xMOF: A Semantics Specification Language for Metamodeling

Tanja Mayerhofer, Philip Langer, Manuel Wimmer

Business Informatics Group, Vienna University of Technology, Austria
{mayerhofer, langer, wimmer}@big.tuwien.ac.at

Abstract. While MOF constitutes a standardized and widely accepted language for formally defining a modeling language's abstract syntax, no standardized language for specifying its behavioral semantics exists. This impedes the efficient development of tools which build upon the behavioral semantics of modeling languages, such as model interpreters, debuggers, and testing environments. To address this shortcoming, we propose to adopt the standardized action language fUML as semantics specification language in metamodeling. In this paper we present tool support integrated with the EMF environment for specifying the behavioral semantics of modeling languages with fUML as well as for executing models based on these specifications. A teaser for the demonstration of this tool support can be found at http://www.youtube.com/watch?v=p4alBfqbjk8.

1 Introduction

In model-driven engineering (MDE), models constitute the central artifacts in the software development process. Thus, the success of MDE depends significantly on the availability of adequate tool support for creating, exploring, analyzing, and utilizing models. In order to develop such tools efficiently, modeling languages (i.e., their *syntax* and *semantics*) have to be defined formally [1]. MOF [4] constitutes a standardized and widely accepted metamodeling language for formally defining the *abstract syntax* of modeling languages and laid the ground for the emergence of a variety of tools building upon the abstract syntax definition of a modeling language, such as techniques for deriving modeling editors from a metamodel and generic components for model serialization, comparison, and transformation. For formally defining a modeling language's *behavioral semantics* no standardized language exists. This hampers the efficient development of tools which build upon the behavioral semantics of a modeling language, such as model interpreters, debuggers, and testing environments.

Due to this lack of a standardized language for specifying the behavioral semantics of modeling languages, we investigated whether the standardized action language fUML [6] can be used as semantics specification language in metamodeling and how it can be integrated with existing metamodeling methodologies and environments. This investigation resulted in the *metamodeling language xMOF* which integrates fUML with Ecore, which is the most prominent implementation of MOF and employed in EMF [7]. Furthermore, we elaborated a *methodology for systematically developing semantics specifications* with xMOF and utilizing them for model execution. This method-

ology integrates seamlessly with existing metamodeling methodologies and environments which enabled us to implement conforming tool support for EMF[1].

In Section 2 we give a brief overview of our metamodeling language xMOF. Section 3 presents our methodology for developing semantics specifications based on xMOF and its tool support integrated with EMF. In Section 4 we give an overview of the tool demonstration. Section 5 concludes this paper.

2 Semantics Specification Language xMOF

fUML [6] is a subset of UML [5] comprising modeling concepts for defining UML classes as well as activities defining the classes' behavior. Furthermore, fUML defines the execution semantics of this UML subset in terms of a virtual machine (VM) capable of executing compliant models. Because UML classes and MOF metaclasses differ only in their intended usage (modeling of systems vs. modeling of languages), we argue that fUML might be well suited for also defining the behavior of metaclasses. Furthermore, as fUML is an object-oriented and imperative action language and well known in the MDE community as it is a subset of UML, which is widely adopted in MDE, it might be intuitive to use for specifying the behavioral semantics of modeling languages. As both MOF and fUML are standardized by OMG, fUML may be considered as promising candidate for becoming a standardized action language in metamodeling.

For integrating fUML with existing metamodeling languages in order to enable the specification of the behavioral semantics of modeling languages we identified two strategies [3]: a transformation-based and an integration-based strategy. Because of the better integration with existing metamodeling environments we decided to apply the integration-based strategy. In this strategy, a metamodeling language is extended with the behavioral part of fUML comprising modeling concepts for activities and actions so that the behavior of metaclass operations defined in the metamodel of a modeling language can be specified in terms of fUML activities. By applying this strategy for integrating fUML with Ecore, which is the most prominent implementation of MOF, we obtained a new metamodeling language which we called *executable MOF (xMOF)*.

The metamodel of xMOF is depicted in Figure 1. For integrating Ecore with the behavioral part of fUML we introduced the metaclasses BehavioredEClassifier, BehavioredEClass, MainEClass, and BehavioredEOperation. The metaclass BehavioredEClassifier is a subclass of EClassifier and can own Behaviors in terms of Activities. BehavioredEClass is a concrete subclass of BehavioredEClassifier and EClass and can therefore own Activities. The class MainEClass is introduced to distinguish one BehavioredEClass in a semantics specification as the main class controlling the execution of a model conforming to the modeling language. We also introduced a subclass of EOperation called BehavioredEOperation whose behavior can be defined by an Activity.

3 Methodology for Developing xMOF Semantics Specifications

With xMOF it is now possible to define the abstract syntax and the behavioral semantics of modeling languages. To foster the systematic and efficient development of behavioral semantics specifications using xMOF as well as the utilization of these specifications

[1] The metamodel of xMOF, the source code of our tool support as well as demos and case studies can be found at our project website `http://www.modelexecution.org`.

