

A Domain Specific Metamodel for Semantic Web enabled Multi-agent Systems

Moharram Challenger, Sinem Getir, Sebla Demirkol and Geylani Kardas

International Computer Institute, Ege University, 35100, Bornova, Izmir, Turkey
moharram.challenger@mail.ege.edu.tr,
{sinem.getir, sebla.demirkol, geylani.kardas}@ege.edu.tr

Abstract. Autonomous, responsive and proactive nature of agents makes development of agent-based software systems more complex than other software systems. A Domain Specific Modeling Language (DSML) may provide the required abstraction and hence support a more fruitful methodology for the development of MASs especially working on the new challenging environments such as the Semantic Web. In this paper, we introduce a domain specific metamodel for MASs working on the Semantic Web. This new metamodel paves the way for definition of an abstract syntax and a concrete syntax for a future DSML of agent systems. Achieved DSML syntax is supported with a graphical modeling toolkit.

Keywords: metamodel, domain specific modeling language, multi-agent system, semantic web

1 Introduction

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 in some environment, and that is capable of flexible autonomous action in this environment in order to meet its design objectives [1]. These autonomous, reactive and proactive agents have also social ability and interact with other agents and humans in order to complete their own problem solving. They may also behave in a cooperative manner and collaborate with other agents to solve common problems. To perform their tasks and interact with each other, intelligent agents constitute systems called Multi-agent systems (MAS).

Considering abovementioned characteristics, the implementation of agent systems is naturally a complex task. In addition, internal agent behaviour model and interaction within the agent organizations become even more complex and hard to implement when new requirements and interactions for new agent environments such as the Semantic Web [2] are taken into account. Semantic Web brought a new vision into agent research. This new generation Web aims to improve World Wide Web (WWW) such that web page contents are interpreted with ontologies. It is apparent that the interpretation in question will be realized by autonomous computational entities –so agents- to handle the semantic content on behalf of their human users.

Software agents are planned to collect Web content from diverse sources, process the information and exchange the results. Autonomous agents can also evaluate semantic data and collaborate with semantically defined entities of the Semantic Web such as semantic web services by using content languages [3].

In [4], we discuss how domain specific engineering can provide easy and rapid construction of Semantic Web enabled MASs and introduce a preliminary domain specific modeling language (DSML), called *Semantic web Enabled Agent Modeling Language (SEA_ML)*, for model driven development of such agent systems. As a domain specific language (DSL), SEA_ML should provide complete definitions for its abstract syntax, concrete syntax and formal semantics.

It is well-known that 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. It consists of a set of provided concepts and their relationships to other concepts [5]. On the other hand, a concrete syntax can be defined as a set of notations that facilitates the presentation and construction of the language. This set of notations can be given in a textual or visual manner. A metamodel that describes the meta-entities and their relationships for a domain can naturally provide a base for the definition of such an abstract syntax and also a concrete syntax. Therefore, in this paper, we introduce a domain specific metamodel for agent systems working on the Semantic Web and describe how it paves the way for the definition of both abstract and visual concrete syntax of SEA_ML. A graphical modeling toolkit for SEA_ML concrete syntax, which is based on Eclipse Graphical Modeling Framework (GMF)¹, is also discussed in this paper.

Rest of the paper is organized as follows: Section 2 discusses the metamodel and SEA_ML's abstract syntax. Section 3 covers the SEA_ML's concrete syntax. Section 4 includes the related work and Section 5 concludes the paper.

2 Metamodel for Semantic Web enabled MASs

The platform independent metamodel, which represents the abstract syntax of SEA_ML, focuses on both modeling the internal agent architecture and MAS organization. Revision of our previous metamodel given in [3] and enhancement of its modeling features have produced the brand new metamodel for SEA_ML abstract syntax within this study. Object Management Group's Ontology Definition Metamodel (ODM)² has been plugged into the new metamodel to help in the definition of ontological concepts. Besides, in addition to the reactive planning, the new metamodel supports modeling of Belief-Desire-Intention (BDI) Agents [6] with new meta-entities and their relations.

To provide clear understanding and efficient use, the new metamodel is divided into six viewpoints each describing different aspects of Semantic Web enabled MASs. These viewpoints are listed as follows:

1. *Semantic Web Agent's Internal Viewpoint*: This viewpoint is related to the internal structures of semantic web agents and defines entities and their

¹ Eclipse Graphical Modeling Framework, <http://www.eclipse.org/gmf> (last access: Feb. 2011)

² Ontology Definition Metamodel, <http://www.omg.org/spec/ODM/1.0/> (last access: Feb. 2011)

A Domain Specific Metamodel for Semantic Web enabled Multi-agent Systems

relations required for the construction of agents. It covers both reactive and BDI agent architectures.

