MODELING LANGUAGE OF MULTI-AGENT SYSTEMS = PROGRAMMING TEMPLATE

Rubén Álvarez-González, Miguel Angel Díaz Martínez

Abstract: The modeling languages are designed to make easier the software development. That is why so many times they are included in the development methodology. In 2001 the OMG proposed model driver architecture for the software development (MDA). In this architecture are transformations which are used between the models to get others. The goal is to show a new way to develop applications using the Agents Oriented paradigm. To do it the MDA is showed and the methodology agent's models are studied. There is no methodology which uses the transformations between the models, so the meta-models group and the transformations between them should be researched.

Keywords: MDA, OMG, Modeling Languages, Meta-Models, Models, MAS, Agents, MDD, MDE.

ACM Classification Keywords: C.2.4 Distributed Systems - Distributed applications, D.2.11 Software Architectures – Languages.

Introduction

The software development evolves all time. It starts when the programmers made the software using machine language. Now, the software with high level programmer languages is built. This evolution tries to make the development software with a language as similar as the human one instead to a machine language. All of these time different paradigms are proposed. The problems are analyzed in a way more natural with these paradigms.

Two of these paradigms are very similar. These are: Object Oriented (OO) and Agent Oriented (AO). The main entity in both paradigms, object to OO and agent to AO, encapsulate their state. The objects use private attributes to save their state and the agents save their state as beliefs. An object might interact with other objects using public method, the first one forces the second to do something when the first object calls for it on a public method. On the opposite, an agent sends messages with the other agent. The agents negotiate with these messages to do something, and an agent ever forces another agent to do something. In other words, these paradigms present a different view of the world. OO analyzes the world as an object group, these objects interact between them. AO interprets the world as an autonomous agent group which collaborates between them. [Bernon et al, 2005]

To develop the software using these paradigms work methodologies were proposed. These methodologies are used to guide the software development process. In the AO case some methodologies exist: Gaia, PASSI, ROADMAP, etc [Wooldridge et al, 2000], [Cossentino, 2005] and [Juan et al, 2002]. All of these use modeling languages to build new models. The solutions are represented with those models. A model is an affirmation collection, which is true or false, about a study system [Seidewitz, 2003].

This document's objective is to present a new way for software development using the AO paradigm. To do this the document has three main parts. The first section presents the Model Driven Architecture (MDA). In the second part the modeling languages which are used in the AO methodologies are studied. And in the third division the differences between the AO languages models and MDA technology are studied. The work finishes with the author conclusions.

MDA

There are some problems in the software development. These appear when the developer group wants to integrate systems which exist already with new technologies [Kent, 2002]. The OMG want to resolve these problems and to do it this group proposed the model driver architecture in 2001.

MDA defines the IT systems (Information Technology) in two parts. The first one is the functionality specification, and the second part is the implement specification of the functionality in a particular technology. This definition is one of the main MDA characteristics. MDA uses platform independent models (PIM) and platform specific models (PSM). The most important advantages of these are [OMG, 2001]:

- To make easier to verify that a model is or is not correct.
- To make easier the generation of an implementation in a different platform with the same structure and behaviour.
- To make possible the definition, of integration mechanism and interoperability between systems in an independent way.

A PIM is a formal specification of the system structures and their functionality. This specification doesn't take into account the technical details. A PSM is a specification model of the system's platform. The relation between PIM and PSM is that defined functionality in the PIN will be executed on the platform which is specified in the PSM.

Like the source code, or like the natural language, the models and the transformations need a proper language for their representation. For the models, these languages are called "modeling languages" or "meta-models". At the same time, another kind of language is necessary to define the meta-models. These new languages are entitled "meta-modeling languages" or "meta-meta-models". The meta-model standard language is MOF [OMG, 2003].

The model driver architecture made up for meta-meta-models, meta-models and models. This architecture has three levels of meta-levels (Fig 1).

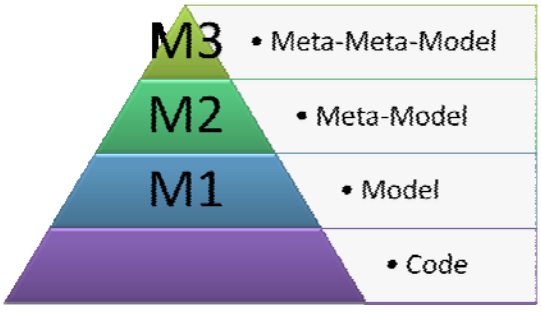

Figure 1. Model's Meta-levels

The IT professionals want to develop applications with models and some transformation between them. However, to build the applications in that way, it will be necessary to transform model to code. So it has three types of transformation: transformation between the same type of models (PIM to PIM), transformation between the different type of models (PIM to PSM or PSM to PIM) and transformation model to text (with this the model could be transformed into a code or into a model) [OMG, 2001].

