MMMDE: Workshop on Mathematical Models for Model-Driven Engineering

(*Read MMMDE as "triple em dee e"*)

Zinovy Diskin[*], Vadim Zaytsev[†], Rick Salay[‡], Bernhard Schätz[§],
[*]Department of Computing and Software, McMaster University, Canada,
[*]Department of Electrical and Computer Engineering, the University of Waterloo, Canada,
[†]Instituut voor Informatica, FNWI, Universiteit van Amsterdam, The Netherlands,
[‡]Department of Computer Science, University of Toronto, Canada,
[§]Institut für Informatik, Technische Universität München, Germany,
[§]Software & Systems Engineering Department, fortiss GmbH, Germany,
[*]diskinz@mcmaster.ca, [†]vadim@grammarware.net, [‡]rsalay@cs.toronto.edu, [§]schaetz@informatik.tu-muenchen.de

***Abstract*—Software engineering (SE) strives to learn from matured engineering disciplines, such as mechanical and electrical engineering (*physical engineering*, PE), and MDE is an essential step in this direction. Mathematical models are fundamental for PE, but should it be so for SE? What are similarities and differences in the development and use of mathematical models in SE vs. PE? How can SE and MDE benefit from a better understanding of these similarities and differences? These questions become even more challenging when we recognize that mathematical modelling and formalisation are not identical (although closely related), and the abundance of formal models in SE may actually hide the lack of mathematical models with all its negative (but perhaps negligible?) consequences.**

Questions above are seldom discussed in the MDE literature, but we believe they deserve special attention. The MMMDE Workshop aims at gathering together MDE experts who are concerned with developing mathematical foundations for MDE, understanding the role of mathematical modelling in engineering in general and SE in particular, and with relating these general thoughts to practical MDE problems. We want to "test the waters", and try to solidify broadly formulated concerns outlined above into several well-focused research questions or directions. For details consult http://mmmde.github.io.

I. Motivation and Scope

There are essential similarities and essential differences between PE and SE, which essentially influence the value and roles of mathematical models. Below we present several observations about the subject (one subsection per observation), and the corresponding questions that we think are useful to discuss at the workshop.

Observation 1: Requirements

While mathematical modelling is a commonplace practice in PE, SE can (arguably) survive without mathematical models. Indeed, building a bridge or a car without an a priori analysis would be too costly, and hence their mathematical modelling is a must. In contrast, testing and debugging can (pretend to) replace modelling, analysis and other mathematical means of providing correct-by-construction software. This leads to the following major question: *what should a mathematical model provide to be accepted and used in SE?*

Observation 2: Formal vs. Mathematical

Discrete domains (the main subject matter of SE, if we forget about cyber-physical systems) are much better amenable to formalisation than continuous ones (in PE); moreover, in design models, the object to be formalised (code) is itself a formal object. This may lead to an abundance of formal models in a typical SE process, but formal models are not necessarily mathematical models. An assembly program or byte code are formal objects, but they need mathematical models of their intended semantics if we are interested in understanding *what* these programs do on the specification level beyond sequences of machine instructions. Similarly, a Java program is a formal (or almost formal) but not a mathematical object if we are looking for a higher-level specification beyond sequences of programming operators. In a sense, the very non-trivial activity of *reverse engineering* can be seen as a transition from the formal to the mathematical world. On the other hand, quasi-mathematical but semi-formal models are widely used in PE. Of course, true mathematical models are themselves formal, but the benefits they provide often go beyond formalisation as such: specification and design patterns, consistent conceptual frameworks and terminological and notational frameworks based on them, ways of thinking about and understanding the domain are typical implications of mathematical rather than just formal modelling.

The differences between mathematical and formal are not always well understood in SE; the latter often lacks mathematical models but the problem is not recognised as mathematical models are substituted by formal ones.

Observation 3: Bridging

An important facet of the issue is how to bridge the (enormous) gap between engineering thinking and intuition on the one side, and mathematical formalisms on the other. Classical PE models like a pendulum and a mass hanging from a spring in mechanics, a bending beam and a shell in mechanics of materials, an RCL-contour or the entire electrohydraulic analogy in electricity are intuitive and manageable by an average PE-engineer, but each of them encapsulates fairly complex mathematical and bulky formal structures. These models can be seen as simple interfaces to the underlying mathematical structures. In fact, the transition from the engineering domain to its formalisation is mediated by a whole chain of models provided by different disciplines: from Engineering theories in Mechanics/Electricity to General Physics to Theoretical Physics to Mathematical Physics (only here we are in the realm of mathematics) to Numerical Methods, which employ their own terminological, conceptual, and mathematical frameworks.

