Modeling Tools for Platform Specific Design of Multi-Agent Systems

Geylani Kardas[1], Erdem Eser Ekinci[2], Bekir Afsar[2], Oguz Dikenelli[2], and N. Yasemin Topaloglu[2]

[1] Ege University, International Computer Institute, 35100 Bornova, Izmir, Turkey
geylani.kardas@ege.edu.tr
[2] Ege University, Department of Computer Engineering, 35100 Bornova, Izmir, Turkey
erdemeserekinci@gmail.com,
{bekir.afsar,oguz.dikenelli,yasemin.topaloglu}@ege.edu.tr

Abstract. In this paper, we introduce platform specific modeling and code generation tools for the model driven development of multi-agent systems (MAS). These tools enable agent developers to model their MASs for the SEAGENT and the JADEX agent platforms based on the semantics and design principles of these platforms. The toolkit also provides automatic code generation for agent developers in order to implement their MASs on the target platforms. Generated codes may vary on type (e.g. Java class files, XML documents or ontologies) according to each platform's requirements.

1 Introduction

Model Driven Development (MDD), which aims to change the focus of software development from code to models, may also provide rapid and easy development of Multi-agent Systems (MAS). However, such a development process should be supported by modeling tools in order to assist developers during their design. Many researchers in Agent-oriented software engineering (AOSE) community propose model driven approaches (e.g. [1], [2] and [3]) in MAS development and also introduce related modeling tools for their approaches. This study contributes to these efforts by introducing new graphical modeling tools for different MAS platforms.

Based on the well-known MDD realization called Model Driven Architecture (MDA)[1], our ongoing work aims to define a MAS development process which will consider the ontologies as the basic components of the MAS architecture. The proposed development process includes definition of metamodels for each layer of the MDA architecture; called *the Computation Independent Model (CIM)*, *the Platform Independent Model (PIM)*, and *the Platform Specific Model (PSM)* and provides modeling software tools for modeling in each layer. This study introduces the software modeling tools which can be used at the PSM level. The developers can use these tools to model MASs for SEAGENT [4] and JADEX [5] platforms and obtain auto-generated software codes of their agent systems for the related platforms.

[1] http://www.omg.org/mda/

The paper is organized as follows: In Section 2, we briefly discuss design and use of the tools. Visual modeling and code generation for SEAGENT and JADEX agents are discussed in Section 3 and 4 respectively. Section 5 covers related work. Conclusion and future work are given in Section 6.

2 Design and Use of the Modeling Tools

The modeling tools introduced in this paper are developed on Eclipse[2] platform by using Graphical Modeling Framework (GMF)[3]. GMF is a framework for building graphical modeling editors for various domains. In our study, we (1) provide domain models of SEAGENT and JADEX agent platforms as Ecore[4] metamodels, (2) prepare graphical elements representing the agent domain elements and their relations, (3) map agent components with the related graphical nodes and (4) generate the agent modeling editors as the Eclipse plug-ins.

The developers use editors for visually modeling their agent systems. Agent domain elements and their relationship links are represented in the editor palettes. The developers choose desired elements and links from palettes and visually create their agent models as will be discussed in the following sections. The editor environment also supports developers in model consistency and prevents wrong relation establishments between agent model elements.

The outputs of the visual modeling are the model documents for the designed agent systems. The next step is the automatic generation of agent software codes, ontology documents and any other system files from visually created agent models. We employ the Abstract Syntax Tree (AST) and related parser in the Eclipse Java Development Tools (JDT) for automatic generation of SEAGENT agent software, plan documents and ontology files. On the other hand, MOFScript[5] is used to generate JADEX agent description files and agent plan codes. Above mentioned code generations are completely abstract from the developers and hence developers do not deal with the generation process.

In order to illustrate practical use of the introduced tools, let us consider a multi-agent based e-barter system. A barter system is an alternative commerce approach where customers meet at a marketplace in order to exchange their goods or services without currency. An agent-based e-barter system consists of agents called *Customer* that exchange goods or services of owners corresponding to their preferences. The *Barter Manager* agent manages all trades in the system. This agent is responsible for collecting barter proposals, matching proper barter proposals and tracking the bargaining process between customer agents. In the following sections, the registration scenario of customer agents with the Barter Manager agent is discussed for the demonstration of the modeling tools. Interested readers may refer to [6] for the complete design and description of the related e-barter MAS system.

