Software and Systems Modeling (2024) 23:1053–1054
https://doi.org/10.1007/s10270-024-01210-z

EDITORIAL

How does your model represent the system? A note on model fidelity, underspecification, and uncertainty

Benoit Combemale[1] · Jeff Gray[2] · Jean-Marc Jézéquel[3] · Bernhard Rumpe[4]

Published online: 23 September 2024

Model-driven engineering (MDE) was elaborated over two decades ago based on a very limited set of seminal concepts: system, model, metamodel, and seminal relationships (i.e., representation of, and conforms to), as shown in Fig. 1.

Since the early development of MDE, the modeling community has produced a large body of knowledge about conformity. This also allows metamodeling techniques to be used as definitions of the modeling languages themselves. Beyond a pure conformity relation between model and metamodel, various detailed "typing" relationships have been explored to support a better decoupling and modularity of models, and thus a certain degree of reuse. Some flexibility in conformity has been explored to support different levels of formality during modeling activities, hence supporting the process from informal to formal modeling.

However, very little literature can be found on the representativity of the model with regard to the system under study, raising the question: "How well does a given model actually represent the system?" Indeed, the core modeling activity has been conceptually defined, e.g., by Stachowiack in terms of characteristics of a model: a model is an abstraction of an original (i.e., the "system"), and fulfill a specific purpose with regard to this original. Several relaxations have been identified. For example, the system might still be under construction and does not exist yet, a model may have many

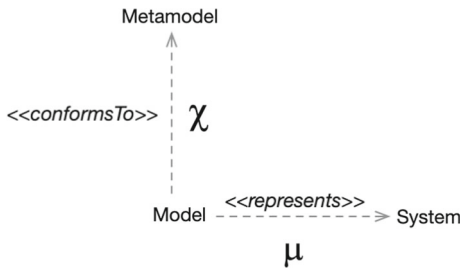

Fig. 1 Seminal MDE Concepts and Relationships

purposes (dependent on the different stakeholders) and a model may be composed of other models. However, all of these techniques do not provide a formal way to capture and reason about the actual representativity of the models built. In this editorial, we discuss some ways of qualifying the representativity relationship in terms of *fidelity* and *underspecification*, leading to some degrees of (possibly explicit) *uncertainty*.

Borrowing a concept from math, the *representativity* is formally defined as a *refinement*, but this view is often too narrow and usually fails if we consider a complex real-world system, either a software system or a cyber-physical system. Fidelity is a useful concept in this context: the model represents the system with a level of *fidelity* depending on the available information on the system, as well as the abstractions used to describe the model. As such, the fidelity can be seen as a distance between a (descriptive) model and the system under study. Many forms of fidelity can be found, such as the model capturing the system correctly within a percentage of actions versus time, average and maximal latency (delay) of reactions, and other manifestations of the concept.

Furthermore, a model can voluntarily capture an *underspecification* of the system, i.e., information that is not precisely captured in the (prescriptive) model, possibly leading to a broader design space. Such underspecification can be captured easily with math (e.g., a value in an interval) or state machines (e.g., several transitions nondeterministically

✉ Bernhard Rumpe
bernhard.rumpe@sosym.org

Benoit Combemale
benoit.combemale@sosym.org

Jeff Gray
jeff.gray@sosym.org

Jean-Marc Jézéquel
jezequel@irisa.fr

[1] University of Rennes, Rennes, France

[2] University of Alabama, Tuscaloosa, AL, USA

[3] IRISA/INRIA and University of Rennes 1, Rennes, France

[4] RWTH Aachen University, Aachen, Germany

triggered by the same stimulus). However, the underspecification of a system presents challenges in executable (and thus deterministic) models. Underspecification leaves open interpretations wherever it is controlled by the developer (through design decisions) or uncontrollable runtime behavior of the system. In practice, an underspecification must also be regarded as uncontrollable when it is "controlled" and decided by the context of the component (i.e., in other development teams), when product variants may have different realizations, or when later upgrades or component replacements change the design decisions.

Both the fidelity level and uncontrollable underspecification lead to the notion of *uncertainty* in modeling. Uncertainty in Software Engineering is a widely studied subject. Padulo and Guenov summarized previous research as twofold: uncertainty *about* the problem (the *epistemic view)*, and uncertainty *within* the problem (the *ontological* view).

Epistemic Uncertainty includes misunderstanding of the user's needs, design space variability, unexpected behavior of APIs, or variable behavior among functionally similar libraries. It may also appear as subtle differences in the semantics of programming languages or even inside the same programming language, such as undefined behavior in C. Thus, epistemic uncertainty is mainly a design topic and needs adequate development actions to reduce or control this kind of uncertainty.

Ontological Uncertainty includes noise in the input data of a program, its memory layout, network delays, the internal state of the processor, the ambient temperature and even the age of the processor (i.e., a given processor microarchitecture evolves over time due to failing subsystems that are compensated). Ontological uncertainty, therefore, needs to be addressed adequately by the resulting system. Consequently, this means that the models representing the system must embody mechanisms to accept, control, and handle ontological uncertainty, e.g., by offering underspecification mechanisms within the model. Thus, instead of reasoning over uncertainty in models, we might "syntactify" uncertainty concepts and embed them within the design models. The models become more complicated, but also more adequate to serve as better formal notions of refinement and fidelity.

Reasoning explicitly on the representativity of a model is of utmost importance to consider the accuracy of the evaluation result of such a model (e.g., simulation). Moreover, capturing explicitly the representativity through fidelity and underspecification is also required to reason over the *substitutability* of one model by another. This is useful in many contexts, such as in a multi-fidelity simulation, or when there is a need for trading off accuracy and performance, but also when replacing a cyber-physical system (CPS) component in a larger, long-living CPS system (e.g., a car or an airplane).

Finally, representativity is also important in the context of model composition scenarios. In that context, model refinement is very important to define composed model/component structures and identify reusable model assets from libraries to insert them in these architectures. This will allow a lively MDE approach to become feasible.

SoSyM welcomes more contributions in this direction because we believe this is strongly needed to make MDE successful.

Funding Open Access funding enabled and organized by Projekt DEAL.