A Multi-Paradigm Modelling Approach for the Engineering of Modelling Languages

Bart Meyers[1]

Modeling, Simulation and Design Lab (MSDL), University of Antwerp, Belgium
bart.meyers@uantwerp.be

1 Problem

The main problem statement in this Ph.D. is in the domain of modelling language engineering:

How can we engineer domain-specific modelling languages (DSMLs) in a more structured way?

We look at this problem from a tool-builder's point of view. This means that our solution is a tool that supports language engineers in modelling DSMLs in a more structured way.

This problem is a very broad one. We have chosen some sub-topics for which the research questions can be described as:

In the case of evolution of a DSML, how can we provide support for (semi-)automatic co-evolution of related artefacts such as instances and transformations? Since the creation of the DSML is part of the software development cycle, it is also subject to evolution. We investigate how we can automate co-evolution on models that are instances of the language, and transformations defined for this language. Automatically co-evolving the latter turns out to be very hard, as the language definition does not contain a formal description of the semantic intent of the language, making it hard to make assumptions on co-evolving related transformations. To simplify the co-evolution problem, we aim at structurally splitting up languages, leading to the next research question.

How can we reuse existing languages or language fragments to compose a new DSML? DSMLs are usually built from scratch, although they may share certain characteristics with existing languages (e.g., it is a state-based formalism, or the language includes simple mathematical expressions in a textual way, etc.). We aim at re-using a DSML in all of its aspects; abstract syntax, concrete syntax and semantics. For abstract and concrete syntax, this turns out to be a feasible research question, but semantics description of a language can be quite extensive. Therefore we choose to investigate the composition of one, yet common, way of describing semantics, leading to the next research question.

How can instances of two different DSMLs run together in a modular way, in the context of simulation? Here, we go a step further into combining operational semantics of DSMLs, in the form of models of computation. This can be considered the most difficult part of the composition of DSMLs, as not only data (e.g., events or signals), but also control (when, during execution, a step needs to be executed for a model of computation) and time (how time scales of models of different computation correspond) need to be adapted to obtain a compositional execution. Additionally, we claim that

verifying the correctness of a composition of two DSMLs can be achieved by describing and verifying properties, leading to the next research question.

How can we provide automatic support for a DSML so that it does not only support the design of systems, but also the precise modelling of properties, and the verification of these properties? DSMLs are generally focused at *designing* systems at a domain-specific level. When requirements need to be verified for this DSL, modellers usually have to resort to general purpose languages such as CTL and LTL. This is in contradiction with the spirit of domain-specific modelling (DSM), as domain experts are expected to express properties in an unfamiliar, mathematical way. We aim at defining a suite of languages we call *ProMoBox* for not only defining the design of a system, but also properties, state, and input and output of a system.

In this paper, we focus on the last two problems because of space constraints and because these topics are currently researched, which will be referred to as *semantic adaptation* and *ProMoBox*. The first problem separates itself from the other three, and was the first major research topic of my Ph.D., which resulted in a number of publications [1–6]. The second problem also resulted in some publications [7, 8], and is now further under research by a master student I am supervising.

2 Related work

For the subject of **semantic adaptation** we chose to focus our study of the state of the art on three different tools for heterogeneous modelling and simulation: Ptolemy II [9], Simulink/Stateflow[1], and ModHel'X [10]. All of them support the joint use of different modelling paradigms in a model and they all use hierarchy as a mechanism for composing the heterogeneous parts of a model. Other types of approaches are described in [11].

In Ptolemy II semantic adaptation of time, control and data can be modelled within one of the models that are composed, but this means that at least one of the models needs to be changed in order to work with the other model. Consequently, modularity is lost. Alternatively, default semantic adaptation can be used, leading to undesired behaviour, as one cannot define a single way to combine two formalisms semantically. The modeller could change this default behaviour by delving into Ptolemy II's code, but this is in contrast to the domain expert-friendly visual environments of Ptolemy II to model in, and the very ambition of DSM.