Agent Modeling Languages

The modeling languages are created to make software development easier. For this reason, they are incorporated in methodology development. The methodology of software development is a group made up of process, techniques and helps to make this type of asset.

Sometimes the methodology's models are described with a natural language. For example, MAS-CommonKADS methodology [Iglesias et al, 1998] uses this kind of description to describe the models and other aspects of it [Gómez Sanz, 2002]. In this way, the automatic analysis of a specification is more difficult [Gómez Sanz, 2002]. It is also possible to define the model with semi-formal languages like UML. UML's language makes feasible to automate the model's analysis, so a model could be checked.

UML is a modeling language or meta-model to develop software using on OO paradigm. In 2000, James Odel, H. Van Dyke Parunak and Bernahard Bauer proposed the AUML modeling language. AUML is an UML extension to MAS (Multi-Agent System) [Odell et al, 2000]. A modeling language's extension redefines the own language, taking off some elements and adding others. The goal of these changes is to adapt the modeling language to new needs.

This modeling language has three levels. Each layer consists of one or more models. In the first layer the protocols are specified as an interaction between roles. In the second layer the interactions between agents are defined. The last layer is used to specify the internal process of each agent.

A protocol is a tidy group of messages that are changed between two entities. In AUML the entities are the roles. An agent could have one or more roles.

To represent the interactions between agents, three different models are used: sequence diagram, collaboration diagram and activity diagram.

The sequence diagrams are used to represent a temporal sequence of messages between agents. With the collaboration diagram, it defines the sequence of messages between agents, but this sequence is not temporal. The activity diagram, which represents a sequence of messages, the sequence diagram are different because with the first one it is able to have an explicit control of the threads. This control is important to model complex models.

To specify the internal processes of agents, the sequence diagram and the state diagram are used. With the sequence diagram, the execution process orders are defined. The state diagram can also be used to specify an agent's process. To do it, the different states of an agent and their transitions are defined.

No Agent Oriented development methodologies use all AUML's models. There are methodologies which use some AUML models, an example is ROADMAP. Following that the models use in the principal AO methodologies are presented. The studied methodologies are: GAIA, RORADMAP, MESSAGE, INGEIAS, TROPOS and SODA.

Gaia can be the most influential methodology to analysis systems as an organization [Bernon et al, 2005]. The Gaia organization's made up of a group of roles which are assigned to an agent. In this methodology the models are used in analysis and in the design phase. These models are: interaction model and role model in the analysis phase and agent model, services model and relations model in the design phase. [Wooldridge et al, 2000]

In figure 2 you can appreciate the different relationships between models. In the interaction model the protocols are defined. A protocol is a relationship between roles, after they are specified in the role model. The agent model is used to identify the kind of system's agents. A kind of agent implements a role group each. The role's activities

are specified in the service model. A service might need one or more protocols. To finish, in the relations model, the agents are associated between them. It is possible to deduce the last model from the iterations model, roles model and agent model. The relations model's objective is to find blocks. These blocks can appear if some kind of agent has too big a task. [Wooldridge et al, 2000]

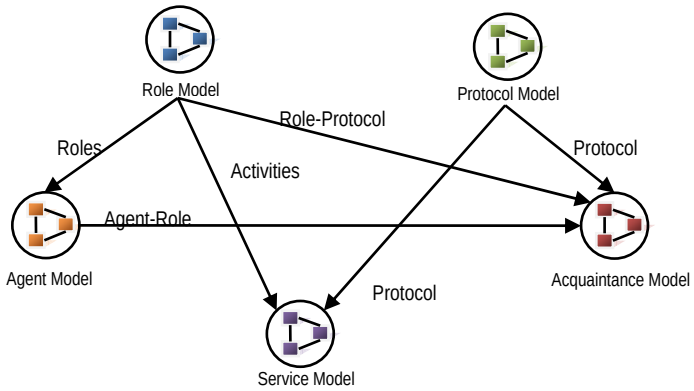

Figure 2. Relationship between GAIA's models [Wooldridge et al, 2000]

GAIA's methodology has a few disadvantages, which are resolved by ROADMAP [Juan et al, 2002]. This methodology modifies GAIA to improve the complex open system's process development.

The methodology ROADMAP has models to specify, analyze and design. Specification phase and analysis phase use a case model, environment model, knowledge model, roles model, protocol model and interaction model. In the design phase, it adds detail into the analysis phase's models until all necessary information for the system development is reflected. The previous phase also uses three new models: agent model, service model and acquaintance model. The last three models are extracted from GAIA.

The use-case model definition does not change in respect to it's specification in UML. The author only modifies its interpretation. In UML, the user interacts with the software to work. In the new interpretation, the user interacts with a group of agents which have all the knowledge and functionalities required. This abstraction is used to make modeling of the problem easier.

