Developing Multi-Agent Systems Based on MDA

Beatriz Alves De Maria Viviane Torres da Silva Carlos J. P. Lucena

Pontifical Catholic University of Rio de Janeiro, Rua Marquês de São Vicente 225, Brazil
{biamar,viviane,lucena}@inf.puc-rio.br

Abstract: In this paper, we propose an MDA based approach for developing multi-agent systems. MDA specifies a structured software development process divided in modeling stages. In the PIM stage, where platform independent models are specified, we propose to use an MAS modeling language called MAS-ML since it does not restrict or specify implementation platforms. In the PSM stage, where platform specific models are defined, we propose to use the UML modeling language. The MAS-ML models defined in the PIM stage are transformed into UML models at the PSM stage, based on an object-oriented framework for implementing MAS. In the last stage, the application code is generated from the UML models.

1 Introduction

Multi-agent systems (MAS) are becoming widely used for building complex and large-scale systems. Different applications' domains, like digital libraries, virtual markets and information systems in general, such as [2,16], were developed by using the MAS approach. For this reason, a variety of methodologies and others MAS modeling and implementing techniques have been proposed with the purpose of helping the developers in building such systems.

This paper proposes an MDA (Model Driven Architecture) [4] based approach for developing MAS. The MDA architecture considers models as the primary artifact and covers the complete life cycle of software systems. Four mains stages defining specific models compose MDA. The tracking of the models is accomplished through transformations of one model into another.

MDA development process explicitly separates models at three abstractions layers The most abstract model is the computational independent model (CIM) that are application conceptual models, independents of the computational characteristics of the solution. A CIM is refined into a platform independent model (PIM), corresponding to the second stage of the development process. PIMs are high-level models that contain already computational information about the system but are independent of implementation platforms. The less abstract model is the platform specific model (PSM), which correspond to the third stage, is model created based on PIM by including specific platform details. In the fourth stage of the MDA development process, PSMs are transformed into application code. In addition, MDA also defines a set of consecutive transformations that should be applied to the models in order to allow the transformation of high-level abstraction models into code.

Proceedings of the CAiSE'05 Forum - O. Belo, J. Eder, J. Falcão e Cunha, O. Pastor (Eds.)

The proposed development process defines the MDA stages and the transitions between the stages according to the characteristics of MAS. The proposed development process defines a MAS PIM, the transformation of these models into PSM, the specification of PSM and the transformation of PSM into code. The representation of CIM as well as the generation of CIM into PIM is beyond the scope of this paper.

In our proposed development process, we use the MAS-ML modeling language [12, 14] to model MAS. MAS-ML is a modeling language that extends UML, including agent-related abstractions. The MAS-ML models will represent the PIMs defined in MDA, since MAS-ML does not restrict or specify implementation platforms. To transform the MAS-ML models into PSMs, we propose to use an object-oriented framework called ASF (Agent Society Framework) [13]. The ASF framework defines a set of object-oriented modules that describes abstract classes that should be extended in order to implement multi-agent applications. By instantiating the ASF framework from MAS-ML models, these models are transformed into UML models that has several characteristics and depend on the framework used. Therefore, the UML models are PSMs. To represent the last stage of MDA approach, the transformation of PSMs into code, we automatically transform the UML models generated in the previous stage into objected-oriented code by using the XMI [6] technology.

The use of MDA in the MAS development process offers several advantages. The MAS-ML models that describe an application are PIMs that are portable to diverse systems and can be used to generate different computational models by applying various implementation platforms. Another advantage is the generation of different MAS viewpoints through the specification of models that have different abstraction levels. The MAS-ML models are high-level models that focus on the problem definition by describing the system in an agent-oriented level. The UML models are low-level models that focus on the solution of the problem by including implementation details and describing the system in an object-oriented level.

The article is organized as follows. Section 2 describes our main contribution by showing the MAS development process using MDA. Section 3 draws some conclusions and discusses future work.

2 Multi-Agent Systems and MDA

The development process of large-scale systems, as MAS, involves the construction of different models based on variety requirements. The transformations of a system specification into models and of these models into code are usually accomplished in a non-organized way, that is not easily adaptable to technology changes.

Since MASs are gaining widespread acceptance, it is necessary to create an MAS development process that clearly specifies the different process stages and the transitions between these stages. The process proposed in this section aims to specify the implementation of agent-oriented systems by applying the MDA approach.

The MAS development process proposed in this paper is illustrated in Fig. 1. In the CIM stage, the system can be described by using domain-specific ontologies or any other computational independent technology. For instance, ontologies define the con-

cepts and relationships related to the problem domain that can be used to specify a system. The definition of such concepts and relationships do not depend on the computational abstractions used to solve the system problem. It is not within the scope of this paper either to exemplify the CIM models of MAS or to demonstrate the transformation from CIMs into PIMs.

