TOWARDS AN INTEGRATIVE METHODOLOGY FOR DEVELOPING MULTI-AGENT SYSTEMS

José M. Gascueña and Antonio Fernández-Caballero

Universidad de Castilla-La Mancha, Escuela Politécnica Superior de Albacete
Instituto de Investigación en Informática de Albacete, 02071-Albacete, Spain

Keywords: Multi-agent systems, Agent development methodologies.

Abstract: A great number of methodologies to develop MAS systems have been proposed in the last few years. But, a perfect methodology that satisfies all the developer necessities can not be found. This is the reason why different methodologies are studied to create a new one. In this article, a methodology that includes all steps from the capture of requirements to the implementation and deployment of an agent-based application is proposed. In first place, an Analysis Overview Diagram is created to obtain an initial sketch of the application. Afterwards, the model obtained - by following the two first stages proposed by Prometheus methodology - is integrated into INGENIAS through UML-AT language. Next, the modelling goes on with INGENIAS. Finally, code is generated for the ICARO-T platform.

1 INTRODUCTION

Multi-agent systems (MAS) technology is adequate for developing open, complex, and distributed systems, and they offer a natural way of operating with legacy systems (Jennings & Wooldridge, 1995). A great number of methodologies to develop MAS systems have been proposed in the last few years. Gaia, Tropos, MaSE, MESSAGE, Prometheus, and INGENIAS are just a few examples.

Nonetheless, a perfect methodology that satisfies all the developer necessities can not be found. Usually, techniques and tools proposed in different methodologies to provide a solution to the specific problem that is being approached are combined. The result is a new methodology fruit of combining several proposals of the analyzed methodologies. In fact, in the literature, methodologies can be found that are influenced by other methodologies that already were proposed previously. For instance, INGENIAS (Pavón, Gómez-Sanz & Fuentes, 2005) together with a methodology to model real-time behaviors were adopted to define a MAS methodology for Holonic Manufacturing Systems (Giret, Botti & Valero, 2005).

In this article, INGENIAS is chosen as the basis, due to its recent direction towards model-driven

development (MDD) (Pavón, Gómez-Sanz & Fuentes, 2006) in order to define a new methodology to develop MAS. But, the two first stages proposed in Prometheus (Padgham & Winikoff, 2004), namely *system specification* and *architectural design*, are previously integrated in order to solve some current deficiencies in INGENIAS (see section 2). The language used by Prometheus is diffeerent from the INGENIAS language. Therefore, in order to use INGENIAS, it is necessary to transform the model obtained with Prometheus into an equivalent INGENIAS model. This transformation is performed with language UML-AT (Fuentes, Gómez-Sanz & Pavón, 2006), (Fuentes, Gómez-Sanz & Pavón, 2007). Later we propose to continue modelling with INGENIAS. Finally, code is generated for the ICARO-T platform (Garijo et. al, 2004), (Garijo et. al, 2008). The result of using the mentioned technologies, Prometheus, INGENIAS, UML-AT and ICARO-T, turns into a new methodology to develop MAS. The process followed in the methodology assists the MAS developer from the capture of requirements to the implementation and deployment of the application.

The article structure is as follows. Section 2 describes the contributions made by INGENIAS and the deficiencies that it presents. Methodologies Prometheus and INGENIAS, as well as the tools that support them, are compared in section 3. In section 4

the phases of the integrative methodology to develop MAS are proposed and described. Finally, some conclusions are offered.

2 WHY STARTING WITH INGENIAS?

In the initial INGENIAS proposal (Pavón, Gómez-Sanz & Fuentes, 2005) there are several contributions to develop MAS. First, it offers a meta-model to specify MAS. A MAS is considered from five complementary viewpoints: *organization*, *agent*, *goals and tasks*, *interaction*, and *environment*. Second, it adopts the unified software development process (USDP) (Jacobson, Booch & Rumbaugh, 1999) as a guideline to define the steps necessary to develop the elements and diagrams of MAS during the analysis and design phases. Third, INGENIAS Development Kit (IDK) is a tool that supports the methodology. IDK has integrated a set of utilities that allow model edition, verification, validation, and automatically generate code and documentation.

