plusieurs protocoles AIP qui sont alors instanciés à l'agent pour un rôle donné.

Enfin, avant d'achever la conception, i.e. finir les spécifications, le concepteur a la possibilité de **prototyper rapidement** les agents afin de détecter un manque quelconque de compétence, d'aptitude ou autre composante par une simulation. Cette activité peut être en interaction permanente avec l'activité précédente afin de concevoir les agents de manière « vivante ».

Actuellement, ADELFE s'arrête à la fin de la conception, laissant libres les concepteurs d'utiliser des méthodes de leur choix pour l'implémentation, et le déploiement, en prenant celle du RUP par exemple. Dans le futur, ADELFE proposera la couverture complète du processus de développement de logiciels des besoins à la maintenance.

4 Notations et outils de support d'ADELFE

La méthode ADELFE, ainsi munie d'un processus, repose aussi sur des notations spécifiques et des outils de spécification. Dans cette section, nous allons discuter de la notation AUML utilisée pour la description de protocoles entre agents après avoir introduit les stéréotypes propres à ADELFE. Enfin, nous présenterons les outils utilisés lors du processus.

4.1 Stéréotypage UML

Afin de cadrer le travail de spécification du concepteur et

de garantir la bonne utilisation des concepts propres à la théorie des AMAS, des stéréotypes UML ont été définis. À chaque stéréotype, un jeu de règle de bonne utilisation du stéréotype a été établi en OCL. Les stéréotypes ADELFE sont les suivants :

- `<<cooperative agent>>` : caractérise une classe comme étant agent au sens de la théorie des AMAS. Un exemple de règles d'utilisation est qu'une classe ainsi stéréotypée doit obligatoirement posséder des règles de coopérations (méthodes ou attributs stéréotypés `<<cooperation>>`) afin de représenter un agent coopératif. Tous les stéréotypes suivants ne peuvent apparaître que dans une classe stéréotypée `<<cooperative agent>>` ;
- `<<cooperation>>` : caractérise des méthodes ou attributs gérant le comportement coopératif de l'agent ;
- `<<perception>>` : caractérise des méthodes ou attributs gérant la perception de l'agent (des capteurs par exemple) ;
- `<<action>>` : caractérise des méthodes ou attributs gérant les actions de l'agent (actionnement de roues par exemple) ;
- `<<skill>>` : caractérise des méthodes ou attributs gérant les compétences de l'agents (une base de règles par exemple);
- `<<aptitude>>` : caractérise des méthodes ou attributs gérant les aptitudes de l'agent (un moteur d'inférences par exemple);
- `<<interaction>>` : caractérise des méthodes ou attributs gérant les interactions de l'agent (les méthodes correspondant au rôle et aux événements apparaissant dans les protocoles AIP associés à l'agent);
- `<<world representation>>` : caractérise des méthodes ou attributs gérant les croyances de l'agent ;
- `<<characteristic>>` : caractéristiques intrinsèque à l'agent (par exemple couleur, numéro,…).

4.2 La notation AUML

Introduite par James Odell, la notation AUML (Agent UML) a pour but de spécifier les interactions entre agents grâce à des diagrammes de séquences modifiés, appelés diagrammes AIP (Agent Interaction Protocole) [10].
La figure 2 présente un exemple de diagramme AIP.

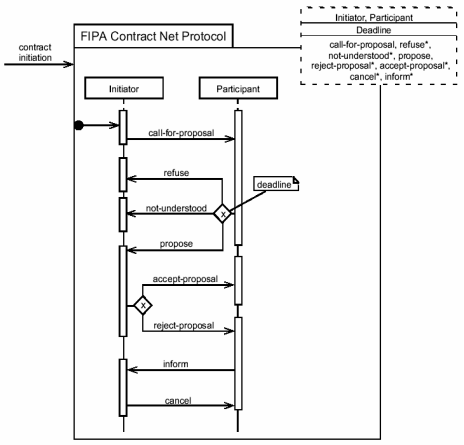

FIG. 3. Un exemple de protocole AIP correspondant au « Contract Net » défini par la FIPA.

