



Software and Systems Modeling journal celebrates its silver anniversary-a look into the future of modeling

Marsha Chechik¹ · Benoit Combemale² · Bernhard Rumpe³ · Martin Schindler³

Published online: 23 September 2026
© The Author(s) 2026

Abstract

SoSyM has turned 25. You can even see that on the cover of this dedicated issue. The 25th anniversary of *Software and Systems Modeling* provides an opportunity to look back—not only at the development of our journal, but also at the evolution of the entire scientific field of software and systems modeling. Or we could look into the future of the journal and thus the field of software and systems modeling. We decided to do both: a short view into the past and a larger view into the future.

1 Twenty-five years of SoSyM and the evolution of software and systems modeling

When SoSyM was founded, explicit software and systems modeling activities had started to become a relevant instrument for managing the growing complexity of software-intensive systems. The emergence of the Unified Modeling Language in the 1990s, model-driven development, and increasingly sophisticated modeling tools had created a need for a dedicated scientific forum. The <<UML>> conference (later renamed MODELS) was established to provide such a forum in 1999: a place in which foundational insights, innovative modeling languages, engineering methods, tools, and experiences from industrial application could be brought together and critically discussed. It soon became clear that an associated journal, namely SoSyM, was needed to sustain these discussions and the results of the scientific activities. So, in autumn 2001, SoSyM was officially announced at the MODELS conference in Toronto and one year later, the first

issue of SoSyM was presented at the MODELS conference in Dresden.

Throughout its history, SoSyM has reflected and helped shape the broadening understanding of modeling software and associated context, such as processes, business, workflows, etc. Models were initially often regarded primarily as abstractions used to document, communicate, and design software systems. This role remains important, but the field has advanced considerably. Models have increasingly become central engineering artifacts that can be analyzed, validated, transformed, executed, simulated, and used to generate substantial parts of an implementation. Models have thus evolved from supporting software development to becoming an integral part of systematic software and systems engineering, at least when they are used at all. As a big “competitor” to the model-based development processes, agile development was pretty successful in preventing the extensive use of explicit models for quite a range of development settings. If modeling tools had been better developed and more efficient in modeling abstract structures and behavior and then generating code from there, then agility and modeling would fit well together. This hasn’t happened as a mainstream methodology so far.

The scientific discussions about modeling have changed over time. This evolution is also visible in the changing scope of the field. Modeling research has progressed from general-purpose notations, such as UML, toward a rich landscape of domain-specific modeling languages, metamodeling techniques, language workbenches, model transformations, and compositional methods. But many issues are still unresolved. For practical purposes, it is still unclear whether a textual, a diagrammatic, or a tabular notation is better for reading or

✉ Bernhard Rumpe
bernhard.rumpe@sosym.org

Marsha Chechik
marsha.chechik@sosym.org

Benoit Combemale
benoit.combemale@sosym.org

Martin Schindler
martin.schindler@sosym.org

¹ University of Toronto, Toronto, ON, Canada

² University of Rennes, Rennes, France

³ RWTH Aachen University, Aachen, Germany

manipulating models. Related to that is the question of how to represent models either in machine-processable UTF-text, as a big binary archive, or in a database. All three approaches have their ups and downs, which is pretty visible in the SysML discussion going on even right now.

While formal semantics and validation techniques have strengthened the precision of modeling languages, there is still a group of people that argue that modeling should be somewhat informal, while others argue that models are just a diagrammatic form of precise specification. However, the above-mentioned tools and techniques can only be developed when the semantics of a modeling language is precise enough so that tools are coinciding in their understanding and thus manipulation of a model. In the early days of SoSyM, we formed pUML, a group of then younger scientists who were pretty interested in using UML as a formal specification technique to gain solid insights into the systems under development (see Kevin Lano's article and the special section dedicated to Bran Selic in this issue). Although the UML, for example, has never become a precisely and formally defined semantics (in terms of "meaning"), the semantics was at least consolidated enough so that people could talk to each other using the same language and many models could be exchanged between semantic-aware tools, like generators and analysis tools. This becomes quite relevant nowadays when agentic AI can talk to humans and explain its design decisions, for example, using UML.

