Electronic Communications of the EASST
Volume 50 (2011)

Recent Advances in Multi-Paradigm Modeling (MPM 2011)

Preface

Vasco Amaral , Cécile Hardebolle, Hans Vangheluwe , László Lengyel, Peter Bunus

10 pages

Guest Editors: Vasco Amaral, Cécile Hardebolle, Hans Vangheluwe, László Lengyel, Peter Bunus
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Preface

Vasco Amaral [1], Cécile Hardebolle[2], Hans Vangheluwe [34], László Lengyel[5], Peter Bunus[6]

[1] CITI-FCT, Universidade Nova de Lisboa, Portugal,vasco.amaral@fct.unl.pt
[2]Supélec, France, cecile.hardebolle@supelec.fr
[3]University of Antwerp, Belgium, Hans.Vangheluwe@ua.ac.be
[4]McGill University, Canada, hv@cs.mcgill.ca
[5]Budapest University of Technology and Economy, Budapest, Lengyel.Laszlo@aut.bme.hu
[6]Linkoping University, Sweden, peter.bunus@liu.se

Abstract: Following the trend of other editions, it was held this year (2011) in Wellington, New Zealand, the 5th International Workshop on Multi-Paradigm Modelling: Concepts and Tools (MPM). Once again has been a satellite event of the International Conference on Model-Driven Engineering Languages and Systems (MoDELS). It aims at further the state-of-the-art as well as to define future directions of this emerging research area by bringing together world experts in the field for an intense one-day workshop. This paper summarizes the results of this year's event.

Keywords: Model-Driven Development, Multi-Paradigm Modeling, Multiple Paradigms, Multi Formalisms, Model Composition, Model Transformation, Software Languages Usability Evaluation

1 Introduction

Multi-Paradigm Modelling (MPM) is a research field focused on solving the challenge of combining, coupling, and integrating rigorous models of some reality, at different levels of abstraction and views, using adequate modelling formalisms and semantic domains, with the goal to simulate (for optimization) or realize systems that may be physical, software or a combination of both. This field promotes that modeling by means of different modeling formalisms, with different perspectives of the system, it can be avoided the tendency to over-design, the modeler works with more manageable models and the system's integration is better supported. The identified key challenges are on finding adequate Model Abstractions, Multi-formalism modelling, Model Transformation and the application of MPM techniques and tools to Complex Systems.

MPM is a series of annual events [1] that has experienced a steady growth in participation, having this year, stabilized around the figure of 25 participants. After a review process that counted with 4 reviewers per paper, 6 high quality contributions out of 13 were selected for oral communications (grouped in two sessions), and 3 papers were accepted for a poster session. This time, a session was dedicated for discussion groups.

[1] The MoDELS MPM workshop was first held in 2006 in Genova, Italy, then in 2007 in Nashville, USA, and 2009 in Denver, USA and the last one in Oslo, Norway, 2010 and finally this year 2011 in Wellington, New Zealand.

2 Communications

1) Session: Multiple Models and Model Composition-MoC

The model-based engineering of a system involves several modeling steps with different purposes. For each purpose, depending (a) on the component of the system under study, (b) on the current activity of the design cycle, (c) on the chosen level of refinement, and (d) on the functional or extra-functional concern at stake, a different modeling paradigm may be used. The domain of Multi-Paradigm Modeling aims at easing the use of several modeling paradigms throughout the design process. The approaches presented during this occurrence of the MPM workshop can be classified in two main groups: (1) approaches addressing the translation of models from a modeling paradigm to another and (2) approaches dealing with the composition of models described using different modeling paradigms.

The first presentation session of this workshop has gathered approaches in the category of model composition. The presented approaches, however, address different situations where composing heterogeneous models is necessary. One of these situations is when one needs to assemble models of different components of a system in order to obtain a global model of the system, for analysis or simulation for instance. The approaches presented in papers [1] and [3] both address this kind of situation. Moreover, both approaches also rely on the concept of Model of Computation (MoC) as a way to represent the semantics of a modeling language. In particular, a Model of Computation is seen as an operational specification of the communication and synchronization aspects of a modeling language in the context of component-based modeling.

