Model-Driven Engineering in Agent-based Modeling and Simulation: a Case Study in the Traffic Signal Control Domain

(Extended Abstract)

Fernando Santos[1,2], Ingrid Nunes[1,3], Ana L. C. Bazzan[1]
[1]UFRGS, Brazil; [2]UDESC, Brazil; [3]TU Dortmund, Germany
{fsantos, ingridnunes, bazzan}@inf.ufrgs.br

ABSTRACT

Model-driven engineering (MDE) is an approach for improving productivity in software development. This approach was exploited in the context of agent-based modeling and simulation (ABMS) only to a certain extent. Previous work has not shown real evidence of the benefits that MDE promotes in ABMS. This paper thus explores the use of MDE in ABMS with a case study in the traffic domain. We propose a domain analysis method to identify domain concepts and a modeling language that provides building blocks for them. Our evaluation gives evidence that our MDE approach reduces the effort to develop agent-based simulations.

Keywords

Agent-based Modeling and Simulation, Model-driven Engineering, Development Effort, Traffic Signal Control

1. INTRODUCTION

Building simulations in which there are multiple interacting agents situated in an environment is a challenging task, which has been widely investigated in the context of agent-based modeling and simulation (ABMS). Alternatives for developing agent-based simulations (ABSs) include platforms, such as NetLogo [21]. However, these platforms demand expertise both in ABMS and programming, thus making ABS development a difficult task for people with little or no ABMS expertise to execute. Researchers have already demonstrated the importance of providing tools and building blocks to ease the development of ABSs [10, 19].

Model-driven engineering (MDE) [2, 16] is a software development approach whose goal is to make domain concepts available for modeling, reducing the abstraction gap and thus increasing the productivity in software development. In MDE, models are first-class citizens, from which source code can be automatically generated [17].

There are MDE approaches for ABMS [5, 7, 9, 11, 13], but these are limited to particular MDE aspects and usually model only high-level ABMS concepts, leaving much left to be further done by developers. Moreover, there is a lack of concrete evidences of the real benefits that MDE approaches promote for developing ABSs.

In this paper, we address these issues by further exploring the use of MDE in the context of ABMS. We present a case study of the development of an MDE approach in the *adaptive traffic signal control* domain [3]. Previous work on MDE showed that the more specific the application domain, the higher the chance of success [12]. We propose a domain analysis method to identify domain concepts, which is applicable to other domains. The proposed MDE approach provides a modeling language and a code generator, to support the development of ABSs in the investigated domain, and potentially in similar domains.

2. MDE FOR ABMS

To create an MDE approach, one must first identify the existing domain concepts, and then make them available for modeling. We propose a domain analysis method to identify key concepts using a set of ABSs in the same domain, given that existing methods [5, 11], despite providing valuable guidelines for identifying agents, do not provide support for identifying the simulation aspects of ABMS. Our domain analysis is composed of the following steps: i) to build a preliminary list of agent-related concepts (following [5] and [11]); ii) to refine the identified concepts using the Overview, Design concepts, and Details (ODD) protocol [8] to incorporate simulation aspects and additional agent capabilities; iii) to find the essence behind each identified concept, abstracting it as a single, essential concept, or generalizing it to a parent concept; and iv) to build the domain model, specifying constraints and relationships between the enumerated concepts.

The proposed domain analysis method was applied to identify concepts in the domain of adaptive traffic signal control. Existing work on self-organizing traffic lights [4, 6] and reinforcement learning for traffic light control [14, 15, 20] was used as domain knowledge source. Consequently, our domain analysis was performed using a *bottom-up* approach, to reduce the bias of individual experts' views while identifying domain concepts. The following are the identified concepts that are related to traffic lights: environment, stages, phases, plans, agents and their capabilities and perceptions, adaptation, and learning.

We provide a modeling language, DSL4ABMS, to facilitate the development of ABSs. The language was built following existing methods for building domain-specific lan-

Appears in: *Proc. of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), S. Das, E. Durfee, K. Larson, M. Winikoff (eds.), May 8–12, 2017, São Paulo, Brazil.*

