The GMF-based Syntax Tool of a DSML for the Semantic Web enabled Multi-Agent Systems

Sinem Getir
Int'l Computer Institute
EGE Univ., Izmir, Turkey
Tel:+90 232 311 3234
sinem.getir@ege.edu.tr

Sebla Demirkol
Int'l Computer Institute
EGE Univ., Izmir, Turkey
Tel:+90 232 311 3233
sebla.demirkol@ege.edu.tr

Moharram Challenger
Int'l Computer Institute
EGE Univ., Izmir, Turkey
Tel: +90 232 311 3252
moharram.challenger@mail.ege.edu.tr

Geylani Kardas
Int'l Computer Institute
EGE Univ., Izmir, Turkey
Tel: +90 232 311 3223
geylani.kardas@ege.edu.tr

Abstract

Internal complexity of agents makes development of agent-based software systems complicated. On the other hand, MAS implementation becomes even more complex when new requirements and interactions for environments such as the Semantic Web is considered. A Domain Specific Modelling Language (DSML) can provide the required abstraction and hence support a more fruitful methodology for the development of MAS especially working on the new challenging environments like the Semantic Web. In this paper, a graphical syntax tool is introduced in which agent developers can model MASs according to a DSML called Semantic web Enabled Agent Modeling Language, SEA_ML.

Categories and Subject Descriptors D.3.1 **[Formal Definitions and Theory]*:*** Formal Definitions and Theory – Semantic, Syntax. D.3.3 **[Languages Constructs and Features]*:*** Constraints. D.2.11 **[Software Architectures]*:*** Domain-specific architectures. D.2.13 **[Reusable Software]*:*** Domain Engineering. I.2.11 **[Distributed Artificial Intelligence]*:*** Intelligent agents, Multi agent system

General Terms Design, Languages.

Keywords Domain Specific Modeling Languages, Meta-model, Multi Agent Systems, Semantic Web.

1. Introduction

SEA_ML is a DSML which supports the development of Multi-agent Systems working on the Semantic Web. In our vision, the "Semantic Web enabled MAS" means that software agents are planned to collect Web content from diverse sources, process the information and exchange the results on behalf of their human users. Autonomous agents can also evaluate semantic data within these MASs and collaborate with semantically defined entities such as Semantic Web services by using content languages. We call the software agents with these capabilities as Semantic Web Agents.

In [1], we discuss how domain specific engineering can provide easy and rapid construction of Semantic Web enabled MASs and introduce SEA_ML. Similar to other DSMLs; SEA_ML should provide complete definitions for its abstract syntax, concrete syntax and formal semantics. We discuss the abstract syntax of SEA_ML in [2]. A meta-model that describes the meta-entities and their relationships for a domain can naturally provide a base for the definition of such an abstract syntax. Our meta-model for the Semantic Web enabled MASs considers various aspects of MAS development (e.g., role, protocol and organizational) and provides both internal modelling of a software agent and interaction of agents and Semantic Web services within the environment. In this demo paper, we present the graphical concrete syntax tool for SEA_ML.

2. Graphical Syntax Tool of SEA_ML

This section introduces the developed GMF-based [3] graphical tool for SEA_ML. Capabilities, constraints, rules and facilities of the tool are discussed in the first subsection. Afterwards, we illustrate the tool properties within an e-barter case study showing how constraints can be provided.

2.1 The Tool Overview

The tool we developed supplies a graphical editor which provides to choose model elements from the palette. These elements can be filled with the labels and properties in order to complete the full features of model elements with optional and different notations and colors of the graphical facilities.

The Ecore [3] models are created in Eclipse. After setting the graphical notations for abstract syntax meta elements, we use Eclipse GMF to tie notations to the domain concepts. Provided GMF tools can be useful for the modelling of MASs and getting an overview of the agents of a complex instance models for agent developers. Some of the graphical notations for Agent internal and Agent-SWS Interaction viewpoints are illustrated in Table 1. Odd columns define the names of the meta elements in abstract syntax and even columns mark notations, or icons in the tool.

It was decided that the elements of the same type or inherited from the same element have similar icons and same or similar geometric notations. For instance, Ontology types and other meta elements mapping are provided when their integrated elements created. As an example, while Role Ontology keeps the basic ontology background; it also holds the Role icon.

Table 1. The concepts and their notations for graphical concrete syntax elements of Agent Internal and Agent-SWS Interaction viewpoints.