The environment model is defined, with the use-case model's information, the different environment zones where the system will execute. These zones are hierarchically specified, at first a small group of zones are defined; later the sub-zones are specified and in this way until enough detail of the environment is presented. The relationship zones are modeled using UML's class diagram: hierarchical relationship, aggregation, etc.

From the two previous models the required knowledge can be deduced and the agent's behavior in each zone for each use-case. These knowledge and behaviors are represented in the acquaintance model.

In ROADMAP add hierarchy to the role model of GAIA. In this new model a father role can use all protocols for its sons and make their activities.

The ROADMAP's interaction model is not the GAIA's interaction diagram. This model is the AUML's interaction model, where the messages are exchanged between roles. The protocol model used in ROADMAP is the GAIA's protocol.

MESSAGE is a generic methodology for MAS development. This methodology is created because it was necessary for the telecommunication industry. To specify the system is need five view: organization view, goal and task view, agent/role view, interaction view and domain view.

To define all mentioned views in the previous paragraph the MESSAGES methodology uses eight nodes and four kind of relationship. Only one node is not a MAS's node, the name of this node is "resource". The resource node represents a non autonomy entity as data base or external programs. [AUML Web Site, 2003]

In the organization diagram the relationship between agents, organization, roles and resources are defined. These elements and their dependences are present in the goal/task view. The agent/role view is used to specify the goals and tasks to each pair of agent/role. Interaction model allows defining the interaction between roles. To finish, the domain view is used to define domain's necessary concepts to construct the system. Every model except one use a meta-model specifies for this methodology, only the domain view use a UML model, the class model.

In 2002 a doctoral thesis from Jorge J. Gómez Sand was introduced [Gómez Sanz, 2002]. This thesis proposed a new methodology which is called INGENIAS, and is for the MAS development. This methodology uses five meta-models for the system definition: organization meta-model, environment meta-model, agent meta-model, interaction meta-mode, goal/task meta-model. The different models have a relation between themselves using them entities, for example, an interaction model and an agent model have a connection if the agents from the agent model are in the interaction model.

The agents, the roles and the groups which make the system can be specified because of the organization's model. This model is based in the AALAADIN meta-model [Ferber et al, 1998]. In this one, an agent can be in one o more groups, and a group can have one or more roles. An agent which is inside a group only can have a role and a role only can have an assigned group. An organization is made up of groups and the agents from those groups.

The environment is defined thanks to the meta-models where the relationship between the agents, the groups, resources and applications are. The resources are in an agent or in a group.

In the goal/task model it is specified how the state of an agent changes in time. The task can change an agent's state. This modification can end the goal. An objective could be made up of the same sub-goals. In the same way, a task can be made up of some sub-jobs. So, an agent can execute a task's group which should help to end the group of objectives.

To define the relationship between the agents the interaction's model is used. This model can be represented for different meta-models: UML's diagram collaboration, GRASIAS's diagram interaction or the AUML's diagram protocol. The authors don't give their preference between any meta-models exposed.

To finish, the meta-model of the agents can specify the relationship between an agent in particular, the task which it can executed and the goals which can found. All of these meta-models have a connection doing a group which can make a whole system definition.

Discussion

It can be shown that the methodologies which were presented in the last section are based, more or less, in the OMG technology. Some of these methodologies redefine the model's language. Other of these methodologies uses MOF to specify their own modeling languages. But none use the transformation that MDA proposes.

UML is a common visual language for the description of any component from an application which is going to be developed with an OO paradigm. In other words, UML can be considered like a language to define the implementation templates [Thomas, 2004]. So, if the meta-models from UML are used, the final goal does not change. These models only can add, delete components or modify the models interpretation.

On the other hand, if the models created are not complement with transformations which were exposed about MDA, the goal found for the language created will be the same as the goal discussed in the last paragraph.

The MDA's objective is the whole definition of an application which is independent of the technology or the paradigm chosen for the implementation. The PSM's models allow you to define the paradigms and technologies which must be used in each implementation. [OMG, 2001]

The modeling used in AO is a way to structure the computer application: data, algorithms, etc. All these elements can be implemented with PIM models. These models allow to the programmers to block out of hardware platform or software. With this, the design software is closer than before to human language. In other words, a developer can use as application's element a car; the developer does not need to know if this car is an object or an agent.

Conclusion

In this work it is presented what model driven architecture is and its advantages. After each model used in the different methodology which was proposed for AO paradigm was studied.

AO paradigm is after OO paradigm. At first this one was programmed using OA paradigm. Later methodologies to make easier the software development with this paradigm appeared. Soon after, textual models were incorporated to the methodologies. At the end these add graphic models to themselves. The graphic models allow to have a visual communication and to have as well a better communication between requirement and implementation.