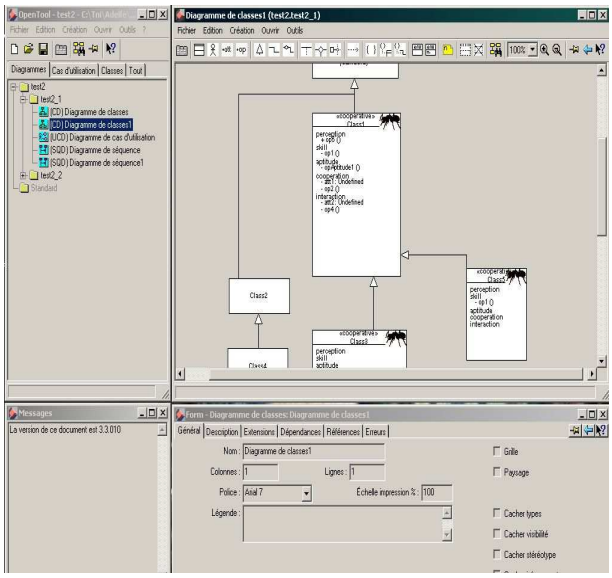

FIG. 2. L'outil OpenTool de spécification UML/AUML utilisé par ADELFE.

Contrairement au diagramme de séquence classique, les lignes de vie correspondent ici à des rôles d'agents et non à des instances de classes. Cette notion de rôles est orthogonale à la notion de classes. Ainsi, il est possible d'attribuer à un agent un ou plusieurs rôles.
L'autre notion importante des protocoles AIP est la possibilité de « brancher » les messages grâce à des nœuds OR, XOR ou AND. Cela permet de préciser l'échelle de la diffusion des messages, et introduit une notion d'indéterminisme caractéristique du paradigme agent.

4.3 OpenTool

Afin de construire les différents modèles apparaissant dans la méthode ADELFE, nous préconisons l'utilisation de l'outil OpenTool© (figure 3) d'édition UML développé par TNI-Valiosys® pour trois raisons majeures :

- il intègre déjà les digrammes AIP de la notation AUML ;
- il utilise et vérifie automatiquement la bonne utilisation des stéréotypes ADELFE ;
- il possède une fonctionnalité de simulation utilisable pour le prototypage rapide.

Les deux premières fonctionnalités ont été ajoutées pour prendre en compte la conception de AMAS. La dernière permet de vérifier le comportement des agents par simulation des machines à états, à partir d'instance de classes apparaissant dans un diagramme de collaboration d'initialisation. Pour cela, la transformation de diagrammes de protocoles AIP en machines à états correspondant aux rôles des protocoles a été automatisée. Ainsi, les machines à états peuvent être simulées pour vérifier le bon déroulement des dialogues entre agents.

4.4 Un processus interactif

Un autre outil utilisé lors du processus ADELFE est un outil interactif de suivi de la méthode permettant aux

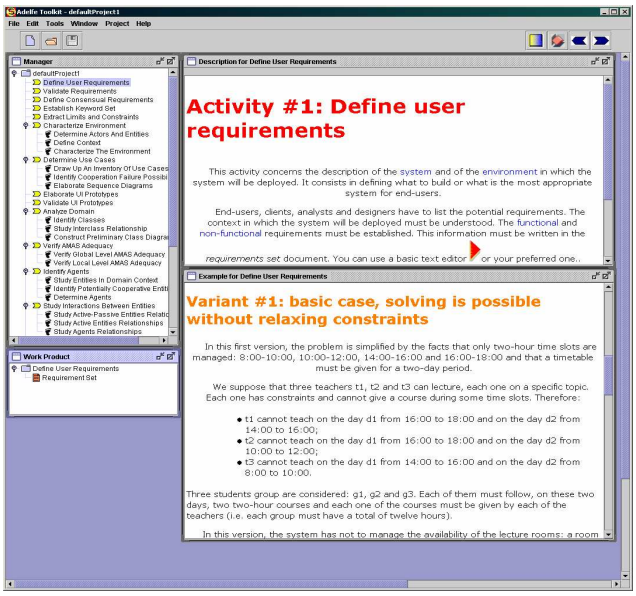

FIG. 4. La plate-forme pour le suivi interactif du processus.

concepteurs de facilement retrouver la description du processus, les documents délivrables produits lors de la conception, des accès directs aux outils d'OpenTool en fonction de l'état d'avancement du projet (figure 4). Cette plate-forme communique avec et pilote OpenTool et d'autres logiciels pour obtenir des diagrammes ou de la documentation générée automatiquement.

5 Conclusion et perspectives

Dans cet article, nous avons présenté la méthode ADELFE sous ses trois dimensions : processus, notations et outils. Cette méthode est dédiée à la construction des systèmes multi-agents adaptatifs (AMAS) en se basant sur les notions d'auto-organisation par coopération et d'émergence. Le processus est basé sur le RUP (Rational Unified Process) et y apporte des modifications ou des ajouts d'activités et d'étapes. Pour prendre en compte la problématique des AMAS, nous avons dû aussi définir des notations spécifiques et réutiliser AUML. Enfin, les outils utilisés OpenTool et la plate-forme ADELFE, ont dû être modifiés ou développés en fonction de ces nouvelles exigences.

