# THE PROTÉGÉ - PROMETHEUS APPROACH TO SUPPORT MULTI-AGENT SYSTEMS CREATION

Marina V. Sokolova[1,2] and Antonio Fernández-Caballero[1]
*[1] Universidad de Castilla-La Mancha, Departamento de Sistemas Informáticos & Instituto de Investigación en Informática de Albacete, Campus Universitario s/n, 02071-Albacete, Spain*

*[2] Kursk State Technical University, ul.50 let Oktiabrya, 94, Kursk, 305040, Russia*

Keywords: Ontologies, Intelligent agents, Mapping, Prometheus, Protégé.

Abstract: The integration of two existing and widely accepted tools, Protégé Ontology Editor and Knowledge-Base Framework, and Prometheus Development Kit, into a common approach, aiming to include the principal stages of MAS development life cycle and offering a general sequence of steps facilitating application creation, is proposed in this paper. The approach is successfully being applied to situation assessment issues, which has concluded in an agent-based decision-support system for environmental impact evaluation.

## 1 INTRODUCTION

Creation, deployment and post implementation of a multi-agent system (MAS) as a software product is a complex process, which passes through a sequence of stages forming its life cycle (Marik and McFarlane, 2005; Vasconcelos et al., 2001). Every step of the life cycle process has to be supported and provided by means of program tools and methodologies. In case of MAS development, in our opinion there is still no solution to a unified approach to cover all the stages. However, there are some works dedicated to this issue (de Wolf and Holvoet, 2005; Konichenko, 2005). For instance, de Wolf and Holvoet have presented a methodology in the context of standard life cycle model, with accent to decentralization and macroscopic view of the process. Within tools and frameworks the authors mention Jade (Bellifemine, Poggi and Rimassa, 1999), Repast (Repast, 2003) and an environment for coordination of situated multi-agent systems (Schelfthout and Holvoet, 2004). The authors offer their approach on the assumption that the research task has already been defined, omitting the problem definition and domain analysis stages of the MAS development process.

The software development in case of MAS is based on the following steps (Konichenko, 2005): (1) domain and system requirements analysis; (2) design; (3) implementation; (4) verification; (5) maintenance.

System analysis and design stages are supported by well known alternative agent-oriented software engineering methodologies, including MaSE (DeLoach, Wood and Sparkman 2001), Gaia (Wooldridge, Jennings and Kinny 2000), MASDK (Gorodetsky et al., 2005), Prometheus (Padgham and Winikoff, 2002), Tropos (Giunchiglia, Mylopoulos and Perini, 2002) among others.

The MAS, after being coded and tested, is ready to be deployed and be further modified or changed, depending from requirements. The existing methodologies do not or only briefly extend over the first stage, as they work under the condition that the developer has already defined the problem and determined the goals and the tasks of the system. However, this stage is a crucial one and has to be carefully examined and planned. Indeed, the whole deployed system functionality and efficiency depends on how precisely the problem was defined and the domain ontology was elaborated.

In this work we introduce our approach to MAS life-cycle support, in which we focus both on creation ontological background and system design and deployment.

The paper is organized as follows. In section 2 the meta-ontology creation realized in Protégé is described and in section 3 the MAS design made in PDT is introduced. In section 4 our intention to

V. Sokolova M. and Fernández-Caballero A. (2008).
THE PROTÉGÉ - PROMETHEUS APPROACH TO SUPPORT MULTI-AGENT SYSTEMS CREATION.
In *Proceedings of the Tenth International Conference on Enterprise Information Systems - AIDSS*, pages 442-445
DOI: 10.5220/0001685304420445

implement the ideas of the integrated methodology for practical issue is briefly explained.

## 2 REQUIREMENTS IN PROTÉGÉ

Ontology creation may be viewed as a crucial step in MAS design as it determines the system knowledge area and potential capabilities (Guarino and Giaretta, 1995). In the first part of this article a model of distributed meta-ontology that serves as a framework for MAS design is proposed. Its components - private ontologies - are described in extensive with respect to an application area and in terms of the used semantics (Sokolova and Fernández-Caballero, 2007).

The model of the meta-ontology that we have created consists of five components, or private ontologies: the "Domain Ontology", the "Task Ontology", the "Ontology of MAS", the "Interaction Ontology" and the "Agent Ontology".

In first place, the "Domain Ontology", includes the objects of the problem area, relations between them and their properties.

$$\mathbf{OD = < C, I, P, V, Rs, Rl>} \qquad (1)$$

where the sets **C** and **I** are classes and their individuals; **P** are class properties; **V** are the properties values; **Rs** are some marginal levels for values (restrictions); **Rl** states rules to receive new individuals for the concrete class.

