MMMDE: Workshop on Mathematical Models for Model-Driven Engineering

http://mmmde.github.io

Workshop Proposal

(*Read MMMDE as “triple em dee e”*)

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

***Abstract*—Software engineering (SE) strives to learn from matured engineering disciplines, such as mechanical and electrical engineering (below *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 a 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. See http://mmmde.github.io/ for details.

I. Objectives and Scope

A. Motivation and Scope

Software engineering (SE) strives to learn from lessons of matured engineering disciplines, such as mechanical and electrical engineering (later *physical engineering*, PE), and MDE is a bold step in this direction. Mathematical models are fundamental for PE, but does it hold for SE as well? What are similarities and differences in the development and use of mathematical models in SE as opposed to PE? How can SE and MDE benefit from a better understanding of these similarities and differences? And how could we reduce the differences when it is really needed? These are general questions underlying the idea of MMMDE. Below we provide a more detailed “questioner” for the workshop participants.

Our *first observation* is that mathematical modelling is a commonplace practice in PE, whereas SE can seemingly survive without mathematical models. Indeed, building a bridge or a car without an a priori analysis would be too costly, and hence their mathematical modeling is a must. In contrast, testing and debugging can replace modelling, analysis and other mathematical means of providing correct-by-construction software. Then the question arises: what a model should really provide to be accepted and used in SE?

The *second observation* is that PE has carefully developed families of models bridging the (enormous) gap between engineering thinking and intuition on the one side, and mathematical formalisms on the other. A pendulum, a mass hanging from a spring, an RCL-contour, or the entire electrohydraulic analogy are simple physical models easily manageable by an average PE-engineer, but each of them encapsulates quite a bit of the underlying mathematics and formalities. Moreover, these simple models are easily composable into networks that can be analysed in standard ways. We have somewhat similar “physical” models bridging the gap in SE too: finite state machines, pushdown automata, Petri nets, Lindenmayer systems, Thue rewriting systems, control flow and data flow nets, and also class diagrams, ER-diagrams, statecharts and message sequence charts constitute the “golden modelling fund” of SE.

There are, however, essential differences. Some of the SE-models above are indeed engineering interfaces to the underlying mathematics, others still lack an agreed formal semantics and often exist in many different versions. Composability of different models can also be an issue. Actually SE models are often quite “asocial” and isolated: deep work on behaviour modelling done with Petri nets is not related to deep work on behaviour modelling done with statecharts, the same for structural modelling with ER-diagrams, class diagrams and ontologies. In general, an experienced modeller could easily find a pair of popular formalisms intended for modeling of basically the same domain, but their close relationships are rarely explicated in the literature. Models in PE are integrated into “forests”, whereas models in SE are isolated trees. Even worse is that the isolationism of models gives rise to the isolationism of the respective communities, and the problem becomes unmanageable.

The model isolationism above could be called *horizontal*. Yet another concern is the *vertical* model isolationism, that is, a discrete jump rather than continuously changing the level of abstraction of SE-models. Indeed, in PE, 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. The opposite direction is also alive and active, when theorists use mathematical instruments to build models to be used by applied specialists, who in turn use their own instruments to build reality-geared tools for the practitioners. It appears that a typical MDE chain is shorter and seemingly more primitive: from the engineering domain to a model to its formalisation.

Our *third observation* is that 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 a chunk of byte code are formal objects, but they can hardly be considered mathematical objects if their semantics is neither well understood nor defined. In a sense, the highly nontrivial activity of reverse engineering can be seen as a transition from the formal to the mathematical world. On the other hand, mathematical but semiformal models are widely used in PE. Of course, mathematical models are often 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 MDE; the latter often lacks mathematical models but the problem is not recognised as mathematical models are substituted by formal ones. Particularly, the lack of mathematical models in the modelling chain contributes to the vertical isolationism described above.

The fuzziness (not to say obscurity) of general foundations (or the lack of them) for mathematical modelling in SE hinders solution of multiple small concrete problems. [ZD: the phrasing should be reworked] Can we evaluate the quality and usefulness of formal models by some semi-formal criteria in order to say beforehand whether a newly proposed formalism has any potential profitable use, or is just of pure academic interest? What patterns can we detect in software artefacts, models, mappings and processes, and how can we improve and reuse them? How can the arrow thinking of category theory be adapted as a useful instrument easy enough to apply for an average CS/SE/MDE researcher? We think that approaching these concrete problems needs greater clarity in understanding the interaction of mathematical modelling and SE than we have today — hence, this workshop.

