Appeared in: 24th International Workshop on Qualitative Reasoning, Portland, USA, 2010

MOM – An Environment for Multiple Modeling

P. Struss, T. Regassa

MQM Group, Technische Universität München
Email: struss@in.tum.de, regassa@in.tum.de

Abstract

Computer-based modeling plays an important role in many domains, such as engineering, biology, sociology. The degree of re-use of models for different systems, scenarios, and purposes is very low, even when they are quite similar, and there are few libraries containing models that are context-independent. Models used for successful projects often have specificities of the task, domain, or even device compiled into them, so that they cannot be integrated into different projects even of similar nature. Another obstacle is the use of different modeling paradigms and incompatible tools that cannot be integrated.

The Model Manipulation Environment (MOM) project has been initiated to develop foundations and a software tool for managing, manipulating, transforming, and integrating models. It should support the user in building and using model libraries with a higher re-use of its elements, even enable the use models implemented in different paradigms and tools in an integrated way, and make the modeling process more transparent.

We present the underlying perspective on the modeling process and the specification of the system and illustrate its utility through the implementation of an automated model abstraction operator.

1. Introduction

As industrial plants and products become more complex and economical and environmental constraints become more tightened, the need for using models to support various tasks during the life-cycle of a product or plant (such as design, control, on-board and off-board diagnosis, testing, and maintenance) increases. Also in other areas, such as biology, ecology, or economics, models are used for testing hypotheses and planning interventions.

There are two fundamental, rivaling requirements on the models and the modeling process: the necessity to keep the modeling process simple and cheap and the objective of obtaining models that effectively and efficiently support solving a particular problem. While the former benefits from the existence of libraries with generic, reusable models, the latter demands for models that capture specificities of a particular domain and task. Although the idea of configuring models from elements of a library is widely appreciated, current practice is dominated by the repeated creation of tailored single-purpose models. The degree of re-use of models for different systems, scenarios, and purposes is very low, even when they are quite similar, and there are few libraries containing models that are context-independent. Moreover, the use of different modeling paradigms and incompatible tools prevent the integration of complementary models.

Creating and organizing multiple, alternative models and supporting the selection of appropriate (combinations of) model fragments has been studied early in the development of the field of qualitative modeling, one key idea being annotating models with their different underlying modeling and simplifying assumptions, e.g. in graphs linking models with edges labeled by such assumptions ([Addanki et. al 89], [Falkenhainer-Forbus 91]). [Struss 92] and [Weld 92] formalized abstraction of models in contrast to model simplification. Assuming that all possibly relevant alternative models have been pre-manufactured is not very realistic. [Nayak 95] searches for the simplest model that provides a causal phenomenon. In [Sachenbacher-Struss 05], a maximal model abstraction reflecting observable and target distinctions is automatically generated, and [Struss 02] discusses practical experiences in automated domain abstraction. [Torta-Torasso 09] present an algorithm for aggregating indistinguishable behavior modes.

Given this background, our work on creating MOM (Model Manipulation Environment) pursues the following goals:

- Building an ontology for a layered representation that distinguishes and explicitly represents alternative modeling decisions (an appropriate structural representation, as well as choices of relevant quantities, variables, and domains) that enables the re-use of those layers that are shared among different models
- Supporting and automating the integration of different fragments of a behavior model, e.g. by mappings between different domains
- Designing abstract interfaces to support the integration of various existing modeling environments, such as Matlab/Simulink, Modelica, Spice, etc
- Developing the theoretical foundations and specification for model operators (e.g. for automated model abstraction, aggregation, structural transformations) that

use the abstract interfaces and, hence, are applicable to classes of modeling environments.

• Developing a model of the modeling process, including problems of distributed development of models and model libraries.

The following section goes through a modeling example to explain the main objectives of MOM. Section 3 presents a systematic structuring of models in MOM derived from this, and introduces the concepts in more detail using UML. We mention the use of the system for automated model abstraction and discuss the current state of the system and future work.

2. The Challenges

The process of modeling has to solve a number of problems: 1) how to represent **structure** of the system to be modeled; 2) how to describe the **behavior** of the **system elements**, and 3) how to use these structural and behavioral descriptions to derive a model of the **behavior of the entire system**. To none of these issues, there is a unique answer; each step involves a number of choices to be made by the modeler.