2. *Protocol Viewpoint*: This aspect of the metamodel expresses the interactions and communications in a MAS by taking agent's roles and behaviours into account.
3. *MAS and Organizational Viewpoint*: This viewpoint solely deals with the construction of a MAS as a whole. It includes main blocks which compose the complex system as an organization.
4. *Role and Behaviour Viewpoint*: This perspective delves into the complex controlling structure of the agents. Agent plans and behaviours with all of their attributes are modeled.
5. *Environmental and Services Viewpoint*: Agents may need to access some resources (e.g. ontologies, knowledgebases, discovery and execution services) in their environment. Use of resources and interaction of agents with their surroundings are covered in this viewpoint.
6. *Agent - Semantic Web Service (SWS) Interaction Viewpoint*: It is probably the most important viewpoint of the metamodel. Interaction of semantic web agents with SWSs is described. Entities and relations for service discovery, agreement and execution are defined. Also internal structure of SWSs is modeled within this viewpoint.

We use Kernel MetaMetaModel (KM3) notation from ATL toolkit³ to define our proposed metamodel (and hence SEA_ML abstract syntax) textually. A KM3 class is provided for each entity and associations between each entity of the metamodel are represented with "reference" labels. Role name of each model element and number of instances are also given for every association. In addition to the neat presentation of the abstract syntax, the utilization of the KM3 notation also enables us to employ our MAS metamodel as source or target models in various model-to-model transformations and provides automatic generation of the platform dependent counterparts of the MAS models for different agent deployment platforms (see [3] and [4] for further information). Each viewpoint of the metamodel is discussed in the following subsections. Due to space limitations only the first and the sixth viewpoints' visual representations are given.

2.1 Semantic Web Agent's Internal Viewpoint

This viewpoint focuses on the internal structure of every agent in a MAS organization. Partial metamodel, which represents this viewpoint, is given in Fig. 1. *Semantic Web Agent* in the SEA_ML abstract syntax stands for each agent in Semantic Web enabled MAS. A Semantic Web Agent is an autonomous entity which is capable of interaction with both other agents and semantic web services within the environment. They play roles and use ontologies to maintain their internal knowledge and infer about the environment based on the known facts. Semantic Web Agents can be associated with more than one *Role* at the same point at any time (multiple classification) and can change roles over time (dynamic classification). An agent can

³ Atlas Transformation Language Toolkit, <http://www.eclipse.org/atl/> (last access: Feb. 2011)

play role in various environments, have various states (*Agent State*) and owns a type (*Agent Type*) during his execution.