Concept	Notation	Concept	Notation
Goal	◎	Belief	📖
Behaviour	➡	SWA	👤
Capabilities	📁	Semantic Web Matchmaker Agent	👤
Role Ontology	🎭	Process	P
Service Ontology	✎	Interface	I
SWS	●	SS_Register Plan	R
Registration Role	🎭	SS_Finder Plan	F
Plan	≡	SS_Agreement Plan	A
Role	🎭	SS_Execution Plan	E

Thus we can see the elements and relationships in visual diagrams which are used to be transformed into GMF tools. During these transformations; icons are determined for palettes, figures and geometrics of icons are described and some constraint checkers and rules are considered. The editor of the tool has a potential to check the designed model for the static and dynamic semantic compatibility of the DSML in respect to prevent the errors like making wrong relationship nexus between MAS elements.

The GMF-based constraints from the Ecore model is provided for any instance models. The constraints according to all viewpoints can be classified as following:

Compartment *constraint:* The composition relationship between the meta-elements in Ecore is transformed to a relationship that one element contains the other. Two elements that do not have this kind of relationship cannot be modelled as nested compartment. For instance, Belief is composed of Facts in the metamodel according to the agent internal viewpoint. Fact is a compartment in Belief which is because of the aggregation relationship between them. Contrarily, this modelling conformation cannot be used between a role and a SWA.

Number of relationships constraint: Due to one-to-one, one-to-many, many-to-many relationships in the Ecore, number of relationships are controlled between the elements in the instance models. As an example, while a SWA can play more than one role, it can have only one Agent Type.

Resource and destination of a relationship constraint: Direction of the relationship defines the resource and destination of that relation. This constraint is defined in Ecore meta-meta level. For example, the relationship between Plan and Goal cannot be created in the instance model in the case of the direction is plan to goal.

Inheritance relationship constraint: The defined inheritance relationships from the metamodel naturally force some constraints while creating the instance model. A subclass in an instance model will include all of the attributes and relationships of its super-class. Agent-SWS Interaction viewpoint is the best example for this issue. Plan has relationships with other elements directly. It has also four subclasses and when they appear in the instance model, all of the relationships are inherited from Plan element.

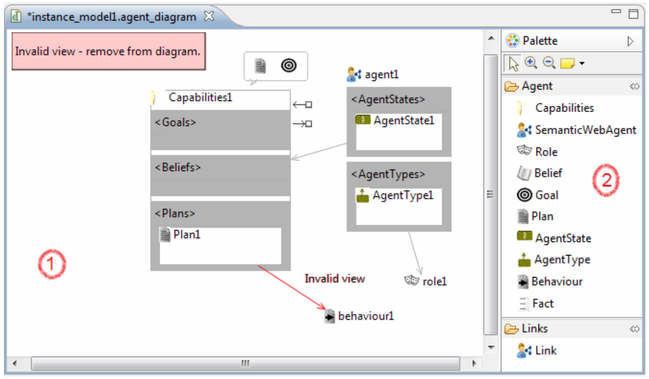
As Protocol viewpoint has the basic communication and messaging relations it does not have compartment structure. However, Compartment constraint is used in all other viewpoints which have the composition relationships between meta elements as compartment structure requires. Some of the constraints are primitive ones, e.g. Number of relationships, Relationship resource and destination and Integrity of relationship-element. These constraints exist in all of the viewpoints on the ground of having the basic entity relationships. Inheritance constraints can be seen in all of the viewpoints except Organization viewpoint as it does not have any super-class relationship in its metamodel.

An overview of how the tool looks like is illustrated in Figure 1. Modelling environment is shown on the left side

(1). Palette is available with icons on the right side (2). The palette provides to complete the aimed model with its user friendly interface. When a user drops and clicks an element in the diagram, the tool automatically shows which elements are related or not related to that element.

The tool also provides the transition of the editor of viewpoints in the modelling environment. It keeps the integration and MAS system momentum, opens the other viewpoints by clicking the transition elements. As an example, in the figure when it is double clicked on the Role element instance (e.g. role1 in the figure), the tool opens the Role viewpoint editor automatically. Additionally the user can create its own diagram file for all viewpoints. Thus, the tool becomes not only flexible but also a good guide to go on completing the instance model for the System.

Figure 1 also demonstrates the control mechanism between model and instance model. It prevents the changes according the metamodel, it keeps the checking for instance model and it warns the user to control relationship by printing it is not a valid relationship anymore. In the upper left corner of the figure, an error message for a relationship that does not exist anymore for agent internal viewpoint concrete syntax tool is shown.


Invalid view - remove from diagram.

Invalid view

Figure 1. Graphical modeling for agent internal viewpoint of an instance model with user error in syntax tool of SEA_ML.