Paper [3] presents a framework called Cometa for heterogeneous modeling based on the concept of MoC. The central element of Cometa is a meta-model for representing models in which components are connected to each other through ports and connectors. Different ”domains”, i.e. different semantics can be attached to different elements of a model (components, ports, connectors) depending on the Model of Computation according to which they should be executed. In Cometa, a domain is made of four parts, called ”schemes”: a behavioral scheme which defines the type of behavior (discrete or continuous), a time scheme which defines the model of time, following the MARTE specification, a data scheme which defines the data types and a communication scheme which defines the types of ports and connectors implementing communication between components. It is possible to mix the domains associated to different elements in a given model. As a result, heterogeneous models can be ”flat” in Cometa, contrary to other heterogeneous modeling frameworks such as Ptolemy II [refptolemy] or ModHel’X [refmodhelx] in which heterogeneity is associated to hierarchy. When different domains are mixed in a model, it is necessary to add ”translator” elements in order to adapt the semantics of the different domains to each other. The authors propose to take into account in the translation the level of compatibility of the semantics of the domains to adapt as defined in a classification proposed in [refclassifmocs]. However, nothing is said about the formalism or technique that should be used to specify the translation itself.

The focus of paper [1] is specifically on this notion of ”adaptation” between models interpreted according to different Models of Computation. In this approach, based on a framework called ModHel’X, different MoCs can only be used in different hierarchical levels of a heterogeneous model. The authors argue that the adaptation between the different semantics of two hierarchical levels involving different MoCs must be modeled explicitly and separately from the models to adapt. To this end, they propose to use a language for specifying clocks and constraints between clocks called CCSL (Clock Constraint Specification Language), defined in the specification of the MARTE UML profile (Modeling and Analysis of Real Time and Embedded Systems). The example of a power window, modeled using two different models of computation, Discrete Events (DE) and Timed Finite State Machine (TFSM), is taken as a case study. The authors show how the semantic adaptation between DE and TFSM can be described declaratively using CCSL. Then, they present a simulation of the behavior of the window obtained using TimeSquare, a solver for CCSL clock contraints. In addition to the explicit modeling of adaptation between heterogeneous models, a benefit of this approach is that the model of adaptation obtained using CCSL is very concise. Moreover, to a certain extent, Timesquare can act as an analysis tool for the adaption model as it detects inconsistencies in the set of constraints. However, as pointed out by the authors, the CCSL language lacks some features such as the modeling of data and it has to be extended or combined to another approach to be fully usable for modeling the adaptation between MoCs.

Another kind of situation where model composition is necessary is when one needs to represent different and potentially overlapping aspects of one component of the system (or of the system itself). Several modeling languages may be necessary to represent these different views of a given component/system, and a major difficulty is to maintain the consistency among the views. The approach presented in [2] is an approach for multi-view modeling. Traditionnaly, multi-view approaches are either ”synthetic”, meaning that a global model of the system does not exist other than as synthesis of the information carried by different views, or ”projective”, meaning that there exists a single model of the system from which views are derived using a projection mechanism. In [2], the authors propose to combine the two methods. They present a methodology in which concern-specific meta-models for describing views can be derived from an initial overall meta-model (projective aspect of the methodology). Each view meta-model comes with automatically generated mechanisms for synchronizing and for preserving the consistency among all the views (synthetic aspect of the methodology). An implementation of the methodology using EMF, Ecore and ATL model transformations is presented. The authors show how Eclipse editors can be generated to allow multiple designers to concurrently create and edit views which conform to different view meta-models. Even if the view meta-models must derive from a global and common meta-model, this approach presents interesting features for heterogeneous multi-view modeling.

