Using fUML as Semantics Specification Language in Model Driven Engineering

Tanja Mayerhofer

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

Abstract. In model-driven engineering (MDE), software is developed based on models which hence constitute the central artifacts in the software development process. Consequently, tools supporting MDE, such as model editors, interpreters, and debuggers are crucial in MDE. For developing such tools efficiently, modeling languages have to be defined formally. While for formally defining a modeling language's syntax standard means exist, this is not the case for defining its semantics. This impedes the efficient development of tools which build upon the modeling language's semantics, such as model interpreters, debuggers, and testing environments. To overcome this limitation, we present an approach for formally defining the semantics of modeling languages making use of the standardized and UML 2 compliant action language foundational UML (fUML).

1 Problem and Motivation

Over the past years, the software development methodology model-driven engineering (MDE) gained significant popularity as it is a promising approach to address the growing complexity of the software systems that have to be built today [17]. In MDE, models are specified using modeling languages to define the structure and behavior of the software to be built. Model transformations and code generation are used to generate different kinds of software artifacts from these models, such as source code and deployment scripts. Because models constitute the central artifacts in MDE, its success depends significantly on the availability of adequate tool support for creating, exploring, analyzing, and utilizing models. In order to develop such tools[1] in an efficient way, modeling languages (i.e., their *syntax* and *semantics*) have to be defined formally [1].

The *abstract syntax* of a modeling language defines the modeling language's concepts and their relations. For formally defining the abstract syntax, metamodels are the standard means. MOF [13] constitutes a standardized, well established, and widely accepted metamodeling language for this purpose. Furthermore, MOF 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.

The *behavioral semantics* of a modeling language defines the execution behavior of conforming models. A formal specification of the behavioral semantics is not only needed for precisely and unambiguously defining the behavior of models, but also to

[1] In the remainder of this paper we will refer to tools for MDE, which are based on the definition of a modeling language, as *model-based tools*.

establish the basis for an efficient development of model-based tools which build upon the behavioral semantics of a modeling language, such as model interpreters, debuggers, and testing environments [1]. Having the behavioral semantics of a modeling language explicitly and formally defined enables to automate the development of such tools. Unfortunately, no standard way for defining a modeling language's behavioral semantics explicitly and formally has been established yet. Hence, formalizing the behavioral semantics of modeling languages remains a core challenge in MDE [1]. To address this challenge we propose to use the standardized and UML 2 compliant action language foundational UML (fUML) [15] as semantics specification language in MDE.

2 Background and Related Work

The behavioral semantics of many modeling languages, including widely adopted languages such as UML [14], are only informally defined in *natural language*. This may lead to several problems, such as to ambiguities in the semantic meaning of the language's modeling concepts. Furthermore, the behavior of models conforming to the modeling language cannot be formally analyzed and the models cannot be executed. Moreover, building tools, such as model debuggers and interpreters, is problematic if the language's semantics is only defined in natural language.

A commonly used approach to make models executable is to utilize *code generators* or to develop *model interpreters* with general-purpose programming languages (GPLs). However, in this approach the behavioral semantics is encoded in the manually constructed code generation templates or model interpreter implementations and hence is only defined implicitly. A problem that arises is that the behavior of models can again not be analyzed easily. Furthermore, the semantics can hardly be extended and reused which makes it costly to create and maintain.

To overcome these limitations, adequate techniques for formally and explicitly specifying the behavioral semantics of modeling languages are needed. This need stimulated intensive research resulting in various approaches proposed in the past which can be divided into *translational approaches* and *operational approaches*.

In *translational approaches*, models conforming to a modeling language are translated into models conforming to another language whose semantics is formally and explicitly defined. The advantage of this approach is that tools available for the target language can be used for the translated models. Its drawback, however, is that a translation from the source into the target language as well as a mapping of results obtained for the translated models to the original models have to be developed. Examples for translational approaches are the work of Chen *et al.* [2] and Di Ruscio *et al.* [4] using the Abstract State Machine formalism as target language, Rivera *et al.* [16] using Maude, Kühne *et al.* [9] using Petri nets, and Rumpe *et al.* [7] using their System Model.

