DSML4BDI: A Modeling Tool for BDI Agent Development

Baris Tekin Tezel[1,2,] Moharram Challenger[1] and Geylani Kardas[1]

[1] International Computer Institute, Ege University
[2] Computer Science Department, Dokuz Eylul University, Turkey
baris.tezel@deu.edu.tr;
{moharram.challenger, geylani.kardas}@ege.edu.tr

Abstract. In this paper, a modeling tool, called DSML4BDI, for the model-driven development of Belief-Desire-Intention (BDI) agents, is introduced. As being an implementation of a domain-specific modeling language, DSML4BDI tool enables graphical modeling of all BDI components and relations with including the automatic construction of logical expressions and rules required for the system. In addition, its operational semantics, based on the Jason platform, leads to the automatic generation of all codes and other artifacts for the exact implementation of the modeled system. Evaluations performed by the users showed that the tool is capable of high generation performance and its use significantly decreases the development time.

Keywords: Domain-specific modeling language, Modeling Tool, Multi-agent System, BDI Agents, DSML4BDI.

DSML4BDI: KİH Etmenlerinin Geliştirilmesi için bir Modelleme Aracı

Barış Tekin Tezel[1,2,] Moharram Challenger[1] ve Geylani Kardaş[1]

[1] Ege Üniversitesi Uluslararası Bilgisayar Enstitüsü
[2] Dokuz Eylül Üniversitesi Bilgisayar Bilimleri Bölümü
baris.tezel@deu.edu.tr;
{moharram.challenger, geylani.kardas}@ege.edu.tr

Özet. Bu bildiride Kanı-İstek-Hedef (KİH) etmenlerinin model-güdümlü geliştirilmesini sağlayan DSML4BDI isimli bir modelleme aracı tanıtılmaktadır. DSML4BDI aracı tüm KİH bileşenlerinin ve ilişkilerinin görsel modellenmesini ve sistemin geliştirilmesi için gerekli mantıksal ifadelerin ve kuralların otomatik inşaasını sağlamaktadır. Bunlara ek olarak, aracın Jason platformuna dayalı işletimsel semantiği, modellenen sistemin eksiksiz uygulanması için gereken tüm kodların ve ürünlerin otomatik üretilmesine imkan vermektedir. Kullanıcılar tarafından gerçekleştirilen değerlendirme sonuçları aracın

CEUR-WS.org/Vol-2201/UYMS_YMO_2018_paper_34.pdf

yüksek üretim performansına sahip olduğunu ve sistem geliştirme süresini ciddi oranda düşürdüğünü ortaya koymuştur.

Anahtar Kelimeler: Alana-özgü modelleme dili, Modelleme Aracı, Çok-etmenli Sistem, KİH Etmenleri, DSML4BDI.

1 Introduction

Multi-agent System (MAS) development based on the belief-desire-intention (BDI) model [1] has found widespread adoption within Agent-oriented software engineering (AOSE) field since this model enables a good representation of agent internals and supports the composition of reactive and/or proactive agent behaviors. In BDI architecture, software agents constantly monitor their environment and respond to the changes in the environment. This reaction depends on agent's mental attitudes. An agent has three types of mental attitudes which are *belief*, *desire* and *intention*. Beliefs are information about an agent's itself, other agents and the environment that the agent is located. Desires express all possible states of affairs which might be achieved by an agent. One desire is a potential trigger for an agent's actions. Simply, desires are often considered as options for an agent. Finally, intentions represent the states of affairs which have been decided to work towards by the agent [2].