[2] http://www.eclipse.org/
[3] http://www.eclipse.org/gmf/
[4] http://www.eclipse.org/modeling/emf/
[5] http://www.eclipse.org/gmt/mofscript/

3 Modeling SEAGENT Agents

SEAGENT [4] is an agent development platform in which Semantic Web enabled MASs can be developed in an interactive and test-driven manner. SEAGENT Agents manage all of their internal knowledge using OWL[6] ontologies and can interact with the semantic web services. Development of SEAGENT MASs includes MAS modeling according to 4 viewpoints called *Organization*, *Plan*, *Protocol* and *Domain*. Within *Organization Model*, agents, organizations, roles, goals and responsibilities of the roles are declared. *Plan Model* provides internal planning of the agents based on the Hierarchical Task Network (HTN) [7] paradigm. *Protocol Models* include protocols for the interactions between SEAGENT agents. Finally, *Domain Model* composes the ontological representation of the related business domain elements**.** Our modeling toolkit includes editors for all 4 base models of the SEAGENT. However, due to space limitations only editor for the Plan model is discussed in here.

In the SEAGENT framework, agents execute their tasks according to HTN. HTN planning creates plans by task decomposition. This decomposition process continues until the planning system finds primitive tasks that can be performed directly. In the HTN formalism, there are two types of tasks: complex tasks called *behaviours* and primitive tasks called *actions*. Each plan has a root task which is a behaviour itself consisting of subtasks (actions) that are composed to achieve a predefined goal. Behaviours hold a “reduction schema” data structure that defines the decomposition of the complex task to subtasks and the information flow between these subtasks and their parent task. Actions, on the other hand, are primitive tasks that can be executed by the SEAGENT planner using the Java Reflection API.

Our toolkit provides the editor called HTN planner for the visual modeling of the SEAGENT agent plans (Fig. 1). In this editor, agent developers can create plan models including agent behaviours, actions and linkage between behaviour and their subtasks by first selecting proper elements from the component palette and then entering their attribute values. Fig. 1 also depicts the HTN plan model of the Barter Manager agent for the customer registration scenario. The Barter Manager has the behaviour “BHResponseRegistration” which can be decomposed into three actions called “ACCheckExistence”, “ACRegisterCompany” and “ACAcceptCompany Registration”. In these actions, the agent first controls whether the customer already has been registered, then checks customer’s registration data according to the related ontology if it is a new customer and finally realizes the registration.

SEAGENT platform executes the artifacts which are stored in two different representation formats: OWL and Java. Considering HTN plans of SEAGENT agents, our editor has built-in model to text translators for automatic generalization of agent behaviour ontologies and template Java class codes of the related behaviour’s actions. As shown in Fig. 1, the developer just needs to right-click on the proper model element, chose code generation and set required parameters (e.g. source code folder, namespace and action method name). Then the editor automatically generates required plan ontologies (for the root behaviours) and action Java codes (for subtasks).