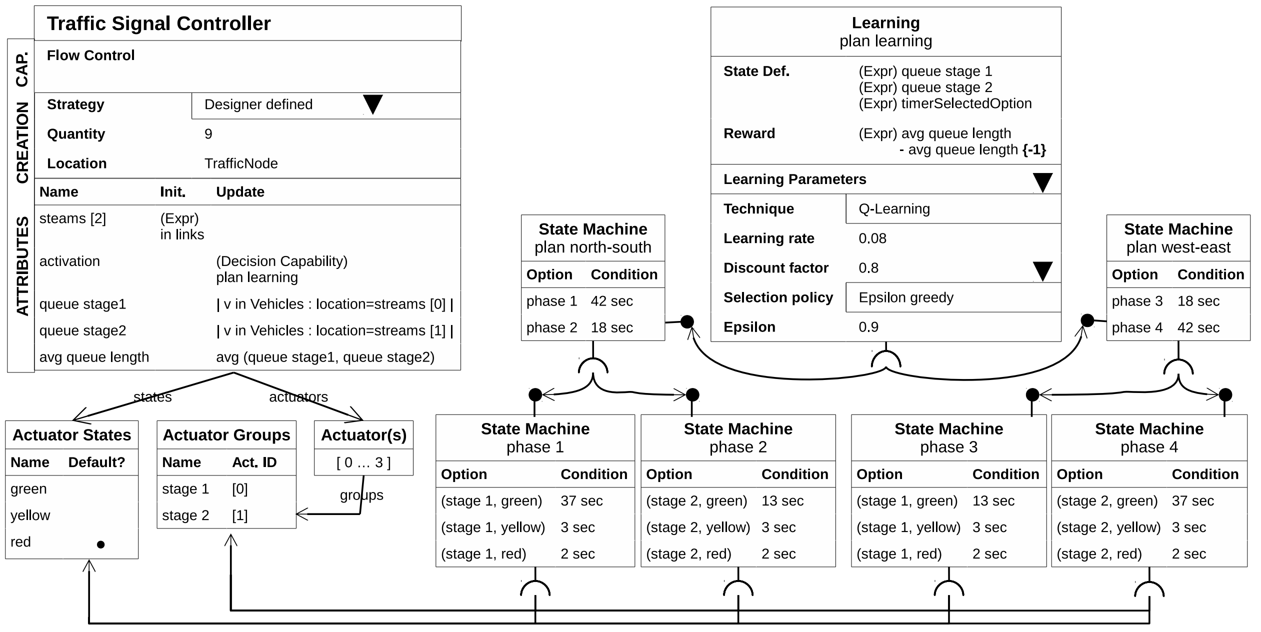

Figure 1: Example of the DSL4ABMS Language Concrete Syntax.

guages [18], in order to make the identified domain concepts available for modeling as building blocks. We adopted a graphical representation to reduce the effort required to identify model elements and their relationships. Figure 1 shows an overview of the concrete syntax of DSL4ABMS. To illustrate all the features of DSL4ABMS, the depicted model shows a (partial) combination of existing simulation elements: traffic signal stages, phases, and plans from [15], and the learning technique used by Mannion et al. [14].

Entities and agents are represented as boxes, with name, attributes, and a creational strategy (to setup how they are created and located in the environment). Such representation is inspired by the Unified Modeling Language (UML) class diagram. Besides name and type, attributes specify how they are initialized and updated during the simulation. The specification is by means of static values, expressions, or other sources. A list of all the available creational strategies and sources of values is available elsewhere [1].

The agent recurrent behaviors and its decision making ability are represented as agent capabilities. Concepts required by an agent capability are represented as elements connected to the agent by an arrow. In Figure 1, the *Traffic Signal Controller* agent has a *flow control* capability, which abstracts the behavior of regulating the flow of a set of streams by means of actuators. Such capability requires *actuators* (which may be arranged into *actuator groups*) and their corresponding *actuator states.*

Decision capabilities are represented as boxes with their content describing the elements required by each particular capability. Supported decision capabilities are: *state machine*, *adaptation*, and *learning.* The input and output of a decision capability are represented by a semicircle and a filled circle connector, respectively. The input that is provided to a decision capability is represented as a connection between the capability's semicircle and any model element that represents a decision option, which includes actuators, their states and groups, and other decision capabilities. In addition to its type and name, each decision capability has specific parameters that must be set: states, reward, and learning parameters for learning capabilities; states and transition rules for state machines; and an adaptation criterion for adaptation capabilities.

Automated code generation is fundamental to reduce the effort to develop ABSs and thus to exploit the benefits of an MDE approach. We provide a code generator that, given a model specified using our DSL4ABMS language, generates code for the NetLogo [21] platform. The produced code is ready-to-use, as opposed to existing MDE approaches for ABMS that usually generate only code skeletons.

3. CONCLUSION AND FUTURE WORK

In this paper, we presented an MDE approach for developing ABSs in the adaptive traffic signal control domain. An empirical study was conducted to evaluate the effort to develop ABSs using our approach. Adopted metrics consider the human effort employed to create a simulation, in terms of lines of code and modeling elements, manually and automatically produced. Results show that our approach requires 60-86% less effort than the NetLogo and ITSUMO simulation platforms. One simulation was considered for each decision capability, and the source code related to traffic lights was fully automatically generated for all of them. This result provides evidence that an MDE approach gives helpful support for developing ABSs.