Programming environments and platforms such as BDI4Jade [3], Jadex [4], and JACK [5] facilitate the implementation of BDI agents. However, developers may still encounter with some difficulties especially on both the exact representation of the logic behind and the creation of agent beliefs and plans due to the limitations of using an imperative programming language like Java to express BDI foundations. Languages like AgentSpeak [6] suits well in logic programming needed for constructing a BDI architecture but this time the developers should deal with the composition of heavy logical and mathematical rule expressions. Jason [7], a Java-based interpreter for AgentSpeak may assist to the developers within this context, however, inefficacy of again using Java still remains as similar to abovementioned Java-based programming environments. Hence, in order to eliminate this deficiency, we present DSML4BDI modeling tool which provides model-driven development (MDD) of Multi-agent Systems (MAS) [8] purely based on the BDI principles, enabling the automatic construction of the required logic structures, rules, beliefs, etc., those are all abstract from the details of real execution environments. The tool is the implementation of a domain-specific modeling language (DSML) [9] with the same name, which has an operational semantics on Jason platform that leads to the automatic generation of executable Jason codes for the corresponding DSML4BDI model instances. A short movie demonstrating the use of the tool is available at: https://youtu.be/KrbgKBIf6us. This demonstration paper first discusses the features and the components of DSML4BDI and then gives a brief evaluation of its use in MAS development.

The rest of the paper is organized as follows: DSML4BDI tool is discussed in Section 2. An empirical evaluation of using DSML4BDI is given in Section 3. Section 4 includes the related work and Section 5 concludes the paper.

2 DSML4BDI Tool

Inside DSML4BDI tool [10], definition of both BDI main elements and their relations are provided with a graphical concrete syntax originating from a BDI metamodel introduced in [9]. Although the viewpoints of the original metamodel [9] remains same, some of the relations between the agent meta-entities are revised and extended in this study to provide the agent platform extensibility of the language. For instance, it is now also possible to construct model transformations from DSML4BDI's extended metamodel to the CArtAgO [11] infrastructure allowing artifact-based environments to be programmed and executed for MAS.

DSML4BDI tool is built on the open-source Eclipse Sirius platform [12] that enables having a graphical editor sourcing from DSML4BDI's abstract syntax encoded with Ecore and allowing the construction of dedicated editors including diagrams and tables.

It is possible to design a BDI MAS with using four different diagram types of DSML4BDI tool, each conforming to an agent-modeling viewpoint defined in the language. A developer may create a MAS diagram that is essential to present the MAS organization of BDI agents with including main elements and relationships. An Agent diagram shows internal agent structure composed of plans, beliefs, rules and goals. Properties and inner components of each agent's plans are modeled inside Plan diagrams. Finally, logical expressions, which can be used in any agent plan or rule, are created in Logical Expression diagrams. Some significant graphical notations pertaining to the abstract syntax elements, covered inside the DSML4BDI diagram types, are listed in Table 1.

Table 1 Some of the concepts and their notations for DSML4BDI

Concept	Notation	Concept	Notation
Agent	👤	MAS	👥
Action	⚙	Mental Note	🧠
Action Set	⚯	Message	✉
Belief	●	Plan	📄
Belief Base	▥	Plan Library	📚
Body	≡	Rule	⊤
Context	◎	Rule Set	▤
Event	📅	Formula	ƒ
Event Set	📅	Formula Set	Σ
External Action	⚙	Logical Expression	●
Internal Action	⚙	Logical Expression Set	●
Goal	◎	Goal Set	⚑

Figure 1 shows the graphical modeling environment of DSML4BDI tool. Current screenshot depicts the view of a BDI Plan diagram. Developers can create general MAS structures by simply drag-and-dropping required items (agents, plans, events etc.) from the palette residing at the right-side of the modeling environment. When the developer double-clicks an element on the MAS diagram, corresponding diagram for the related BDI element (e.g. agent, plan) is opened for modeling. Any change made in a view is immediately reflected to all other models of the MAS without any additional user intervention. Constraint checks and static semantics controls are automatically made by the tool. Figure 1 also contains a partial model covering BDI entities and relations required for the implementation of the well-known garbage collector MAS [4] in which the destructor agents inform the collector agents on the location of garbage in an environment while the collector agents pick the garbage and bring them to the destructor agents.

Figure 2 shows the modeling of a collector agent's BDI plan specifications inside a DSML4BDI plan diagram. Following the graphical modeling of MAS, DSML4BDI tool can automatically generate software codes and artifacts including ASL files, MAS2J specifications and Java classes which are all required for the exact implementation of the modeled BDI agents on Jason platform [7]. Generated codes for all AgentSpeak files are complete and ready-to-use and they can be directly executed on Jason platform without any code addition by the developers.