For illustration, modeling an electrical resistor (see Fig. 1) involves choices at the level of

1. **structure**: what are the potential paths of interaction of the resistor with other elements? Besides two obvious terminals that establish an electrical connection, does it interact with its environment via heat radiation, i.e. a third terminal?
2. **quantities**: which properties and quantities are required to characterize the relevant aspects of its behavior? Should both voltage and current be included? Does temperature have to be included, because it might influence the resistance? Note that these are purely conceptual units, **not** variables. This is the next choice.
3. **variables**: which aspects of the chosen quantities need to be represented by variables? Do consider only the magnitude of current, or also its derivative? We might want to have also (or only) a variable representing the deviation of a quantity. There could be more (mean value, increment...). The association quantity:variable is 1:n.
4. **model decomposition**: what are meaningful groups of variables needed to represent different "laws" of the behavior? We call them model frames. For the resistor, we probably have Ohm's and Kirchhoff's law with the respective set of magnitude variables. Instead or additionally, we may include the laws for the deviation variables.
5. **domains**: what are the possible values of variables that need to be distinguished? Is the deviation in current expressed as a numerical value or simply by its sign? Note, so far, we only spanned the space for describing the behavior.
6. **behavior descriptions**: how do we implement the "laws" of behavior? There is a whole variety of different formalisms and modeling tools.

The steps and decisions in the sequence above constitute an ideal top-down modeling process. In practice, it will usually not work this way. There may exist a structural description of a circuit and a set of models of electrical components implemented in Matlab/Simulink, and the modeling environment should support linking these separate parts. Also, the process may be completely bottom-up.

Identification of the decision levels and enumeration of the alternative options at each level enables provision of generic model building blocks that can be reused for different systems and purposes. For instance, if we wish to model the same electrical circuit without temperature-dependent resistance and using magnitudes only, but alternatively at a qualitative and a numerical level. Then we can reuse all the model elements and respective decisions down to the level of model frames and branch only to different behavior descriptions.

Furthermore, if we have two different numerical simulation tools that implement the same system, then everything at the higher level remains the same, and it is possible to specify the use of these numerical models (for simulation, diagnosis, or transformations of such models) in a single way independently of the particular tool used.

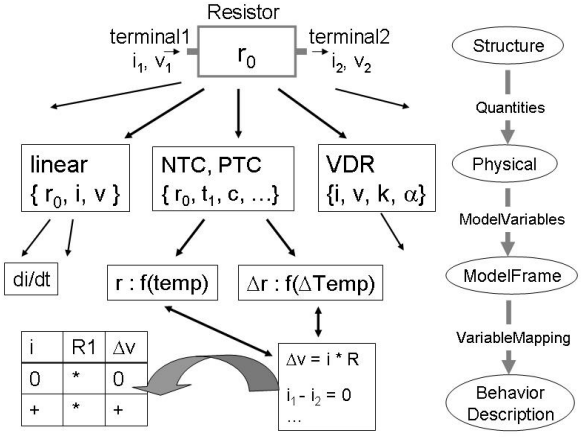

Figure 1: An example illustrating the decision steps taken while modeling

3. Specification of MOM

3.1 Overall Structure

In perspective, there may be two or more actors involved in the modeling process: one building a library of re-usable model elements, and one using these elements to build a model of a system of higher complexity for the purpose of behavior prediction.

Figure 2 shows schematically the idea of decision-based model building, consisting four main layers. The concept rests upon the assumption that the transition from one layer to the next involves a decision. They are taken and characterized by the user when specifying the respective model elements. The work of the model builder then

consists of fitting suitable elements together and selecting among alternative decisions at each transition to create the desired modeling unit. A number of such units are then connected according to the physical/logical part-of relation and hierarchy representing the complexity of the task at hand.

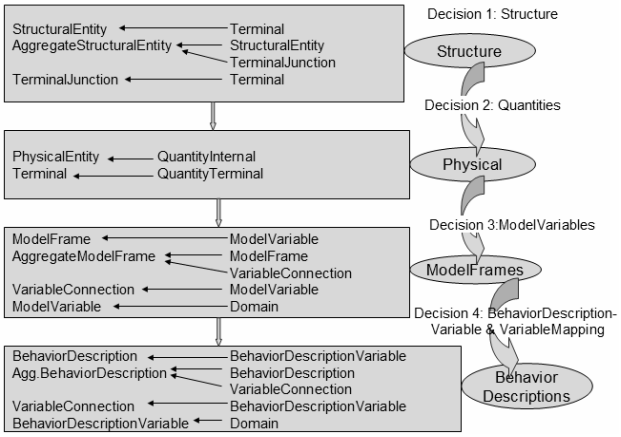

Figure 2: Decision-based model building steps

Fig. 2 contains an overview of the essential classes and their associations at each layer:

- **StructuralEntity** with Terminals associated and can be an aggregate with a structure specified by linking terminals.
- **PhysicalEntity** with associated Quantities.
- **ModelFrame** with ModelVariables and their (basic) Domains. It may be an aggregate one with a structure specified by linking variables (refining the physical structure)
- **BehaviorDescription** containing the respective variables and domains used for the specific implementation.