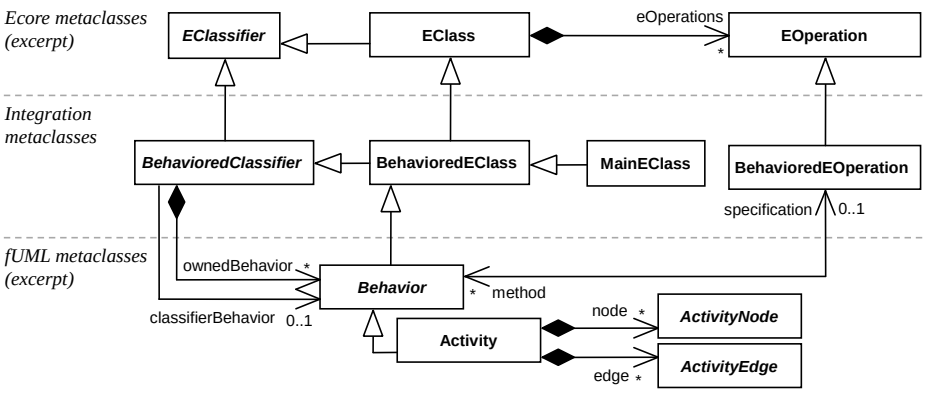

Fig. 1: Metamodel of xMOF (excerpt)

for executing models, we elaborated a dedicated methodology which is accompanied by EMF-based tool support. An overview of our methodology is depicted in Figure 2.

In the **semantics specification**, the behavioral semantics of a modeling language is developed starting from its *Ecore-based metamodel*. The behavioral semantics is defined in an own artifact called *xMOF-based configuration* which is automatically initialized. In this initialization one subclass (BehavioredEClass) of each metaclass defined in the metamodel is generated. They are called *configuration classes* and can now be extended with additional attributes, references, operations, and activities for specifying the behavioral semantics of the metaclasses. Also additional configuration classes can be defined. Furthermore, one BehavioredEClass called Initialization is generated which can be used to define supplementary data that is necessary as additional input for executing models based on this semantics specification. This class can be extended with attributes, references, and additional contained classes called *initialization classes*.

In the **model execution preparation**, preparatory tasks for executing a model according to the xMOF-based behavioral semantics specification of the used modeling language are carried out. In EMF, models consist of instances of the metaclasses defined in the Ecore-based metamodel of the modeling language. As the semantics of these metaclasses is defined by the operations introduced in the configuration classes, the model to be executed has to be represented in terms of instances of the respective configuration classes. This representation is called *xMOF-based model* and is generated automatically. Obviously, the supplementary data needed as input for executing a model defined by the initialization classes has to be instantiated manually by the modeler.

The xMOF-based model can be executed by leveraging the fUML VM (**model execution**). Therefore, it is automatically converted into an *fUML-based model* conforming to the format required by the fUML VM. During the execution the fUML VM interprets the activities specifying the behavioral semantics of the modeling language and manipulates the fUML-based model accordingly. The result of the execution consists of the manipulated fUML-based model (*fUML extensional values*) representing the runtime state of the executed model after the execution finished. For enabling its visualization *model annotations* are generated for the executed model. For this we make use of EMF Profiles [2] which is an mechanism for annotating EMF-based models.

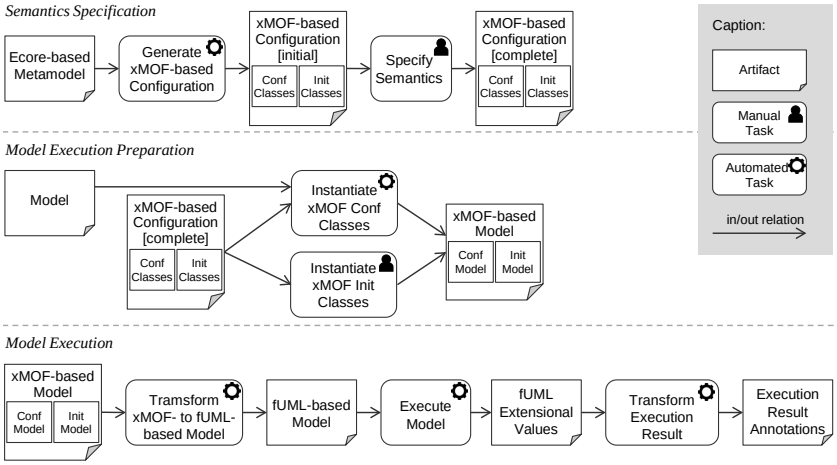

Fig. 2: Overview of our semantics specification approach based on fUML

4 Demonstration

In the demonstration of our tool support for specifying the behavioral semantics of modeling languages, we will develop a *Petri Net modeling language (PNML)* following the methodology introduced in Section 3. We will showcase how the following tasks are accomplished using our EMF-based tool support: (i) initialization of the xMOF-based configuration from the Ecore-based metamodel of PNML, (ii) specification of the behavioral semantics of PNML by extending the xMOF-based configuration, (iii) execution of PNML models based on the xMOF-based configuration of PNML.