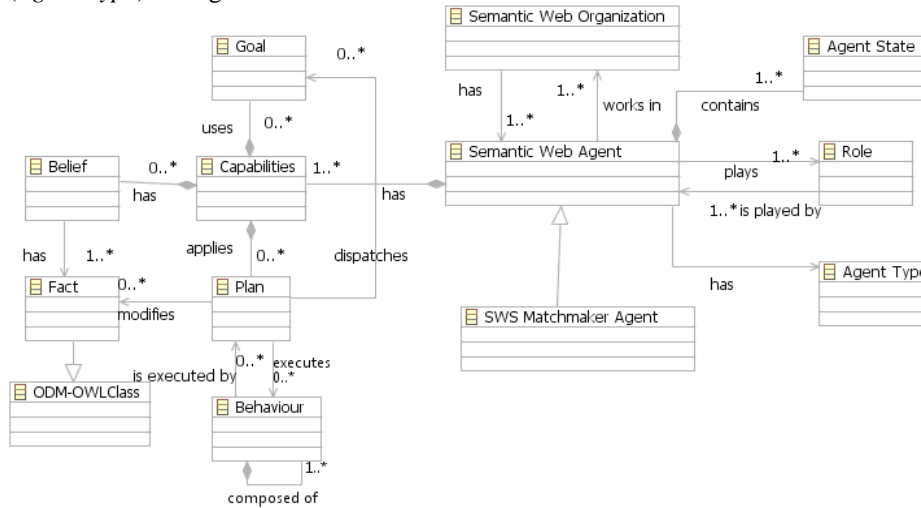


Fig. 1. Semantic Web Agent’s Internal Viewpoint

Metamodel supports both reactive and BDI agents. BDI was first proposed by Bratman [6] and used in many agent systems. In a BDI architecture, an agent decides on which Goals to achieve and how to achieve them. Beliefs represent the information an agent has about its surroundings, while Desires correspond to the things that an agent would like to see achieved. Intentions, which are deliberative attitudes of agents, include the agent planning mechanism in order to achieve goals. Taking into consideration of concrete BDI agent frameworks (such as JADEX⁴ and JACK⁵), we propose an entity called *Capabilities* which includes each agent’s *Goals*, *Plans* and *Beliefs* about the surrounding. Each Belief is composed of one or more *Facts*. For the Semantic Web environment, each fact is an ontological entity and they are modeled as an extension of *OWLClass* from ODM in the metamodel. Semantic Web Agents apply *Plans* to perform their tasks. Each plan executes one or more agent *Behaviours* and goals are achieved during this execution.

On the other hand, agents need to communicate with a service registry in order to discover service capabilities. Hence, the model includes a specialized agent entity, called *SWS Matchmaker Agent*. This entity represents the matchmaker agents which store the capability advertisements of semantic web services within a MAS and match those capabilities with service requirements sent by the other platform agents.

2.2 Protocol Viewpoint

This viewpoint focuses on agent communications and interactions in a MAS and defines entities and relations such as *Protocol*, *Interaction*, *Message* and *Message*

⁴ JADEX BDI Agent, <http://jadex-agents.informatik.uni-hamburg.de/> (last access: Feb. 2011)

⁵ JACK Autonomous Software, <http://www.agent-software.com.au/> (last access: Feb. 2011)

Type. Agents communicate with each other based on their social ability. A *Protocol* in a role uses several *Interactions* to be realized. Each interaction, by itself, consists of some *Message* submissions which are triggered by agent behaviours. Each of the messages should have a Message Type such as inform, request, and acknowledgement. Within a behaviour definition an agent can send or receive a message from other agents.

Protocol viewpoint supports abstraction of commonly-used MAS interaction and messaging approaches. For instance, interactions of agents, which apply the well-known “Contract Net Protocol” (CNP) [7], can be modeled by using this metamodel. CNP as its own can be an instance of the *Interaction* entity. Each communication between initiator and participant agents can be a *Message* and has *Message Types* such as call-for-proposal (cfp), refuse, propose, reject, and accept. Likewise, messages and message types, defined by IEEE Foundation for Intelligent Physical Agents (FIPA) Agent Communication Language (ACL) specification⁶, can be described in meta-level by using the entities given in this metamodel.

2.3 MAS and Organizational Viewpoint

Structure and organization of a MAS are modeled within this viewpoint. *Semantic Web Organization* entity of SEA_ML metamodel is a composition of Semantic Web Agents which is constituted according to the organizational roles of those agents. An agent cooperates with one or more agents inside an organization and he may also reside in more than one organization. Moreover, a Semantic Web Organization can include several agents at any time and each organization can be composed of several sub-organizations. A Semantic Web Organization is inconceivable without ontologies. An ontology represents any information gathering and reasoning resource for MAS members. Collection of the ontologies creates knowledgebase of the MAS that provides domain context. These ontologies are represented in SEA_ML models as *Organization Ontology* instances.

2.4 Role and Behaviour Viewpoint

Semantic web agents can play roles and use ontologies to maintain their internal knowledge and infer about the environment based on the known facts. They can also use several roles at any time and can alter these roles over time. Task definitions and related task execution processes of Semantic Web agents are modeled with *Behaviour* concepts. *Role* is a general model entity and it should be specialized in the metamodel according to task definitions of architectural and domain based roles: An *Architectural Role* defines a mandatory Semantic Web enabled MAS role (e.g. registration or ontology mediator) that should be played at least one agent inside the platform regardless of the organization context whereas a *Domain Role* completely depends on the requirements and task definitions of a specific Semantic Web Organization created for a specific business domain. Inside a domain role, an agent

⁶ FIPA Agent Communication Language Message Structure Specification, <http://www.fipa.org/specs/fipa00061/> (last access: Feb. 2011)

uses a *Role Ontology* which is defined for the related domain concepts and their relations. An agent participates within a communication or task *Scenario* over the role(s) he plays. One role includes several agent Behaviours and each Behaviour is composed of many *Tasks*. Each Task also covers one or more atomic *Actions* (such as sending a message to another agent or querying an ontology).