Next, the CIM models should be transformed into PIMs by using any platform independent modeling language, such as AUML [3] and MAS-ML. In our approach, we propose to use the MAS-ML modeling language. The main difference between MAS-ML and other MAS modeling languages is that MAS-ML defines organizations, roles, agents and environment as first-class entities. By defining these abstractions as first-class entities it is possible to define their properties, relationships, internal executions and interactions.

After the creation of PIMs, PSMs should be generated by choosing an implementation platform. In order to implement MAS, several architectures, frameworks and platforms, such as ASF, Jadex [8] and RETISINA [15], can be used. In our approach, the MAS-ML models are transformed into UML models by using the ASF framework. The ASF framework was chosen since it is possible to implement all MAS-ML abstractions. Other implementation platforms do not deal with the implementation of such abstractions. To implement MAS with Jadex from MAS-ML models, new transformation rules to convert the MAS-ML models in Jadex structure are requested, that can become very difficult since Jadex does not use the same abstractions as MAS-ML [1]. Finally, PSMs are transformed into code. The programming language will depend on the platform used to implement the system. In the proposed approach, MAS will be implemented in Java since this is the language used by ASF.

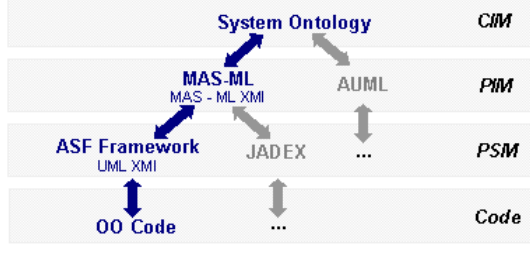

Fig. 1. MAS development process based on MDA

2.1 PIM

MAS-ML is a modeling language focused to MAS. This language is an extension of UML by including agent-related concepts presented in the TAO conceptual framework [11]. Thus, by using MAS-ML it is possible to model not only objects, but also agents, environments, organizations and roles that are entities defined in TAO and usually found in MAS.

MAS-ML was chosen to model MAS PIMs due to three main factors. First, this language is not a platform dependent modeling language. Although MAS-ML uses agent and object-oriented abstraction, MAS-ML does not restrict the implementation

of its models to a specific implementation platform. Second, MAS-ML is a modeling language that is MOF [5] compliant. This characteristic facilitates the MAS development process since XMI [6] can be used during the development process. XMI provides a mapping from MOF to XML and, therefore, can be used to describe MOF compliant models. XMI is used in our approach to represent the MAS-ML models, to assist in the transformation from MAS-ML models into UML models, to represent the UML models and to help the transformation from UML models into code. Finally, as stated before, MAS-ML models abstractions such as organizations, environments and roles that are not adequately modeled in other MAS modeling languages [2,3]. Without modeling these abstractions it is not possible to model, for instance, agents playing different roles in different organizations.

2.2 Transforming PIMs into PSMs

After using MAS-ML to model PIMs, the models should be transformed into PSMs. The transformation of MAS-ML models into UML models occurs in two stages.

Fist Stage: Creating the MAS-ML XMI
The first stage consists of describing the all the MAS-ML models in a textual description by using the XMI format. In order to do so, it was necessary to create an MAS-ML DTD that specifies the MAS-ML models in XMI. The MAS-ML DTD extends the UML DTD according to the extensions proposed by MAS-ML to the UML metamodel. The XMI file created by using the MAS-ML DTD represents the MAS-ML models and is called MAS-ML XMI.

Second Stage: Creating the UML XMI
In this stage, the MAS-ML XMI generated in the previous phase is converted into a UML XMI through the instantiation of the ASF Framework. This transformation uses a UML XMI file that contains the framework specifications. This file describes all classes and relationships among classes that are defined in the framework. During the transformation, the UML XMI file of the framework is extended with the application details that are described in MAS-ML XMI. The extended file characterizes the instantiation of the framework and the implementation of an application. The UML XMI has details related to the implementation platform.

2.3 PSM

The UML XMI generated in the previous stage represents a PSM of the application. By using graphical tools, such as Rational Rose [10] and Poseidon [9], which import XMI files, it is possible to generate UML models of the application.

The UML model created in this stage is a UML class diagram that contains the ASF framework classes and the classes related to the application that instantiate the framework. All application entities, properties and relationships modeled on the three MAS-ML structural diagrams generate the UML class diagram of the application.

2.4 Transforming PSMs into code

In the final stage of the MAS development process, the UML models are transformed into code. This kind of transformation corresponds to the last stage of the MDA approach that transforms PSMs into code. Almost all graphic tools that import UML XMI, such as Rational Rose and Poseidon, are capable of automatically generating code from a UML XMI.

