A Multi-Agent Systems Engineering Tool based on Ontologies

Artur Freitas, Lucas Hilgert, Sabrina Marczak
Felipe Meneguzzi, Rafael H. Bordini and Renata Vieira

Graduate Program in Computer Science - Faculty of Informatics (FACIN)
Pontifical Catholic University of Rio Grande do Sul (PUCRS) - Porto Alegre, Brazil
{artur.freitas,lucas.hilgert}@acad.pucrs.br,
{sabrina.marczak,felipe.meneguzzi,rafael.bordini,renata.vieira}@pucrs.br

Abstract. This paper describes a tool that supports the development of agent-oriented software on the basis of an ontology. The tool implements techniques for transforming instantiations from conceptual models into MAS implementations.

Keywords: Model-driven engineering, ontology, multi-agent system

1 Introduction

Model-Driven Engineering (MDE) employs conceptual models as the cornerstone of software development processes [3] in order to improve productivity, portability, interoperability, maintenance, and so on. MDE provides abstractions and notations to improve understanding and to support the modelling of applications for specific domains. These advantages can also be employed in the development of Multi-Agent Systems (MAS) given their complexity and the need for integrating several components that are often addressed from different angles. For example, the JaCaMo [1] framework for MAS programming combines three separate technologies: Jason for coding agents in AgentSpeak, CArtAgO for programming the environment as artifacts in Java, and Moise for specifying MAS organisations in XML. These three distinct starting points to code the MAS should work as a single conceptual model, thus combining these separate MAS dimensions is desirable. The use of models is present in most agent methodologies [3], and MDE techniques for agent-oriented software engineering emerges naturally. Prometheus [5] is one example of MDE for MAS; however, differently from our work, formal ontologies are not explored as part of such models. Since MDE and ontologies share a number of principles and goals, and since there is much work combining ontologies and MAS, these synergies led us to investigate their uses in tools for MAS modelling and development, considering features such as code generation, support during programming, and reasoning about the system. Although the advantages of ontologies for agents are clear, few MAS platforms currently incorporate ontology techniques throughout the entire systems development life cycle. We are not aware of any previous exploitation of ontologies in the global modelling of MAS where such model is used also in a tool during the programming phase. It is our claim that tools

for designing complex systems should consider models that are clear to communicate, allow reuse and reasoning over the specification, and provide support during programming. Thus we propose a model and tool that: *(i)* covers MAS design and development as a whole in an integrated formalism; and *(ii)* enables techniques for transforming instantiations from such conceptual model represented as an ontology into effective MAS implementations. This paper presents our proposal of using MAS conceptual modelling and an Eclipse plug-in that provides features such as drag-and-drop and auto-complete from ontologies for coding MAS using agent-oriented programming languages.

2 Ontology-based Modelling of Multi-Agent Systems

We designed an ontology for MAS modelling that represents the main dimensions usually considered in MAS development frameworks: agents, environments, and organisations. Our MAS ontology employs a single and integrated formalism for these dimensions, which allows us to model, reuse, and transform it into effective MAS implementations. The methodology for modelling MAS consists of creating instances of the concepts and properties from such ontology, which can be done with any ontology editor. Our ontology was defined from analysing and combining the meta-models of Prometheus [5] and JaCaMo [1], which was also done by other, e.g. [6], but without using an ontology and without presenting model-based development tools. Our approach allows designers to gradually move from high-level abstract views to elements directly available in concrete technical MAS programming platforms. The MAS initial model is presented in Fig. 1. In the development of a MAS, the main concepts of our ontology are instantiated by the designers. They specify the agents in their particular system, what is observable in the environment, and the characteristics and roles of the agents in the organisation. For example, suppose the design of a MAS to simulate a soccer match. Instances of *Agent* will be *player* and *coach*. *Actions* such as *move*, *pass ball*, and *change strategy* could be modelled, and *Actions* can be related to *Agents* through the *has-action* property. The *ball* could be modelled as an *Artifact*, containing an *Observable Property* such as *position*, associated to it through the *has-property* relationship. The team can be organised in *Groups* that include *Roles* (e.g., *defender* and *midfielder*) by means of *contains-role*. These examples show how to specify MAS elements with our ontology. Our proposal has been tested with a group of five students; the results show that it improves how MAS are expressed, communicated, and converted into effective MAS implementations. It allows designing without going into specifics of programming languages, and it groups together what is developed using separate technologies.

3 Ontology-based Multi-Agent System Development Tool

As mentioned above, the design process consists of instantiating a top-ontology with details of a particular MAS. Our tool is an Eclipse plug-in that uses the instantiated model to support MAS programming. Our plug-in (to be used in the "Jason Perspective", provided by the Jason or JaCaMo plug-ins) loads OWL ontologies and provides two model-based programming features to develop Jason [2] agent code: drag-and-drop and auto-complete from ontologies. The plug-in was developed using the OWL API [4]

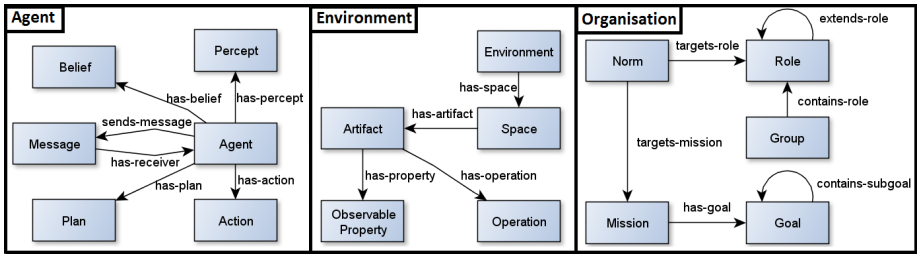

Fig. 1. Main concepts and properties of the MAS ontology model.