[6] http://www.w3.org/2004/OWL/

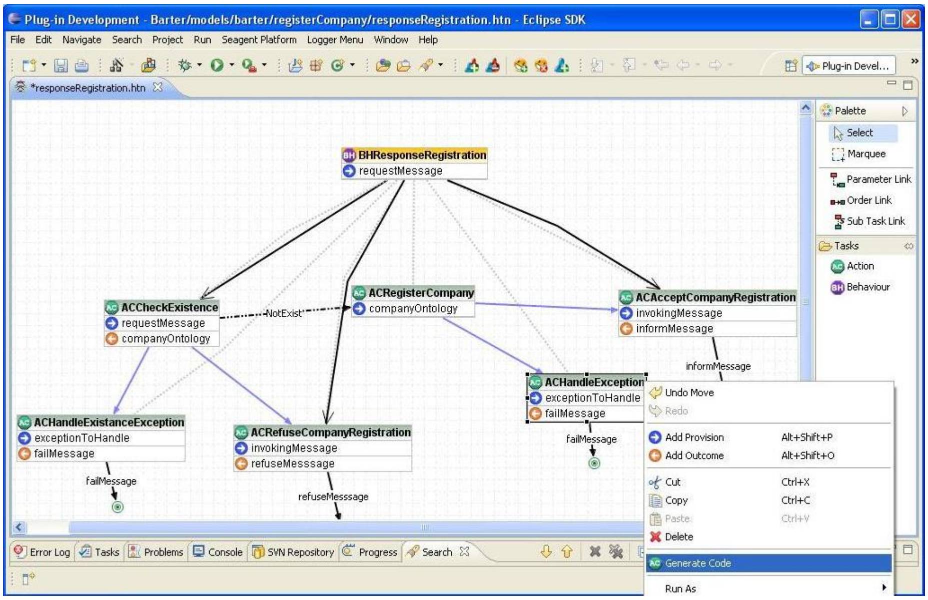
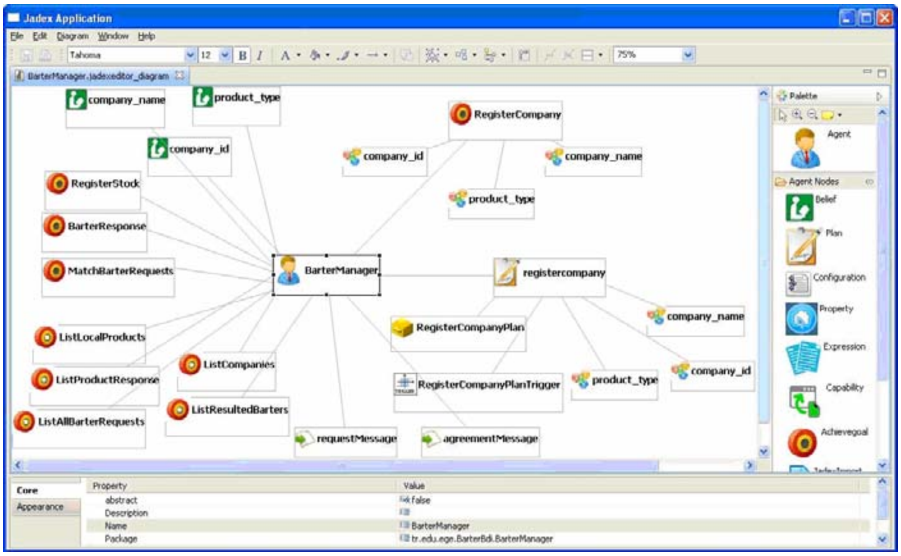


Fig. 1. Modeling SEAGENT Agent HTN plans

4 Modeling JADEX Agents

Our toolkit includes modeling and automatic code generation utilities for another agent development platform called JADEX. JADEX [5] provides an engine and a programming platform for developing well-known Belief-Desire-Intention (BDI) [8] agents. The development of JADEX agents is based on a hybrid approach in which declaration of static agent properties and programming of executable agent plans take place. Declaration of static agent properties is given in files called Agent Definition Files (ADF). An ADF file is written using XML and specifies the BDI model of the related agent. On the other hand, agent plans are executable components and they are given in Java program files. In order to assist agent developers in design and development of JADEX BDI agents, we provided a graphical modeling editor and an automatic code generation tool based on the design principles discussed in Section 2. Developers can model the BDI architecture of the agents by choosing appropriate model elements from the component palette, drawing the relationship links between elements and setting model attributes. The editor environment (Fig. 2) provides visualization of the BDI model of the agents and supports easy and efficient development of JADEX agents. The editor also prevents users from wrong relationship establishments between BDI elements. For example, *Body* of the plan node can only be linked with the *Plan* node according to the BDI semantics of the JADEX metamodel. Hence the editor does not allow linking *Body* nodes with any other nodes except the Plan node.