As a conclusion, the three papers presented during this session on model composition illustrate well the current challenges in this domain. One of them is the modeling of the semantics of modeling languages. Two of the presented approaches use the concept of Model of Computation as a way to represent the operational semantics of modeling languages. In the Cometa approach, in particular, the different components of MoCs are modeled separately, which is a step towards more modularity and reusability in the description of MoCs. Ongoing research targets the model-based verification and validation of MoCs. Another major issue in model composition is the modeling of the composition mechanism itself. As we have seen, composing models may serve two different purposes: composing models of different components of the system, or composing models that are different views of the system. In the former case, the composition must include an adaptation layer to take into account the differences between the semantics of the models to compose. The CCSL based approach described in [1] is an attempt to provide a modeling formalism to describe adaptation layers explicitely and independantly from the models that are composed. In the latter case, the composition must preserve the consistency of the composed models. In [2], the authors present a methodology for synchronizing and for maintaining the consistency among views of a given system.

2) Session: Model Transformations

Paper[BA11] investigates techniques to prove that model transformations are correct. They observe that even though it is now possible to explicitly build the set of syntactic correspondences from a given transformation, it is still not clear how to reason about the correctness of these syntactic correspondences w.r.t. to both the source and target languages underlying semantics. In the paper, correctness of a model-to-model transformation is analyzed by establishing a semantic relation between the respective operational semantics. The approach is demonstrated through a concrete translation between two languages: State Machines and Petri Nets.

The second paper [WKR+11] addresses the problem of model transformation re-use in different context. Their work is inspired by generic programming and proposes *generic model transformations*. Such generic transformations are defined over meta-model *concepts* which are later bound to specific meta-models. Current binding mechanisms lack automated resolution support for recurring structural heterogeneities between meta-models. Therefore, based on a systematic classification of meta-model heterogeneities, the paper proposes a flexible binding mechanism able to automatically resolve recurring structural heterogeneities between metamodels. For this, the binding model is analyzed and required adaptors are automatically added to the transformation.

Finally, paper [DCB+11] addresses the complexities of deployment-space exploration, by means of refinement (model) transformations. In particular, a high-level architecture description provides the basis for the choice of a low-level implementation. In this paper, the focus is on real-time systems. All possible solutions of a deployment step are generated using a refinement transformation while the non-conforming results are pruned as early as possible using a simulation model or analytical method. The feasibility of the approach was demonstrated by deploying part of an automotive power window, optimized for its realtime behaviour using an AUTOSAR-like meta-model.

3) Session: Posters

The work presented in [RP11] is about on-going work on an approach for querying, selection and pruning models. Under the assumption that models are often treated wholly or in part as trees, the authors propose a novel approach to model querying-by-example, treating models as trees. It exploits tree-based patterns in expressing queries, where the results of the queries are also trees. Thus, it provides means to compose (conjoin) queries without requiring intermediate manipulations.

Poster [BAGB11] concentrates on the issue of Languages Usability. Departing from the statement that usability evaluation of new languages is often skipped or relaxed [GGA10], although being a key factor for its successful adoption, the authors argue that a systematic approach based on User Interface experimental validation techniques should be used to assess the impact of those same languages. The poster goes a little bit further discussed the quality criteria, proposes a development and evaluation process that could be used to achieve usable languages.

Finally, poster [LZST11] presents work on verifying access control policies in statecharts. The approach is based on the transformation of a statechart into an Algebraic Petri net to enable checking access control policies and identifying potential inconsistencies with an OrBAC set of access control policies.

4) Discussion Group: Modular design of modelling languages

The working group on modular design of modelling languages recognized that there is a growing need for the principled and modular design, not only of models, but also of modelling languages.

In software-intensive systems for example, the need is increasingly felt for rigorously designed modelling languages to describe, analyze, simulate, optimize, and where appropriate synthesize their physical, control, and software components, as well as the often intricate interplay of those components. In addition, requirements/property languages for these highly diverse formalisms should be carefully engineered to match the design languages and to form the basis for automated analyzes. The need for combining modelling languages to form new ones goes beyond Domain-Specific Modelling Languages. Even simple State Machines are in practice already combined languages as their use requires not only the basic automata with states and transitions, but also an action language to describe what the effects of transitions are on the values of variables.

The starting point for the modular design of modelling languages is the realization that to define a language, its abstract syntax (AS), concrete syntax (CS), and semantics (SEM), including both semantic domain and semantic mapping, need to be explicitly modelled. In the working group, the different combinations of modelling languages were discussed. At the level of abstract syntax (modelled in the form of meta-models), this led to the insight that meta-models may need to be Reduced (when only part of a language is needed), Augmented (when new notions need to be incorporated) and Combined. The latter can be through the introduction of Associations, through merging, inheritance, and embedding.