In Simulink/Stateflow, which is a powerful and efficient simulation tool for heterogeneous models, semantic adaptation is even more diffuse. The global semantics of a heterogeneous model (for instance a Simulink model including a Stateflow submodel) is given by one solver which is used to compute its execution.

ModHel'X is a framework for modelling and executing heterogeneous models [10]. ModHel'X tries to resolve these issues by allowing modular semantic adaptation by explicitly defining an *interface block* between the parent model and the embedded model. This block adapts the data, control and time between the different models [12]. Semantic adaptation is specified using calls to a Java API in different methods of an interface block. Therefore, constructing an interface block is a tedious and error prone process.

[1] `http://www.mathworks.com/products/simulink/`

In conclusion we can state that the adaptation of data, control and time is either scattered over both models, or is hard-coded. No approaches exist that allow a domain-specific syntax at the right level of abstraction like the models, while at the same time respecting the modularity of the composition.

With respect to the contribution of **ProMoBox**, we distinguish two threads of related work. First, we consider approaches that translate models to formal representations to specify and verify properties that are created specifically for one modelling language. Second, we discuss approaches that have a more general view on providing specification and verification support for different modelling languages.

A plethora of language-specific approaches have been presented to define properties and verification results for different kind of design-oriented languages. For instance, Cimatti et al. [13] have proposed to verify component-based systems by using scenarios specified as Message Sequence Charts (MSCs). Li et al. [14] also apply MSCs for specifying scenarios for verifying concurrent systems. The CHARMY approach [15] offers amongst other features, verification support for architectural models described in UML. Collaboration and sequence diagrams have been applied to check the behaviour of systems described in terms of state machines [16–18]. Rivera et al. [19] map the operational semantics of DSMLs to Maude, and thus, benefit from analysing methods provided out-of-the-box of Maude environments such as checking of temporal properties specified in LTL. These mentioned approaches are just a few examples that aim at specifying temporal properties for models and verifying them by model checkers (see [20] for a survey). They have in common that they offer language-specific property languages or that LTL properties have to be defined directly on the formal representation. Thus, these approaches are not aiming to support DSMLs engineers in the task of building domain-specific property languages.

There are some approaches that aim to shift the specification and verification tasks to the model level in a more generalised manner. First of all, there are approaches that propose OCL extensions, often referred to Temporal OCL (TOCL), for defining temporal properties on models [21–23]. As OCL may be combined with any modelling language, TOCL can be seen as a generic model-based property language as well. In [24, 25] the authors discuss and apply a pattern to extend modelling languages with events, traces, and further runtime concepts to represent the state of a model's execution and to use TOCL for defining properties that are verified by mapping the design models as well as the properties expressed in TOCL to formal domains that provide verification support. In addition, not only the input for model checkers is automatically produced, but also the output, *i.e.,* the verification results, is translated back to the model level. The authors explain the choice of using TOCL to be able to express properties at the domain level, because TOCL is close to OCL and should be therefore familiar to domain engineers. However, they also state that early feedback of applying their approach has shown that TOCL is still not well suited to many domain engineers and they state in future work that more tailored languages may be of help for the domain engineers. In this Ph.D. I directly go in this direction by enabling domain engineers to use their familiar notation for defining properties and exploring the verification results.

Another approach that aims to define properties on the model level in a generic way is presented in [26]. The authors extend a language for defining structural patterns based

on Story Diagrams [27] to allow for modelling temporal patterns as well. The resulting language allows to define conditionally timed scenarios stating the partial order of structural patterns. They authors argue that their language is more accessible for domain engineers, because their language allow decomposition of complex temporal properties into smaller ones by if-then-else decomposition and quantification over free variables. Their approach is tailored to engineers that are familiar to work with UML class diagrams and UML object diagrams as their notation is heavily based on the concepts of these two languages. Furthermore, they explain how the specification patterns of Dwyer et al. [28] are encoded in their language, but there is no language-inherent support to explicitly apply them. In our work, we tackle these two issues in the context of DSM by reusing the notation of domain engineers for specifying properties and providing explicit language support for specification patterns.