B. *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 make them accessible top 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.

C. *Relevance and context*

A similarly focused workshop was organised in summer 2014 for the Network for Engineering of Complex Software-Intensive Systems for Automotive Systems (NECSIS)[1], which encompasses eight research institutions across Canada and three industrial partners (GM Canada, IBM Canada, and Malina Software). The workshop was organized by Zinovy Diskin with an active participation of Bran Selic and Tom Maibaum. It turned out successful and seemingly deserving its extension beyond NECSIS, ultimately leading to the present proposal.

There exist several conferences primarily devoted to applications of formal methods to computer science and software engineering: FM, IFM, TASE, ICFEM, SEFM, FSEN; and some that focus on formal proofs, verification and validation: TAP, CAV, TACAS, SPIN, ISSTA. Normally, however, these forums do not specially distinguish between mathematical and formal models, and do not specially address the interaction of mathematical modelling and engineering, which we think is important for MDE. There has been another similarly themed edition of SFM (School on Formal Methods for the Design of Computer, Communication and Software Systems) which led to publishing “Formal Methods for Model-Driven Engineering” in 2012 [1].

We provide an illustrative list of papers that address the problems similar or close to those in the focus of the workshop (of course, the list is in no way comprehensive):

- the use of specification patterns in MDE [4], [9]
- assigning formal semantics to MDE processes and artefacts [8], [10], [11], [13]

[1]http://www.apc-pac.ca/About-Renseignements/Project-Project_eng.asp?ID=6

- verifying model transformations and proving their properties [3], [14]
- using abstract approaches to find and model similarities in methods and techniques [15], [16]
- using category theory in MDE [5]
- integrating formal methods with MDE [6]
- systematic reviews on methods developed in the formal community with their usefulness for software engineers [7], [12]
- similarity and differences between mathematical modelling and programming [2]

II. Organisation Details

A. Organisers

Zinovy Diskin is Senior Research Scientist with NECSIS (see the beginning of Section C above). He is cross-appointed as Research Associate with McMaster and the University of Waterloo. His area of expertise is mathematical models for MDE, particularly, metamodelling, multimodelling and model management. He worked as a mechanical engineer and database designer in Latvia, business analyst and consultant in the US, and as a researcher and mentor in Canada. He has served in the PCs of MoDELS14, ECMFA14–13, SLE14, MODELSWARD14–13, BX14–12, and is an external reviewer for SoSyM, FAOC, SOCP, DKE, MSCS and several other journals. In summer 2014, he organised a one-day NECSIS workshop on MMMDE.

Bernhard Schätz received his Ph.D. and Habilitation degree in Informatics from the Technische Unversität München. At the fortiss Transfer Institute associated with the Technische Universität München, he leads the research department "Software & Systems Engineering" with the fields of Analysis and Design of Dependable Systems, Optimised Design Space Exploration, Model-Based Engineering Tools, with Smart Grid, Automotive, and Automation as fields of application. Besides his scientific activities, he is Lecturer at the Technische Unversität München, co-founder and member of the advisory board of the Validas AG, and works as a consultant (including BMW, Bosch, Eurocopter) in the field of Software and Systems Engineering.

Vadim Zaytsev is a software language engineer with background in applied mathematics and interest in bridging technological spaces, software artefact quality, transformational techniques and grammars in a broad sense; creator and maintainer of the Grammar Zoo, http://slebok.github.io/zoo; OOPSLE workshop co-organiser (2013–15); Programme Chair at WCN (2011–12) and SATToSE (2014); Tool Track Chair at WCRE (2013); Hackathon Chair at SoTeSoLa (2012) and SATToSE (2013); Publicity or Social Media Chair at STAF (2015), MoDELS (2013), SLE (2011), GTTSE (2009–2015); weekly PEM Colloquium organiser (2012–13); PC member for GTTSE (2015), SATToSE (2015), ICSME ERA (2015), SANER ERA (2015), CSMR-WCRE ERA (2014), DADA (2014), SCAM (2010–15), LDTA (2012), SQM (2012–15), DYLA (2010), WCN (2011–14), ACM SRC (2013), XM (2013–14); an external reviewer for 20+ other venues.