In the *operational approach*, the behavioral semantics is directly introduced into the modeling language without moving to another language. This can be done by utilizing *graph transformations* as for instance proposed by Engels *et al.* [6]. Another way is to introduce operations into the metaclasses of a modeling language's metamodel and provide their implementations using a dedicated *action language*. Several action languages and GPLs have been proposed for this purpose, such as Kermeta [12], Smalltalk [5], xCore [3], and EOL [8].

3 Approach and Uniqueness

Despite the fact that several approaches for formally and explicitly specifying the behavioral semantics of modeling languages exist, none of these approaches is widely adopted especially in industry. Moreover, the challenge of automatically generating model-based tools from semantics specifications is only addressed by few approaches which provide only partial solutions [1]. We advocate for *operational semantics* specification approaches because instead of translating models into a different language, the behavioral semantics is directly attached to the modeling language. We also believe that a *standardized* action language should be employed for this purpose. The action languages proposed until now are either GPLs (e.g., Smalltalk), proprietary languages (e.g., Kermeta), or adapted versions of standardized languages (e.g., xOCL integrated in xCore). In contrast, we propose to use fUML, a standardized and UML 2 compliant action language, for specifying the behavioral semantics of modeling languages. fUML defines the semantics of a subset of UML consisting of concepts for modeling classes and activities formally and provides a virtual machine (VM) for executing compliant models. As both MOF and fUML are standardized by OMG, fUML may be considered as promising candidate for becoming a standardized action language in metamodeling.

To use fUML as semantics specification language and leverage the semantics specification of a modeling language to execute conforming models, the following challenges had to be addressed. First, fUML had to be *integrated with existing metamodeling languages*. Second, state-of-the-art metamodeling methodologies and environments had to be adequately extended to *support a systematic and efficient development of behavioral semantics specifications* based on fUML. Third, a *generic model interpreter* had to be developed to enable the execution of models based on the behavioral semantics specification of the modeling language they conform to.

To enable the usage of fUML for specifying the behavioral semantics of modeling languages, we identified two strategies: a transformation-based and an integration-based strategy [11]. Because of the better integration with existing metamodeling environments, we decided to apply the integration-based strategy. In this approach, a metamodeling language is extended with the behavioral part of fUML in a way that enables to specify the behavior of the operations introduced for the metaclasses of a modeling language using fUML activities. By applying this approach we extended Ecore, which is integrated with the Eclipse Modeling Framework (EMF) [18] and which constitutes the most prominent implementation of MOF. Thereby we obtained a new metamodeling language which we called executable MOF (xMOF) that enables to specify a modeling language's abstract syntax using the modeling concepts stemming from Ecore and its behavioral semantics using the modeling concepts stemming from fUML.

Based on xMOF we elaborated a tool-supported methodology for developing the behavioral semantics of modeling languages (cf. Figure 1). The input for the semantics specification is the Ecore-based metamodel of a modeling language. As preparatory step we automatically generate a subclass for each metaclass defined in the metamodel. The language designer then adds operations and their implementations in the form of activities to these subclasses, which specify the behavior of the metaclasses. By subclassing the metaclasses for specifying their behavioral semantics, the behavioral semantics specification is clearly separated from the abstract syntax specification. This

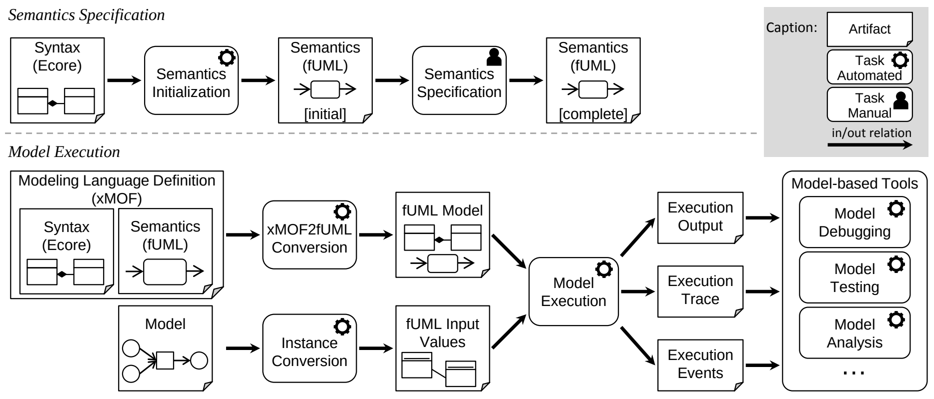

Fig. 1: Overview of our semantics specification approach based on fUML.

fosters reusability and flexibility in the semantics specification and facilitates the seamless integration of the development of a modeling language's behavioral semantics into existing metamodeling methodologies and environments which are by our methodology only extended concerning the semantics specification.