Now, INGENIAS is being reformulated in terms of the MDD paradigm (Pavón, Gómez-Sanz & Fuentes, 2006). Nowadays the use of model-driven engineering (MDE) techniques along the life cycle of software development is gaining more and more interest (Schmidt, 2006). The key idea underlying this paradigm is that if the development is guided by models there will be important benefits in fundamental aspects such as productivity, portability, interoperability and maintenance. Therefore, in the MAS field, it seems quite useful to use a methodology such as INGENIAS, which supports this approach. There are some other works using MDE in the area of MAS (Perini & Susi, 2005), (Rougemaille et. al, 2007), (Jarraya & Guessoum, 2007), among others.

Indeed, there are other reasons for studying the methodology INGENIAS and the tools created around. The INGENIAS engineer, connoisseur of the INGENIAS meta-model, can (a) define the meta-model for the domain of a concrete application, (b) personalize the IDK for a specific application domain, and, (c) create transformations to generate source code for the final platform on which the agents will run. There exist some previous experiences to adapt the INGENIAS language to more specific systems. For example, the IDK framework has been used to construct an editor for Holonic Manufacturing Systems (Giret, Botti & Valero, 2005). Also the INGENIAS language has been adapted for social simulation environments (Sansores, Pavón & Gómez-Sanz, 2004).

Unfortunately, in our opinion, the process followed in INGENIAS during the analysis and design phases of MAS is very complex and difficult to understand, because it is not clear how the different models are being constructed along the phases, despite the documented general guidelines. Moreover, INGENIAS does not provide any mechanism to discover which will be the agents of the system and their interactions. Thus, it is necessary to raise a process of alternative development that makes system development simpler. In order to make the MAS methodology easy to use for non expert people in the development of such systems, it is necessary that it offers a collection of detailed guidelines, including examples and heuristics, which help better understanding what is required in each step of the development process used in the methodology. These guidelines also serve as a help to the experts in MAS development. They will be able to transmit their experience to other users explaining why and how they have obtained the different elements (agents, interactions, etc.) of the agent-based application.

3 COMPARISON

INGENIAS has several advantages as opposed to Prometheus (see Table 1): (a) it follows an MDD approach, (b) it facilitates a general process to transform the models generated during the design phase into executable code. The advantages of Prometheus can be used (following the process to discover which be the agents of the system and its interactions) to enhance INGENIAS. In Table 2 the Prometheus Design Tool (PDT) (Padgham, Thangarajah & Paul) and INGENIAS Development Kit (IDK) (Gómez Sanz & Pavón) tools are compared. It may be observed that PDT only has one advantage with respect to IDK: it has a mechanism to prioritize parts of a project. In the rest of considered characteristics, IDK equals or surpasses PDT. Thus, the tool used to support the new methodology proposed is IDK as it is independent from the development process and it may be personalized for the application under development.

Table 1: Comparing Prometheus and INGENIAS.

