The OsMoSys approach to multi-formalism modeling of systems

V. Vittorini¹, M. Iacono², N. Mazzocca², G. Franceschini³

¹Dip. di Informatica e Sistemistica, Università di Napoli, Napoli, Italy
²Dip. di Ing. dell’Informazione, Seconda Università di Napoli, Napoli, Italy
³Dip. di Informatica, Università del Piemonte Orientale, Alessandria, Italy

Abstract. Analysis and simulation of complex systems are facilitated by the availability of appropriate modeling formalisms and tools. In many cases, no single analysis and modeling method can successfully cope with all aspects of a complex system: a multi-formalism multi-solution approach is very appealing, since it offers the possibility of applying the most suitable formalisms and solution techniques to model and analyze different components or aspects of a system. Another important feature that a successful modeling approach should include is the possibility of reusing (sub)models: by composing parameterized submodels and then instantiating the parameters, complete models of different scenarios can be obtained and analyzed.

This paper introduces an innovative approach to multi-formalism modeling of systems that is part of the OsMoSys (Object-based multi-formalism MOdeling of SYStems) framework. OsMoSys uses the proposed modeling approach to build multi-formalism models, and workflow management to achieve multi-solution. Our modeling approach is based on meta-modeling, allowing to easily define and integrate different formalisms, and on some concepts from object orientation. Its main objectives are the interoperability of different formalisms and the definition of mechanisms to guarantee the flexibility and the scalability of the modeling framework.

Keywords: Multi-formalism modeling – Meta-languages – Object orientation – Compositionality

1 Introduction

Formal modeling allows to specify, analyze and verify hardware and software systems. It is based on the application of a formal method to describe the system properties; such method, in turn, is said to be “formal” if it is based on a mathematical sound specification language [25]. In the past forty years, the formal methods research field has experienced the development of a wide spectrum of languages, nevertheless their effective usage in practice has been strongly penalized by the following factors [4]: mathematical basis require proper advanced skills, techniques do not scale as necessary, available tools do not adequately support the techniques and/or they are too difficult to use, in many cases no single analysis and modeling method can cope with all different aspects of a complex real system.

For these reasons, more than new formal languages and techniques, it seems necessary to develop new approaches and tools to let the available means to be effectively used and to inter-operate. This paper is focused on the research efforts in achieving formalisms and techniques integration, in particular the multi-formalism multi-solution approaches in which an unified framework implements more than one formalism and solution technique. Hence we do not mention the work done to improve the applicability and scalability of single notations and methods or to integrate tools.

Of course the integration of different formalisms in a single model requires a rigorous definition of the semantics of such integration. It is interesting to observe that despite the fact that composition techniques of distinct classes of formalisms seem different, they often have several common aspects, for example, the synchronous composition of terms in Process Algebras is very close to synchronous composition of Petri Net (PN) models by transition superposition; and the substitution of a Queuing Network (QN) model with an “equivalent server” (with load dependent service rate) is conceptually very similar to the substitution of a component with another behaviorally equivalent component in Process Algebras. Surprisingly enough, it can be easy to define a sound inte-
gration semantics also between formalisms of very different nature: e.g., a Fault Tree may be used to compute the fault rate of some component whose behavior (with and without fault) could be modeled by a Stochastic Petri Net (including a timed transition with firing rate equal to the above mentioned fault rate). It is thus natural to conceive a framework supporting multi-formalism modeling that does not pose any a-priori constraint on which formalisms can be combined and how.

The objective of combining multiple modeling formalisms into an unified methodological framework and a single tool can be pursued by adopting different strategies that differ in the multi-formalism modeling mode and the solution integration time.

