Engineering Applications of Artificial Intelligence 28 (2014) 111–141

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

On the use of a domain-specific modeling language in the development of multiagent systems

Moharram Challenger [a], Sebla Demirkol [a], Sinem Getir [a], Marjan Mernik [b], Geylani Kardas [a,*], Tomaž Kosar [b]

[a] International Computer Institute, Ege University, Bornova, 35100 Izmir, Turkey
[b] Faculty of Electrical Engineering and Computer Science, University of Maribor, Smetanova 17, 2000 Maribor, Slovenia

ARTICLE INFO

Article history:
Received 3 June 2012
Received in revised form
8 November 2013
Accepted 19 November 2013
Available online 21 December 2013

Keywords:
Agent
Multiagent system
Model driven development
Domain-specific modeling language
Metamodel
Semantic web

ABSTRACT

The study of Multiagent Systems (MASs) focuses on those systems in which many intelligent agents interact with each other. The agents are considered to be autonomous entities which contain intelligence that serves for solving their selfish or common problems, and to achieve certain goals. However, the autonomous, responsive, and proactive natures of agents make the development of agent-based software systems more complex than other software systems. Furthermore, the design and implementation of a MAS may become even more complex and difficult to implement when considering new requirements and interactions for new agent environments like the Semantic Web. We believe that both domain-specific modeling and the use of a domain-specific modeling language (DSML) may provide the required abstraction, and hence support a more fruitful methodology for the development of MASs. In this paper, we first introduce a DSML for MASs called SEA_ML with both its syntax and semantics definitions and then show how the language and its graphical tools can be used during model-driven development of real MASs. In addition to the classical viewpoints of a MAS, the proposed DSML includes new viewpoints which specifically support the development of software agents working within the Semantic Web environment. The methodology proposed for the MAS development based on SEA_ML is also discussed including its example application on the development of an agent-based stock exchange system.

1. Introduction

The development of intelligent software agents keeps its emphasis on both artificial intelligence and software engineering research areas. In its widely-accepted definition, an agent is an encapsulated computer system (mostly a software system) situated within a certain environment, and which is capable of flexible autonomous action within this environment in order to meet its design objectives (Wooldrige and Jennings, 1995; Bădică et al., 2011). These autonomous, reactive, and proactive agents also have social ability and can interact with other agents and humans in order to solve their own problems. They may also behave in a cooperative manner and collaborate with other agents for solving common problems. In order to perform their tasks and interact with each other, intelligent agents constitute systems called Multiagent Systems (MASs).

The implementation of agent systems is naturally a complex task when considering the aforementioned characteristics. In addition, the internal agent behavior model and any interaction within the agent organizations become even more complex and hard to implement when new requirements and interactions for new agent environments like the Semantic Web (Berners-Lee et al., 2001; Shadbolt et al., 2006) are taken into account.

The Semantic Web improves the current World Wide Web such that the web page content can be organized in a more structured way by tailoring it towards the specific needs of end-users. The web can be interpreted with ontologies (Berners-Lee et al., 2001) that help machines to understand web content. It is apparent that the interpretation in question can be realized by autonomous computational entities, like agents, to handle the semantic content on behalf of their human users. Software agents can be used to collect Web content from diverse sources, process the information, and exchange the results.

* Corresponding author at: International Computer Institute, Ege University, Bornova, 35100 Izmir, Turkey. Tel: +90 232 311 3223, Fax: +90 232 388 7230.
E-mail addresses: moharram.challenger@mail.ege.edu.tr, m.challenger@gmail.com (M. Challenger), sebla.demirkol@ege.edu.tr (S. Demirkol), sinem.getir@ege.edu.tr (S. Getir), marjan.mernik@uni-mb.si (M. Mernik), geylani.kardas@ege.edu.tr (G. Kardas), tomaz.kosar@uni-mb.si (T. Kosar).

0952-1976/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.engappai.2013.11.012

In addition, autonomous agents can also evaluate semantic data and collaborate with semantically-defined entities of the Semantic Web, like semantic web services, by using content languages (Kardas et al., 2009a). Semantic web services can be simply defined as the web services with semantic interfaces to be discovered and executed (Sycara et al., 2003). In order to support the semantic interoperability and automatic composition of web services, the capabilities of web services are defined in service ontologies that provide the required semantic interfaces. Such interfaces of the semantic web services can be discovered by software agents and then these agents may interact with those services to complete their tasks. Engagements and invocations of a semantic web service are also performed according to the service's semantic protocol definitions. For instance, the dynamic composition of heterogeneous services for the optimal selection of service providers can be achieved by employing agents and ontologies, as proposed in Bădică et al. (2012).