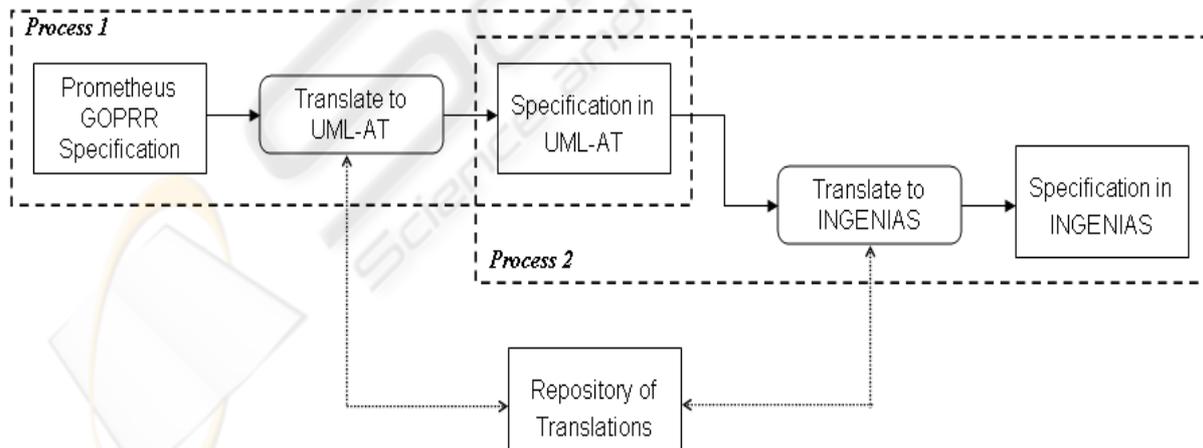|	Prometheus	INGENIAS
Proper development process	YES	NO: Based in the USDP (analysis and design phases)
General process to generate code from the models	NO: Only obtains code for JACK language	YES: Based in template definitions
Iterative development process	YES	YES
Model-driven development (MDD)	NO: Only proposes a correspondence between design models and JACK code	YES
Requirements capture	YES: A version of KAOS is used to describe the system's goals (van Lamsweerde, 2001) complemented with the description of scenarios that illustrate the operation of the system. In addition, in (Cysneiros & Zisman, 2004) guidelines appear to generate the artefacts of the Prometheus system specification from organizational models expressed in i*	YES: Performed by means of use case diagrams. Then, use cases are associated to system goals, and a goals analysis is performed to decompose them into easier ones; and finally tasks are associated to get the easiest goals
Meta-model	YES (Dam, Winikoff & Padgham, 2006)	YES
Mechanisms to discover agents and interactions among agents	YES: Groups functionalities through cohesion and coupling criteria.	NO
Agent model	BDI-like agents	Agents with mental states

Figure 1: Mapping Prometheus into INGENIAS.

Table 2: Comparing PDT and IDK.

	PDT	IDK
Supported methodology	Prometheus	INGENIAS
Interface references the development process	YES: Diagrams are grouped in three levels according to the three Prometheus phases	NO: Possibility to create packets that correspond to the diverse phases of the process. Models of each phase are added to the corresponding packet
Mechanisms to prioritize parts of the project.	YES: Three scope levels (essential, conditional and optional) (Perepletchikov & Padgham, 2005)	NO
Code generation	YES: JACK *http://www.agent-software.com/*	YES: JADE http://jade.tilab.com/
Report generation of the MAS specification in HTML	YES	YES
Model fragmenting in various pieces	NO: For instance, only one diagram may be created to in order to gather all the objectives of the system	YES
Save a diagram as an image	YES	YES
Deployment diagrams	NO	YES
Agent communication	Defined in basis of messages and interaction protocols. Does not use a specific communication language. For JACK, there is a module compliant with FIPA (Yoshimur, 2003).	Defined in accordance with communication acts of the agent communication language (ACL) proposed by FIPA http://www.fipa.org/specs/fipa00061/
Utility to simulate MAS specifications before generating the final code	NO	YES: Realized on the JADE platform. It is possible to manage interaction and tasks, and to inspect and modify the agents' mental states

4 PHASES OF THE NEW METHODOLOGY

First Phase. In the first stage of the methodology proposed an *analysis overview diagram* is created. This diagram is used to develop a high level view of the system requirements. This diagram will specify, in main lines, which are the actors - entities (human or software/hardware) external to the system – that interact with our system, where the perceptions that enter the system come from, which are the responses of the system (actions), an initial proposal of which might be the system roles, what messages are sent,

and some used data. This kind of diagram appeared for the first time in PDT version 2.5.

Second Phase. Prometheus defines a proper detailed process to specify, implement and test/debug agent-oriented software systems. This process incorporates three phases: (1) *system specification* identifies the basic goals and functionalities of the system, develops the use case scenarios that illustrate the functioning of the system, and specifies which are the inputs (percepts) and outputs (actions); (2) *architectural design* uses the outputs produced in the previous phase to determine the agent types that exist in the system and how they interact; and, (3) *detailed design*

centres on developing the internal structure of each agent and how each agent will perform its tasks within the global system. Finally, Prometheus details how to obtain the implementation in the agent-oriented programming language JACK.