2.5 Environmental and Services Viewpoint

This viewpoint focuses on agents' use of resources and environmental interactions. SEA_ML's core concepts defined for this viewpoint can be listed as follows: *Environment*, *Resource*, *Permission Table*, *Service*, *Semantic Web Service* and *Service Ontology*. An agent can access many *Environments* during his execution and an environment can include many *Resources*, (e.g. database, network device) with their access permissions in *Permission Tables*.

An environment also includes *Semantic Web Services*. A Semantic Web Service represents any service (except agent services) whose capabilities and interactions are semantically described within a Semantic Web enabled MAS. A Semantic Web Service composes one or more *Service* entities. Each service may be a web service or another service with predefined invocation protocol in real-life implementation. But they should have a semantic web interface to be used by autonomous agents of the platform. It should be noted that association between the semantic web agents and the services is provided over the agent *Role* entities in the metamodel. Because agents interact with semantic web services, depending on their roles defined inside the organization [3]. Semantic interfaces and capabilities of Semantic Web Services are described according to *Service Ontologies*.

2.6 Agent - Semantic Web Service Interaction Viewpoint

Perhaps the most important viewpoint of SEA_ML metamodel is the one which models the interaction between agents and SWSs. Concepts and their relations for appropriate service discovery, agreement with the selected service and execution of the service are all defined. Furthermore, internal structure of SWSs is modeled inside this viewpoint.

Fig. 2 portrays the agent – SWS interaction viewpoint of SEA_ML. Since, *Semantic Web Agent*, *Role*, *Registry Role*, *Plan*, *Behaviour* and *SWS Matchmaker Agent* concepts are imported from above discussed viewpoints, they will only be referenced when their relations between core concepts of this viewpoint are discussed.

Semantic web service modeling approaches (e.g. OWL-S⁷) mostly describe services by three semantic documents: Service Interface, Process Model and Physical Grounding. Service Interface is the capability representation of the service in which service inputs, outputs and any other necessary service descriptions are listed. Process Model describes internal composition and execution dynamics of the service. Finally, Physical Grounding defines invocation protocol of the web service. These Semantic

⁷ OWL-S: Semantic Markup for Web Services, <http://www.w3.org/Submission/OWL-S/> (last access: Feb. 2011)

A Domain Specific Metamodel for Semantic Web enabled Multi-agent Systems

Web Service components are given in our metamodel with *Interface*, *Process* and *Grounding* entities respectively. *Input*, *Output*, *Precondition* and *Effect* (a.k.a. IOPE) definitions used by these Semantic Web Service components are also defined. The metamodel imports *OWLClass* meta-entity from the OMG's ODM as the base class for the semantic properties (mainly IOPE) of the semantic web services. Since the operational part of today's semantic services is mostly a web service, *Web Service* concept is also included in the metamodel and associated with the grounding mechanism.

Semantic Web Agents apply *Plans* to perform their tasks. In order to discover, negotiate and execute Semantic Web Services dynamically, three extensions of the *Plan* entity are defined in the metamodel. *Semantic Service (SS) Finder Plan* is a *Plan* in which discovery of candidate semantic web services takes place. *SS Agreement Plan* involves the negotiation on QoS metrics of the service (e.g. service execution cost, running time, location) and agreement settlement. After service discovery and negotiation, the agent applies the *SS Executor Plan* for executing appropriate semantic web services.

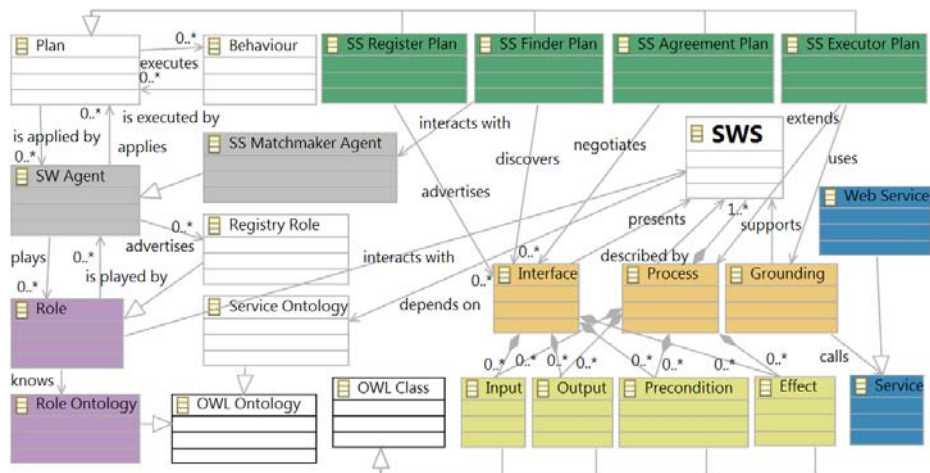


Fig. 2. Agent - Semantic Web Service Interaction Viewpoint











As discussed before, *Semantic Service Matchmaker Agents* represent service registries for agents to discover service capabilities. During their executions, they apply *SS Register Plans*.