3 Conclusion and Future Work

The MAS development process presented in this paper intends to provide an approach for modeling and implementing MAS by using MDA. The proposed process is composed of three stages. In the first stage, our approach proposes the use of MAS-ML to model MAS by creating PIMs. MAS-ML was chosen since it models several MAS models that are not (appropriately) modeled in other MAS modeling languages. In the second stage, the MAS-ML models are transformed into UML models by using the ASF framework. The ASF framework was chosen since it defines a set of object-oriented models that implement the MAS entities modeled in MAS-ML. Finally, the UML models are automatically transformed into code in the third stage proposed by the MAS development process.

The MAS development process takes advantage of the MDA approach used to define the process. Four main characteristics can be enumerated: (i) portability and reusability: MAS-ML models are portable to different implementing platforms because such models describe MAS without including implementation details. Thus, MAS-ML models can be reused by several developers to implement the system by using different platforms; (ii) interoperability: The use of XMI to describe models created during the process provides interoperable PIM and PSM models. MAS-ML models as well as UML models are interoperable due to the use of MAS-ML XMI and UML XMI; (iii) low coupling: Due to the intrinsic characteristics of MDA, PIMs (or conceptual models) and PSMs (computation models) are low coupling. In our approach, MAS-ML models are PIMs that do not include details related to a specific platform. Such details are present just in UML models, PSMs.

To make the construction of MAS feasible by using our approach, a modeling tool was developed to support the design and implementation of MAS. The tool allows the designer to graphically model MAS systems by using MAS-ML and to implement them while generating Java code by using the ASF framework.

With the aim of enhancing the tool that gives support to the development of MAS by using MDA, several important improvements should be carried out. First of all, the transformer that generates code from MAS-ML models should also consider the MAS-ML dynamic diagrams. Second, the tool should make the visualization and also the modification of the UML models that represent the system implementation feasible. In addition, the tool should provide a model checker to analyze and verify the consistence among the different models (MAS-ML models and UML models).

References

1 Braubach, L., Pokahr, A. and Lamersdorf, W. *Jadex: A Short Overview.* Main Conference Net.ObjectDays, AgentExpo, 2004.

2 He,M., Jennings,N.R. and Leung,H. *On Agent-Mediated Electronic Commerce*. IEEE Trans on Knowledge and Data Engineering, 15:4, 2003.

3 Odell,J., Dyke, Parunak,H. and Bauer,B. *Extending UML for Agents*. AOIS Workshop at AAAI, 2000.

4 OMG, *OMG MDA Guide*. Version 1.0.1. 2002. Available at: http://www.omg.org/docs/omg/03-06-01.pdf. Accessed in: 11/2004

5 OMG, *Meta Object Facility (MOF) 2.0 Core Specification*. Version 20, 2003. Available at: http://www.omg.org/docs/ptc/03-10-04.pdf. Accessed in: 11/2004.

6 OMG, *OMG XML Metadata Interchange*. Version 1.2. 2003. Available at: http://www.omg.org/docs/formal/02-01-01.pdf. Accessed in: 11/2004.

7 OMG, *OMG Unified Modeling Language Specification*. Version 1.5, 2003. Available at: http://www.omg.org/docs/formal/03-03-01.pdf. Accessed in: 11/2004.

8 Pokahr, A., Braubach, L. and Lamersdorf and W. *Jadex: Implementing a BDI-Infrastructure for Jade Agents*. Exp in Search of Innovation, 3:3, 2003. Available at: http://exp.telecomitalialab.com. Accessed in: 11/2004.

9 Poseidon. Available at: http://www.gentleware.com. Accessed in: 11/2004.

10 Rational Rose. Available at: http://www-306.ibm.com/software/rational/. Accessed in: 11/2004

11 Silva,V.; Garcia,A.; Brandao,A., Chavez,C., Lucena,C. and Alencar, P. *Taming Agents and Objects in Software Engineering.* Software Engineering for Large-Scale Multi-Agent Systems, LNCS 2603, 2003.

12 Silva,V. and Lucena,C. *From a Conceptual Framework for Agents and Objects to a Multi-Agent System Modeling Language*. Journal of Autonomous Agents and Multi-Agent Systems, Kluwer Academic Publishers, ISSN 1387-2532, 9:1 and 2, 2004.

13 Silva,V., Cortés,M. and Lucena,C. *An Object-Oriented Framework for Implementing Agent Societies*. MCC32/04. Technical Report, PUC-Rio. Rio de Janeiro, Brazil, 2004.

14 Silva, V., Choren, R. and Lucena, C. *Using the MAS-ML to Model a Multi-Agent System*. Lucena, C., Garcia, A.; Romanovky, A., Castro, J., Alencar, P. (Eds.) Software Engineering for Large-Scale Multi-Agent Systems II, LNCS 2940, Springer, 2004.

15 Sycara,K., Paolucci,M., van Velsen,M. and Giampapa,J. *The RETSINA MAS Infrastructure.* Special joint issue of Autonomous Agents and MAS, 7:1 and 2, 2003.

16 Wooldridge,M. and Jennings,N. *Applications of Intelligent Agents*. Agent Technology: Foundations, Applications, and Markets, 3:28, 1998.