International Research Journal of Computer Science (IRJCS)
Issue 5, Volume 2 (May 2015)
ISSN: 2393-9842
www.irjcs.com

Multi-agent systems engineering: A survey and analysis

Jihad CHAKER, Mohamed KHALDI and Souhaib AAMMOU
LIROSA, Faculty of Sciences, Abdelmalek Essaadi University, MOROCCO

Abstract— Recently, we have witnessed a significant emergence of multi-agent systems in the field of artificial intelligence (AI). Faced with this situation, new needs appear throughout the development processes as well management and quality. A wide range of agent-oriented methodologies have been proposed and to better understand the application of these theories, a research area called Agent Oriented Software Engineering (AOSE) is fully dedicated to research of best practices to guide the user in the development of multi-agent systems and to provide means and tools to verify, validate and test the functionalities of the system. This article is aiming to study different methodologies for the design of multi-agent systems (MAS). Thus, we will be presenting both sides the positives and the negatives. Moreover, the study will cover the most common methods, based on several criteria: documentation, coverage of life cycle, dedicated tools, citation frequency, standard integration and Areas coverage.

Keywords— Multi-agent system, Agent Oriented Software Engineering, Design methodology, Life cycle, UML

I. INTRODUCTION

In the evolution of programming approaches, modular programming replied to the increase in terms of complexity, application memory requirements, and the need to reusability but the modules are under external control by appeal instructions. Which leads to the decomposition of programs into objects, but these objects are still liabilities. The logical step was to agent-oriented programming that offers to design agents as active objects, autonomous with its own behavior. The engineering object is the main source of influence for agent-oriented methods, in addition, knowledge engineering, requirements engineering. Another remarkable source of influence, are multi-agent platforms for simulation and development.

According to Booch [1], a methodology is a collection of methods applied across the software development life cycle and unified by some general, philosophical approach. Methods are important for several reasons. Foremost, they instill a discipline into the development of complex software systems. They define the products that serve as common vehicles for communication among the members of a development team. Additionally, methods define the milestones needed by management to measure progress and to manage risk. Shehory [2] specifies: as part of AOSE research we can find methodologies and modeling techniques that present concepts of software agent modeling at different levels, i.e., different lifecycle stages. In the “Handbook on Agent-Oriented Design Processes” [3], a methodology for the design of MAS, has several constituent parts, including a full life cycle process, a comprehensive set of concepts, a set of rules, heuristics and guidelines, a set of metrics, information on quality assurance, a set of coding and other organizational standards, techniques for reuse and project management procedures .

As methodologies development of MAS are many and evolving rapidly, it is necessary to make a comparison to help the designer to choose the most suitable methodology for the given problem or to exhibit the essential differences between different methodologies.

This paper is divided into six sections: Section 2 presents the basic concepts related to agents and multi-agent systems. Section 3 describes the criteria for this analysis. Section 4 presents the proposed comparison matrix followed by summaries of the different methods that we judge comparable. Section 5 shows the results of the analysis using the Kiviat diagram followed by a discussion. Finally, Section 6 concludes the paper.

II. OVERVIEW OF MULTI-AGENT SYSTEMS

A. Agent: definition and properties

An agent is an autonomous and intelligent, real or abstract entity, who is capable of acting on itself and on her environment in a multi-universe agent also, she can communicate with the other agents and whose behavior is the consequence of her observations, her knowledge and the interactions with the other agents [4]. Russell [5] summarizes: an agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors.

Jennings and Wooldridge [6] define the agent as a computer system operating in an **environment** which is able to act **independently** and **flexibly** in this environment in order to achieve its objectives. In what follows we show the concepts: “environment”, “independently” and “flexibly”.

In the online course “Intelligent Agents” [10], the authors see an agent is located in an environment. To model the structure of the agent must have a model of the environment. The environment can be seen as being in a state “e” from a set of states E = {e1, .., e, ..} The environment can change its state either spontaneously or as a result of the actions of the agent. The main distinctions about the types of environments:

- Accessible environment or inaccessible.
- Deterministic and non-deterministic environment
- Static or dynamic environment.
- Discrete or continuous environment.