The "Task Ontology" contains information about tasks and respective methods, about the pre-task and post-task conditions, and informational flows for every task. The "Task Ontology" has the following model:

$$\mathbf{OT=<T, M, In, Ot, R>} \qquad (2)$$

where **T** is a set of tasks to be solved in the MAS, and **M** is a set of methods or activities related to the concrete task, **In** and **Ot** are input and output data flows, **R** is a set of roles that use the task. Component **R** is inherited from the "Ontology of MAS" through the property *belong to role*. The tasks are shared and accomplished in accordance with an order. The "Ontology for MAS" architecture is stated as:

$$\mathbf{OA=<L, R, IF, Or>} \qquad (3)$$

where **L** corresponds to the logical levels of the MAS (if required), **R** is a set of determined roles, **IF** is a set of the corresponding input and output information represented by protocols. Lastly, the set **Or** determines the sequence of execution for every role (orders).

The interactions between the agents include an initiator and a receiver, a scenario and the roles taken by the interacting agents, the input and output information and a common communication language. They are stated in the "Interaction Ontology" as:

$$\mathbf{OI=<In, Rc, Sc, R, In, Ot, L>} \qquad (4)$$

Actually, as **In** and **Rc** Initiator and Receiver, respectively, of the interaction we use roles. The component **Sc** corresponds to protocols. **R** is a set of roles that the agents play during the interaction. **In** and **Ot** are represented by informational resources, required as input and output, respectively. Language **L** determines the agent communication language (ACL).

In our approach BDI agents, which are represented by the "Agent Ontology", are modelled. Hence, every agent is described as a composition of the following components (Georgeff et al., 1998):

$$\mathbf{Agent = <B, D, I>} \qquad (5)$$

Every agent has a detailed description in accordance with the given ontology, which is offered in a form of BDI cards, in which the pre-conditions and post-conditions of agent execution and the necessary conditions and resources for the agent successful execution are stated. Evidently, **B**, **D** and **I** stand for Believes, Desires and Intentions, respectively.

Thus, the "Agent Ontology" incorporates the following components: **B** is related to "Domain Ontology", which determines information data resources, necessary for every agent (its believes), **D** includes methods stored in the "Task Ontology" (agent desires), and **I** calls for intentions necessary for every activity or task specification. There is also a collaborator, in case there are two or more agents needed to solve the task.

Private ontologies mapping is made through slots of their components. So, the "Agent Ontology" has four properties: *has intentions* - which contains individuals of the methods **M** class from the "Task Ontology"; *has believes* - which contains individuals from the "Domain Ontology"; *has desires* - which contains individuals from the "Task Ontology"; *has type* - which contains variables of String type.

There is a real connection between the "Task Ontology" (OT) and the "Domain Ontology" (OD) through *believes*. The OT, in turn, refers to the "Ontology of MAS" (OA), which is formally described by four components. The first two are:

*level value* and *order*; contain values of Integer type, which determine the logical level number and the order of execution for every role. Roles (R) are the individuals of the named ontology. The next two properties: *has input* and *has output* refer to individuals of "Interaction Ontology" (OI); in particular, to protocols, which manage communications. Their properties are of type String: *has scenario, language, roles at scenario.*

The "Interaction Ontology" slots named *has initiator* and *has receiver* are the individuals of the "Agent Ontology" (Agent). Thus, agents are linked to the proper protocols within the MAS. The OD, by means of its individuals - which contain data records - is connected to Agent , which uses the knowledge on the domain area as its believes. This way, the proposed meta-ontological model realized in Protégé covers the first steps of the software development life cycle.

## 3 SYSTEM DESIGN WITH PROMETHEUS DEVELOPMENT TOOL

In order to validate the next step of our approach, we introduce a running example, consisting in an agent-based decision support system (ADSS) dedicated to monitoring environmental pollution information, analyzing this data, simulating with respect to health consequences and making decisions for responsible system users. The general view of the ***Classes*** belonging to the domain of interest includes regions, which are characterized with some environmental pollution, and morbidity level.

The accent is made on regions (Sokolova and Fernández-Caballero, 2007). The MAS is a logical three layer architecture. The first layer is aimed for meta-data creation, the second one is responsible for hidden knowledge discovery, and the third level provides real-time decision support making, data distribution and visualization. This architecture satisfies all the required criteria for decision support systems.

The first level is named Information fusion and it acquires data from diverse sources, and pre-processes the initial information to be ready for further analysis. The second layer is named Data Mining and there are three roles at this level, dedicated to knowledge recovering through modelling, and calculation impact of various pollutants upon human health. The third level, Decision Making, carries out a set of procedures including model evaluation, computer simulation, decision making and forecasting, based on the models created in the previous level. The main function of this level is to provide a user - actually, a person who makes decisions - with the possibility to run online real-time "what - if" scenarios. The end-user, that is to say the person making decisions, interacts with the MAS through a System-User Interaction protocol, which is responsible for human-computer interaction. The user chooses the indicator that he wants to examine and initiates a computer simulation.

The system resembles a typical organizational structure. The agents are strictly dedicated to work with the stated sets of data sources. They solve the particular tasks and are triggered when all the necessary conditions are fulfilled, and there are positive messages from previously executed agents (Weiss, 2000). The system includes a set of roles, correlated with the main system functions and a set of agents related to each role.

## 4 IMPLEMENTATION IN JACK

The system design in the Prometheus design tool ends with generation of the skeleton code for JACK Intelligent Agents™ that facilitates further coding, testing and deployment.