3 Evaluation

Usability of DSML4BDI tool was evaluated by benefiting from the MAS DSML evaluation framework proposed in [13] for the systematic assessment of both language constructs and tool usage. Our evaluation had two parts: (1) quantitative analysis including generation performance and development time measurement; and (2) qualitative assessment including user feedbacks via a questionnaire. We followed the study protocol again described in [13] for our case study preparation (including case study design, team selection and team preparation), case study execution, analysis and reporting. Development of a garbage collector MAS was considered as the case study. Two evaluator groups were employed each including four graduate students having at least 2 years MAS design and implementation experience. Group A utilized DSML4BDI while Group B did not use any domain-specific modeling tool during the case study. We compared the results considering generated code and development time for both groups to complete part (1) of the evaluation. Group A also filled a questionnaire for the usability assessment (for part (2)). Due to the space limitations, the results are briefly reported in this paper.

Varying from minimum 75% to maximum 100%, the average rate of generated Lines of Code (LoC) is 89% comparing with a complete implementation. It is also worth indicating that the distribution of the generation performance is directly related to the composition of MAS models created by each developer. Moreover, 92.7% of the overall required artifacts were automatically generated on average among Group A by just modeling with DSML4BDI. Group A completed the whole development process about 3 times faster than Group B on average. Gain in speed up was much more when specifically, implementation/generation and test phases are considered on where it was approximately 6 and 9 times respectively.