The agent is capable of acting independently means that the agent is able to act without the intervention of a third party (human or agent) and controls its own actions and its internal state.

The flexibility of the agent means it is able to respond in time (it must be able to perceive their environment and develop a response within the required time), proactive (it should require behavior while being able to take the initiative at the "right" moment) and social (it must be able to interact with other agents (software and human) when the situation requires to complete its tasks or assist these agents to do their own).

B. Classification of agents

Ferber [4] is based on the design methodology agents or as "smart" entities, ie able to solve problems by themselves, or should we assimilate them to be very simple directly reacting to environmental changes. This gives two types:

- Cognitive agents, most of which are intentional, each have a knowledge base containing all information and know-how to help them achieve their task and manage interactions with other agents and their environment. They are able to advance and can plan their behavior.
- Reactive agents are able to anticipate or plan their behavior, it is not necessary that agents are intelligent individually if the system has an intelligent global behavior.

Mill and Chaib-Draa [8] characterize agents by their ability to solve problems: a reactive agent, an intentional agent and one social.

In the book "Ontology Based Multi-Agent Systems" [9], the authors propose another classification based on the functions of the agent, they distinguish the following types of agents: Interface Agent, Agent Manager, Executive Agent and intelligent agent.

C. Multi- agent system: definition and properties

A Multi-Agent System is a set of agents operating in a common environment, which means the real world or the virtual world. A Multi-Agent system is defined by Sycara [11] as the emergence of a global behavior produced by a set of interactions between agents to solve problems that individually exceed their reasoning abilities. According to Ferber [10] A Multi-Agent System is a system consisting of:

- *E environment, a space with a generally metric.*
- *A set of objects O. These objects are located, that is to say that, for any object, it is possible, at a given time, to associate a position in E. These objects are passive; they can be perceived, created, destroyed and modified by the agents.*
- *A set A of agents, which are specific objects ($A \subseteq O$), which represent the active entities of the system.*
- *A set of relations R that unite objects (and thus agents) to each other.*
- *A set of operations Op allowing agents of A to perceive, produce, consume, transform and manipulate objects from O.*
- *Operators responsible for representing the application of these operations and the world's reaction to this attempt to change, that the laws of the universe will be called.*

Therefore, a multi-agent system should have the following skills: social organization, coordination, cooperation, negotiation and communication.

D. Communications between agents

The communication between agents is defined by the parts involved in the communication, the place where the communication parts are located, the time in which communication takes place and the language used. A Communication Language Agent must be designed as a high-level language that provides first exchange of mental states and the meaning of the vocabulary [12]. The format for the exchange of knowledge is given by a content language, independent of ACL language. The common vocabulary concerning definitions specified in ontology.

Several attempts at normalization of inter-agent communication were carried out in the multi-agent community in recent years include, communication language KQML[1] and communication language FIPA-ACL[2].

III. CRITERIA ANALYSIS

To appear maturity of the studied methodologies, we have referred to various criteria:

- Standards integration: The purpose of this criterion is to visualize the degree of use of standards in the general process of the methodology, included: UP[3], SPEM[4], UML[5], FIPA, RUP[6], USDP[7], MOF[8]…

[1] KQML: Knowledge Query and Manipulation Language
[2] Foundation for Intelligent Physical Agents (FIPA). Fipa2000 agent specification
[3] UP: Unified Process
[4] SPEM: Software process engineering metamodel specification

- Coverage of Life Cycle: This is precisely coverage steps known in agile methods, eg requirements analysis (This step can also be divided into two stages: preliminary needs and final needs), design architecture, detailed design, development and maintenance.
- Dedicated tools: This criterion is very useful because it focuses on the pragmatic aspect of the method, including IDE[9], platforms and libraries.
- Citation frequency: This aspect is subject to many scientific citations of the official paper of the methodology, based on the results generated by the most popular index as google Scholar[10] and IEEE Xplore Digital Library[11]…
- Documentation: This criterion covers the official documentation of the method, in addition to the tutorials and user guides, especially those aimed at beginners getting started MAS design field.
- Areas of coverage: The aim is to distinguish between generic methodologies and specific methodologies to one or more areas.