The working group explored practical examples and took initial steps towards a classification of techniques for the modular design of modelling languages.

5) Discussion group: Modeling Model Compositions - MoCs

A major challenge in model composition is to obtain a meaningful result from models with heterogeneous semantics. Tackling this issue implies that the semantics of the modeling languages used in the composed models is described in a way such that it can be processed and, more importantly, such that it can be composed. In other words, the semantics of the modeling languages must be explicitly and formally described in order to enable model composition. One approach for representing semantics is the concept of Model of Computation. The objective of this discussion group was to work on a methodology to describe Models of Computation.

The first action of the group has been to work on the definitions of the main concepts relating to models, modeling languages and models of computation. It has been chosen to focus on models which describe the behavior of systems and two hypothesis were made:

- the abstract syntax of the modeling language used to describe a model is described by a *meta-model*. Therefore, there is a conformance relation between a model and the meta-model of its language.

- the semantics of the language must enable the *execution* of the model. Executing a model means computing one of the possible behaviors described by the model. The result of the execution of a model is a series of observation, i.e. a trace which represents of the computed behavior.

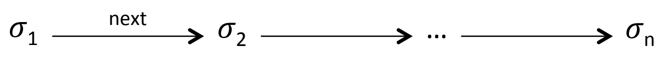

Figure 1: Execution of a model by a MoC. σ_i is an observation of the state of the model.

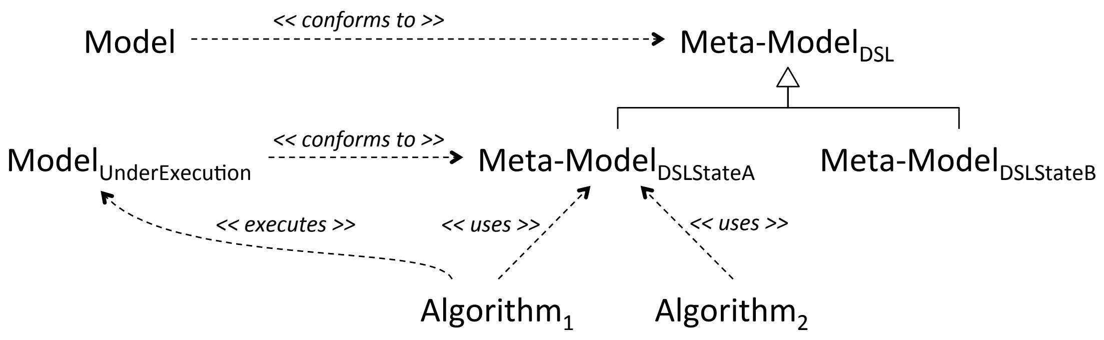

Figure 2: Notions of model and model under execution.

In this context, the discussion group has agreed on the following definition:

> **Model of Computation:** A Model of Computation is a constructive definition of the behavior of a model obtained by composition of the behavior of its elements.

In this definition, "elements" refers to the modeling elements used in the model, which are therefore defined in the meta-model of the modeling language. A consequence of this definition is that a Model of Computation is seen as a kind of algorithm which computes the trace representing the behavior of the model. When the model is in a given state, the role of such an algorithm is to compute an observation of the state of the model (a part of the trace) and its next state, as illustrated on Fig. 1.

Such an algorithm needs to manipulate the *state of the model*. However, modeling the state of the model may require modeling concepts that are not defined originally by the modeling language. That is why the group has chosen to distinguish between the model and the *model under execution*:

> **Model under Execution:** A "model under execution" includes an explicit representation of its state for execution purpose.

The modeling language with which the model under execution is described is therefore an extension of the modeling language used in the original model. In particular, the meta-model of its abstract syntax must be extended to include concepts for representing the state of the model. Figure 2 shows a model and the meta-model defining the corresponding abstract syntax. This meta-model is then extended to include concepts for representing the state of the model when under execution. Since choosing an execution semantics for a model has an impact on the way its state is modeled, several meta-models may be created for the model under execution, depending on the chosen execution semantics. At last, Figure 2 also illustrates that a given execution semantics may be implemented by different algorithms, just as the resolution of differential equations may be implemented by different kind of solvers.