3 SEA_ML's Concrete Syntax

While specification of abstract syntax includes the concepts that are represented in the language and the relationships between those concepts, concrete syntax definition provides a mapping between meta-elements and their textual or graphical representations. In this study, we propose a graphical concrete syntax for SEA_ML. KM3 representation of SEA_ML metamodel has been converted to an Ecore

representation and graphical notation for each language concept and relation has been chosen. Table 1 lists graphical notations for some SEA_ML concepts.

Table 1. Some of the concepts and their notations for the concrete syntax of SEA_ML

Concept	Notation	Concept	Notation
Semantic Web Agent		Semantic Web Organization	
Role		Belief	
Goal		Semantic Web Service	
Plan		Agent State	
Capabilities		Behaviour	

After choosing the graphical notation, we used Eclipse GMF to tie the domain concepts (supplied by the abstract syntax of SEA_ML) in Ecore format and their notations together. Achieved artifact is a graphical editor in which agent developers may design models for each viewpoint of required MAS conforming to the concrete syntax of SEA_ML. Screenshot in Fig. 3 illustrates use of the editor for modeling agent - semantic web service viewpoint of a multi-agent electronic barter (e-barter) system.

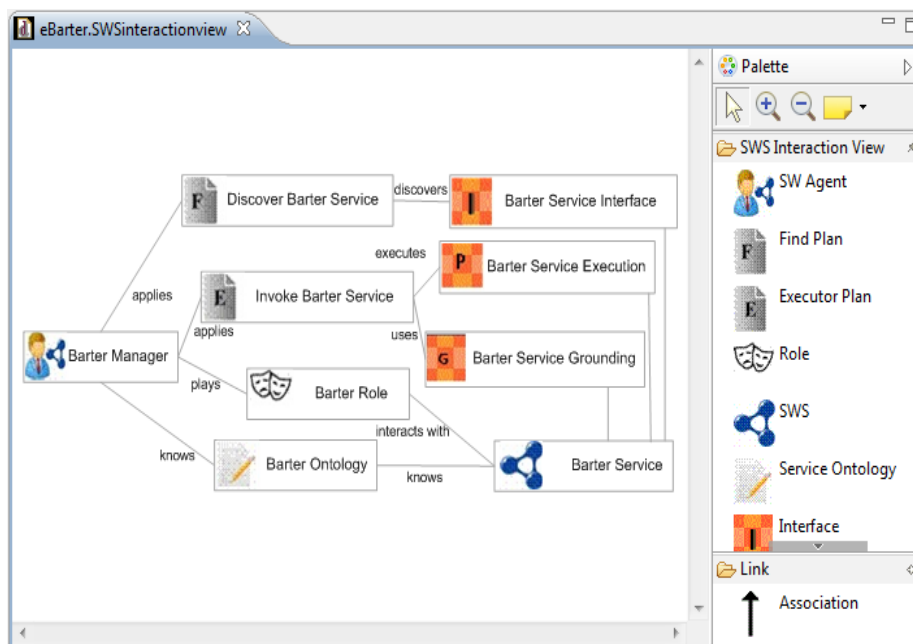


Fig. 3. Modeling agent-SWS viewpoint of a multi-agent e-barter system

A Domain Specific Metamodel for Semantic Web enabled Multi-agent Systems

An agent-based e-barter system consists of agents that exchange goods or services for their owners without using any currency. In our example, Barter Manager agent (shown in Fig. 3) manages all trades in the system. This agent is responsible for collecting barter proposals, matching proper barter proposals and tracking the bargaining process between customer agents. 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. Conforming to his Barter Role definition, Barter Manager needs to discover the proper SWS, interacts with the candidate service and realizes the exact execution of the SWS after an agreement. Fig. 3 shows how the related interaction can be modeled by using the concrete syntax constructs of SEA_ML. More information on this case study can be found in [4].