We have somewhat similar "physical" models bridging the gap in SE too: automata of different types, Petri nets, grammars and rewriting systems of different types, different sorts of tables providing interfaces to different logics, control flow and data flow nets, dependence graphs, and also class diagrams, ER-diagrams, statecharts and message sequence charts constitute the "golden modelling fund" of SE. There are, however, essential differences between engineering models in PE and SE. Some of the SE-models above are indeed engineering interfaces to the underlying mathematics, others still lack an agreed formal semantics. Also, it appears that a typical MDE chain from engineering to mathematical is shorter and seemingly more primitive than in PE: from the engineering domain to an engineering model to its formalisation in code.

Observation 4: Connections

Models in PE are integrated into systems—model libraries, in which each model has its precisely defined goals and application conditions. It seems that models in SE are less integrated, and their applicability is less accurately specified. Software engineers often tend to master one modelling framework and supporting tooling, and apply it as broadly as possible without too much worrying about the very applicability of the framework to a given domain problem. Models in PE are integrated into "forests", whereas models in SE look more like isolated trees. For example, work on behaviour modelling done with Petri nets is not well related to work on behaviour modelling done with statecharts, the same for structural modelling with ER-diagrams, class diagrams and ontologies. Even worse is that the isolationism of models gives rise to the isolationism of the respective communities, and the problem becomes unmanageable.

Observation 5: MDE Paradox

Model-driven methods and techniques are centred around the foundational concept of usefulness of models in general. Models of behaviour, models of structure, models of communication, as well as many others, were made first-class citizens in the software development process by the MDE ideology. Many researchers and practitioners learnt to appreciate abstraction through observations of the technical benefits it provides in the context of model-based requirement engineering, software development, testing, etc. However, even though this appreciation of abstraction was widely promoted and praised, the use of actual mathematical models never received any significant attention. Instead of bringing additional mathematical rigour to software development, the opposite happened: the level of mathematical correctness of modelling theories and especially in models as such, dropped. MDE enabled us to create, manipulate and use many kinds of models with extreme ease and unreasonable effectiveness[1], which also means many models are incorrect, inconsistent, partly manually adjusted, etc, and many modelling theories have limitations that are left unexplored or unknown to general public.

Observation 6: Categorical Thinking

Category theory (CT) is an obvious candidate to mediate the transition from a structurally complex engi-

[1]cf. Wigner, *The Unreasonable Effectiveness of Mathematics in the Natural Sciences*, 1960.

neering domain to its mathematical models to formalisation. Moreover, categorical modelling by its very nature tends to integrate mathematical models into a consistent mathematical framework. However, CT is not a part of a common SE curriculum, its methods are (although quite natural but) unusual, and there are no good textbooks suitable for a software developer; the result is that CT is often considered as being excessively complicated.

These observations show enough differences in the value and application of mathematical models in PE and SE to justify (re)thinking and discussing the interaction of mathematical modelling and MDE and SE. Such a discussion seems need greater clarity in understanding the core issues than we have today — hence, this workshop.

II. Objectives and the Intended Audience

Questions above are seldom discussed in the MDE literature, but we believe they deserve a special attention and discussion. The MMMDE Workshop aims at gathering together MDE experts who are concerned with developing mathematical foundations for MDE, understanding the role of mathematical modelling in engineering in general and SE in particular, and with relating these general thoughts to practical MDE problems. We want to "test the waters" and try to solidify broadly formulated concerns above into several well-focused research questions or directions, and perhaps make them accessible to the community via a publication. Perhaps, we could continue the workshop with the next edition of MoDELS in a more traditional setting with paper submission and reviewing process.

The intended audience is assumed encompassing three main groups.

1) MDE researchers and practitioners with an affinity to mathematical and formal methods.
2) Applied mathematicians or computer scientists applying (or wishing to apply) their research skills to MDE and having trouble in bridging the conceptual gap.
3) SE/MDE practitioners who seek help in mathematical techniques but do not want to pursue mastery in the underlying theories.

III. Tentative Schedule

The workshop will consist of three parts: the keynote (first session, 9:00–10:30); the invited talks (11:00–12:30 and 14:00–15:30) and the panel (16:00–18:00), the detailed tentative schedule is found below on section III.

The topic of the panel is: "*What should a mathematical model provide to be accepted and used in SE?*". The panel will be followed by a conclusive general discussion.

Time	**Presenter**	**Topic**
9:00–9:30	Organisers	Introduction and goals
9:30–10:30	Tom Maibaum	TBA
11:00-11:30	TBA	TBA
11:30-12:00	TBA	TBA
12:00-12:30	TBA	TBA
14:00-14:30	TBA	TBA
14:30-15:00	TBA	TBA
15:00-15:30	TBA	TBA
16:00-18:00	Panel	What should a mathematical model provide to be accepted and used in SE?

TABLE I. Tentative schedule of MMMDE 2015.