IV. SURVEY OF METHODOLOGIES

A. *Classification system*

To give a synthetic vision of the studied solutions, we have positioned the solutions relative to functionalities groups described above. The results are presented as a graph that is originally a Kiviat diagram, here is an example:

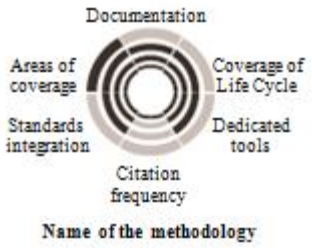

Fig. 1 Example of Analysis results representation

The placement of solutions on functional lines is a subjective placement depending on the richness and originality of the features implemented. However, given the multitude of features and number of approaches, these matrices can be used only point of comparison between these solutions.

B. *List of studied methodologies*

We chose the methodologies commonly used in the field of multi-agent systems engineering, the most cited methodologies in the research, which are still improving especially in scalability. The methods studied are unique or based on other methods. Here are the methods studied [13] - [24]:

1) ADELFE[12]: The name of the methodology ADELFE comes from the French acronym meaning "Toolkit for Designing Software with Emergent Functionalities", this methodology aims to help and guide any designer during the development of an adaptive multi-agent system, it is based on the RUP, UML agent-oriented extensions and some specific steps to adaptive system design.

2) *ANEMONA[13]*: Is a multi-agent system (MAS) methodology for holonic manufacturing system analysis and design, based on HMS requirements. ANEMONA defines a mixed top-down and bottom-up development process, and provides HMS-specific guidelines to help the designer in identifying and implementing holons. In ANEMONA, the specified HMS is divided into concrete aspects that form different "views" of the system.

3) *ASPECS[14]:* Is based on a holonic organisational metamodel and provides a step-by-step guide from requirements to code allowing the modelling of a system at different levels of details using a set of refinement methods. This paper details the entire aspecs development process and provides a set of methodological guidelines for each process activity. A complete case study is also used to illustrate the design process and the associated notations.

---

[5] UML: Unified Modeling Language
[6] RUP: Rational Unified Process
[7] USDP: Unified Software Development Process
[8] MOF: Meta Object Facility
[9] IDE: Integrated Development Environment
[10] http://scholar.google.com/
[11] http://ieeexplore.ieee.org/Xplore/home.jsp
[12] ADELFE: A methodology for adaptive multi-agent systems engineering
[13] ANEMONA: A multi-agent methodology for holonic manufacturing systems
[14] ASPECS: An agent-oriented software process for engineering complex systems

4) Aspecs uses UML as a modelling language. Because of the specific needs of agents and holonic organisational design, the uml semantics and notation are used as reference points, but they have been extended by introducing new specific profiles.

5) *GAIA*[15]*:* Is both general, in that it is applicable to a wide range of multi-agent systems, and comprehensive, in that it deals with both the macro-level (societal) and the micro-level (agent) aspects of systems. Gaia is founded on the view of a multi-agent system as a computational organisation consisting of various interacting roles. We illustrate Gaia through a case study (an agent-based business process management system).

6) *GORMAS*[16]*:* This guideline covers the requirement analysis, the structure design and the organization-dynamics design steps, in which software designers mainly specify the services that the system under study is requested to offer, the internal structure of this system and the norms that control its behavior, taking into account the specific features of open multi-agent systems.

7) *INGENIAS*[17]*:* It combines agent research results with concepts and methods established in MESSAGE/UML. The result is a development process in the line of conventional software engineering processes, like object oriented software development paradigm or structured paradigm. INGENIAS defines deliverables and default activities to help in planning effort along a project. INGENIAS also provides with tools that facilitate the production of these deliverables. As a result, INGENIAS can be considered a valuable alternative in the Agent Oriented Software Engineering research field.

8) *O-MASE*[18]*:* Is a customisable agent-oriented methodology based on consistent, well-defined concepts supported by plug-ins to an industrial strength development environment, agentTool III.

9) *PASSI* [19]*:* Is a step-by-step requirements-to-code method for developing multi-agent software that integrates design models and philosophies from both object-oriented software engineering and MAS using UML notation.

