The DSML4MAS Development Environment

Stefan Warwas, Christian Hahn
German Research Center for Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3
66123 Saarbrücken, Germany
{stefan.warwas, christian.hahn}@dfki.de

ABSTRACT

This paper presents the DSML4MAS *Development Environment* (DDE) which is a model-driven framework for the development of multiagent systems based on the *Domain Specific Modeling Language for Multiagent Systems* (DSML4-MAS).

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems; D.2.6 [**Programming Environments**]: Graphical environments

General Terms

Design

Keywords

Agent Oriented Software Engineering, Development Environment, Model-driven, Multiagent System

1. INTRODUCTION

Even if the research on multiagent systems (MASs) is a very active area, only little research has been done with respect to the development of adequate tools to support the design of agent-based systems. In particular, integrated development environment (IDE) support for developing agent systems is rather weak, and existing agent tools do not offer the same level of usability as state-of-the-art object-oriented IDEs [4].

The DSML4MAS *Development Environment* (DDE) bases on the *Domain Specific Modeling Language for Multiagent Systems* [1] (DSML4MAS) and is a model-driven framework for the development of MASs. The abstract syntax of DSML4MAS is defined by the *Platform Independent Metamodel for Agents* (PIM4AGENTS).

The functionality of DDE encompasses (i) the platform independent specification of MASs, (ii) model validation, (iii) model transformation and code generation, and (iv) execution of generated source code. The code generation facilities support the agent execution environments Jack[1] and Jade[2].

Section 2 provides an overview of the features of the development environment. An overview of the workbench is given in Section 3. Section 4 provides information about the execution platform support. Followed by Section 5 concluding this paper.

Cite as: The DSML4MAS Development Environment, Stefan Warwas, Christian Hahn, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009),* Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. 1379 – 1380

2. FEATURES

This section summarizes the core features of DDE.

Model-driven approach. DDE uses a model-driven development process to close the gap between design and code. PIM4AGENTS models are transformed to platform specific Jack and Jade models. In a second step, source code for Jack and Jade is generated. Finally, the generated source code can be edited and executed.

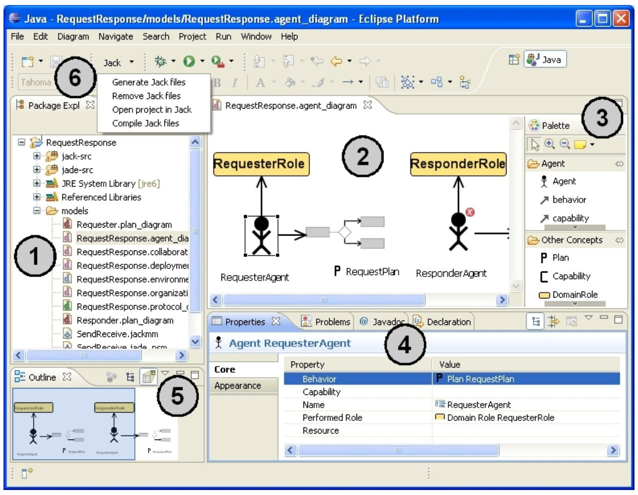
Reduction of complexity. To reduce the complexity of MAS design is one of the main objectives of the research area of agent-oriented software engineering. For this purpose, DDE offers several views on the MAS. Each view (e.g. agent view, protocol view, deployment view, etc.) focuses on a certain aspect and abstracts from others. Changes that affect several views are automatically propagated to the others.

Model validation. Many design errors of a MAS can already be captured at the model level. DSML4MAS offers a formal semantics (cf. [2]) that can be used to check the syntactic correctness of the created models. For this purpose, constraints based on the *Object Constraint Language*[3] (OCL) have been manually derived from the formal Object-Z specification of the DSML4MAS language to check the static semantics of the models. These constraints are automatically evaluated during design time and support the developer to produce well-formed models.

Service Oriented Architectures (SOAs). MASs do not exist in pure isolation. Instead, they need to be integrated and combined with other related technologies like for instance SOAs. Therefore, in [3] we demonstrated how to integrate Semantic Web services into DDE. The service description can be defined at design-time and invoked by agents.

Reusable components. DDE allows the user to reuse components (like plans, protocols, organizational structures, etc.) across several projects. This reduces development time and cost, and increases the quality of the components.