3 Proposed solution

The solution we aim at in this research takes the form of a tool. We will use our in-house tool AToMPM [29], a tool for meta-modelling and model transformation. We consider these as the traditional methods of language engineering we will build on.

For the subject of **semantic adaptation**, we will design a DSML for modelling the interaction or adaptation between simulators. This includes a transformation to an execution platform and back, such as ModHel'X [10]. We focus on semantic adaptation of models of computation, meaning that we consider adaptation at run time. As described in [10] we adapt data, control and time at runtime, with a hierarchy in the two models: there is always one "outer" model and one "inner" model. Our goal is to first describe this adaptation in detail for one case, *i.e.,* the Discrete Events model of computation to the Synchronous Data Flow model of computation. Different variants of the adaptation can be devised, and we can distil an adaptation DSML that allows describing these variants in their essence. A next step is to generalise this adaptation DSML by applying it to more models of computation. As a consequence, more and more features will be added to the DSML. Not only a usable DSML will be the result, but also a clear classification of the adaptation possibilities.

For the subject of **ProMoBox**, we will design a collection of transformations and model templates in ATOMPM, and transformation to an execution platform and back, such as SPIN [30]. We will mainly focus on behavioural DSMLs and temporal properties, as we can consider these to entail structural properties of one state of a system. The ProMoBox framework consists of the following three parts:

Generic languages for modelling all artefacts that are needed for specifying and verifying properties. For a given DSML, *ProMoBox* defines a family of five sub-languages that are required to modularly support property verification, covering (i) design modelling as supported by traditional DSMLs, (ii) run-time state representation, (iii) event-based input modelling (to model the behaviour of an environment), (iv) state-based output representation (to model an execution trace of the system or verification results), and (v) property specification. Property languages generated by *ProMoBox* are specifically tailored to ease the development of temporal patterns as well as structural patterns needed to describe the desired properties of the system's design by domain engineers in the DSML's concrete syntax. To allow to formulate temporal properties at a high-level of abstraction, we formalise Dwyer's specification patterns [28] for defining temporal

patterns as a DSML. With the help of this DSML, domain engineers are able to express temporal properties for finite state verification such as absence, existence, or universality. To ease the development of structural patterns to be checked on snapshots of the system's execution states, we propose an automated technique based on [31, 32] that is able to produce a specialised language from a given DSML tailored to express structural patterns. The language for defining structural patterns is inspired by *PaMoMo* [33, 34], a language supporting several pattern kinds such as enabling, positive, and negative patterns. Finally, we introduce the possibility to define quantifiers for temporal properties to express complex properties in a more concise manner, *e.g., every* element of a certain type has to fulfil a certain property.

A *fully automated method* to specialise and integrate these generic languages to a given DSML. We extend meta-modelling and model transformation languages with annotations, to add necessary information for every language construct and semantic step. This additional information enables the fully automatic generation of the five sub-languages and necessary transformations between the sub-languages, thus minimising the effort of the language engineer. Because of their generative definition, consistency between the languages and their models is guaranteed by construction. We use templates that describe the generic part of each language, and that are subsequently woven with the DSML. By using templates, we allow the ProMoBox framework to be configurable for different types of DSMLs.

A *verification backbone* based model checking directly plug-able to DSM environments. Properties in ProMoBox are translated to LTL and a Promela system is generated that includes a translation of the initialised system, the environment, and the rule-based operational semantics of the system. The properties are checked by SPIN [30]. The verification results (in case of a counter-example) are translated back to the DSM level.