The two first phases proposed in Prometheus (*system specification* and *architectural design*) are used to be the next phase of the new integrative methodology. The user identifies the agents and their interactions following the guidelines offered by Prometheus in these phases. In general terms, the mechanism provided by Prometheus to identify agents consists in identifying the goals during the system specification phase, and then in grouping the goals to obtain functionalities. Next, in the architectural design phase, functionalities are grouped to obtain the system agents, using cohesion and coupling criteria to decide which the best groupings are. These two concepts are essential in Software Engineering to obtain a good software development (the one that has maximum cohesion and minimum coupling) and to ease its further maintenance. Thus, the MAS developer gets an initial model according to Prometheus, following its two first stages (*system specification* and *architectural design*).

Afterwards, mappings are used to obtain an equivalent model in INGENIAS. From this point on the advantages offered by model-driven software development are used. The mappings are defined leaning in an intermediate language denominated Unified Modelling Language for Activity Theory (UML-AT) (Fuentes, Gómez-Sanz & Pavón, 2007). UML-AT allows establishing bidirectional transformations between models of different languages. There exists a previous experience in integrating two methodologies, Tropos and INGENIAS, using UML-AT (Fuentes, Gómez-Sanz & Pavón, 2006).

The process of transforming Prometheus into INGENIAS methodology elements, with the help of intermediate language UML-AT, is shown in Figure 1. In first place (process 1), we start from the Prometheus meta-model specified with language GOPRR (Graph Object Property Relationship Role) (Lyytinen & Rossi, 1999). Translation rules to obtain elements, expressed in UML-AT language, equivalent to the ones selected in Prometheus, are created and used. Next (process 2), translation rules are used to obtain the specification in INGENIAS equivalent to the one obtained in UML-AT language. The Repository of Translations contains tuples indicating the matches and instantiation functions used in the translation, as well as the

elements participating in it (either used in the process or created as a result of it) and an identifier of the specification to which each one belongs to.

A match represents the translation between two sets of structures, the source pattern (it is described in the source language) and the target pattern (it is described in the target language). An instantiation function describes the correspondence of the variables in the source patterns with the elements in the current specifications according to the matching that is presented.

ATLAS Transformation Language (ATL) (Bézivin, Jouault & Touzet, 2005), a model transformation language compliant with the OMG MOF/QVT (Queries / Views /Transformation), can be used as an alternative to approach the problem of transforming Prometheus to INGENIAS. A meta-model and a model expressed in original language (in this case, Prometheus), a destination meta-model (in this case, INGENIAS) and rules defined with ATL to perform the transformation are needed in this case. The result is a destination model (INGENIAS, in this case) equivalent to the original model. The meta-models and models can be defined in Ecore, the language used by Eclipse Modelling Framework (EMF). The INGENIAS meta-model, defined initially in GOPRR, has been migrated to Ecore (García-Magariño, Gómez-Sanz & Pavón, 2007). Thus, it is only necessary to define: (1) the Prometheus meta-model with Ecore, and, (2) the transformation rules in ATL. We have decided to use UML-AT because it is a technology related to the research group that has developed INGENIAS. In addition, it supposes the same service load as using ATL: to define a meta-model (for Prometheus in GOPRR) and translation rules (to transform a Prometheus specification into a UML-AT specification). The corresponding part to transform UML-AT into INGENIAS is solved in (Fuentes, Gómez-Sanz & Pavón, 2007). A tool called *Activity Theory Assistant* (ATA) has been developed to help using the techniques based on the theory of the activity and to support the translation process. ATA is embedded in a plug-in of the IDK. Notice, however, that the current IDK version available in SourceForge, *http://sourceforge.net/projects/ingenias*, does not include it.

In Prometheus, in order to describe the interactions among agents, interaction protocols using a reviewed version of Agent UML (AUML) denominated AUML-2 are developed. UML-AT has already been applied to establish correspondences between FIPA protocols designed with AUML

models (Bauer & Odell, 2005) and INGENIAS models (Fuentes, Gómez-Sanz & Pavón, 2007).

This work is taken as the starting point to transform interaction protocols obtained with Prometheus into the equivalent notation used in INGENIAS. The IDK tool, which provides support to INGENIAS, allows representing protocols according to the AUML annotation (Gómez Sanz & Pavón). This means that the interaction protocols created with Prometheus could be used directly in INGENIAS, with no need to use any transformation. Nevertheless, its development has not evolved enough. In fact, in version 2.6 of the IDK this utility no longer appears.