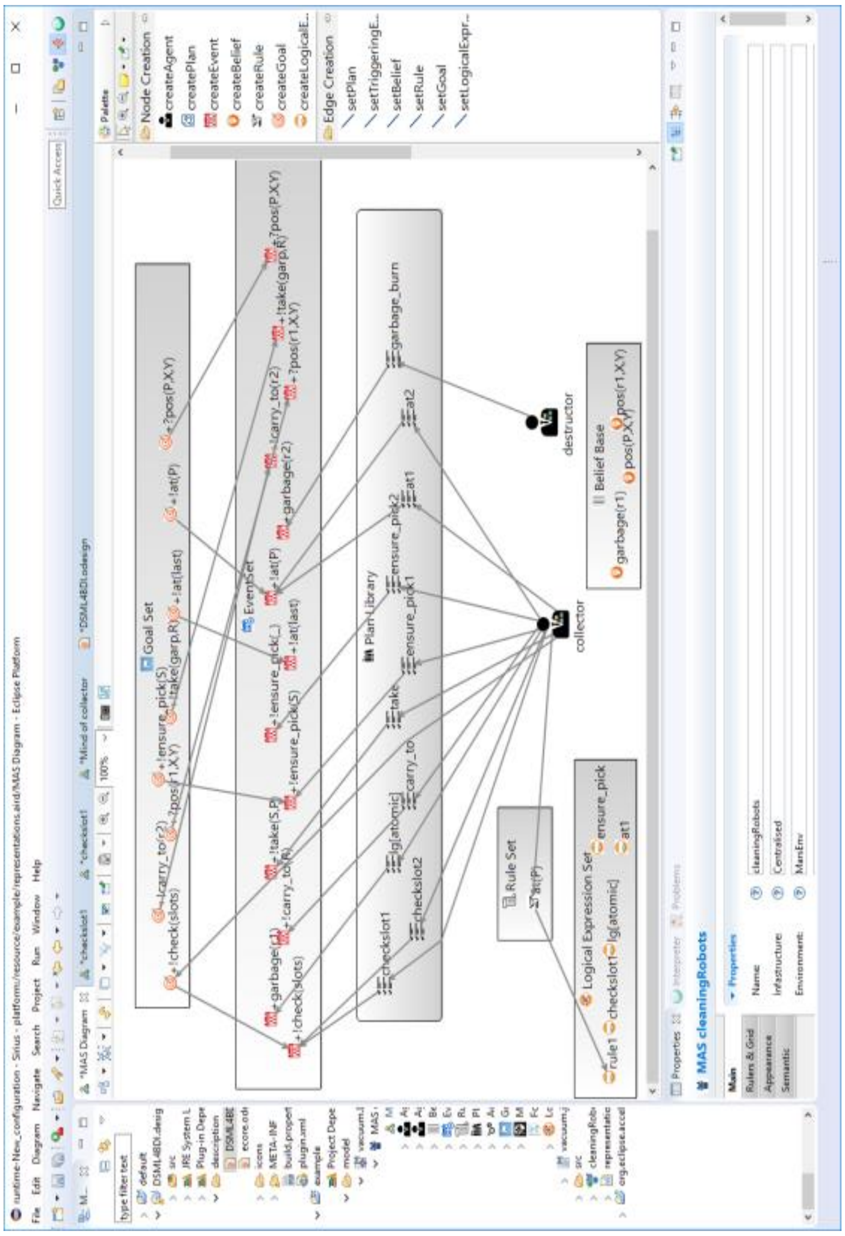

Figure 1. DSML4BDI graphical modeling environment.

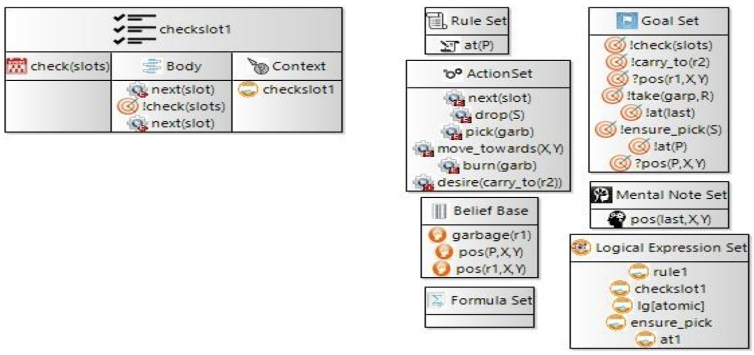

Figure 2. Modeling agent plans in DSML4BDI.

Group A developers, who experienced the use of DSML4BDI, were asked to fill a questionnaire (mixed with marking and open-ended questions) for evaluating usability, comprehensiveness and easiness of the tool. For marking questions, average result was 4.30 out of 5 (0 is nothing and 5 is at most level) indicating the tool was generally found handy and easy-to-use. An extended discussion of this evaluation and achieved results can be found in [9].

4 Related Work

AOSE researchers have significant efforts on model-driven MAS development [14]. While various agent metamodel proposals [15-17] and MDD approaches [8, 18, 19] exist, perhaps the most popular way of MDD of MASs is based on providing DSMLs (e.g. [20–25]) in which both MAS modeling and implementation can be performed. Within these studies, some of them [21-23, 25] specifically considers MDD of BDI agents. However, most of them are not supported with proper tools and the remaining ones with tool support [21, 23] mostly do not evaluate both the user's adoption and the generation performance of the tool. Our evaluation results indicate that the developers found DSML4BDI providing an all-embracing model of BDI elements and the use of the tool both led to automatic generation of most BDI artifacts required for exact MAS implementation and substantial decrease in time needed for developing a MAS from scratch.

5 Conclusions

An MDD tool for developing BDI agents was introduced. The tool both provides modeling all MAS structures and relations visually and is capable of achieving the MAS implementation via automatic code generation for Jason platform. Comparative evaluations showed the tool has a high generation performance and causes significant decrease in the development time. Feedbacks gained from the users also supported the claim on its usability in general. Our future work is to enhance DSML4BDI's organization and environment modeling capabilities by integrating it with JaCaMo platform [26].

Acknowledgements

This work was funded by the Scientific and Technological Research Council of Turkey (TUBITAK) under grant 115E591.

