EVALUATING MAS ENGINEERING TOOLS

Emilia Garcia, Adriana Giret and Vicente Botti
Department of Information Systems and Computation, Technical University of Valencia
Camino de Vera, Valencia, Spain

Keywords: Multiagent systems, agent oriented software engineering, multiagent systems developments tools.

Abstract: Recently a great number of methods and frameworks to develop multiagent systems have appeared. Nowadays there no an established framework to evaluate environments to develop multiagent system and choosing between one framework or another is a difficult task. The main contributions of this paper are an analysis of the state of the art in the evaluation of MAS engineering and a complete list of criteria that helps in the evaluation of multiagent system development environments.

1 INTRODUCTION

Multiagent systems (MAS) are complex and distributed systems with autonomous components. The development of MAS is a very complex task, so it needs the use of software engineering techniques. This techniques help developers to get complete, robust and functional systems decreasing the development time.

Nowadays, there is a great number of methods and frameworks to develop MAS, almost one for each agent-research group (Wooldridge and Ciancarini, 2001). This situation makes the selection of one or another multiagent development tool, a very hard task. The main objective of this paper is to provide a mechanism to evaluate these kind of tools. This paper shows a list of criteria that allows a deep and complete analysis of multiagent development tools. Through this analysis, developers can evaluate the appropriateness of using a tool or another depending on their needs.

The rest of the paper is organized as follows: Section 2 briefly summarizes the state of the art of the evaluation of MAS engineering. Section 3 details some important features to develop multiagent systems. Finally, Section 4 presents some conclusions and future works.

2 BACKGROUND

Shehory and Sturm (Sturm and Shehory, 2003) provide a list of criteria that includes software engineering related criteria and criteria relating to agent concepts. Also they add a metric evaluation. Cernuzzi and Rossi (Cernuzzi and Rossi, 2002) present a qualitative evaluation criteria employing quantitative methods for the evaluation of agent-oriented analysis and design modeling methods. The related works focus their efforts on the analysis of methodologies, but do not analyzes the tools that provide support for these methodologies. It is a very important feature because a well-defined methodology loses a great part of its functionality if there is no tool to apply it easily.

Eiter and Mascardi (Eiter and Mascardi, 2002) analyzes environments for developing software agents. They provide a methodology and general guidelines for selecting a MASDK. Their list of criteria includes agent features, software engineering support, agent and MAS implementation, technical issues of the MASDKs and finally economical aspects.

Bitting and Carter (Morales et al., 2003) use the criteria established by Eiter and Mascardi to analyze and compare five MASDKs. In order to obtain objective results from the evaluation Bitting and Carter add a quantitative evaluation. Sudeikat and Braunch (Sudeikat et al., 2004) presents an interesting work in which they analyzes the gap between modeling and platform implementation. Their framework allows the evaluation of the appropriateness of methodologies with respect to platforms.

This paper is based on the related works. The objective of this paper is to offer a list of evaluation criteria that allows to analyze and compare methods, techniques and environments for developing MAS. These criteria focus on the gap between the theorical guide-

Garcia E., Giret A. and Botti V. (2008).
EVALUATING MAS ENGINEERING TOOLS.
In *Proceedings of the Third International Conference on Evaluation of Novel Approaches to Software Engineering*, pages 181-184
DOI: 10.5220/0001763101810184

lines of the methodologies and what can be modeled in the MASDKs. Furthermore, these criteria analyze the gap between the model and the final implementation, i.e., which implementation facilities provide the MASDKs and which model elements have no direct translation in the implementation platform.

3 CRITERIA

The following features allow a complete analysis of a MASDK and the selection between one and another. They are grouped in five categories.

3.1 Concepts and Properties of MAS

As it is well known, there is no complete agreement on which features are mandatory to characterize an agent and a MAS. This is the reason why an analysis of the basic notions (concepts and properties) of agents and MAS are necessary at the beginning of the evaluation. This section deals with the question whether a methodology and it associated MASDK adhere to the basic concepts and properties of agents and MAS.

- AGENT FEATURES

These features are grouped into basic features that represent the core of agenthood, and advanced features that represent specific and desirable agent characteristics.

Basic Features

Agent architecture: It represents the concepts that describe the internals of an agent. The importance of this feature is not to say which approach is better than other, but this feature is very useful to know if the approach is appropriate to specific requirements.

Properties: Agents are supposed to be autonomous, reactive, proactive and socials. In this section which agent properties are supported by the methodology and by the MASDK is analyzed.

Advanced Features

Mental Attitudes: The agent has mental notions like beliefs, desires, intentions and commitments.