The concepts presented in this section were defined by the discussion group in an attempt to lay the foundations for a methodology to represent models of computation. The group has tried to make explicit the notion of state of a model and to define means to model it. Much work remains to obtain a complete framework for describing models of computation, i.e. for describing execution semantics for models.

A major challenge in MoCs is to obtain a meaningful result from models with heterogeneous semantics implying that the semantics of the modeling languages must be explicitly and formally described in order to enable model composition. One approach for representing semantics is the concept of Model of Computation. The working group tackled the definitions of the main concepts to use in the methodology to describe Models of Computation. For that, the group focused the discussion on models which describe the behavior of systems and two hypothesis were made: i) the abstract syntax of the modeling language used to describe a model is described by a *meta-model*(i.e. there is a conformance relation between a model and the meta-model of its language) ii) the semantics of the language must enable the *execution* of the model (executing a model means computing one of the possible behaviors described by the model, and the result of its execution is a series of observation, i.e. a trace which represents of the computed behavior).

In this context, the discussion group has agreed on the following definition: *Model of Computation:* A Model of Computation is a constructive definition of the behavior of a model obtained by composition of the behavior of its elements.

The concepts defined by the discussion group is an attempt to lay the foundations for a methodology to represent models of computation. The group has discussed how to explicit the notion of state of a model and to define means to model it. In spite of the enthusiastic discussion held by this discussion group, much work remains to obtain a complete framework for describing models of computation, i.e. for describing execution semantics for models.

6) Discussion group: Evaluating Usability in the context of MPM

Departing from a perspective that MPM, as a methodology, thrives for removing the accidental complexity in software development, by choosing adequate formalisms at the right level of abstraction, the working group raised the following questions: Once decided the modeling formalisms how do we know that they are adequate? What is a good formalism adequate for a given problem?an finally, how do we measure that same adequacy to the problem?

It was generally accepted, by the discussion group, that to answer those questions we have to define Quality criteria. In addition, we need to proceed with empirical studies with real users and have a sound process for evaluation. The discussion proceeded by highlighting that an evaluation process can be (from a loose approximation to a more serious and strong one): a toy example, industrial case study, or properly conceived empirical studies. For the last one, the Quality criteria, supporting the evaluation process, implies metrics and measurement.

From this general vision, several questions arose as being still open for discussion: Does the composition of demonstrated usable Languages implies to be still usable?and, what is the difference between usability and Re-usability? Is MPM itself a good solution for Usability purposes?

The rest of the working group discussion was focused on sketching the road-map for a sound methodology and tool support to determine Usability in MPM.

3 Conclusion

The workshop was deemed very successful by the participants, and we plan to continue organizing future workshops. It is a vibrant forum for research and industry experts to join together and discuss fundamental concepts and tools for Multi-Paradigm Modelling. At the end of the event the attendants voted for papers [BDH+11] and [CCL11] for the best two papers award.

This workshop would not been possible without the help of many people besides the authors. We wish to acknowledge our Programme Committee to whom we thank:

- Antonio Vallecillo (Universidad de Málaga)
- Arnaud Cuccuru (CEA LIST)
- Bruno Barroca (UNL)
- Bernhard Westfechtel (U. Bayreuth)
- Cécile Hardebolle (Supélec)
- Chris Paredis (Georgia Tech)
- Christophe Jacquet (Supélec)
- Didier Buchs (U. Geneva)
- Dirk Deridder (Free U. Brussels)
- Esther Guerra (U. Carlos III de Madrid)
- Eugene Syriani (U. Alabama)
- Franck Fleurey (SINTEF)
- Frédéric Boulanger (Supélec)
- Gergely Mezei (Budapest University of Technology and Economics)
- Hessam Sarjoughian (U. Arizona State)
- Hans Vangheluwe (U. McGill and U. Antwerp)
- Holger Giese (Hasso-Plattner-Institut)
- Jeff Gray (U. Alabama)
- Jeroen Voeten (Eindhoven University of Technology)
- Jonathan Sprinkle (U. Arizona)
- Laurent Safa (Silver Egg Technology)
- László Lengyel (Budapest University of Technology and Economics)
- Luís Pedro (DAuriol Assets)
- Mamadou K. Traoré (FR Sciences et Technologies)
- Manuel Wimmer (Vienna University of Technology)
- Mark Minas (U. Federal Armed Forces)
- Martin Törngren (KTH Royal Institute of Technology)
- Matteo Risoldi (U. Geneva)