The Ecore-based metamodel of PNML (cf. Figure 3a) defines that a Net consists of Places and Transitions whereas a Transition is associated with at least one input and one output Place. The xMOF-based configuration of PNML (cf. Figure 3b) consists of the configuration classes NetConfiguration, PlaceConfiguration, and TransitionConfiguration generated for the metaclasses of PNML. For the configuration class TransitionConfiguration the operation fire() was introduced whose behavior is specified by the activity depicted in Figure 3c. This activity specifies that for the output places of a transition the operation addToken() is called while for the input places removeToken() is invoked. Furthermore, the initialization class Token was introduced which has to be used for defining the initial token distribution in the net to enable the executing of a PNML model.

5 Conclusion

We have presented the metamodeling language xMOF integrating fUML with Ecore. It enables to define the behavioral semantics of modeling languages in an operational way. Furthermore, we gave an overview about our methodology for developing xMOF-based semantics specifications and utilizing them for model execution and presented accompanying tool support for EMF.

To evaluate the applicability of our semantics specification approach, we carried out several case studies in which we developed the behavioral semantics specifications of distinct modeling languages and utilized these specifications to execute conforming

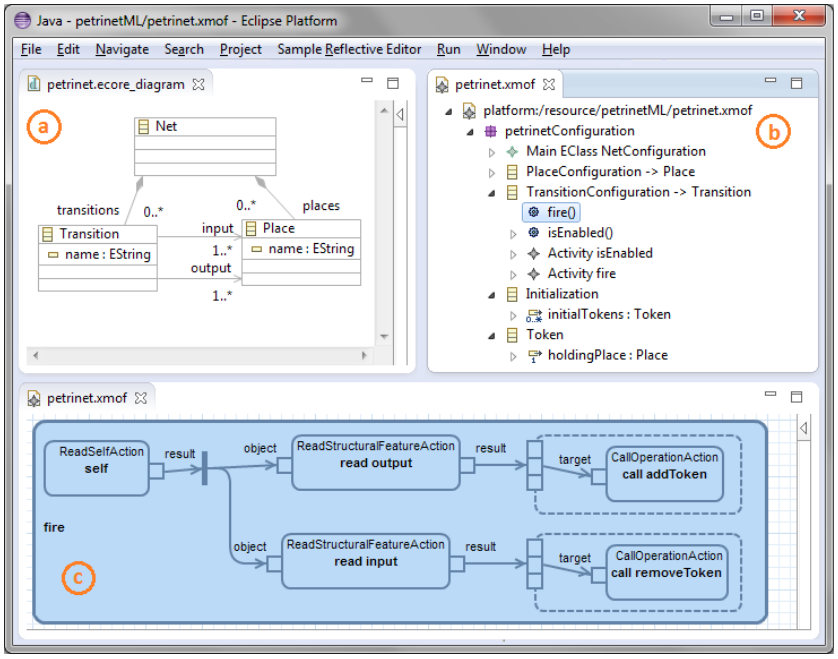

Fig. 3: Tool support for specifying behavioral semantics with fUML

models using the implemented tool support. In summary, the case studies confirmed that xMOF, its accompanying methodology, and tool support are applicable for defining the behavioral semantics of different kinds of modeling languages. Regarding the suitability of fUML as semantics specification language, we come to the conclusion that due to its object-oriented and imperative nature, fUML is highly suitable.

References

1. B. R. Bryant, J. Gray, M. Mernik, P. J. Clarke, R. B. France, and G. Karsai. Challenges and Directions in Formalizing the Semantics of Modeling Languages. *Computer Science and Information Systems*, 8(2):225–253, 2011.
2. P. Langer, K. Wieland, M. Wimmer, and J. Cabot. EMF Profiles: A Lightweight Extension Approach for EMF Models. *Journal of Object Technology*, 11(1):1–29, 2012.
3. T. Mayerhofer, P. Langer, and M. Wimmer. Towards xMOF: Executable DSMLs based on fUML. In *Proc. of the 12th Workshop on Domain-Specific Modeling*, pages 1–6. ACM, 2012.
4. Object Management Group. OMG Meta Object Facility (MOF) Core Specification, Version 2.4.1, August 2011. Available at: http://www.omg.org/spec/MOF/2.4.1.
5. Object Management Group. OMG Unified Modeling Language (OMG UML), Superstructure, Version 2.4.1, August 2011. Available at: http://www.omg.org/spec/UML/2.4.1.
6. Object Management Group. Semantics of a Foundational Subset for Executable UML Models (fUML), Version 1.0, February 2011. Available at: http://www.omg.org/spec/FUML/1.0.
7. D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks. *EMF: Eclipse Modeling Framework*. Addison-Wesley Professional, 2nd edition, 2008.