2.2 An Example of the Tool Usage

In order to illustrate the use of the introduced tool, we consider the modelling of a simple multi-agent based e-barter system which provides an alternative commerce approach where customers meet at a marketplace in order to exchange their goods or services without currency [1]. In this application, Customer agents are responsible for adding and evaluating barter proposals. Barter Manager is responsible for collecting barter proposals, matching proper barter proposals and tracking the bargaining process between Customer agents. After bargaining, Customer agents send engagement message to the Barter Manager agent. Then, the Barter Manager agent notifies the Cargo agent [4].

Screenshot in Figure 2 illustrates the internal agent structure of a Barter Manager Agent which is an instance of SWA (Semantic Web Agent) with GMF editor.

Screenshot in Figure 3 illustrates the use of the GMF editor we provide for modelling Agent-SWS (Semantic Web Service) Interaction viewpoint of a multi-agent electronic barter system. In order to infer about semantic closeness between offered and purchased items based on the defined ontologies, barter manager may use a SWS called Barter Service. Due to the space limitations, SEA_ML viewpoints and concepts cannot be discussed here but can be found in [2] in detail.

For the agent internal viewpoint, it can be figured out some strict compartment relationships. Goals, beliefs, plans instances such as best matching ,system rules, financial plan respectively can be exist only as compartment in the instance of Capabilities named Barter. However Barter cannot include Barter Role. Barter manager can only have one agent type due to number of relations constraint however it can play more than one Role such as Barter Role.

Basic instance constraints for SWS Interaction viewpoint can be the compartment relationships between SWS instance Barter Service and Web Services instance Trading Service because these meta-elements have the composition relationship in the metamodel. Plan types instances such as Invoke Barter Service and Discover Barter Service (They are the instances of Executor Plan and Discover Plan respectively) can be dropped with all the relationships Plan has. Similar and basic relationships are protected as shown in Figure 3. Barter Service can be as a compartment or independent in the environment.

3. Conclusion

In this paper, the graphical syntax tool required for domain specific modelling of MASs which are enabled to work and interact with the Semantic Web is presented. Also use of the tool is illustrated within an e-barter case study. As the future work of this study, our aim is to enhance the tool with semantic model checking and code generation for various MAS development frameworks.

4. Acknowledgments

This study is funded by The Scientific and Technological Research Council of Turkey (TUBITAK) under grant 109E125.

5. References

[1] Kardas, G., Demirezen, Z., and Challenger, M. “Towards a DSML for Semantic Web enabled Multi-agent Systems”, FML Workshop in ECOOP 2010, ACM Press, pp. 1-5 (2010).

[2] Challenger, M., Getir, S., Demirkol, S., and Kardas, G., “A Domain Specific Metamodel for Semantic Web enabled Mul-

Multi-agent Systems”, Lecture Notes in Business Information Processing 83: 177-186 (2011).

[3] “Eclipse Graphical Modeling Project”, www.eclipse.org/modeling/gmp/ (last access: Sep. 2011)

[4] Demirkol, S., Getir, S., Challenger, M. and Kardas, G., “Development of an Agent based E-barter System”, INISTA 2011, IEEE Computer Society, pp. 193-198 (2011)