10) *ROMETHEUS*[20]*:* Is intended to be practical; in particular, it aims to be complete and detailed, and to be usable by industrial software developers and undergraduate students.

11) *ROMAS*[21]*:* An agent-oriented methodology that guides developers on the analysis and design of systems of this kind. Contracts and norms are used to formalize the normative context and the interactions. This methodology has been described using the template proposed by the FIPA Design Process Documentation and Fragmentation Working Group.

12) *SODA*[22]*:* Is based on the core notion of task, SODA promotes the separation of individual and social issues, and focuses on the social aspects of agent-oriented software engineering. In particular, SODA allow the agent environment to be explicitly modelled and mapped onto suitably-defined agent infrastructures.

13) *TROPOS*[23]*:* Is based on two key ideas. First, the notion of agent and all related mentalistic notions (for instance goals and plans) are used in all phases of software development, from early analysis down to the actual implementation. Second, Tropos covers also the very early phases of requirements analysis, thus allowing for a deeper understanding of the environment where the software must operate, and of the kind of interactions that should occur between software and human agents. The methodology is illustrated with the help of a case study. The Tropos language for conceptual modeling is formalized in a metamodel described with a set of UML class diagrams.

[15] GAIA: A methodology for agent-oriented analysis and design
[16] GORMAS: An Organizational-Oriented Methodological Guideline for Open MAS.
[17] INGENIAS: An agent oriented software engineering methodology for Multi-Agent Systems development.
[18] O-MASE: A customisable approach to designing and building complex, adaptive multi-agent systems.
[19] PASSI: Process for Agent Societies Specification and Implementation.
[20] Prometheus: A Practical Agent-Oriented Methodology.
[21] ROMAS : Regulated Open multiagent Systemsbased on contracts.
[22] SODA : Societies in Open and Distributed Agent spaces.
[23] Tropos : An agent-oriented software development methodology.