For both research tracks, a formal, theoretical framework will be elaborated, and a prototype is created as an extension of our tool AToMPM. Using the prototype, the approach is tested and a comparison is made with existing approaches. This whole research process is repeated many times, whenever our vision is changed because of e.g., feedback, new literature or unsatisfactory results. When the results are satisfactory, they should be published in a journal. During this project, we aim for close collaboration with the research community to allow cross-fertilisation of our research. In this context, I have been invited by Prof. Juan de Lara for a research visit to the Universidad Autónoma de Madrid (Spain), by Prof. Frédéric Boulanger at Supélec, by Dr. Manuel Wimmer at the Technische Universität Wien (Austria), and by Dr. Alexandre Petrenko at the Computer Research Institute of Montréal (Canada). Other collaborations are evidenced by publications with several people from the research community.

4 Preliminary work

I have six year research experience in the field of modelling language engineering. In 2008-2009, I worked in the context of my master's thesis on transformation languages, which resulted in two international peer-reviewed workshop papers [35, 36] and a thesis [37]. In 2009-2010, I worked on the problem of evolution of modelling languages. During my research, I attended top conferences and workshops on model-driven engineering (i.e., MoDELS, ASE, ICMT), and fruitfully worked together with international researchers to write several internationally refereed papers [1–6].d

We investigated the composition of DSMLs for textual models using metaDepth [38] in [8], where we proposed techniques for the modular definition and composition of languages, including their abstract, concrete syntax and semantics. These techniques are based on (meta-)model templates rather than aspects, where interface elements and requirements for their connection can be established. As a side-track we did research on how to add aspects to Petri nets [7], where we looked into adding a "aspect-oriented language module" to an existing language, in this case Petri nets.

The research for the tracks semantic adaptation and ProMoBox were outlined in the section "Proposed solution". We will briefly go over what has been done. For the subject of semantic adaptation we investigated how we can describe the semantic adaptation using a textual DSML from the Discrete Events model of computation to the Synchronous Data Flow model of computation as described in [39]. For the subject of ProMoBox, we showed in a first step how we can generate a properties language for Statecharts [40]. In a next step, we broaden our scope to any event-based formalism, for which we want to check temporal properties with the expression power of LTL. We are finishing this chapter, and we are in the process of writing a technical report and a paper we will submit to the SLE conference.

5 Expected contributions

The contributions will be formal models, validated by a prototype implementation (see also "proposed solution"). We aim for publication in SoSyM (Software and Systems Modeling, published by Springer), JVLC (Journal of Visual Languages and Computing, published by Elsevier), SCP (Science of Computer Programming, published by Elsevier), TSE (IEEE Transactions on Software Engineering), MoDELS (ACM/IEEE International Conference on Model Driven Engineering Languages and Systems), SLE (International Conference on Software Language Engineering), etc.

6 Plan for evaluation and validation

The evaluation and validation of this research will be done mainly in the form of tool prototypes (see "proposed solution"). We will touch on some larger case studies, but we will refrain from industry case studies or empirical studies, as our focus is on defining and implementing a formal model. To conclude, we aim at preparing a prototype that could be used for industry case studies or empirical studies or to re-implement on different platforms.

7 Current status

A large part of the work already has been done (see "Preliminary work"), yet we plan to finish the two main research tracks that are presented here.

The cooperation with Supélec is a good basis to work on semantic adaptation. We intend to improve the DSML for semantic adaptation that support more types of DSMLs. Although we cannot be complete and support every existing DSML, we want to be able to support "core" DSMLs that together form a broad range of all behavioural modelling languages (Discrete Event, Synchronous Data Flow, Petri nets, and Timed Finite State Machines). We intend to publish our final results in a journal like Software and Systems Modeling.

We are currently finishing a ProMoBox paper for SLE, and in a next step we want to broaden our scope to different types of DSMLs (as we want to do for semantic adaptation), and different types of properties, with consequently different types of execution platforms. We also intend to showcase the approach using a realistic case study with the "GISMO" DSL [41]. Because of the computational limitations of model checking, we wish to investigate the employment of test case generation techniques with Dr. Alexandre Petrenko, an expert in the domain. As we believe that this is a relevant research track and that we have nice results, we intend to publish the results in a journal like Transactions on Software Engineering.