[1]http://www.aosgrp.com/products/jack/index.html
[2]http://jade.tilab.com/
[3]http://www.omg.org/docs/ptc/03-10-14.pdf



Property	Value
Behavior	Plan RequestPlan
Capability	
Name	RequesterAgent
Performed Role	Domain Role RequesterRole
Resource	

Figure 1: The DSML4MAS **development environment.**

Refinement. To support developers in specifying behaviors that conform to a certain protocol, we provide refinement functionalities that transform PIM4AGENTS protocols into PIM4AGENTS behaviors. The transformation saves a lot of redundant work.

Extensibility. DDE is seamlessly integrated into the Eclipse workbench. This implies that other researches can easily develop own extensions for DDE (e.g. transformations, views, model validation, etc.) and plug them into the Eclipse workbench. Furthermore, DDE directly benefits from new developments around the very active Eclipse modeling project[4] and other Eclipse tools.

Open source. At the time of writing this paper, we are finalizing a version that integrates all described features. We will launch an open source project until AAMAS'09. The source code will be published under LGPL. DDE will be available at http://dsml4mas.sourceforge.net.

3. THE DEVELOPMENT ENVIRONMENT

DSML4MAS covers the most import aspects of MASs such as agents, roles, collaborations of agents, protocols, behaviors, deployment aspects, and their environment. For each of these views, DDE offers a diagram that allows modeling MASs with the according language constructs.

Figure 1 depicts the graphical interface of DDE. It consists of several parts: (1) depicts the project explorer, the *models* folder that contains all diagrams and models, and the source folders for the generated Jack and Jade code. (2) shows the modeling area and (3) the palette that contains the language constructs and relations that are available in the current diagram. The properties view which shows the details of the currently selected diagram element is depicted in (4). The diagram outline that is useful for large diagrams is shown at (5). Finally, (6) depicts the Jack menu which offers special actions for the generated Jack code (see Section 4). The icon close to ResponderAgent visualizes a validation error. The error message can be looked up in the problems view of Eclipse.

[4]http://www.eclipse.org/modeling/

4. CODE GENERATION SUPPORT

To close the gap between design and implementation is one of the main objectives of DDE. The code generation within DDE is achieved through model transformations in accordance to the principles of Model-Driven Architecture. Two agent-based execution platforms are supported by DDE, however, other platforms can easily be supported in the same manner.

Jade. The code generation for Jade consists of (i) a transformation of the PIM4AGENTS model to a platform specific model (PSM) for Jade and (ii) a code generation step from the Jade PSM to Jade Java code. The generated Java code can be edited, executed, and debugged like any other Java code in Eclipse. We provide a mechanism for protecting manual changes at the generated source code.

Jack. The code generation for Jack consists of (i) a transformation of the PIM4AGENTS model to a Jack PSM and (ii) a code generation step from Jack PSM to Jack gCode. The generated gCode can be opened with the Jack development environment (e.g. to refine it). We are also able to generate Jack files from the Jack gCode within DDE. Jack files are extended Java files. Finally, the Jack files are compiled by the Jack compiler into Java files (inside Eclipse).

5. CONCLUSION

This paper discusses a novel development tool for designing MAS called DDE. The modeling language is based on a platform independent metamodel for MAS called PIM4AGENTS that defines the vocabulary of the language. Beside several features, code generation facilities are offered by DDE to seamlessly transfer the generated design into executable code. The demo will demonstrate (i) how to specify MASs with DDE, (ii) model validation, (iii) model transformation and code generation for Jack and Jade, and (iv) the execution of the generated code.

6. ACKNOWLEDGEMENTS

We want to thank Cristián Madrigal-Mora, Torsten Gründel, and Stefan Nesbigal who contributed to DDE.

7. REFERENCES

[1] C. Hahn. A domain specific modeling language for multiagent systems. In *Proceedings of 7th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2008)*, pages 233–240, 2008.

[2] C. Hahn and K. Fischer. The static semantics of the domain specific modeling language for multiagent systems. In *Proceedings of the 9th International Workshop on Agent-Oriented Software Engineering (AOSE 2008). Workshop at AAMAS'08*, 2008.

[3] C. Hahn, S. Nesbigall, S. Warwas, I. Zinnikus, K. Fischer, and M. Klusch. Integration of multiagent systems and semantic web services on a platform independent level. In *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Intelligent Agent Technologies (IAT 2008)*, 2008.

[4] M. Luck, P. McBurney, and J. Gonzalez-Palacios. Agent-based computing and programming of agent systems. In *Proceedings of Programming Multi-Agent Systems, Third International Workshop, ProMAS 2005*, volume 3862 of *Lecture Notes in Computer Science*, pages 23–37, Berlin et al., 2006. Springer.