Third Phase. The new methodology does not reuse the last phase of Prometheus (*detailed design*) because it is too much centred in BDI-like agents. Moreover, Prometheus also describes how the obtained entities are transformed in the design phase into the concepts used for a specific implementation language (JACK). These two aspects, centring in a single type of agent and defining a mechanism to generate code for a particular implementation language, suppose, in principle, a loss of generality. In the new methodology, once the equivalent model in INGENIAS has been obtained, the architecture of each type of identified agent is provided. The possible types of agents are the ones available in ICARO-T: cognitive agents and reactive agents. In this phase the necessary guidelines for completing all the INGENIAS models already exist.

Fourth Phase. With respect to code generation, the INGENIAS proposal is followed. INGENIAS generalizes a process to transform the models, generated in the phase of design, in running code for any destination platform (Pavón, Gómez-Sanz & Fuentes, 2006). It is based in the definition of templates for each destination platform and procedures for extracting information present in the models. Once the code has been obtained, the developer refines the resulting code completing any information that was not contained in the specifications (models) or in the templates. Finally, the application is deployed.

ICARO-T is the platform selected for running the agents (Garijo et. al, 2004), (Garijo et. al, 2008). It offers four categories of reusable component models: *agent organization models* to describe the overall structure of the system, *agent models*, *resource models* to encapsulate computing entities providing services to agents, and *basic computing entities*.

There are several reasons for selecting the multi-agent platform ICARO-T. The use of its components has allowed to significantly reducing time and effort in the design and implementation phases by an average of a 65 percent. In the phases of testing and correction cycles the errors are also reduced. Consequently, the applications require less resources and lower implementation time (Pavón, Garijo & Gómez-Sanz, 2007). The ICARO-T components have been used in telecommunications company Telefonica for developing several voice recognition services. At the moment, it is also being used by other research teams. This is the case, for example, in an e-learning project denominated ENLACE (Celorrio & Verdejo, 2007).

Figure 2 shows the technology and tools used in the integrative methodology proposed.

5 CONCLUSIONS AND FUTURE WORK

The combination of current technologies (Prometheus, INGENIAS, UML-AT and ICARO-T) has given rise to a new integrative methodology for the development of agent-based systems. It uses the guidelines offered by Prometheus to identify agents and their interactions. Later, the obtained model, following Prometheus methodology, is transformed into INGENIAS to continue the development. This transformation is performed under UML-AT. Once modelling has ended up, code is generated for the ICARO-T platform.