ADELFE est un projet en cours de développement dont les perspectives sont les suivantes :

- Validation de la méthode ainsi que des outils associées par la société ARTAL Technologies ®, par le développement d'une application de gestion et de construction interactive et auto-organisée d'emplois du temps (ETTO, *Emergent Timetable Organization*) ;
- L' « agentification » du processus afin de rendre la méthode plus flexible en fonction des applications. Le principe est le suivant : à partir du modèle SPEM qui nous à permis de spécifier le processus, nous allons construire un système multi-agent composé d'agents activités et étapes. L'auto-organisation de ces agents permettra d'obtenir un processus en fonction de l'utilisateur et de l'application qu'il désire concevoir. Ce système multi-agent sera ensuite intégré à la plate-forme ADELFE.

Références

[1] Bernon C., Gleizes M-P., Peyruqueou S., Picard G. – ADELFE, a Methodology for Adaptive Multi-Agent Systems Engineering – *In Pre-proceedings of Third International Workshop on Engineering Societies in the Agents World (ESAW-2002)*, Madrid, 16-17 Sept. 2002.

[2] Bernon C., Gleizes M-P., Picard G., Glize P. – The Adelfe Methodology For an Intranet System Design – In *Fourth International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2002)*, Toronto (Ontario, Canada) at CAiSE'02, 27-28 May, 2002.

[3] Caire G., Leal F., Chainho P., Evans R., Garijo F., Gomez J., Pavon G., Kearney P., Stark J. & Massonet P. – Agent Oriented Analysis using MESSAGE/UML – *In Proc. of the 2nd International Workshop on Agent-Oriented Software Engineering, AOSE'01*. LNAI, vol. 2222. Springer Verlag, 2002, pp. 119-135.

[4] Castro J., Kolp M. & Mylopoulos J. – A Requirements-driven Development Methodology – *In Proceedings of the 13th International Conference on Advanced Information Systems Engineering (CAiSE'01), 2001*. LNAI, vol. 2068. Springer Verlag, 2002, pp. 108-123.

[5] Collinot A.,Drogoul A. and Ploix L. – Application de la méthode Cassiopée à l'organisation d'une équipe de robots – In *Intelligence Artificielle Distribuée et Systèmes Multi-Agents*, J.P. Muller et J. Quinqueton (Eds), pp. 136-152, Hermès, Paris, 1996.

[6] Costa A.C.R., Demazeau Y. – Toward a formal model of multi-agent systems with dynamic organizations – *In Proceedings of the Second International Conference on Multi-Agent Systems,* AAAI Press/ The MIT Press, Kyoto, 1996, pp. 431.

[7] Dunin-Keplicz B. and Treur J. – Compositional Formal Specification of Multi-Agent Systems – In M. Wooldridge, N. Jennings (eds.), *Intelligent Agents, Proc. of the ECAI'94 Workshop on Agent Theories, Architectures and Languages*, Lecture Notes in AI, vol. 890, Springer Verlag, pp. 102-117, 1995.

[8] Gleizes M-P., Camps V., Glize P. – A Theory of emergent computation based on cooperative self-organization for adaptive artificial systems – *In Fourth European Congress of Systems Science*, Valencia, 1999.

[9] Jacobson I., Booch G. and Rumbaugh J. – *The Unified Software Development Process: the complete guide to the Unified Process from original designers* – Addison-Wesley, ISBN-0-201-57169-2, 1999.

[10] Odell J., Van Dyke Parunak H. and Bauer B. – Representing Agent Interaction Protocols in UML – *In Proceedings of the Second International Workshop On Agent Oriented Software Engineering (AOSE'01)*, Paolo Ciancarini and Michael Wooldridge eds., LNAI, vol. 2222. Springer Verlag, 2002, pp. 1-16.

[11] O'Malley S. A. and Deloach S. A. – Determining When to Use an Agent-Oriented Software Engineering Paradigm – *In Proceedings of the Second International Workshop On Agent-Oriented Software Engineering (AOSE'01), Montreal, Canada, May 29th 2001*. LNAI, vol. 2222. Springer Verlag, 2002, pp. 188-205.

[12] Object Management Group– *Software Process Engineering Management : Software Process Engineering Metamodel (SPEM), Version 1.0* – OMG, 2002

[13] Russel S. and Norvig P. – *Artificial Intelligence: a Modern Approach* – Prentice-Hall, 1995.