The latter has subclasses representing different types of external modeling environments, such as directed numerical computation or finite extensional, and specifying an abstract interface that has to be implemented for each tool, e.g. for Matlab or a constraint system. This allows us to specify and implement algorithms and operators for such classes, rather than individual tools, and to integrate them easily.

In the following, we present the classes in more detail using UML static structure diagrams.

3.2 Structural Entity

From a compositional point of view, a real-world system can be seen as a structural hierarchy of Entities (physical components, objects, processes, etc.) with part-of-relationships, and in terms of how their internal states are reciprocally influenced. A StructuralEntity represents one of the units that are the building blocks of a (physical) system to be modeled. As usual, it has one or more points of contact with its surroundings – called Terminals, through which all of its interactions with the outside world take place.

A StructuralEntity can be elementary or an aggregated one. An elementary structural entity represents an atomic entity, whereas an aggregate one is made up of other structural entities connected through their terminals in a composite pattern (and has terminals to use it as a higher-level building block). A node where such terminals make contact is called a TerminalJunction. Figure 3 depicts StructuralEntity and its components.

3.3 Physical Entity

PhysicalEntities are StructuralEntities with chosen Quantities. They can be associated directly with the entity, representing their StateVariables and Parameters. StateVariables are internal attributes of an entity that may change due to external influences, e.g. temperature and temperature-dependent resistance, whereas Parameters are attributes or properties of the entity that can be taken as constant, e.g. resistance assumed to be fixed. Or they are associated with Terminals and, hence, sharable among Entities.

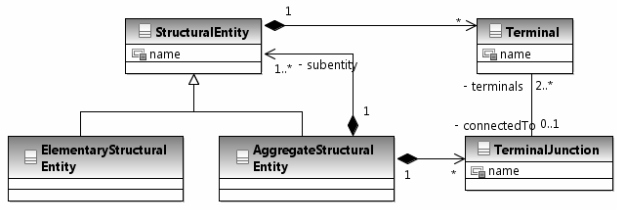

Figure 3: StructuralEntity

When a PhysicalEntity is created by associating the desired type and combination of internal and terminal quantities to a StructuralEntity, this choice is explicitly represented as a QuantityDecision as shown in Figure 4, which yields a set of Quantity-Associations - a set of compatible quantities needed for the particular modeling scenario.

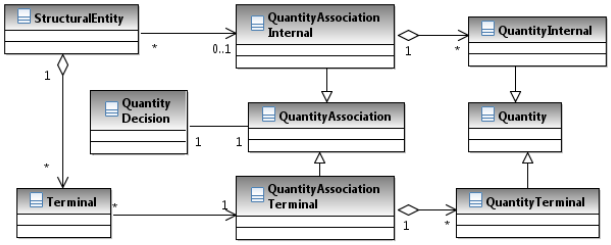

Figure 4: QuantityAssociation and QuantityDecision in relation to StructuralEntity

At this level, we only have a conceptual representation of some constituent of the physical phenomenon we would like to model, **not** a model fragment with particular variables.

3.4 Model Frame

A given PhysicalEntity may be modeled in many different ways. Different QuantityAspects can be considered in different systems and tasks involving the same quantity. Such common aspects are magnitude, derivative and deviation. But additional ones can be defined by the modeler. Each pair (Quantity, QuantityAspect) specifies a ModelVariable to be used in the behavior modeling (Fig. 5).

ModelFrames are generic, just provide sets of variables to be used, do not contain any actual description of behavior of the PhysicalEntity they are associated with, and, especially, do not presuppose a particular mathematical or computational form of such a description. In particular, their domains have to be chosen as generic and the most fine-grained ones, which can then be mapped to more specific representations.

A ModelFrame can be structured in a hierarchical way, where structure is captured by linking ModelVariables, which is straightforward and not shown in Fig. 5.

