Chapter 10
Designing User-Defined Modeling Languages with SIMTHESys

Mauro Iacono
Seconda Università degli Studi di Napoli, Italy

ABSTRACT

The availability of modeling languages that best suit the needs of modelers from case to case can enhance the applicability and immediateness of a model-based approach in the design and evaluation of systems, and can meet the modelers' expertise and customs. The goal of SIMTHESys is to enable the design and implementation of user-defined modeling languages (dubbed formalisms) and their native integration in multiformalism models, so that it is possible to develop language enhancements for existing formalisms, domain-oriented languages, extensions for existing languages, and abstraction tools to empower the modeling process. The key of this feature relies on the SIMTHESys metamodel and the description framework that is founded onto it. This chapter presents the foundation of SIMTHESys formalisms and demonstrates the possibilities it offers by examining a complete modeling example of a classical problem and a number of different formalisms, chosen to suggest how to exploit the framework.

INTRODUCTION

Abstractions are the key tool by which the essential aspects of a problem to be solved can be isolated from the richness of complex but negligible details characterizing reality. Such essential aspects can be explored at different level of detail, to help dominating the problem in different phases of its analysis or in different perspectives of is nature. The mean by which abstractions are practically used to support reasoning, analysis and solution of a problem are usually proper representations, formal or informal, textual, mathematical or graphical, called models. A model describes a part of a real problem, materializing an abstraction of it

DOI: 10.4018/978-1-4666-4659-9.ch010

to allow the modeler to focus on that specific part and dominate it by understanding its explicit and implicit properties, evolution and characteristics.

A modeler can identify the causes and effects that are important for the evolution of a model, and describe them by appropriate modeling languages. The advantages resulting from the correct choice of a modeling language (a formalism) for a certain part of the system are twofold. From point of view of the modeler, it is possible to model each subsystem using the most appropriate language; from point of view of the analysis, the right (combination of) formalism(s) results in a proper mapping of the model concepts onto the primitives in which the analysis tool (a solver) is articulated.

The correctness criteria of the choice are a matter of the specific problem to be faced, and should be determined case by case by the modeler: the point is to provide a modeler with the right language, whichever the criteria. A possible, reference, general interpretation of what a right language is can be given by ease of use, where ease of use can result e. g. from:

- Ease of representation: choice of a language the constructs of which are semantically or syntactically close to the objects that are relevant in the domain of the problem that the modeler wants to represent: an example (furtherly detailed in this chapter) can be given by designing a language to support modeling of software aging and rejuvenation problems that represents by specific constructs a critical error due to accumulation of errors over time, a rejuvenation strategy, a degradation condition.
- Ease of learning and application: choice of a language in which new aspects of a problem are represented by additional elements for an existing language with which the modeler is already familiar: an example is given in (Codetta Raiteri et al. 2004) where a language able to analyze reliability and availability of repairable systems (Repairable Fault Trees) is defined by extending the common Fault Trees language with a state-space component, namely Repair Box, that hides all repair-related aspects and transparently applies them for the analysis[1] of the resulting model.
- Time efficiency in modeling: choice of a language that allows rapid development of models from parts of the real world problem: an example is given by the SIMTHESysHQL language (Barbierato et al. 2013c), that allows the direct use of queries defined in a Big Data applications query language in performance models, thus enabling an immediate modeling of complex operations such as massively distributed data retrieval operations.

Width of choice possibility is thus a factor that can improve the overall efficiency of the modeling process: the problem is the availability of a multitude of modeling languages, or, better, the availability of a mean to extend the possibility of designing, experimenting and customizing specialized modeling languages to meet the needs that every case will manifest; and, consequently, the availability of a mean to rapidly develop proper analysis algorithms for the needed modeling languages. A further, desirable characteristic is the possibility of defining models in which different parts are modeled with different modeling languages, to ensure flexibility in the approach.

SIMTHESys (Structured Infrastructure for Multiformalism modeling and Testing of Heterogeneous formalisms and Extensions for SYStems (Gribaudo & Iacono 2011a, Iacono & Gribaudo 2010, Iacono et al. 2012)) is a framework for the definition and solution of multiformalism models. It is based on the automatic generation of implementations of specific analysis algorithms (solvers), starting from the rules contained in the definitions of modeling languages (formalisms).