4 Related Work

Recent work on metamodeling of agent systems mostly considers definition of metamodels specific for some MAS development methodologies or generation of platform independent agent metamodels. For instance, Bernon et al. [8] give metamodels for MAS development methodologies called ADELFE, Gaia and PASSI. They introduce a unified metamodel composed by merging the most significant contributions of these methodologies. A similar study [9] introduces a metamodel for SODA agent development methodology. The study aims to model interaction and social aspects of the SODA and defines a metamodel considering these aspects. However, those metamodels are just formal representations for the concepts of the related methodologies and they are not suitable for the general MAS modeling.

FAML metamodel, introduced in [10], is in fact a synthesis of various existing metamodels for agent systems. Design time and runtime concepts for MASs are given and validation of these concepts is provided with their use in again various MAS development methodologies. Platform independent metamodel, proposed in [11] groups agent modeling concepts in various viewpoints in the same manner with our study. But neither [10] nor [11] consider a DSML specification for MAS development.

The syntax proposed in [5] is perhaps the most related work with our study. At first, the abstract syntax of a MAS DSML is represented by a platform independent metamodel and then a visual concrete syntax is defined based on the given concepts and their notations. However generated syntax does not support both agents on the Semantic Web and the interaction of Semantic Web enabled agents with their environment.

5 Conclusion

A metamodel for the domain of MASs working on the Semantic Web is discussed. Taking into consideration of internal agent architectures, metamodel supports both reactive and BDI agent structures. For MAS perspective, agent communication protocols can be modeled with the proposed metamodel. Also interactions of agents

Moharram Challenger, Sinem Getir, Sebla Demirkol and Geylani Kardas

with each other and semantic entities such as SWS are supported. Proposed metamodel presents an abstract syntax and causes the derivation of a graphical concrete syntax for a DSML for Semantic Web enabled MASs, called SEA_ML. Achieved syntax is supported with an Eclipse GMF-based toolkit. Future work consists of the formal representation of the semantics for SEA_ML and development of an integrated tool to support all features of SEA_ML.

Acknowledgements

This study is funded by The Scientific and Technological Research Council of Turkey (TUBITAK) Electric, Electronic and Informatics Research Group (EEEAG) under grant 109E125.

References

1. Wooldrige, M., Jennings, N.R.: Intelligent agents: theory and practice. *Knowl. Eng. Rev.* 10(2), 115--152 (1995)
2. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. *Sci. Am.* 284(5), 34--43 (2001)
3. Kardas, G., Goknil, A., Dikenelli, O., Topaloglu, N.Y.: Model Driven Development of Semantic Web Enabled Multi-agent Systems. *Int. J. Coop. Inf. Syst.* 18(2), 261--308 (2009)
4. Kardas, G., Demirezen, Z., Challenger, M.: Towards a DSML for Semantic Web enabled Multi-agent Systems. In: International Workshop on Formalization of Modeling Languages, held in conjunction with the 24th European Conference on Object-Oriented Programming (ECOOP 2010), pp. 1-5, ACM Press (2010)
5. Warwas, S., Hahn, C.: The concrete syntax of the platform independent modeling language for multiagent systems. In: Agent-based Technologies and applications for enterprise interoperability, held in conjunction with the 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008) (2008)
6. Bratman, M.E.: *Intention, Plans, and Practical Reason*. Harvard University Press: Cambridge, Massachusetts (1987)
7. Smith, R.G.: The Contract Net Protocol: High-level Communication and Control in a Distributed Problem Solver. *IEEE T. Comput.* 29(12), 1104--1113 (1980)
8. Bernon, C., Cossentino, M., Gleizes, M-P., Turci, P., Zambonelli, F.: A Study of some Multi-Agent Meta-Models. In: J. Odell et al. (eds.) AOSE 2004. LNCS, vol. 3382, pp. 62-77. Springer, Heidelberg (2005)
9. Molesini, A., Denti, E., Omicini, A.: MAS Meta-models on Test: UML vs. OPM in the SODA Case Study. In: Pechoucek, M., Petta, P., Varga, L.Z. (eds.) CEEMAS 2005. LNAI, vol 3690, pp. 163-172. Springer, Heidelberg (2005)
10. Beydoun, G., Low, G.C., Henderson-Sellers, B., Mouratidis, H., Gómez-Sanz, J.J., Pavon, J, Gonzalez-Perez, C.: FAML: A Generic Metamodel for MAS Development. *IEEE T. Software Eng.* 35(6), 841--863 (2009)
11. Hahn, C., Madrigal-Mora, C., Fischer, K.: A platform-independent metamodel for multiagent systems. *Auton. Agent. Multi-Ag.* 18(2), 239--266 (2009)