V. ANALYSIS RESULTS AND DISCUSSION

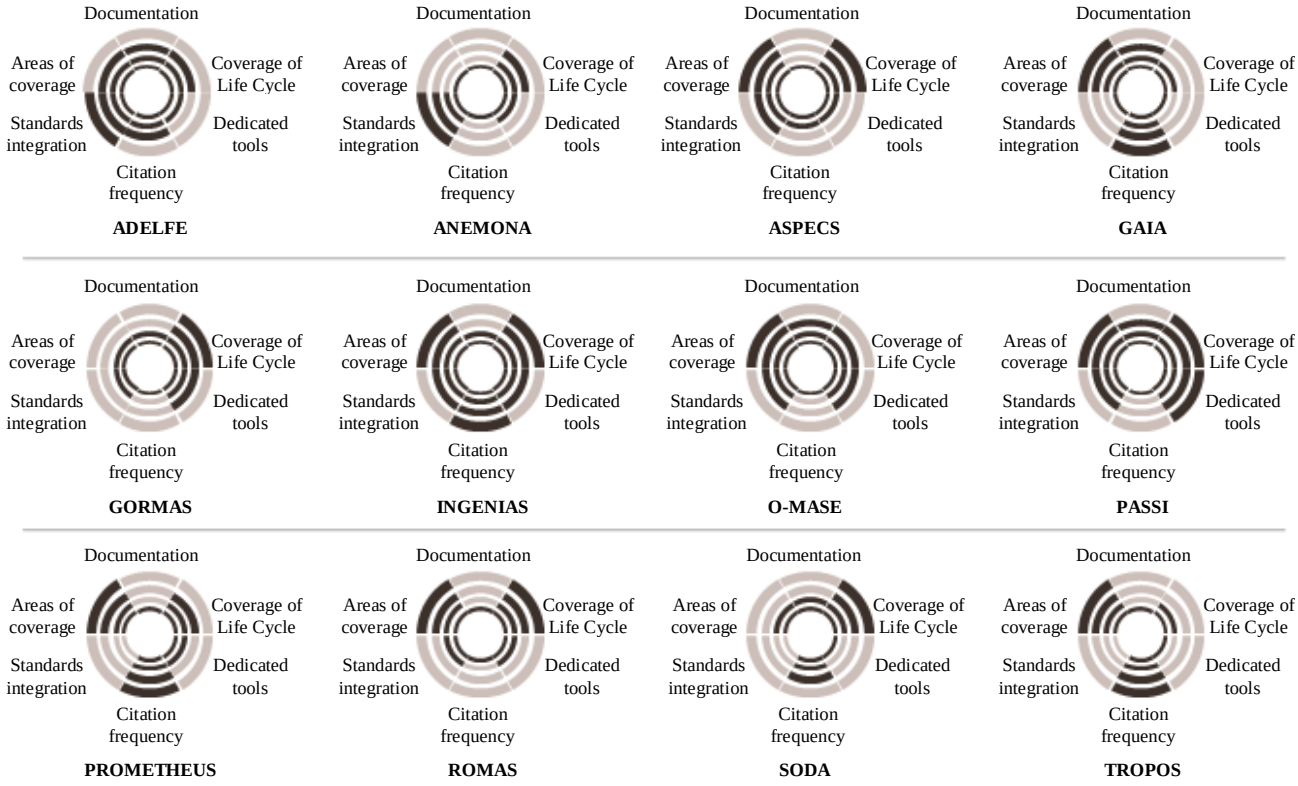

Fig. 2 Analysis results

The analysis results above illustrate the strengths and weaknesses of each methodology, consider the example of the methodology ADELFE, it incorporates a large number of standard versus other, include: UP, FIPA SPEM and UML .The ASPECS methodology integrates least standards, specifically: SPEM, UML, same thing for INGENIAS methodology. By against, they exist the methodologies that not rich in this criterion: the case of PROMETHEUS, SODA and TROPOS. Another criterion that seems to us very important when choosing a design methodology: Areas of coverage. There are some methodologies that are generic as ASPECS, GAIA, PASSI and TROPOS. Other methodologies are specific to an area, for example the methodology ANEMONA is dedicated for holonic manufacturing system .

We point out that the methodologies PASSI and O-MASE, are rich in documentation under different sorts, which is not the case for methodologies ROMAS and TROPOS. Generally documentation stills a real challenge for the startup of beginners in domain of MAS design. For the criterion of covrage of life cycle, we observe the presence of agile methods in the life cycle methodologies, eg methodologies ANEMONA and INGENIAS uses the process UP which positively reflects at the project management. For the frequency citation of methodologies papers, we note the popularity of the GAIA methodology: 2659 citations on Google Scholar until the moment of writing this paper , same for TROPOS with 1514 citations.

VI. CONCLUSION

We have presented in this paper a study of MAS design methodologies; this study was helpful to make the right choices regarding the establishment of a new ontology matching system through a multi-agent architecture [25].

The next step is to establish a detailed version of this study, this time based on the criteria for modeling concepts of the agents (roles, goals, organization, etc.), characteristics of agents and their applications.

REFERENCES

[1] BOOCH, Grady. Object Oriented Analysis & Design with Application. Pearson Education India, 2006.
[2] SHEHORY, Onn et STURM, Arnon. Evaluation of modeling techniques for agent-based systems. In : *Proceedings of the fifth international conference on Autonomous agents*. ACM, 2001. p. 624-631.
[3] COSSENTINO, Massimo, HILAIRE, Vincent, MOLESINI, Ambra, *et al.* (ed.). *Handbook on Agent-Oriented Design Processes*. Heidelberg : Springer, 2014.