- Peter Bunus (Linkoping University)
- Pieter van Gorp (Eindhoven University of Technology)
- Stefan Van Baelen (K.U. Leuven)
- Steve Hostettler (U. Geneva)
- Thomas Feng (Oracle)
- Thomas Kühne (Victoria University of Wellington)
- Vasco Amaral (UNL)

Bibliography

[BA11] B. Barroca, V. Amaral. Asserting the Correctness of Translations. In *MPM'11, ECEASST*. 2011.

[BAGB11] A. Barisic, V. Amaral, M. Goulão, B. Barroca. How to reach a usable DSL? Moving toward a Systematic Evaluation. In *MPM'11, ECEASST*. 2011.

[BDH+11] F. Boulanger, A. Dogui, C. Hardebolle, C. Jacquet, D. Marcadet, J. Prodan. Semantic Adaptation using CCSL Clock Constraints. In *MPM'11, ECEASST*. 2011.

[BHJM11] F. Boulanger, C. Hardebolle, C. Jacquet, D. Marcadet. Semantic Adaptation for Models of Computation. In *Proceedings of the 11th International Conference on Application of Concurrency to System Design*. Pp. 153–162. IEEE Computer Society, June 2011.

[CCL11] A. Cicchetti, F. Ciccozzi, T. Leveque. A hybrid approach for multi-view modeling. In *MPM'11, ECEASST*. 2011.

[DCB+11] J. Denil, A. Cicchetti, M. Biehl, P. D. Meulenaere, R. Eramo, S. Demeyer, H. Vangheluwe. Automatic Deployment Space Exploration Using Refinement Transformations. In *MPM'11, ECEASST*. 2011.

[DCL11] P. I. Diallo, J. Champeau, V. Leilde. Model Based Engineering for the support of Models of Computation: The Cometa Approach. In *MPM'11, ECEASST*. 2011.

[EJL+03] J. Eker, J. W. Janneck, E. A. Lee, J. Liu, X. Liu, J. Ludvig, S. Neuendorffer, S. Sachs, Y. Xiong. Taming heterogeneity – The Ptolemy approach. *Proceedings of the IEEE, Special Issue on Modeling and Design of Embedded Software* 91(1):127–144, January 2003.

[GBA+07] A. Goderis, C. Brooks, I. Altintas, E. A. Lee, C. Goble. Composing Different Models of Computation in Kepler and Ptolemy II. In *Proceedings of the 7th international conference on Computational Science, Part III: ICCS 2007*. ICCS '07, pp. 182–190. Springer-Verlag, 2007.

[GGA10] P. Gabriel, M. Goulão, V. Amaral. Do Software Languages Engineers Evaluate their Languages? In *XIII Congreso Iberoamericano CIbSE2010 (Ecuador), Xavier Franch, Itana Gimenes, Juan Pablo Carvallo*. 2010.

[LZST11] L. Lucio, Q. Zhang, V. Sousa, Y. L. Traon. Verifying Access Control in Statecharts. In *MPM'11, ECEASST*. 2011.

[RP11] A. Radjenovic, R. Paige. An Approach for Model Querying-by-Example Applied to Multi-Paradigm Models. In *MPM'11, ECEASST*. 2011.

[WKR+11] M. Wimmer, A. Kusel, W. Retschitzegger, J. Schoenboeck, W. Schwinger, J. S. Cuadrado, E. Guerra, J. D. Lara. Reusing Model Transformations across Heterogeneous Metamodels. In *MPM'11, ECEASST*. 2011.