The JACK Design Tool provides a visual interface, which supports application creation directly created in Jack Development Environment imported from Prometheus. This last possibility supports our proposal. Therefore, we propose to implement the coding, testing and deployment stages of the MAS development process within the JACK Development Environment.

## 5 CONCLUSIONS

The integration of two existing and widely accepted tools, Protégé Ontology Editor and Knowledge-Base Framework, and Prometheus Development Kit, into a common methodology has been introduced in this paper. To provide the following stages with tools, we have tested different methodologies, and finally decided to use the Prometheus Development Tool, which offers a wide range of possibilities for MAS planning and implementation, namely the system architecture, the system entities, their internals and communications within the system and with outer entities.

The integrated approach covers all the stages of MAS planning and implementation, supporting them with tools and frameworks. The proposed fusion of methodologies, Protégé and Prometheus, was chosen because of the wide range of functions offered and their conformance to international standards. However, other combinations of agent-oriented tools could be used, whenever it helps getting the same result and support during MAS development, deployment and maintenance.

## ACKNOWLEDGEMENTS

Marina V. Sokolova is the recipient of Postdoctoral Scholarship 0000253836, Program II.E (Becas MAE) awarded by the Agencia Española de Cooperación Internacional of the Spanish Ministerio de Asuntos Exteriores y de Cooperación.

This work is supported in part by the Spanish Ministerio de Educacion y Ciencia TIN2004-07661-C02-02 and TIN2007-67586-C02-02 grants, and the Junta de Comunidades de Castilla-La Mancha PBI06-0099 grant.

## REFERENCES

Bellifemine, F., Poggi, A., Rimassa, G. (1999). Jade - A FIPA-compliant agent framework. Proceedings of the Practical Applications of Intelligent Agents, pp. 97-108.

DeLoach, S.A., Wood, M.F., and Sparkman, C.H. (2001). Multiagent systems engineering. International Journal of Software Engineering and Knowledge Engineering, 11, 231-258.

de Wolf, T., Holvoet, T. (2005) Towards a full life-cycle methodology for engineering decentralised multi-agent systems. Proceedings of the Fourth International Workshop on Agent-Oriented Methodologies, pp. 1-12.

Georgeff, M., Pell, B., Pollack, M., Tambe, M., Wooldridge, M. (1998). The Belief-Desire-Intention model of agency. Intelligent Agents V: Agent Theories, Architectures, and Languages. Lecture Notes in Computer Science, 1555, pp. 1-10.

Giunchiglia, F., Mylopoulos, J., Perini, A. (2002). The Tropos software development methodology: Processes, models and diagrams. Third International Workshop on Agent-Oriented Software Engineering, pp. 162-173.

Gorodetsky, V., Karsaev, O., Konushy, V., Mirgaliev, A., Rodionov, I., Yustchenko, S. (2005). MASDK software tool and technology supported. International Conference on Integration of Knowledge Intensive Multi-Agent Systems, pp. 528-533.

Guarino, N., Giaretta, P. (1995), Ontologies and knowledge bases: Towards a terminological clarification. In: Towards Very Large Knowledge Bases, IOS Press 1995, pp. 25-32.

Konichenko A.V. (2005). Distribution Information Systems Design Management. Rostov-Don Press, Russia.

Marík, V., McFarlane, D. (2005). Industrial adoption of agent-based technologies. Intelligent Systems, 20, pp. 27–35.

Padgham, L., Winikoff, M. (2002). Prometheus: A pragmatic methodology for engineering intelligent agents. Proceedings of the Workshop on Agent Oriented Methodologies (Object-Oriented Programming, Systems, Languages, and Applications), pp. 97-108.

Repast home page. (2003). http://repast.sourceforge.net.

Schelfthout, K., Holvoet, T. (2004). ObjectPlaces: an environment for situated multi-agent systems. Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 1500-1501.

Sokolova, M.V., Fernández-Caballero, A. (2007). A multi-agent architecture for environmental impact assessment: Information fusion, data mining and decision making. 9th International Conference on Enterprise Information Systems, ICEIS 2007, vol. AIDSS, pp. 219-224.

Sokolova, M.V., Fernández-Caballero, A. (2007). An agent-based decision support system for ecological-medical situation analysis. 2nd International Work-conference on the Interplay between Natural and Artificial Computation. Lecture Notes in Computer Science, 4528, pp. 511-520.

Vasconcelos, W.W., Robertson, D.S., Agusti, J., Sierra, C., Wooldridge, M., Parsons, S., Walton, C., Sabater, J. (2001). A lifecycle for models of large multi-agent systems. Proceedings of the Second International Workshop on Agent-Oriented Software Engineering. Lecture Notes in Computer Science, 2222, pp. 297-318.

Weiss, G. (2000). Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. The MIT Press.

Wooldridge, M., Jennings, N.R., Kinny, D. (2000). The Gaia methodology for agent-oriented analysis and design. Journal of Autonomous Agents and Multi-Agent Systems, 3, 285-312.