Let us first consider multi-formalism modeling mode: we distinguish between explicit and implicit multi-formalism. **Explicit multi-formalism** is visible at the user level: the user may build a model by explicitly using different formal languages to express different parts of the model. According to this definition, the DEDS toolbox (Discrete Event Dynamic System) supports explicit multi-formalism, even if the user model, initially expressed in several formalisms such as QNs and stochastic and colored PNs, is then translated into a common abstract PN notation [2], hence models can be solved by using a PN solution approach. Another tool supporting explicit multi-formalism is Möbius [5]: a model can be constructed by composing sub-models that may be expressed through different formalisms (Stochastic Activity Networks, Markov chains, Process Algebra); any formalism can be integrated in Möbius, provided that a suitable mapping from such formalism to an underlying predefined common semantics can be implemented [6].

**Implicit multi-formalism** is a form of multi-formalism that is exploited at the tool/framework level and it is not visible by the user: he/she initially builds a model described through a modeling language which allows to define what parts are (implicitly) intended to be expressed by different formalisms. Multi-formalism is subsequently exploited in order to solve the models. We will see an example of this mode in Sect. 3.3.

In this paper we propose an approach that provides a practical and flexible mean to build both explicit and implicit multi-formalism models.

Let us now consider the second aspect mentioned above: solution integration time, that concerns the way in which models may share results. We distinguish two cases: models may share results a) in a dynamic manner, during the solution process e.g., by message passing as in DEDS; b) in a static manner, after the execution of the solution process, by exchanging results to obtain the overall solution, as in SHARPE\(^1\) (Symbolic Hierarchical Automated Reliability and Performance Evaluator) [21] and SMART\(^2\) (Stochastic Model-checking Analyzer for Reliability and Timing) [3]. Möbius supports both types of solution integration (the former applies to composed models, the latter to connected models). The framework proposed in this paper supports different types of solution integration in a very flexible way, by allowing to define customized solution processes (see Sect. 4); a solution process may involve appropriate filters (automatic model translators, automatic model generators, model composers) embedding the multi-formalism integration semantics.

A very recent trend in multi-formalism modeling is to take advantage from meta-modeling approaches. Such approaches are based on a three layered structure [17, 24]: (1) meta-metamodel level, (2) metamodel level, (3) model level. At the model level we describe an entity by using a specific formalism; e.g. we can describe a set of interacting processes by means of Petri Nets or Process Algebras. At the metamodel level, formalisms are defined, allowing to express models in the next level; for example the Petri Nets, Process Algebras and Fault Tree formalisms are defined at this level. Finally, at the meta-metamodel level languages are defined for describing formalisms; e.g. a language to define graph-based formalisms could be defined at this level, including the possibility of listing the type of nodes and arcs allowed in a given graph-based formalism, as well as the constraints on how to connect them. In other words, the idea is to define a language that may describe many different formalisms, so that a model of such modeling language is a formalism, which in turn can be used to construct models of concrete systems.

Among the many benefits of meta-modeling, is important to mention that it allows to easily implement visual tools to describe formalisms, as well as models expressed in such formalisms (e.g. [9, 12, 17]), for example AToM\(^3\) [9] is used as a graphical front-end for the multi-formalism hybrid\(^3\) systems modeling approach described in [23] and to process PNs and Statecharts [10].

Meta-modeling promotes the availability of tools that support multi-paradigm modeling. **Multi-paradigm** [19, 24] modeling is a more general concept than multi-formalism. It includes the combined usage of different modeling languages, meta-modeling and model abstraction. The multi-paradigm approaches nearest to our research field (multi-formalism) are described in [20, 23].

In summary, some results in the direction of combining multiple modeling formalisms into an unified methodological framework and a single tool have been achieved. The most flexible approaches seem to be the Möbius approach and the multi-paradigm approach implemented by means of the AToM\(^3\) tool since they do not define a-priori what formalisms are to be used and

---

1 SHARPE is a tool for specifying and analyzing performance, reliability and performability models that deals with both combinatorial and state-space models.

2 SMART is a software package that integrates various high-level logical and stochastic modeling formalisms in a single modeling framework and provides both simulation capabilities on Markov models and symbolic model-checking.

3 Here “hybrid” stands for both continuous and discrete models.