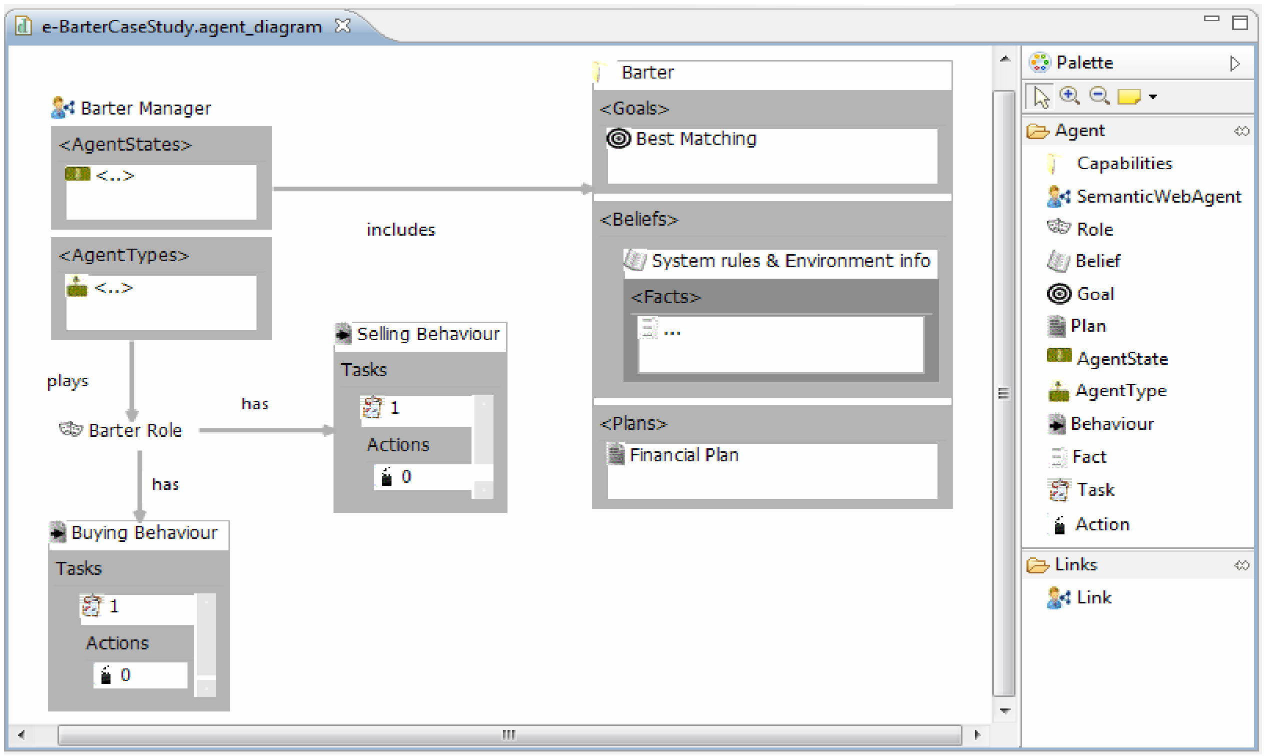


Figure 2. Graphical modeling for agent internal viewpoint of a multi-agent e-barter system in syntax tool of SEA_ML.

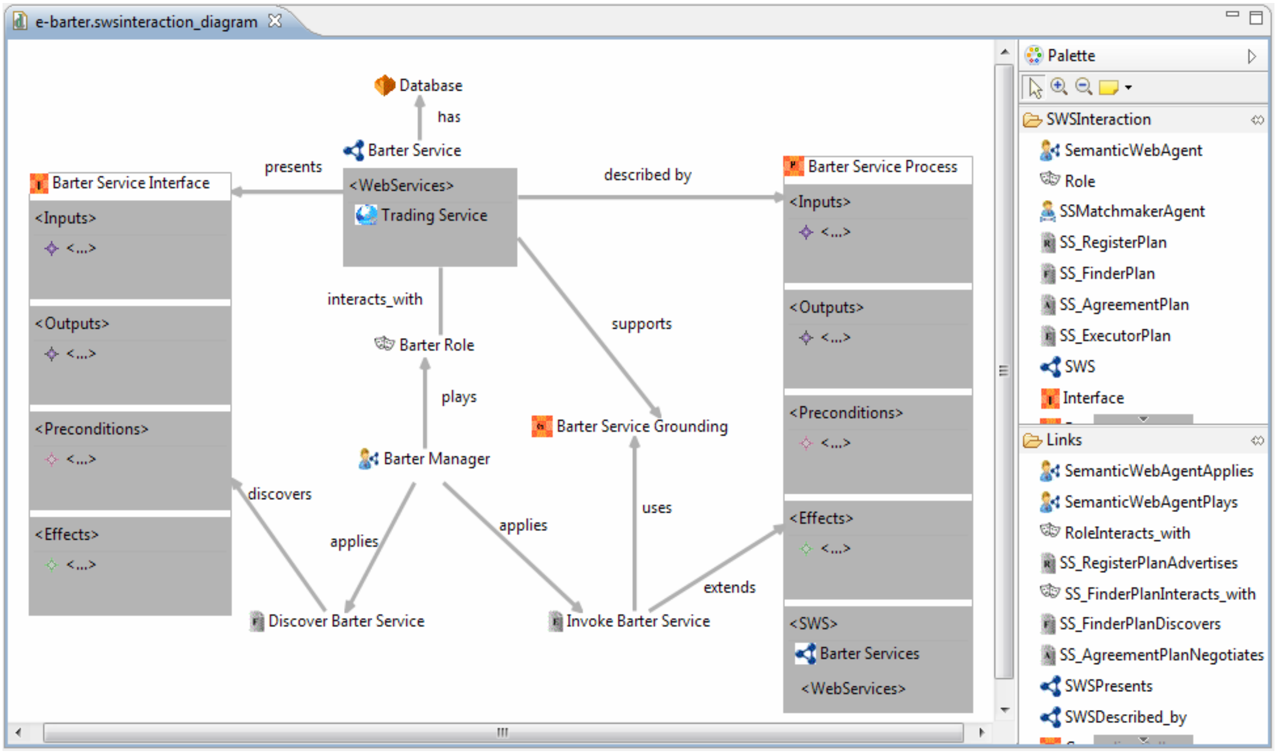


Figure 3. Graphical modeling for agent-SWS viewpoint of a multi-agent e-barter system in syntax tool of SEA_ML