Deliberative Capabilities: The agent is able to select some possible plans to solve a problem and to choose the most appropriate in this situation.

Adaptivity: It may require that the modeling technique be modular and that it can activate each component according to the environmental state.

Meta-management: The agent is able to reason about a model of itself and of other agents.

- MULTIAGENT SYSTEMS FEATURES

Support for MAS Organizations. At this point will be analyzed only which kind of organizations are supported by the evaluated methodology or development environment, the other specific characteristics of the organizations will be analyzed in the following categories.

Support for the Integration with Services. Some MAS software engineering has been expanded to the integration with services (P.Singh and N.Huhns, 2005). At this point is interesting to analyze which kind of integration is supported by the approach (agents invoke services, services invoke agents or bidirectional) and the mechanisms used to facilitate the integration. Related with this, it is very interesting to know which services communication and specification standards are supported.

3.2 Software Engineering Support

The development of a multiagent systems is a complex task that can be simplified with the use of MAS engineering techniques. This section will analyze how MASDKs support these techniques.

- APPLICATION DOMAIN

There are some methodologies and MASDKs that can be used to develop any kind of MAS, but other approaches are specialized in a particular application domain.

- MODEL-CENTRAL ELEMENT

Traditionally, agents are the model-central element in most MAS models, but in the last years there are an evolution to the organization-oriented modeling and service-oriented modeling.

- METHODOLOGY

It can be analyzed using the following criteria:

Based on Metamodels. Meta-model presents relationships, entities, and diagrams, which are the elements to build MAS models.

Models Dependence. A high dependence on some specific models of a modelling method may imply that if they are not well-designed it may affect all the design process; hence, lower dependence is better.

Development Process. It indicates which software-development process follows the methodology.

Lifecycle Coverage. In complex systems such as MAS it is desirable to use tools that facilitate the development of the application throughout the entire process.

Development Guides. They facilitate the developers work and make the methodology more easy to understand and follow.

Platform Dependent. Some methodologies are focused on the development in a specific deployment

platform.
Organization Support. The methodology includes agent-organization concepts in the development life cycle.
Service Support. The methodology provides support to integrate services and agents at the different stages of the life cycle.

- MODELING LANGUAGE
The methodology should use a complete and unambiguous modeling language. It can be formal, informal or a mix of them. It should be expressive enough to represent MAS structure, data workflow, control workflow, communication protocols, concurrent activities and different abstraction level views. Other advanced features are the possibility to represent restrictions in the resources, mobil agents, the interaction with extern systems and the interaction with human beings.

- SUPPORT FOR ONTOLOGIES
It is analyzed if the MASDK offers the possibility to model, implement or import ontologies is analyzed.

- VERIFICATION TOOLS
The verification process can be analyzed from two points of view:
Static Verification. It involves to check the integrity of the system, i.e., that the specification of all model elements and the relationships between those elements are correct. The MASDK must be able to detect inconsistencies and to notify when the model is incomplete. In the best cases, the application not only detects these mistakes, but also propose solutions.
Dynamic Verification. It involves testing the system using simulations, i.e., the MASDK creates a simplified system prototype and test their behavior.

- THE GAP BETWEEN METHODS AND DEVELOPMENT TOOL
Three conflicted areas have been highlighted.
Complete Notation. The MASDK should provide the possibility to model all the methodology elements and their relationships. All the restrictions defined in the methodology should be defined in the modeling language and should be taken into account in the MASDK.
Lifecycle Coverage. This criterion identifies which methodology stages are supported by the MASDK.
Development Guidelines. These guides are very useful to develop MAS and if they are integrated in the MASDK the development task become more intuitive and easier. This integration reduces the modeling time and facilitate the development of MAS to developers.

3.3 MAS Implementation

This section analyzes how the MASDK helps the developer to transform the modeled system into a real application.

- IMPLEMENTATION FACILITIES
Graphical Interfaces. It represents the possibility to generate graphical interfaces using the MASDK.
Limited Systems. The MASDK may support the development of system with some limitations, i.e., the development of system that are going to be executed in limited devices like mobile phones.
Real Time Control. Some application need real time control, so it must be supplied for the MASDK.
Security Issues. The MASDK can provide security mechanism to ensure that agents are not malicious and do not damage other agents, that their agents are not be damaged and has to avoid the interception or corruption of messages.
Physical Environment Models. These are a library of simulators of physical parts of some kinds of systems for testing.
Code Implementation. The MASDK allows to implement or complete the agents code in the same tool.
Debugging Facilities. They are necessary for developing correct, reliable and robust system, due to the complex, distributed and concurrent nature of MAS.