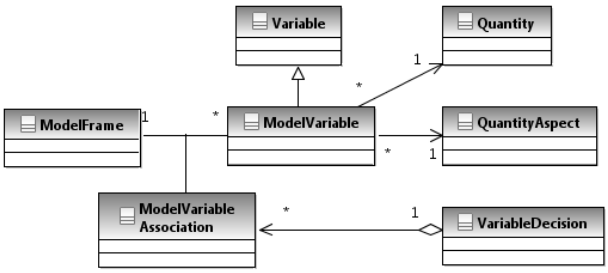

Figure 5: Concepts of ModelFrame and ModelVariables

3.5 Behavior Description

A BehaviorDescription (Fig. 6) is like a container for an actual description of a portion of the behavior of an entity. It could be some quantitative (numerical) model in the form of differential equations or inequalities, or a relational model in terms of a finite set of qualitative values, or some other abstract characterization. The BehaviorDescription fixes the granularity and structure of the Domains of the variables actually used in the computation. Mappings specify the correspondence of variables of the ModelFrame to the variables used in the BehaviorDescription (with possibly totally different names) and how their domains are related. Since a ModelFrame can be implemented by different BehaviorDescriptions (e.g. the numerical and the qualitative version of Kirchhoff's law), the respective association is explicitly labeled by an explicit decision (omitted in Fig. 6).

Note that one BehaviorDescription may also be used to implement different ModelFrames: to give a trivial example, a Matlab™ block out1= - in1 implements both the magnitude and the deviation version of Kirchhoff's Law.

MOM contains interfaces to various *classes* of BehaviorDescription, which allows the builder of models and model-based systems to use these interfaces regardless of the underlying computational system being used. The main issue here is to provide a generic definition for an interface that can be a common denominator for the different formalisms being used to represent a behavior of an entity.

BehaviorDescriptions are classified, e.g. depending on whether the relation is computational or symbolic, and whether computation is directed or undirected. Operations in connection with BehaviorDescription include evaluation of some output for a given set of input, addition or elimination of a variable, etc. In case of tabular relations, operations may include addition or removal of a tuple of such a relation.

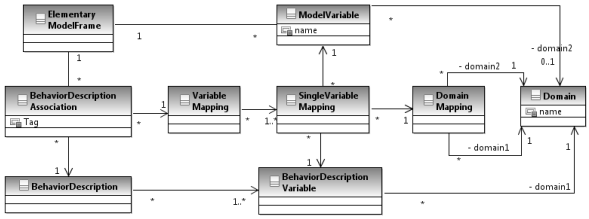

Figure 6: Association between ModelFrame and Behavior-Description and the related VariableMapping

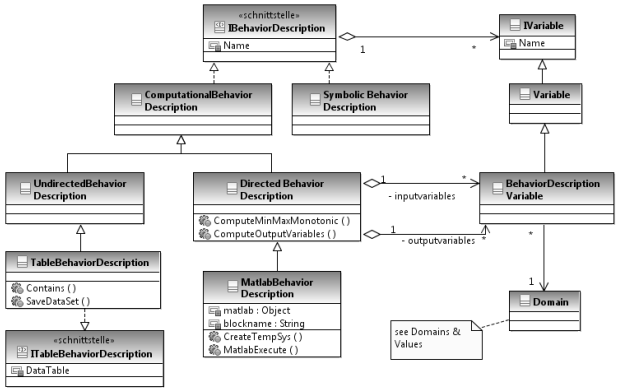

Figure 7: Classification of BehaviorDescription

3.6 Domains

A Domain is a set that contains all the possible values of a Variable. Implicitly, a domain encompasses a range of values of a certain type. Such values can be broadly divided into numeric or non-numeric (i.e. symbolic) types, and they may be discrete or continuous over a bounded range.

Depending on the type of the values and their range, domains can be finite or infinite. Symbolic values are inherently finite, so InfiniteDomain refers usually to numeric value types. A FiniteDomain may contain a finite (discrete) set of numerical values, or may simply represent a list of objects without any numerical connotation, or it may represent qualified references to an interval of numerical values (see Figure 8).