Considering our case study, the BDI model of the Barter Manager agent can be visualized as given in Fig. 2. Registration information received from the customer agents is given as the initial facts (beliefs) for the Barter Manager (at the upper left of



Property	Value
abstract	false
Description	
Name	BarterManager
Package	tr.edu.ege.BarterBdi.BarterManager

Fig. 2. Modeling JADEX BDI Agents

the model in Fig. 2). The model also includes agent's goals to be achieved (at the lower left of the model), the plan structure (at the right of the model) and their relations. After modeling of the JADEX Agent system is completed, developers can obtain the ADF and related agent plan's Java classes by using the integrated code generator. Agent BDI models are stored as Ecore files. The code generator uses these files as input, applies a model-to-text transformation using MOFScript and finally outputs related ADF and template Java class files for the designed agent system.

5 Related Work

Several MAS software tools exist for modeling agent systems according to various AOSE methodologies e.g. agentTool[7] for O-Mase, IDK[8] for INGENIAS, TAOM4e[9] for Tropos and PDT[10] for Prometheus. These tools mostly cover analysis and design of MASs and a few of them only consider implementation and provide code generation for specific agent platforms. On the other hand, studies like [1] and [2] utilize tools for MDD of MASs but only one MAS development platform and its PSM are taken into consideration. The graphical modeling editor introduced in [9] has the same design principles with our toolkit. The editor is GMF based and supports MDD of MASs at the PIM level. The toolkit introduced in this paper contributes to those noteworthy studies by supporting platform specific modeling and implementation of MASs according to MDA principles.

[7] http://agenttool.cis.ksu.edu/
[8] http://sourceforge.net/projects/ingenias/
[9] http://sra.itc.it/tools/taom4e/
[10] http://www.cs.rmit.edu.au/agents/pdt/

6 Conclusion and Future Work

In this paper, platform specific modeling and code generation tools for the MDD of SEAGENT and JADEX agents are introduced. These tools enable agent developers to design and implement their MASs on different target platforms which vary on the design principles such as BDI, HTN and Semantic Web integration. In our future work, we plan to develop another modeling toolkit for the platform independent modeling of MASs. The toolkit will be again based on the GMF and use the enhanced version of the PIM discussed in [3]. This new toolkit will be integrated into the development environment introduced in this paper.

Acknowledgements

This work is funded by The Scientific and Technological Research Council of Turkey (TUBITAK) Electric, Electronic and Informatics Research Group (EEEAG) under grant 108E141.

References

[1] Perini, A., Susi, A.: Automating Model Transformations in Agent-Oriented Modeling. In: Müller, J.P., Zambonelli, F. (eds.) AOSE 2005. LNCS, vol. 3950, pp. 167–178. Springer, Heidelberg (2006)

[2] Pavon, J., Gomez, J., Fuentes, R.: Model Driven Development of Multi-Agent Systems. In: Rensink, A., Warmer, J. (eds.) ECMDA-FA 2006. LNCS, vol. 4066, pp. 284–298. Springer, Heidelberg (2006)

[3] Kardas, G., Goknil, A., Dikenelli, O., Topaloglu, N.Y.: Modeling the Interaction between Semantic Agents and Semantic Web Services using MDA Approach. In: O'Hare, G.M.P., Ricci, A., O'Grady, M.J., Dikenelli, O. (eds.) ESAW 2006. LNCS (LNAI), vol. 4457, pp. 209–228. Springer, Heidelberg (2007)

[4] SEAGENT MAS Development Framework, `http://seagent.ege.edu.tr/`

[5] JADEX BDI Agent System, `http://jadex.informatik.uni-hamburg.de/`

[6] Cakirlar, I., Ekinci, E.E., Dikenelli, O.: Exception Handling in Multi-Agent Systems. In: 9th Workshop on Engineering Societies in the Agents World, Saint-Etienne (2008)

[7] Williamson, M., Decker, K., Sycara, K.: Unified Information and Control Flow in Hierarchical Task Networks. In: AAAI 1996 Workshop, pp. 142–150 (1996)

[8] Rao, A., Georgeff, M.: BDI Agents: From Theory to Practice. In: First International Conference on Multi-Agent Systems, San Francisco, pp. 312–319 (1995)

[9] Warwas, S., Hahn, C.: The concrete syntax of the platform independent modeling language for multiagent systems. In: ATOP 2008 Workshop, Estoril, pp. 94–105 (2008)