Having the behavioral semantics of a modeling language defined using xMOF, our approach enables to execute conforming models based on this semantics specification by leveraging the fUML VM[2]. For this, we automatically convert the modeling language definition into an fUML compliant model. In this conversion, metaclasses are converted into fUML classes and the implementations of the metaclasses' operations (in the form of activities) are converted into fUML activities. Furthermore, the elements of the model to be executed are converted into corresponding fUML objects, i.e., instances of the fUML classes generated for the metaclasses of the model elements, and are provided as input to the fUML VM. The fUML VM interprets the activities and manipulates the objects, which represent the executed model, accordingly. The primary output of the model execution is the set of manipulated objects which represents the runtime state of the executed model after the model execution finished and is provided to the user in the form of model annotations. For enabling the observation and analysis of a model execution being carried out we extended the fUML VM in previous work [10] with an event mechanism and a trace model. These outputs of the model execution can be utilized by model-based tools building upon the modeling language's behavioral semantics, such as model debuggers, testing environments, and analyzers.

4 Results and Contributions

Formalizing the behavioral semantics of modeling languages and leveraging this formalization for automatically generating model-based tools building upon the modeling languages' semantics is an open challenge in MDE [1]. Our previous research aiming at addressing this challenge resulted in the following contributions[3]. *(i)* A strategy was

[2] Our implementation is based on the reference implementation of the fUML VM provided at `http://fuml.modeldriven.org`

[3] 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`.

proposed to integrate the standardized action language fUML with existing metamodeling languages enabling the specification of the behavioral semantics of modeling languages in an operational way. This strategy was applied to integrate fUML with Ecore, which is the most prominent implementation of MOF, resulting in the metamodeling language xMOF. *(ii)* Based on xMOF, a tool-supported methodology was elaborated for developing behavioral semantics specifications of modeling languages which integrates seamlessly with existing metamodeling methodologies and environments. *(iii)* A generic model interpreter was developed which enables the execution of models based on the modeling language's semantics specification by leveraging the fUML VM.

To qualitatively evaluate the applicability of our semantics specification approach based on fUML, we carried out case studies in which we developed the behavioral semantics specifications of distinct modeling languages using xMOF. In summary, the case studies confirmed that the proposed metamodeling language xMOF as well as its accompanying tool-supported methodology are applicable for defining the behavioral semantics of different kinds of modeling languages. Moreover, by utilizing the generic model interpreter based on the fUML VM it was possible to execute models conforming to the modeling languages by interpreting their semantics specifications. Regarding the suitability of fUML as action language for metamodeling we come to the conclusion that due to its object-oriented and imperative nature, fUML is highly suitable as semantics specification language. The case studies also revealed possible improvements of our approach regarding the reusability of semantics specifications and tooling. Furthermore, we recognized that the runtime information about the execution of a model provided by our generic model interpreter in the form of execution events and an execution trace helps in debugging and analyzing the xMOF-based semantics specification as they provide detailed information about the executed fUML activities contained in the semantics specification. However, to also provide this detailed runtime information to the modeler, the execution events and trace should be tailored to the respective modeling language as the modeler is only concerned with the concepts of the modeling language and not with its semantics specification in terms of fUML activities.

5 Conclusion and Future Work

Our research is concerned with formally specifying the behavioral semantics of modeling languages using fUML and utilizing the semantics specifications for efficiently developing model-based tools building upon the behavioral semantics of modeling languages. Our implementation of a generic model interpreter capable of executing models based on the semantics specification of the used modeling language defined with fUML is only the first step towards addressing the challenge of generating model-based tools from a modeling language's semantics specification. We will, in future work, further investigate methods for generating model-based tools from fUML-based semantics specifications. In particular, we are currently investigating the semi-automatic generation of model debuggers. Furthermore, we will carry out a large-scale case study applying our proposed semantics specification approach to fUML itself. This case study serves two purposes; first, we will evaluate the applicability and scalability of our semantics specification approach and second, having defined the behavioral semantics of fUML in fUML enables to bootstrap fUML's model execution capabilities for instance to support a larger subset of UML without extending the fUML VM itself.