To finish both research tracks, writing two more journal papers as well as writing the thesis, I have time until January 2016. We plan to finish the research on ProMoBox first, as we are currently working together productively with Romuald Deshayes (Université de Mons, "GISMO" case study), Dr. Manuel Wimmer (Technische Universität Wien), Prof. Eugene Syriani (Université de Montréal), Dr. Levi Lucio (McGill University), and in the near future Dr. Alexandre Petrenko (Computer Research Institute of Montréal). I believe that such international co-operations are key to the success of a Ph.D.

References

1. Meyers, B., Vangheluwe, H.: Evolution of modelling languages. In: 8th BElgian-NEtherlands software eVOLution seminar (BENEVOL), UCL (2009)
2. Meyers, B., Vangheluwe, H.: Evolution of modelling languages. In: 7th international Fujaba days, Eindhoven, The Netherlands, Technische Universiteit Eindhoven (2009)
3. Meyers, B., Ebraert, P., Janssens, D.: Intensional changes avoid co-evolution! In: 7th ECOOP'2010 Workshop on Reflection, AOP and Meta-Data for Software Evolution. (2010)
4. Meyers, B., Vangheluwe, H.: A framework for evolution of modelling languages. Science of Computer Programming **76** (2011) 1223 – 1246
5. Wimmer, M., Cicchetti, A., Meyers, B.: Abstract and concrete syntax migration of instance models. In: ICMT 2010 Transformation Tool Contest. (2010)
6. Meyers, B., Wimmer, M., Cicchetti, A., Sprinkle, J.: A generic in-place transformation-based approach to structured model co-evolution. ECEASST (2011)
7. Molderez, T., Meyers, B., Janssens, D., Vangheluwe, H.: Towards an aspect-oriented language module: Aspects for petri nets. In: DSAL '12, New York, USA, ACM (2012) 21–26
8. Meyers, B., Cicchetti, A., Guerra, E., de Lara, J.: Composing textual modelling languages in practice. In: MPM'12, Innsbruck, Austria (2012)
9. Eker, J., Janneck, J., Lee, E., Liu, J., Liu, X., Ludvig, J., Neuendorffer, S., Sachs, S., Xiong, Y.: Taming Heterogeneity - The Ptolemy Approach. Proc. of the IEEE **91** (2003) 127–144
10. Boulanger, F., Hardebolle, C.: Simulation of Multi-Formalism Models with ModHel'X. In: ICST 2008. (2008) 318–327
11. Hardebolle, C., Boulanger, F.: Exploring multi-paradigm modeling techniques. SIMULATION **85** (2009) 688–708
12. Boulanger, F., Hardebolle, C., Jacquet, C., Marcadet, D.: Semantic Adaptation for Models of Computation. In: ACSD'11. (2011) 153–162
13. Cimatti, A., Mover, S., Tonetta, S.: Proving and Explaining the Unfeasibility of Message Sequence Charts for Hybrid Systems. In: FMCAD. (2011) 54–52
14. Li, X., Hu, J., Bu, L., Zhao, J., Zheng, G.: Consistency Checking of Concurrent Models for Scenario-Based Specifications. In: SDL. (2005) 1171–1180
15. Pelliccione, P., Inverardi, P., Muccini, H.: CHARMY: A Framework for Designing and Verifying Architectural Specifications. TSE **35** (2008) 325–346