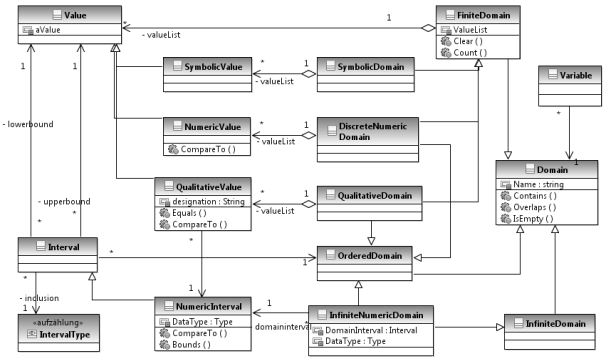

Figure 8: The Interrelation between Domains and Values

3.7 Summary

Fig. 9 provides a selective overview of the main classes of the various layers. The associations that are subject to potential alternative choices (in particular, different sets of Quantities, different QuantityAspects and, hence, Variables, and different Domains) are supported by explicitly represented modeling decisions. Navigation can be done by specifying a unit at some layer and a set of decisions that yield units at the related layers. Logical relations between modeling decisions can be stated: exclusiveness helps to configure a coherent model, and implication supports a compact representation of choices.

Note that the model decisions in MOM are more fine-grained than the ones in the other referenced work on multiple modeling. This forms the basis for a higher degree of reuse of the various layers. For instance, a qualitative and a numerical model of a circuit may share everything from the structure representation down to the ModelFrames, and branch only to different BehaviorDescriptions. An absolute and a deviation model of some device still reuse the structural description.

Furthermore, the environment provides the basis for integrating various modeling and simulation tools and even supports the mixed use of them.

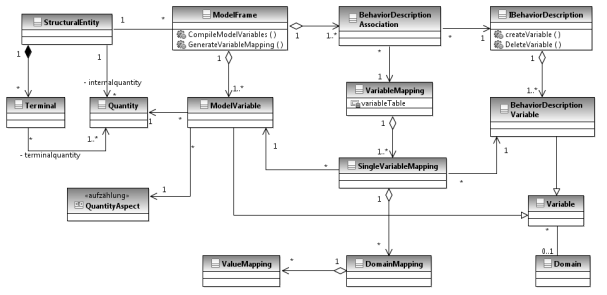

Figure 9: Overall interrelation between StructuralEntity, Model-Frame, Variables and BehaviorDescription

4. Current State and Future Work

So far, a kernel of the overall system, which is described in section 3, has been implemented and used to realize the automated abstraction operator and also to support mappings between different representations in an application project. Also, an interface to Matlab™/Simulink™ has been implemented in MOM (MatlabBehaviorDescription) and used to automatically produce qualitative abstractions of Matlab models that then form alternative BehaviorDescriptions in MOM. Details are described in [Fraracci 09].

To create a complete, useful modeling environment, future work will address a number of issues:

- **Library**: we need a persistent storage of the model fragments or, rather, of templates used to create instances for usage in different structures, scenarios, and contexts. This is more complex compared to the component model libraries we are maintaining right now: firstly, there are additional types of fragments, such as model frames, and secondly, there are complex templates, basically specified by one fragment at some level and a set of modeling decisions which may include different sets of variables and domains and mappings between them required for the instantiation of the template.
- **Browsing, selecting, and composing multiple models**: the existence of various modeling decisions and the respective alternative model fragments generates the necessity for a transparent and informative presentation of the content of a library and the structure of complex model, including the involved modeling decisions and their relations, and powerful tools for navigating through a potentially huge and confusing space of alternatives.
- Specific **modeling ontologies**, esp. component-oriented modeling: one may be tempted to believe that the modeling paradigm presented here is simply a component-oriented one. Actually, this is not true: it is more general, providing a frame for a quite large class of compositional approaches modeling. Indeed, component-oriented modeling can be realized in two fundamentally different ways: the more natural one appears to be the one that introduces Component as a subclass of StructuralEntity, which allows to maintain one component structure and requires the introduction of behavior modes (OK and faulty behaviors) for reasoning for diagnostic and related purposes. This approach views a component as a certain element in the structure, which modifies its behavior under fault conditions. Alternatively, one can view the various component modes as the structural constituents, since, after all, when a component transitions to a fault mode, this is like replacing a particular StructuralElement, say a correctly working resistor, by a different one, e.g. an open electrical dipole. We have not yet analyzed whether the environment is

generic enough to support the implementation of other modeling ontologies, including those with an explicit causal annotation, such as process-oriented modeling or Bond graphs.