B. Programme Committee

The programme committee will be composed (if the workshop is accepted) from several experts with whom we have already discussed the ideas underlying the workshop. This list includes Alfonso Pierantonio (L'Aquila), Antonio Vallecillo (Málaga), Bran Selic (Malina Software), Harald König (Hannover), Jacques Carette (McMaster), Jürgen Dingel (Queen's), Krzysztof Czarnecki (Waterloo), Martin Gogolla (Bremen), Perdita Stevens (Edinburgh), Ralf Lämmel (Koblenz-Landau), and Richard Paige (York). Several members of this list combine academic and industrial experience, and can thus provide an objective and multidimensional consideration of the subject. The role of the programme committee will be to help to distil an efficient and attractive programme, to disseminate information about the workshop, and to facilitate participation.

Possible Merge. Our main motivation for taking initiative in organising this workshop is that there is no similar event, but there should be. In case this proposal is assessed as uninteresting, we would rather *not* merge with an unrelated workshop for size considerations.

III. Workshop Format

We intend to run the workshop in the following format. The first half of the workshop (before lunch) will consists of a 45 min keynote followed by 4–5 interactive 0.5 hour invited talks. We are proud to announce that Tom Maibaum has confirmed his willingness to deliver a keynote if the workshop is accepted. The other speakers will be recruited from the PC list above depending on time and other constraints (our potential speakers may be involved in other workshops, and time clashes are possible), or by suggestions from the PC members. We intend to assign each prospective speaker some mild requirements on what topic we would like her/him to address.

The second half of the workshop (after lunch) will feature two panels followed by a conclusive general discussion.

- Panel 1: *"What is bad with a bad mathematical model?"*
- Panel 2: *"From trees to forests: How to correct asocial life of models and modeling communities?"*

The PC also provides us with a pool of panelists, again depending on time constraints.

Equipment needed: the classic academic setup with a whiteboard/chalkboard; a beamer; a microphone/sound system.

References

[1] M. Bernardo, V. Cortellessa, and A. Pierantonio, Eds., *Formal Methods for Model-Driven Engineering. 12th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM 2012. Advanced Lectures*, ser. LNCS, vol. 7320. Springer, 2012.

[2] D. M. Berry, "The Essential Similarity and Differences Between Mathematical Modeling and Programming," *Science of Computer Programming*, vol. 78, no. 9, pp. 1208–1211, 2013.

[3] F. Büttner, M. Egea, J. Cabot, and M. Gogolla, "Verification of ATL Transformations Using Transformation Models and Model Finders," in *Proceedings of the 14th International Conference on Formal Engineering Methods: Formal Methods and Software Engineering*, ser. ICFEM'12. Springer, 2012, pp. 198–213.

[4] Z. Diskin, S. Kokaly, and T. Maibaum, "Mapping-Aware Megamodeling: Design Patterns and Laws," in *Proceedings of the 6th International Conference on Software Language Engineering*, ser. LNCS, M. Erwig, R. F. Paige, and E. Van Wyk, Eds., vol. 8225. Springer, 2013, pp. 322–343.

[5] Z. Diskin and T. S. E. Maibaum, "Category Theory and Model-Driven Engineering: From Formal Semantics to Design Patterns and Beyond," in *Proceedings of the 7th Workshop on Applied and Computational Category Theory, ACCAT 2012*, ser. EPTCS, U. Golas and T. Soboll, Eds., vol. 93, 2012, pp. 1–21.

[6] A. Gargantini, E. Riccobene, and P. Scandurra, "Integrating formal methods with model-driven engineering," in *Fourth International Conference on Software Engineering Advances, ICSEA 2009*, Sep. 2009, pp. 86–92.

[7] C. A. Gonzlez and J. Cabot, "Formal Verification of Static Software Models in MDE: A Systematic Review," *Information and Software Technology*, vol. 56, no. 8, pp. 821–838, 2014.

[8] E. Guerra and J. de Lara, "An Algebraic Semantics for QVT-Relations Check-only Transformations," *Fundamenta Informaticae*, vol. 114, no. 1, pp. 73–101, 2012.

[9] E. Guerra, J. de Lara, and F. Orejas, "Inter-modelling with Patterns," *Software and System Modeling*, vol. 12, no. 1, pp. 145–174, 2013.