- THE GAP BETWEEN MODELING AND IMPLEMENTATION
Match MAS Abstractions with Implementation Elements. It is analyzed which agent architecture elements, mental attitudes and deliberative attitudes are translated into elements of the final implementation. Most times the approaches allow modeling with this elements but they are not reflected in the implementation.
Automatic Code Generation. It is an important feature because it reduces the implementation time and the errors in the code. Nowadays, there are a great number of techniques and languages to transform automatically from models to code. Which technology is used and if the generation code is complete or if it only generate the skeletons of the agents are important features. Furthermore is important to remark for which agent platform and using which code language is generated the code. Some MASDKs generate utility agents. They offer services that do not depend on the particular application domain (for example yellow and white pages). Another MASDKs offer reengineering techniques and furthermore, some of them provides mechanisms to integrate services and agents.

3.4 Technical Issues of the MASDK

This criteria selection is related to the technical characteristics of the development environment.
Programming Language. The language used to implement the MASDK and the language used to store the models are important keys.
Resources. System requirements to the MASDK which include in which platforms can be executed and if it is light-weight.
Required Expertise. It indicates if it is necessary be a expert modeler and developer to use the MASDK.
Fast Learning. It indicates if the MASDK is easy to use and does not need much training time.
Possibility to Interact with Other Applications. For example this can provide the possibility to import or export models developed with other applications.
Extensible. The MASDK is prepared to include other functional modules in an easy way.
Scalability. This issue analyzes if the MASDK is ready to develop any scale of applications (small systems or large-scale applications).
Online Help. A desirable feature in a MASDK is that it helps developers when they are modeling or implementing, i.e., the MASDK takes part automatically or offer online suggestions to the developer.
Collaborative Development. This functionality may be very interesting to develop complex systems in which there are a group of developers which cooperates.
Documentation. An important aspect when dealing with new proposals is how they are documented. A good documentation and technical support should be provided.
Examples. If the MASDK presents complete case study is another feature to evaluate. The fact that the MASDK has been used in business environments also demonstrate the usefulness of the MASDK.

3.5 Economical Aspects

Economical characteristics are important to choose between one or another MASDK. Obviously, one key in the evaluation is the cost of the application, the cost of it documentation and a technical service is provided. Also, the vendor organization gives an idea about the reliability and the continuity of the application.

4 CONCLUSIONS

This paper summarizes the state of the art in the evaluation of methods and tools to develop MAS. These studies are the base of the presented evaluation framework. This framework helps to evaluate MASDKs by the definition of a list of criteria that allows to analyze the main features of this kind of systems. This list covers traditional software engineering needs and specific characteristics for developing MAS. This study allows the evaluation of the gap between the methods and the modeling tool, and the gap between the model and the implementation.

As future work, this framework will be used to evaluate and compare a large set of MASDKs and their methodologies. Also the MASDKs support for agent organizations and service-oriented MAS systems will be studied in depth [1].

REFERENCES

Cernuzzi, L. and Rossi, G. (2002). On the evaluation of agent oriented modeling methods. In *In Proceedings of Agent Oriented Methodology Workshop*.

Eiter, T. and Mascardi, V. (2002). Comparing environments for developing software agents. *AI Commun.*, 15(4):169–197.

Morales, P. C., Moreno, J. C. G., Rodriguez, A. M. G., and Martinez, F. J. R. (2003). A Framework for Evaluation of Agent Oriented Methodologies. *In Taller de Agentes Inteligentes en el tercer milenio (CAEPIA'2003)*.

P.Singh, M. and N.Huhns, M. (2005). *Service-Oriented Computing Semantics, Processes, Agents*. John Wisley and Sons Ltd.

Sturm, A. and Shehory, O. (2003). A framework for evaluating agent-oriented methodologies. In *AOIS*, volume 3030 of *LNCS*, pages 94–109. Springer.

Sudeikat, J., Braubach, L., Pokahr, A., and Lamersdorf, W. (2004). Evaluation of agent-oriented software methodologies examination of the gap between modeling and platform. *AOSE-2004 at AAMAS04*.

Wooldridge, M. and Ciancarini, P. (2001). Agent-Oriented Software Engineering: The State of the Art. In *Agent-Oriented Software Engineering: First International Workshop*, volume 1957/2001 of *LNCS*, pages 55–82. Springer.

[1]TIN2006-14630-C03-01, GV06/315, PAID-06-07/3191, CONSOLIDER INGENIO 2010 under grant CSD2007-00022.