In order to use this methodology definitively it is necessary: (1) to specify the Prometheus meta-model in GOPRR, (2) to create the rules to translate Prometheus concepts in UML-AT language, and, (3) to define templates for the IDK to generate code for the ICARO-T platform. A first proposal in order to transform Prometheus models into equivalent INGENIAS models using an informal language has recently been considered (Gascueña & Fernández-Caballero, 2008).
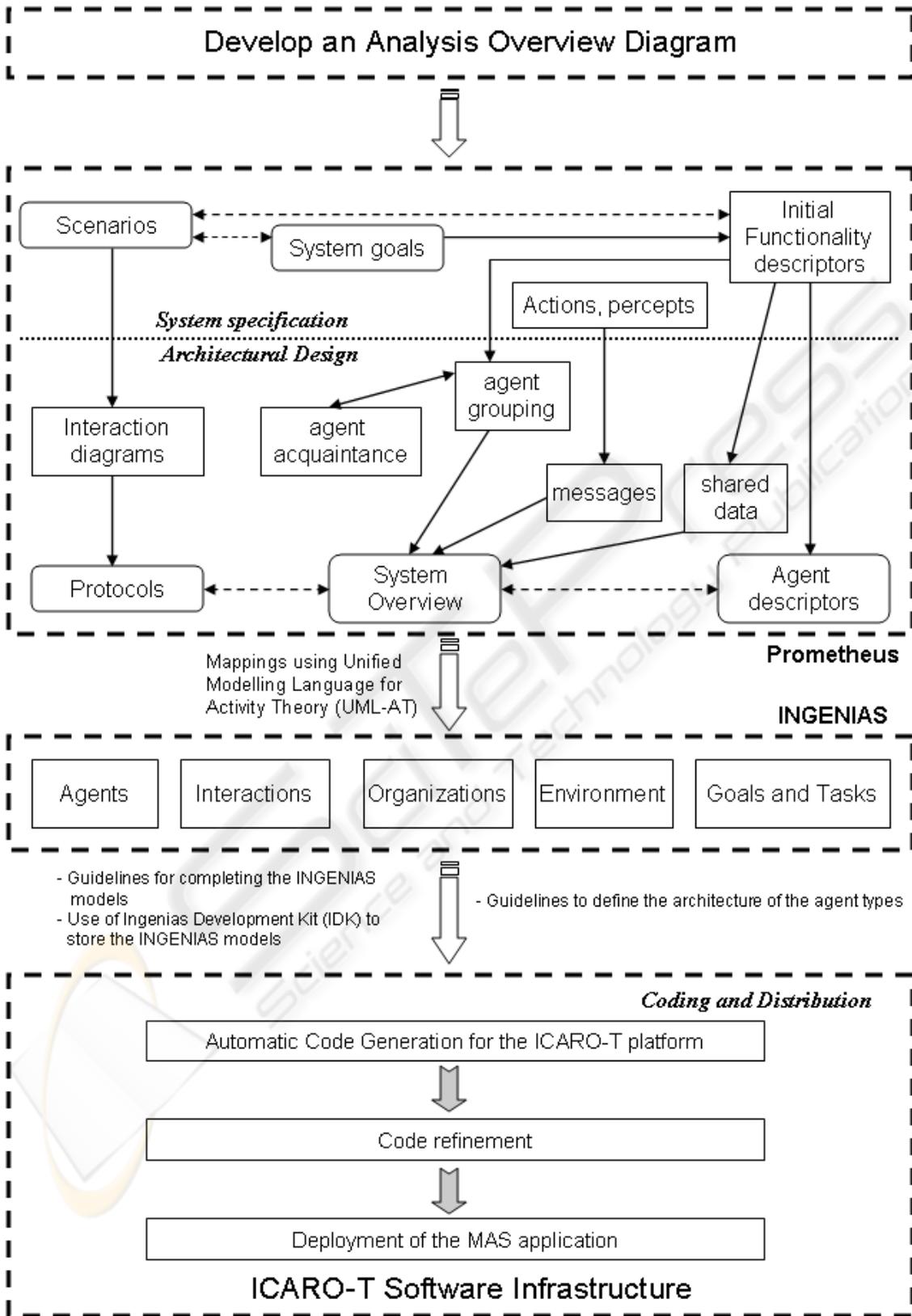
ACKNOWLEDGEMENTS

Figure 2: MAS development methodology.

REFERENCES

Bauer, B., & Odell, J. (2005). UML 2.0 and agents: How to build with the new UML standard. Journal of Engineering Applications in Artificial Intelligence, Vol. 18, No. 2, pp. 141-157.

Bézivin, J., Jouault, F., & Touzet, D. (2005). An introduction to the ATLAS model management architecture. Research Report LINA, (05-01). February 2005.

Celorrio, C., & Verdejo, M.F. (2007). Adapted activity deployment and configuration in a pervasive learning framework. Proceedings in Pervasive Learning, Workshop at Pervasive 2007, PL2007, pp. 51-58.

Cysneiros, G., & Zisman, A. (2004). Refining Prometheus methodology with i*. 3rd International Workshop on Agent-Oriented Methodologies, OOPSLA 2004.

Dam, K.H., Winikoff, M., & Padgham, L. (2006). An agent-oriented approach to change propagation in software evolution. Proceedings of the Australian Software Engineering Conference, ASWEC 2006, pp. 309-318.

Fuentes, R., Gómez-Sanz, J.J., Pavón, J. (2006). Integrating agent-oriented methodologies with UML-AT. Proceedings of the 5th International. Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS 2006, pp. 1303-1310.

Fuentes, R., Gómez-Sanz, J.J., Pavón, J. (2007). Model integration in agent-oriented development. International Journal of Agent-Oriented Software Engineering, Vol. 1, No.1, pp. 2-27.

García-Magariño, I., Gómez-Sanz, J.J., & Pavón, J. (2007). Representación de las relaciones en los metamodelos con el lenguaje Ecore. Desarrollo de Software Dirigido por Modelos, DSDM07.