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. K. Chen, J. Sztipanovits, S. Abdelwalhed, and E. Jackson. Semantic Anchoring with Model Transformations. In *Proceedings of the 1st European Conference on Model Driven Architecture - Foundations and Applications (ECMDA-FA)*, pages 115–129. Springer, 2005.
3. T. Clark, A. Evans, P. Sammut, and J. Willans. *Applied Metamodelling: A Foundation for Language Driven Development*. Ceteva, Sheffield, 2004.
4. D. Di Ruscio, F. Jouault, I. Kurtev, J. Bézivin, and A. Pierantonio. Extending AMMA for Supporting Dynamic Semantics Specifications of DSLs. Technical Report, INRIA/LINA, Università degli Studi di L'Aquila, 2006.
5. S. Ducasse and T. Gîrba. Using Smalltalk as a Reflective Executable Meta-language. In *Proceedings of the 9th International Conference on Model Driven Engineering Languages and Systems (MoDELS)*, pages 604–618. Springer, 2006.
6. G. Engels, J. H. Hausmann, R. Heckel, and S. Sauer. Dynamic Meta Modeling: A Graphical Approach to the Operational Semantics of Behavioral Diagrams in UML. In *Proceedings of the 3rd International Conference on the Unified Modeling Language (UML)*, pages 323–337. Springer, 2000.
7. H. Grönniger, J. O. Ringert, and B. Rumpe. System Model-Based Definition of Modeling Language Semantics. In *Proceedings of the Joint 11th IFIP International Conference FMOODS and 29th IFIP International Conference FORTE (FMOODS/FORTE)*, pages 152–166. Springer, 2009.
8. D. S. Kolovos, R. F. Paige, and F. Polack. The Epsilon Object Language (EOL). In *Proceedings of the 2nd European Conference on Model Driven Architecture - Foundations and Applications (ECMDA-FA)*, pages 128–142. Springer, 2006.
9. T. Kühne, G. Mezei, E. Syriani, H. Vangheluwe, and M. Wimmer. Explicit Transformation Modeling. In *Models in Software Engineering: Reports and Revised Selected Papers of Workshops and Symposia at MoDELS 2009*, pages 240–255. Springer, 2009.
10. T. Mayerhofer, P. Langer, and G. Kappel. A Runtime Model for fUML. In *Proceedings of the 7th Workshop on Models@run.time (MRT) @ MoDELS'12*, pages 53–58. ACM, 2012.
11. T. Mayerhofer, P. Langer, and M. Wimmer. Towards xMOF: Executable DSMLs based on fUML. In *Proceedings of the 12th Workshop on Domain-Specific Modeling (DSM) @ SPLASH'12*, pages 1–6. ACM, 2012.
12. P.-A. Muller, F. Fleurey, and J.-M. Jézéquel. Weaving Executability into Object-Oriented Meta-languages. In *Proceedings of the 8th International Conference on Model Driven Engineering Languages and Systems (MoDELS)*, pages 264–278. Springer, 2005.
13. Object Management Group. OMG Meta Object Facility (MOF) Core Specification, Version 2.4.1, August 2011. `http://www.omg.org/spec/MOF/2.4.1`.
14. Object Management Group. OMG Unified Modeling Language (OMG UML), Superstructure, Version 2.4.1, August 2011. `http://www.omg.org/spec/UML/2.4.1`.
15. Object Management Group. Semantics of a Foundational Subset for Executable UML Models (fUML), Version 1.0, February 2011. `http://www.omg.org/spec/FUML/1.0`.
16. J. E. Rivera, F. Durán, and A. Vallecillo. On the Behavioral Semantics of Real-Time Domain Specific Visual Languages. In *Proceedings of the 8th International Workshop on Rewriting Logic and Its Applications (WRLA) @ ETAPS'10*, pages 174–190. Springer, 2010.
17. D. C. Schmidt. Guest Editor's Introduction: Model-Driven Engineering. *IEEE Computer*, 39(2):25–31, 2006.
18. D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks. *EMF: Eclipse Modeling Framework*. Addison-Wesley Professional, 2nd edition, 2008.