References

1. Rao, A.S. and Georgeff, M.P.: Decision procedures for BDI logics. J Logic Comput, 8(3):293-343, (1998).
2. Tezel, B. T., Challenger, M., and Kardas, G.: A Metamodel for Jason BDI Agents. In Proceedings of the 5th Symposium on Languages, Applications and Technologies (SLATE 2016), pages 8:1-8:9, June 20-21 (2016).
3. BDI4JADE. http://www.inf.ufrgs.br/prosoft/bdi4jade
4. Jadex. https://www.activecomponents.org/#/project/news
5. JACK. http://aosgrp.com/products/jack/
6. Rao, A. S.: AgentSpeak(L): BDI agents speak out in a logical computable language. In Proceedings of the 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW 1996), pages 42-55, January 22–25, (1996).
7. Bordini, R.H., Hübner, J.F., Wooldridge, M.: Programming Multi-Agent Systems in AgentSpeak using Jason. John Wiley & Sons, Ltd, Chichester, UK (2007).
8. Kardas, G.: Model-driven development of multi-agent systems: a survey and evaluation. The Knowledge Engineering Review, 28(4): 479-503, (2013).
9. Kardas, G.,Tezel, B.T., Challenger, M.: Domain-specific modelling language for belief–desire–intention software agents. IET Softw., 12(4): 356-364 (2018).
10. DSML4BDI. http://serlab.ube.ege.edu.tr/Bundles/dsml4bdi.zip
11. CArtAgO. http://cartago.sourceforge.net/
12. Sirius Modeling Platform. https://eclipse.org/sirius/
13. Challenger, M., Kardas, G., Tekinerdogan, B.: A systematic approach to evaluating domain-specific modeling language environments for multi-agent systems. Softw. Qual. J., 24:755–795, (2016).
14. Kardas, G., Gomez-Sanz, J.J.: Special issue on model-driven engineering of multi-agent systems in theory and practice, Comput. Lang. Syst Str. 50, 140-141 (2017).
15. Omicini, A., Ricci, A., and Viroli M.: Artifacts in the A&A meta-model for multi-agent systems. Autonomous Agents and Multi-Agent Systems, 17(3): 432-456, (2008).
16. Hahn, C., Madrigal-Mora, C., and Fischer, K.: A Platform-Independent Metamodel for Multiagent Systems. Auton Agent Multi-Ag, 18(2): 239-266, (2009).
17. Beydoun, G., Low, G., Henderson-Sellers, B., Mouratidis, H., Gomez-Sanz, J. J., Pavon, J., and Gonzalez-Perez, C.: FAML: A Generic Metamodel for MAS Development. IEEE Transactions on Software Engineering, 35(6): 841-863, (2009).
18. Pavon, J., Gomez-Sanz, J. J., and Fuentes, R.: Model driven development of multi-agent systems. In Proceedings of the 2nd European Conference on Model Driven Architecture – Foundations and Applications (ECMDA-FA 2006), pages 284–298, July 10-13, (2006).
19. Bergenti, F., Iotti, E., Monica, S., and Poggi, A.: Agent-oriented model-driven development for JADE with the JADEL programming language. Comput Lang Ssyt Str, 50: 142-158, (2017).
20. Hahn, C.: A Domain Specific Modeling Language for Multiagent Systems. In: 7th Int.'l Conf. on Autonomous Agents and Multiagent Systems, pp. 233–240 (2008).

21. Gascueña, J.M., Navarro, E., Fernández-Caballero, A.: Model-driven engineering techniques for the development of multi-agent systems. Eng. Appl. Artif. Intell. 25, 159-173 (2012).
22. Cossentino, M., Chella, A., Lodato, C., Lopes, S., Ribino, P., Seidita, V.: A Notation for Modeling Jason-Like BDI Agents. In: 6th International Conference on Complex, Intelligent, and Software Intensive Systems. pp. 12–19 (2012).
23. Challenger, M., Demirkol, S., Getir, S., Mernik, M., Kardas, G., Kosar, T.: On the use of a domain-specific modeling language in the development of multiagent systems. Eng. Appl. Artif. Intell. 28, 111–141 (2014).
24. Gonçalves, E.J.T., Cortés, M.I., Campos, G.A.L., Lopes, Y.S., Freire, E.S.S., da Silva, V.T., de Oliveira, K.S.F., de Oliveira, M.A.: MAS-ML 2.0: Supporting the modelling of multi-agent systems with different agent architectures. J. Syst. Softw.108,77-109 (2015)
25. Wautelet, Y., Kolp, M.: Business and model-driven development of BDI multi-agent systems. Neurocomputing. 182, 304–321 (2016).
26. JaCaMo. http://jacamo.sourceforge.net/