Some authors affirmed that AO is closer to owner world's view that OO is [Jennings, 2000]. Anyway AO is still based in variables and loops used (for, while, etc). *It* can also say that to develop applications with AO paradigm it is necessary to think in declare functions and variables instead use the natural environment's elements (bread, cars, etc.).

It is obvious that the methodologies and the models studied in this document have improved the AO applications development. But as it happen with the paradigms this is far from the way that a person understands the process which wants to automate.

To finish we can affirm that it is necessary to study the way to build a group of meta-models and the transformations to allow the model driven development uses AO paradigm.

Bibliography

[AUML Web Site, 2003] AUML Web Site Modeling Notation Source MESSAGE. In AUML Web Site. - AUML, Marzo 12, 2003. - Febrero 20, 2009. - http://www.auml.org/auml/documents/.

[Bernon et al, 2005] C. Bernon, M. Cossentino and J. Pavón. An Overview of Current Trends in European AOSE Research. In Informatica. - Ljubljana : [s.n.], 2005. - Vol. 29. - pp. 379-390 .

[Cossentino, 2005] M. Cossentino. From requirements to Code with the PASSI Methodology. In Agent-Oriented methodologies / ed. Henderson-Sellers B and Giorgini P. - [s.l.] : Idea Group Publishing, 2005.

[Ferber et al, 1998] F. Jacques and G. Olivier. A meta-model for the analysis and design of Organizations in multi-agent systems. In Third International Conference on Multi Agent Systems (ICMAS'98). - 1998. - pp. 128-135.

[Gómez Sanz, 2002] J. Gómez Sanz. Multi-Agent System Modelling (In Spanish: MODELADO DE SISTEMAS MULTI-AGENTE). In Tesis Doctoral / Sistemas Informáticos y Programación. - Madrid : [s.n.], 2002.

[Iglesias et al, 1998] C. Iglesias, M. Garijo, J. Gonzales, J. Velasco. Analysis and Design of Multiagent Systems using MAS-CommonKADS. In 4th International Workshop on Agent Theories, Architectures, and Languages.. - Londres : illustrated, 1998. - pp. 313-328.

[Jennings, 2000] N. Jennings. On agent-based software engineering. In Artificial Intelligence. - [s.l.] : Elsevier Science B.V., 2000. - 117. - pp. 277–296.

[Juan et al, 2002] T. Juan, A. Pearce and L. Sterling. ROADMAP: Extending the Gaia Methodology for Complex Open Systems [Conference] // First International Joint Conference on Autonous Agents & Multi-Agent systems. - [s.l.] : ACM Press, 2002. - pp. 3-10.

[Kent, 2002] S. Kent. Model Driven Engineering. In Proceedings of the Third International Conference on Integrated Formal Methods / ed. Butler M, Petre L and Sere K. - Turku : [s.n.], 2002. - pp. 286-298.

[Odell et al, 2000] J. Odell, H. Van dyke Parunak and B. Bauer. Extending UML for Agents. In Proc. of the Agent-Oriented Information Systems, Workshop at the 17th National conference on Artificial Intelligence. - Austin : [s.n.], 2000.

[OMG, 2003] OMG The Object Management Group (OMG). OMG's MetaObject Facilit. – OMG. In - http://www.omg.org/docs/formal/02-04-03.pdf . Abril 02, 2003. - 1.4. - Enero 07, 2009.

[OMG, 2001] OMG The Object Management Group (OMG). MDA Specification. - OMG, 2001. - 1.0.1.

[Seidewitz, 2003] E. Seidewitz. What the model's mean?. In IEEE Software. - 2003. - Vol. 20. - pp. 26-32.

[Thomas, 2004] D. Thomas. MDA: Revenge of the Modelers or UML Utopia? In IEEE SOFTWARE. - [s.l.] : I E E E Computer Society, 2004. - p. 3.

[Wooldridge et al, 2000] M. Wooldridge, N. Jennings and D. Kinny. The Gaia Methodology for Agent-Oriented Analysis and Design. In Journal of Autonomous Agents and Multi-Agent Systems. – 2000, Vol. 3, Num.3 - pp. 285-312.

[Shannon, 1949] C. Shannon. The Mathematical theory of communication. In: The Mathematical Theory of Communication. Ed. C.E.Shannon and W.Weaver. University of Illinois Press, Urbana, 1949.

Authors' Information

Rubén Álvarez-González – Studient of Natural Computing Group. Faculty of Computer Science. Technique University of Madrid. ruben.alvarez.gonzalez@gmail.com

Miguel Angel Díaz Martínez – Professor of Computer Science School. Technique University of Madrid. mdiaz@eui.upm.es