However, agent interactions with semantic web services add further complexities when designing and implementing MASs. Therefore, it is natural that methodologies are being applied to master the problem of defining such complex systems. One of the possible alternatives is represented by domain-specific languages (DSLs) (van Deursen et al., 2000; Mernik et al., 2005; Varanda-Pereira et al., 2008; Fowler, 2011) that have notations and constructs tailored towards a particular application domain (e.g. MAS). The end-users of DSLs have knowledge from the observed problem domain (Sprinkle et al., 2009), but they usually have little programming experience. Domain-specific modeling languages (DSMLs) further raise the abstraction level, expressiveness, and ease of use. The main artifacts of DSML are models instead of software codes and they are usually specified in a visual manner (Schmidt, 2006; Gray et al., 2007). Note that graphical notation for DSMLs is not mandatory and textual DSMLs also exist (Sánchez Cuadrado and Garcia Molina, 2007; Mernik, 2013), although the majority of DSMLs do indeed use visual notation.

DSMLs are used in Domain-specific Modeling (DSM), a software engineering methodology which is a particular instance of model-driven engineering (MDE) (Schmidt, 2006). A DSML's graphical syntax offers benefits, like easier design, when modeling within certain domains. The development of DSML is usually driven by language model definition (Strembeck and Zdun, 2009). That is, concepts and abstractions from the domain which need to be defined in order to reflect the target domain (the language model). Then, relations between the language concepts need to be defined. Both form an abstract syntax of modeling language. Usually, a language model is defined with a metamodel. The additional parts of a language model are constraints that define those semantics which cannot be defined using the metamodel. Constraints are usually defined in a certain dedicated language (e.g. Object Constraint Language (OCL) (OMG, 2012)). Domain abstractions and relations need to be presented within a concrete syntax and serve as a modeling block within the end-user's modeling environment. This modeling environment can be generated automatically if dedicated software is used (e.g. MetaEdit+ (MetaCase, 1995) and Eclipse GMF (The Eclipse Foundation, 2006)), otherwise, modeling editor must be provided by hand (Kos et al., 2011). Then, the model transformations need to be defined in order to call the domain framework, which is a platform that provides the functions for implementing the semantics of DSMLs within a specific environment. Usually, the semantics is given by translational semantics (Bryant et al., 2011).

We are convinced that both DSM and the use of a DSML may provide the required abstraction and support a more fruitful methodology for the development of MASs. Hence, in this paper, we first introduce a DSML for MASs with both its syntax and semantics definitions, and then show how the language and its graphical tools can be used during the model-driven development of real MASs. The domain of our study is “Semantic Web enabled MASs” in which autonomous agents can evaluate semantic data and collaborate with semantically-defined entities of the Semantic Web, like Semantic Web Services. In order to support MAS experts when programming their own systems, and to be able to fine-tune them visually, our DSML covers all aspects of an agent system from the internal view of a single agent to the complex MAS organization. In addition to these classical viewpoints of a MAS, the proposed DSML also includes new viewpoints which specifically pave the way for the development of software agents working on the Semantic Web environment. We refer to this DSML as a Semantic web Enabled Agent Modeling Language (SEA_ML). Our concrete motivation for SEA_ML is to enable domain experts to model their own MASs on the Semantic Web without considering the limitations of using existing MAS development frameworks (e.g. JADE (Bellifemine et al., 2001), JADEX (Pokahr et al., 2005) or JACK (Howden et al., 2001)).

The remainder of the paper is organized as follows: Sections 2 and 3 discuss the abstract and concrete syntaxes of SEA_ML. The model-to-model and model-to-text transformations which are the building blocks of SEA_ML's semantics are discussed in Section 4 and Section 5, respectively. Section 6 discusses the methodology proposed for the MAS development based on SEA_ML, including its example application on the development of an agent-based stock exchange system. Related work is given in Section 7. Finally, Section 8 concludes the paper and states the future work.

2. Abstract syntax

The abstract syntax of a DSML describes the concepts and their relations to other concepts without any consideration of meaning. In other words, the abstract syntax of a language describes the vocabulary of concepts provided by the language and how they may be combined to form models or programs (Clark et al., 2004). In terms of Model Driven Development (MDD), the abstract syntax is described by a metamodel which defines what the models should look like. Hence, in this section, we discuss the metamodel that constitutes the SEA_ML's abstract syntax.

In a Semantic Web enabled MAS, software agents can gather Web contents from various resources, process the information, exchange the results, and negotiate with other agents. Within the context of these MASs, autonomous agents can evaluate semantic information and work together with semantically-defined entities like semantic web services using content languages. The work in Kardas et al. (2009a) defined a conceptual architecture that contained an overview of such a MAS. Originating from this architecture, Kardas et al. also provided a metamodel which covers those meta-elements that belong to Semantic Web enabled MASs. Reengineering of that metamodel enabled us to form the platform independent metamodel (PIMM) which represents the abstract syntax of SEA_ML. Resulting metamodel focuses on both modeling the internal agent architecture and MAS organization.

When the SEA_ML's metamodel is overviewed, one can detect that one of the main elements within the metamodel is the Semantic Web Agent (SWA). A SWA works within a Semantic Web Organization (SWO) and can interact with various services. Another meta-element is the Semantic Web Service (SWS) which represents a web service (except agent service) defined semantically. Also, a SWS is