Garijo, F., Bravo, S., Gonzalez, J., & Bobadilla, E. (2004). BOGAR-LN: An agent based component framework. Lecture Notes in Artificial Intelligence, Vol. 3040, pp. 207-220.

Garijo, F., Polo, F., Spina, D., Rodríguez, C. (2008). ICARO-T User Manual. Internal Report, Telefonica I+D, May 2008.

Gascueña, J.M., & Fernández-Caballero, A. (2008). Prometheus and INGENIAS agent methodologies: A complementary approach. Proceedings of the 9th International Workshop on Agent-Oriented Software Engineering, AOSE'08.

Giret, A., Botti, V., & Valero, S. (2005). MAS methodology for HMS. Lecture Notes in Artificial Intelligence, Vol. 3593, pp. 39-49.

Gómez Sanz, J.J., & Pavón, J. (2008). INGENIAS Development Kit (IDK) Manual. Version 2.5.2 (corresponds to version 2.5 of the IDK). http://heanet.dl.sourceforge.net/sourceforge/ingenias/ ingeniasmanual.v2.5.pdf

Jarraya, T., and Guessoum, Z. (2007). Towards a model driven process for multi-agent system. Lecture Notes in Computer Science, Vol. 4696, pp. 256-265.

Jacobson, I., Booch, G. & Rumbaugh, J. (1999). The unified software development process. Reading, MA: Addison-Wesley.

Jennings, N.R., & M. Wooldridge. (1995). Applying agent technology. International Journal of Applied Artificial Intelligence, Vol. 9, No. 4, pp. 351-359.

Lyytinen, K.S., & Rossi, M. (1999). METAEDIT+ - A fully configurable multi-user and multi-tool CASE and CAME environment. Lecture Notes in Computer Science, Vol. 1080, pp.1-21.

Padgham, L., & Winikoff, M. (2004). Developing Intelligent Agents Systems: A Practical Guide. John Wiley and Sons.

Padgham, L., Thangarajah, J., & Paul, P. (2008). Prometheus Design Tool. Version 2.5. User Manual. http://www.cs.rmit.edu.au/agents/pdt/docs/PDT-Manual.pdf.

Pavón, J., Gómez-Sanz, J.J., & Fuentes, R. (2005). The INGENIAS methodology and tools. In: Agent-Oriented Methodologies. Idea Group Publishing.

Pavón, J., Gómez-Sanz, J.J., & Fuentes, R. (2006). Model driven development of multi-agent systems. Lecture Notes in Computer Science, Vol. 4066, pp. 284-298.

Pavón, J., Garijo, F., & Gómez-Sanz, J. (2007). Complex systems and agent-oriented software engineering. In Engineering Environment-Mediated Multiagent Systems, EEMMAS 2007, pp. 3-16.

Perepletchikov, M., & Padgham, L. (2005). Systematic incremental development of agent systems, using Prometheus. In Fifth International Conference on Quality Software, QSIC'05, pp. 413-418.

Perini, A., and Susi, A. (2005). Automating model transformations in agent-oriented modelling. Lecture Notes in Computer Science, Vol. 3950, pp. 167-178.

Rougemaille, S., Migeon, F., Maurel, Ch., Gleizes, M.P. (2007). Model driven engineering for designing adaptive multi-agent systems. Proceedings of the 8th Annual International Workshop Engineering Societies in the Agents World, ESAW07.

Sansores, C., Pavón, J. & Gómez-Sanz, J.J. (2006). Visual modeling for complex agent-based simulation systems. Lecture Notes in Artificial Intelligence, Vol. 3891, pp. 174-189.

Schmidt, D.C. (2006). Guest editor's introduction: Model-driven engineering. Computer, Vol. 39, No. 2, pp. 25-31.

van Lamsweerde, A. (2001). Goal-oriented requirements engineering: A guided tour. Proceedings of the 5th IEEE International Symposium on Requirements Engineering, RE'01, pp. 249-263.

Yoshimura, K. (2003). FIPA JACK: A plugin for JACK Intelligent Agents[TM]. Technical Report, RMIT University. September 2003.