- **Prediction, simulation, and generic model-based reasoning**: this paper (and the currently finished implementation) focuses on the structuring and representation of models. It does not address the question of **how to use** the models for a particular purpose. Of course, the goal is to provide an interface to MOM models and a set of generic reasoning and problem solving algorithms exploiting this interface. There are basic ones, such as prediction, simulation and consistency check, and task-oriented ones like consistency-based diagnosis or test generation. The aim is to be able to implement the algorithms independently of the particular modeling tool used, which would include the possibility of using models implemented in different tools in a mixed way, e.g. simulating a model including BehaviorDescriptions implemented in Matlab™/Simulink™, Spice, Modelica, and even a Finite Constraint Satisfaction Algorithm.
- **Representation of pragmatics and an application context**: what has been presented in this paper is restricted to an abstract modeling of the structure and behavior of a system. Using such a model for particular problem solving task requires representing properties of the real physical system and the context for the problem solving task, such as (e.g. in a diagnostic setting) which quantities are observable and the observation granularity, the physical lay-out of a structure, which actions are required to provide input to the system and measuring its response, the cost of these actions, an assessment of the practical utility of the results generated by an algorithm, etc. It is important to strictly separate these aspects from the generic representation of a device topology and the behavior. However, any problem solver requires the representation and exploitation of such information and, perhaps, special strategies to exploit the generic and the pragmatic level in an integrated way.
- **Implementing various model operators**: so far, we have used the system for automated model abstraction. Many more useful, generic operators are anticipated (and some of them have already been specified), such as the generation of deviation models from an existing (numerical) model, structural model compilation (turning a compositional model into a black-box model by eliminating intermediate variables), and others.
- More interfaces to **special modeling environments**: this is needed to support the integration of models from several origins and use of heterogeneous models. So far, interfaces to Matlab™/Simulink™ and to a relational representation have been implemented (for the automated abstraction operator). Interfaces to other modeling tools, such as Modelica or various constraint satisfaction algorithms are on the agenda.
- **Human-Machine-Interaction and GUI**: The current tool has a very confined GUI, which mirrors the limited functionality and the stage of the project. We would like to emphasize that the main emphasis is not on a GUI, but well-developed concepts for the interaction with the users, which are actually different types of actors, such as the builder(s) of the library and the modelers using the library. The specification of the interaction has to be based on a model of the modeling process(es), and we believe that the foundation of MOM allows us to develop this.

We cannot and do not intend to solve the last two problems ourselves. Rather, we plan to make the core of the modeling environment available as public software in order to allow the model-based systems community to implement a diversity of modeling ontologies, predictors, simulators, problem solving algorithms, modeling GUIs etc. within this environment and benefiting from this through the easy integration and combined use of models, model operators, and algorithms developed by the community.

References

[Addanki et. al 89] Addanki, S., Roberto Cremonini, J. Scott Penberthy, Graphs of models, Artificial Intelligence (51) 1-3, 1991 p.145-177

[Falkenhainer-Forbus 91] Falkenhainer, B., and Forbus, K. Compositional modeling: Finding the right model for the job. Artificial Intelligence, 51, 1991, p. 95-143

[Fraracci 09] Fraracci, A.: Model-based Failure-modes-and-effects Analysis and its Application to Aircraft Subsystems. Dissertationen zur Künstlichen Intelligenz DISKI 326, AKA Verlag/IOS Press, 2009

[Nayak 95] Nayak, P., Causal approximations. Artificial Intelligence (70), Issue 1-2, 1994, p. 277 – 334

[Sachenbacher-Struss 05] Sachenbacher, M. , Struss, P. Task-dependent qualitative domain abstraction, Artificial Intelligence, (162)1-2, 2005, p. 121-143

[Struss 92] Struss, P., What's in SD? Towards a Theory of Modeling for Diagnosis. In: Hamscher et al. (eds.), Readings in Model-based Diagnosis. Morgan Kaufmann Publishers, 1992 p. 419-450

[Struss 02] Struss, P.: Automated Abstraction of Numerical Simulation Models - Theory and Practical Experience. In: Sixteenth International Workshop on Qualitative Reasoning, Sitges, Catalonia, Spain, 2002

[Torta-Torasso 09] Torta, G., Torasso, P., Parametric abstraction of behavioral modes for model-based diagnosis, AI Communications (22) 2, 2009, p.73-96

[Weld 92] Weld, D., Reasoning about model accuracy, Artificial Intelligence (56) 2-3, 1992, p. 255-300