[10] L. Hamann and M. Gogolla, "Endogenous Metamodeling Semantics for Structural UML 2 Concepts," in *Proceedings of the 16th International Conference on Model-Driven Engineering Languages and Systems*, ser. LNCS, A. Moreira, B. Schätz, J. Gray, A. Vallecillo, and P. J. Clarke, Eds., vol. 8107. Springer, 2013, pp. 488–504.

[11] G. Simko, D. Lindecker, T. Levendovszky, S. Neema, and J. Sztipanovits, "Specification of Cyber-Physical Components with Formal Semantics — Integration and Composition," in *Proceedings of the 16th International Conference on Model-Driven Engineering Languages and Systems*, ser. LNCS, A. Moreira, B. Schätz, J. Gray, A. Vallecillo, and P. J. Clarke, Eds., vol. 8107. Springer, 2013, pp. 471–487.

[12] A. Stevenson and J. R. Cordy, "A Survey of Grammatical Inference in Software Engineering," *Science of Computer Programming*, vol. 96, pp. 444–459, 2014.

[13] H. Störrle, "Semantics and Verification of Data Flow in UML 2.0 Activities," *ENTCS*, vol. 127, no. 4, pp. 35–52, 2005.

[14] J. Troya and A. Vallecillo, "A Rewriting Logic Semantics for ATL," *Journal of Object Technology*, vol. 10, pp. 5:1–29, 2011. [Online]. Available: http://www.jot.fm/contents/issue_2011_01/article5.html

[15] V. Zaytsev, "Formal Foundations for Semi-parsing," in *Proceedings of the IEEE Conference on Software Maintenance, Reengineering and Reverse Engineering, Early Research Achievements Track, CSMR-WCRE 2014 ERA*, S. Demeyer, D. Binkley, and F. Ricca, Eds. IEEE, Feb. 2014, pp. 313–317.

[16] V. Zaytsev and A. H. Bagge, "Parsing in a Broad Sense," in *Proceedings of the 17th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2014)*, ser. LNCS, J. Dingel, W. Schulte, I. Ramos, S. Abrahão, and E. Insfran, Eds., vol. 8767. Springer, Oct. 2014, pp. 50–67.

Appendix A
Call for Participation (Draft)

MDE has reached a certain level of maturity, but still seems lacking a sound underlying foundational framework based on mathematics. Creation of such a framework needs mathematical models of modelling itself and, perhaps, some rethinking of the role and nature of software models (as it often happens when mathematics is applied). We need a better understanding of that mysterious process that converts the ever changing and contra-formal real world into its semi-formal models later transformed into formal models and finally to code.

The value and role of modelling are well understood in matured engineering disciplines such as mechanical and electrical engineering (MEE), but the case of software engineering (SE) is dramatically different. In MEE, mathematical modelling is a must, as building a bridge or a car without an a priori analysis would be too costly. In contrast, SE can live without mathematical models, which may create problems, even significant ones, but is not, in general, a life-threatening issue for SE: testing and debugging can replace correct-by-construction modelling and analysis. Then the question arises what a model should really provide to be accepted and used by software engineers or by SE-researchers.

Another important distinction between MEE and SE is that discrete domains are much better amenable to formalisation than continuous ones, and, 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 necessary mathematical models! An assembly program or byte code are formal objects, but they can hardly be considered mathematical objects if their semantics is not well understood. In a sense, the highly non-trivial activity of reverse engineering can be seen as a transition from formal to mathematical. On the other hand, mathematical but semi-formal models are widely used in MEE. Of course, mathematical models are often themselves formal, but the benefits they provide often go beyond formalisation as such: specification and design patterns, consistent conceptual frameworks and based on them terminological and notational frameworks, 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 MDE; the latter often lacks mathematical models but the problem is not recognised as mathematical models are substituted by formal ones.

Questions above are seldom discussed in the MDE literature, but we believe deserve a special attention and discussion. The MMMDE Workshop aims at gathering together MDE experts who are concerned with developing mathematical foundations for MDE, and are willing to share their thoughts about the issue. The first half of the workshop will consist of several invited presentations and the second half will feature two panels and a general discussion.

[NB: We plan to extend the CFP with some ideas and details from the Section A of the Proposal.]