It is our ongoing work the addition of other agent capabilities and aspects from other domains to our MDE approach. Our long-term goal is to use MDE to allow people with little or no ABMS expertise to build agent-based simulations.

Acknowledgments

This work is supported by funds from CNPq ref. 442582/2014-5 and 305062, CAPES 7619-15-4, and Alexander von Humboldt BRA 1184533 HFSTCAPES-P.

REFERENCES

[1] DSL4ABMS. `http://www.inf.ufrgs.br/prosoft/resources/dsl4abms/`. Accessed: 2017-02-22.

[2] C. Atkinson and T. Kühne. Model-driven development: A metamodeling foundation. *IEEE Software*, 20:36–41, 2003.

[3] A. L. C. Bazzan. Opportunities for multiagent systems and multiagent reinforcement learning in traffic control. *Autonomous Agents and Multiagent Systems*, 18(3):342–375, June 2009.

[4] S.-B. Cools, C. Gershenson, and B. D'Hooghe. Self-organizing traffic lights: A realistic simulation. In M. Prokopenko, editor, *Advances in Applied Self-Organizing Systems*, pages 45–55. Springer, London, 2013.

[5] A. Garro and W. Russo. easyABMS: A domain-expert oriented methodology for agent-based modeling and simulation. *Simulation Modelling Practice and Theory*, 18(10):1453–1467, 2010.

[6] C. Gershenson. Self-organizing traffic lights. *Complex Systems*, 16(1):29–53, 2005.

[7] A. Ghorbani, P. Bots, V. Dignum, and G. Dijkema. MAIA: a framework for developing agent-based social simulations. *Journal of Artificial Societies and Social Simulation*, 16(2):9, 2013.

[8] V. Grimm, U. Berger, D. L. DeAngelis, J. G. Polhill, J. Giske, and S. F. Railsback. The odd protocol: a review and first update. *Ecological modelling*, 221(23):2760–2768, 2010.

[9] C. Hahn. A domain specific modeling language for multiagent systems. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 1*, pages 233–240. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[10] L. Hamill. Agent-based modelling: The next 15 years. *Journal of Artificial Societies and Social Simulation*, 13(4):7, 2010.

[11] S. Hassan, R. Fuentes-Fernandez, J. M. Galan, A. Lopez-Paredes, and J. Pavon. Reducing the modeling gap: On the use of metamodels in agent-based simulation. In *In 6th conference of the european social simulation association (ESSA 2009)*, pages 1–13, 2009.

[12] J. Hutchinson, M. Rouncefield, and J. Whittle. Model-driven engineering practices in industry. In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, pages 633–642, New York, NY, USA, 2011. ACM.

[13] F. Klügl and P. Davidsson. AMASON: Abstract Meta-model for Agent-Based SimulatiON. In M. Klusch, M. Thimm, and M. Paprzycki, editors, *MATES*, pages 101–114, 2013.

[14] P. Mannion, J. Duggan, and E. Howley. An experimental review of reinforcement learning algorithms for adaptive traffic signal control. In L. T. McCluskey, A. Kotsialos, P. J. Müller, F. Klügl, O. Rana, and R. Schumann, editors, *Autonomic Road Transport Support Systems*, pages 47–66. Springer, 2016.

[15] D. de. Oliveira and A. L. C. Bazzan. Multiagent learning on traffic lights control: effects of using shared information. In A. L. C. Bazzan and F. Klügl, editors, *Multi-Agent Systems for Traffic and Transportation*, pages 307–321. IGI Global, Hershey, PA, 2009.

[16] D. Schmidt. Model-driven engineering. *Computer-(IEEE Computer Society*, 39(2):25–31, feb 2006.

[17] T. Stahl, M. Völter, J. Bettin, A. Haase, and S. Helsen. *Model-driven software development: technology, engineering, management.* John Wiley & Sons, 2006.

[18] M. Strembeck and U. Zdun. An approach for the systematic development of domain-specific languages. *Software: Practice and Experience*, 39(15):1253–1292, 2009.

[19] M. P. Wellman. Putting the agent in agent-based modeling. *Autonomous Agents and Multi-Agent Systems*, 30(6):1175–1189, 2016.

[20] M. Wiering. Multi-agent reinforcement learning for traffic light control. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, pages 1151–1158, 2000.

[21] U. Wilensky. NetLogo, 1999. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.