16. Brosch, P., Egly, U., Gabmeyer, S., Kappel, G., Seidl, M., Tompits, H., Widl, M., Wimmer, M.: Towards Scenario-Based Testing of UML Diagrams. In: TAP. (2012) 149–155
17. Knapp, A., Wuttke, J.: Model checking of UML 2.0 interactions. In: MoDELS'06. (2006) 42–51
18. Schäfer, T., Knapp, A., Merz, S.: Model Checking UML State Machines and Collaborations. ENTCS **55** (2001) 357–369
19. Rivera, J.E., Guerra, E., de Lara, J., Vallecillo, A.: Analyzing Rule-Based Behavioral Semantics of Visual Modeling Languages with Maude. In: SLE. (2008) 54–73
20. Gabmeyer, S., Kaufmann, P., Seidl, M.: A classification of model checking-based verification approaches for software models. In: VOLT. (2013)
21. Ziemann, P., Gogolla, M.: OCL Extended with Temporal Logic. In: PSI. (2003) 351–357
22. Kanso, B., Taha, S.: Temporal Constraint Support for OCL. In: SLE 2012. Volume 7745 of Lecture Notes in Computer Science., Springer (2012) 83–103
23. Bill, R., Gabmeyer, S., Kaufmann, P., Seidl, M.: OCL meets CTL: Towards CTL-Extended OCL Model Checking. In: OCL Workshop. Volume Vol-1092. (2013) 13–22
24. Zalila, F., Crégut, X., Pantel, M.: Leveraging Formal Verification Tools for DSML Users: A Process Modeling Case Study. In: ISoLA. (2012) 329–343
25. Combemale, B., Crégut, X., Pantel, M.: A Design Pattern to Build Executable DSMLs and Associated V&V Tools. In: APSEC. (2012) 282–287
26. Klein, F., Giese, H.: Joint structural and temporal property specification using timed story scenario diagrams. In: FASE. (2007) 185–199
27. Fischer, T., Niere, J., Torunski, L., Zündorf, A.: Story Diagrams: A New Graph Rewrite Language Based on the Unified Modeling Language and Java. In: TAGT. (2000) 296–309
28. Dwyer, M.B., Avrunin, G.S., Corbett, J.C.: Patterns in Property Specifications for Finite-State Verification. In: ICSE. (1999) 411–420
29. Syriani, E., Vangheluwe, H., Mannadiar, R., Hansen, C., Mierlo, S.V., Ergin, H.: AToMPM: A Web-based Modeling Environment. In: MoDELS Demonstrations. (2013)
30. Holzmann, G.J.: The Model Checker SPIN. TSE **23** (1997) 279–295
31. Kühne, T., Mezei, G., Syriani, E., Vangheluwe, H., Wimmer, M.: Explicit transformation modeling. In: MoDELS Workshops. (2009) 240–255
32. Syriani, E.: A Multi-Paradigm Foundation for Model Transformation Language Engineering. PhD thesis, McGill University Montreal, Canada (2011)
33. Guerra, E., de Lara, J., Kolovos, D.S., Paige, R.F.: A Visual Specification Language for Model-to-Model Transformations. In: VL/HCC. (2010) 119–126
34. Guerra, E., de Lara, J., Wimmer, M., et al.: Automated verification of model transformations based on visual contracts. ASE **20** (2013) 5–46
35. Meyers, B., Van Gorp, P.: Towards a hybrid transformation language: Implicit and explicit rule scheduling in story diagrams. In: Sixth International Fujaba Days, Germany (2008)
36. Muliawan, O., Meyers, B., Janssens, D.: BPMN2BPEL in MoTMoT. In: 5th International Workshop on Graph-Based Tools, Zürich (Switzerland) (2009)
37. Meyers, B.: Hybrid rule scheduling in story driven modeling - a tool-independent approach. Master's thesis, University of Antwerp, University of Antwerp (2009)
38. de Lara, J., Guerra, E.: Deep meta-modelling with metadepth. In: TOOLS (48). Volume 6141 of LNCS. Springer Berlin Heidelberg (2010) 1–20
39. Meyers, B., Denil, J., Boulanger, F., Hardebolle, C., Jacquet, C., Vangheluwe, H.: A dsl for explicit semantic adaptation. In: MPM'13, MODELS 2013 (2013) 47–56
40. Meyers, B., Wimmer, M., Vangheluwe, H., Denil, J.: Towards domain-specific property languages: The promobox approach. In: DSM '13. (2013) 39–44
41. Deshayes, R.: A domain-specific modeling approach for gestural interaction. In Kelleher, C., Burnett, M.M., Sauer, S., eds.: VL/HCC, IEEE (2013) 181–182