for manipulating the ontology. The drag-and-drop from the ontology to the agent code is illustrated in Fig. 2, which depicts the Jason perspective in Eclipse. On the right-hand side of the image, the developer can navigate the MAS model encoded as ontology concepts, instances, and properties. These elements can be dragged to the left-hand side that contains the AgentSpeak code of Jason agents (in this case, file *player.asl*). As exemplified in Fig. 2, the programmer is selecting an instance, named *pass_ball*, from the *Agent Action* concept, to be inserted into a plan body. Similarly, our tool provides auto-complete from ontology to agent code, which is activated when the developer is typing Jason code (or pressing the shortcut "ctrl+space"). Then, the available options based on the ontology are presented to the programmer as suggestions. One example is when coding a plan context, which may be composed of ontology-based queries (e.g., checking if an individual belongs to a concept). The generation of agent code from ontologies takes into consideration the context where an element is dragged into the code. For example, when dropping an instance of "*Role*" inside a plan's body, the action to adopt that role can be automatically created; when a "*Message*" is dropped, an action to send the corresponding message can be generated. Moreover, as the MAS is being

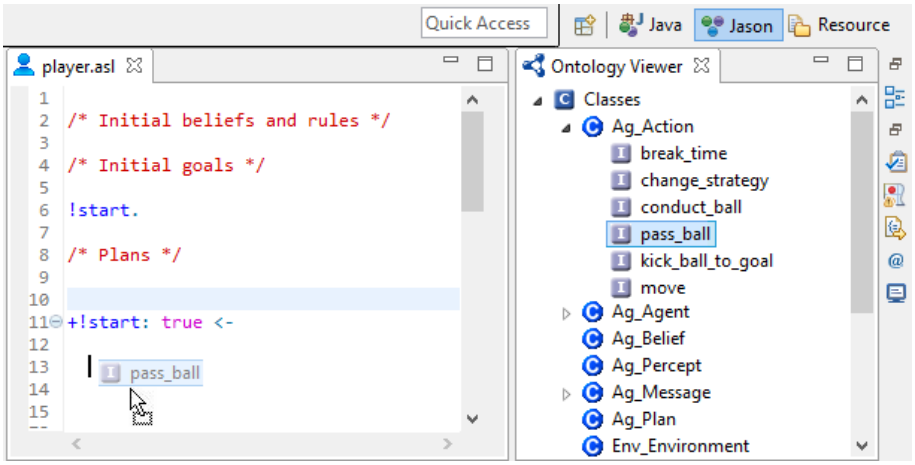

Fig. 2. Drag-and-drop in Eclipse from ontology model to MAS implementations.

coded, updated information about the MAS has to be reflected in the model. One of the next steps in our tool development is the use of semantic reasoning for purposes such as consistency checking or inferring new knowledge about the MAS specified in the ontology. The evaluation of our tool with participants of an experiment pointed out that it is intuitive to use, easy to understand, easy to visualise, useful in programming, and improves model-program approximation. It helps with code consistency (as it encourages developers to consistently use terms from the ontology) and provides an overview of the MAS in the programming context, combined with features such as dragging content from a model into MAS code. Further MAS code could be generated automatically from this model, and the ontology can be more restrictive during MAS coding (i.e., indicating errors or mismatches between model and code). Also, the Eclipse plug-in could allow users to edit the MAS model (e.g., to include new instances) without having to use an ontology editing tool.

4 Final Remarks

Ontologies offer significant advantages for MAS model-based development, such as interoperability, reusability, support for MAS development activities (such as system analysis and agent knowledge modelling) and support for MAS operations (such as agent communication and reasoning). In terms of MAS design, our model provides a global conceptual view, often missing in agent platforms, which is extended by MAS engineers in the initial phases of their development processes. As a result, developers obtain new features for developing complex software systems with an infrastructure that combines and applies modelling, software, and knowledge engineering principles. For example, reasoners can validate meta-models automatically or generate MAS code from models, all of which contribute to more principled ways to develop MAS. As future work, we plan to assess some of these features and advantages derived from our proposal. Further details about our model, tool, and their evaluation can be found at https://demoproposal.wordpress.com/.

References

1. Boissier, O., Bordini, R.H., Hübner, J.F., Ricci, A., Santi, A.: Multi-agent oriented programming with JaCaMo. Science of Computer Programming 78(6), 747–761 (2013)
2. Bordini, R.H., Hübner, J.F., Wooldridge, M.: Programming multi-agent systems in AgentSpeak using Jason. John Wiley & Sons (2007)
3. Gascueña, J.M., Navarro, E., Fernández-Caballero, A.: Model-driven engineering techniques for the development of multi-agent systems. Engineering Applications of Artificial Intelligence 25(1), 159–173 (2012)
4. Horridge, M., Bechhofer, S.: The OWL API: a Java API for OWL ontologies. Semantic Web 2(1), 11–21 (2011)
5. Padgham, L., Winikoff, M.: Prometheus: A methodology for developing intelligent agents. In: Giunchiglia, F., Odell, J., Weiß, G. (eds.) Agent-Oriented Software Engineering III. LNCS, vol. 2585, pp. 174–185. Springer (2003)
6. Uez, D., Hübner, J.: Environments and organizations in multi-agent systems: From modelling to code. In: Dalpiaz, F., Dix, J., van Riemsdijk, M. (eds.) Engineering Multi-Agent Systems. LNCS, vol. 8758, pp. 181–203. Springer (2014)