[4] FERBER, Jacques et PERROT, Jean-François. *Les systèmes multi-agents: vers une intelligence collective*. InterEditions, 1995.
[5] RUSSELL, Stuart. Rationality and intelligence. In: *Foundations of rational agency*. Springer Netherlands, 1999. p. 11-33.
[6] JENNINGS, Nicholas R. et WOOLDRIDGE, Michael. Applications of intelligent agents. In : *Agent technology*. Springer Berlin Heidelberg, 1998. p. 3-28.
[7] BENMERZOUG, Djamel. Modèles et outils formels pour l'intégration d'applications d'entreprises. 2009. Thèse de doctorat. Paris 6.
[8] MOULIN, Bernard et CHAIB-DRAA, Brahim. An overview of distributed artificial intelligence. *Foundations of distributed artificial intelligence*, 1996, vol. 1, p. 3-55.
[9] WONGTHONGTHAM, Pornpit, DILLON, Tharam S., et CHANG, Elizabeth. *Ontology-based multi-agent systems*. Heidelberg : Springer, 2009.
[10] FLOREA, A., KAYSER, Daniel, PENTIUC, Stephan, *et al*. Cours Web Interactif: Agents Intelligents.
[11] SYCARA, Katia, NORMAN, Timothy J., GIAMPAPA, Joseph A., *et al.* Agent support for policy-driven collaborative mission planning. *The Computer Journal*, 2009, p. bxp061.
[12] BOISSIER, Olivier, BALBO, Flavien, et BADEIG, Fabien. Controlling multi-party interaction within normative multi-agent organizations. In : *Coordination, Organizations, Institutions, and Norms in Agent Systems VI*. Springer Berlin Heidelberg, 2011. p. 357-376.
[13] BERNON, Carole, GLEIZES, Marie-Pierre, PEYRUQUEOU, Sylvain, *et al.* ADELFE: a methodology for adaptive multi-agent systems engineering. In : *Engineering Societies in the Agents World III*. Springer Berlin Heidelberg, 2003. p. 156-169.
[14] BOTTI, Vicent et BOGGINO, Adriana Giret. *ANEMONA: A multi-agent methodology for Holonic Manufacturing Systems*. Springer Science & Business Media, 2008.
[15] COSSENTINO, Massimo, GAUD, Nicolas, HILAIRE, Vincent, *et al*. ASPECS: an agent-oriented software process for engineering complex systems. *Autonomous Agents and Multi-Agent Systems*, 2010, vol. 20, no 2, p. 260-304.
[16] WOOLDRIDGE, Michael, JENNINGS, Nicholas R., et KINNY, David. The Gaia methodology for agent-oriented analysis and design. *Autonomous Agents and multi-agent systems*, 2000, vol. 3, no 3, p. 285-312.
[17] ARGENTE, Estefanía, BOTTI, Vicent, et JULIAN, Vicente. Gormas: An organizational-oriented methodological guideline for open mas. In : *Agent-Oriented Software Engineering X*. Springer Berlin Heidelberg, 2011. p. 32-47.
[18] PAVÓN, Juan et GÓMEZ-SANZ, Jorge. Agent oriented software engineering with INGENIAS. In : *Multi-Agent Systems and Applications III*. Springer Berlin Heidelberg, 2003. p. 394-403.
[19] GARCIA-OJEDA, Juan C., DELOACH, Scott A., OYENAN, Walamitien H., *et al*. *O-MaSE: a customizable approach to developing multiagent development processes*. Springer Berlin Heidelberg, 2008.
[20] COSSENTINO, Massimo et SEIDITA, Valeria. PASSI: Process for agent societies specification and implementation. In : *Handbook on Agent-Oriented Design Processes*. Springer Berlin Heidelberg, 2014. p. 287-329.
[21] PADGHAM, Lin et WINIKOFF, Michael. Prometheus: A practical agent-oriented methodology. *Agent-oriented methodologies*, 2005, p. 107-135.
[22] GARCIA, Emilia, GIRET, Adriana, et BOTTI, Vicente J. Developing Regulated Open Multi-agent Systems. In : *AT*. 2012. p. 12-26.
[23] OMICINI, Andrea. SODA: Societies and infrastructures in the analysis and design of agent-based systems. In : *Agent-oriented software engineering*. Springer Berlin Heidelberg, 2001. p. 185-193.
[24] BRESCIANI, Paolo, PERINI, Anna, GIORGINI, Paolo, *et al*. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 2004, vol. 8, no 3, p. 203-236.
[25] CHAKER, Jihad, KHALDI, Mohamed, et AAMMOU, Souhaib. Towards a new ontology matching system through a multi-agent architecture. *International Journal of Emerging Technologies in Computational and Applied Sciences (IJETCAS)*, 2014, vol. 3, no 9, p. 295-299.