Models in the last 25 years have become connected with requirements engineering, software architecture, testing, verification, deployment, and operation. And beyond individual projects, variability modeling has supported the systematic development of product lines and software ecosystems. Consequently, modeling today addresses not only individual development activities but, increasingly, the entire lifecycle of software-intensive systems.

At the same time, the systems being modeled have changed substantially. Software has become deeply embedded in physical, organizational, and societal contexts. Cyber-physical systems, cloud platforms, autonomous systems, digital twins, and highly interconnected ecosystems require engineers to combine structural, behavioral, physical, and operational perspectives. Modeling has responded by extending its concepts, languages, and methods beyond traditional software boundaries. The integration of heterogeneous models and the coordination of multiple viewpoints have therefore become recurring challenges—and relevant subjects of research published in SoSyM. While the journal name from its beginning indicates that it considers papers about modeling "software" as well as "systems," SoSyM today is mainly a software-related journal; while some papers report scientific analysis on SysML, its application to deeply embedded physical systems remains relatively little.

Over these 25 years, the journal has accompanied the field through phases of enthusiasm, consolidation, critical reflection, and also renewal. We are proud that we could offer space for theoretical foundations as well as practical methods, tools, empirical studies, and reports on industrial experience. Modeling ultimately derives its relevance from its ability to improve the understanding, construction, analysis, and evolution of real systems in practical development projects. Conversely, sustainable progress in practice depends on sound concepts, precise terminology, explicit semantics, smart analytical, transformational and synthesizing algorithms (i.e., conditions checking, code generation, etc.), and analytical or evidence-based evaluation. SoSyM has continually sought to connect these perspectives.

We believe that SoSyM and with it the authors, reviewers, editors, and all other stakeholders play a substantial role in establishing modeling and, especially, the use of explicit modeling languages, as a scientific discipline with its own body of knowledge, research methods, technological foundations, and community. SoSyM documents many of the debates that have shaped the field: the appropriate level of abstraction, the semantics of models, the relationship between models and programs, the role of transformations and code generation, the still missing usability of modeling tools, to name a few. These discussions have helped the community refine its claims, identify limitations, and develop more mature approaches. And many approaches nowadays are also used outside the community for software development and for systems development. For example, LowCode approaches have benefited a lot, allowing less experienced coders to develop mature complex software.

SoSyM's first 25 years are therefore closely intertwined with the history of modern modeling. This accomplishment rests on the sustained contributions of all our authors, reviewers, editors, readers, tool developers, industrial partners, and the wider modeling community. Their engagement has enabled SoSyM to remain a lively forum for a field that continuously adapts while preserving its central objective: to master the complexity of products and their development.

Because it would be too much to name everyone, we want to name some people who have helped to make SoSyM such a success over the years (which includes us as Editors-In-Chief; in the order of appearance):

- Robert France, Editor-in-chief and co-founder from 2002 until his passing in 2015
- Bernhard Rumpe, Editor-in-chief and co-founder from 2002
- Geri Georg, Assistant Editor from 2003 to 2016
- Martin Schindler, Assistant editor from 2004
- Marsha Chechik, Associate Editor-in-Chief from 2014, Editor-in-Chief from 2025
- Jean-Marc Jezequel, Associate Editor-in-Chief from 2015

- Martin Gogolla, Associate Editor-in-Chief from 2015 to 2022
- Jeff Gray, Editor-in-chief, from 2015 to 2026
- Tao Yue, China SoSyM Ambassador from 2019
- Vinay Kulkarni, India SoSyM Ambassador from 2019
- Benoit Combemale, Editor-in-Chief from 2022
- Stéphanie Challita, Assistant editor from 2022
- Jordi Cabot, Editor for the Tool Paper Segment since 2024
- Abel Gómez, Editor of the Tool Papers Segment 2024
- Sophia Jit, Assistant editor since 2025

We want to thank the SoSyM Advisory Board Egidio Astesiano, Jean Bezivin, Grady Booch, Manfred Broy, Martin Gogolla, Gregor Engels, David Harel, Michael Jackson, Tom Maibaum, Pierre-Alain Muller, Wolfgang Reisig, and Bran Selic.

We also want to thank all the current and former editors who over the years continuously helped in getting all the papers handled: Silvia Abrahao, Perry Alexander, Shaukat Ali, Daniel Amyot, Egidio Astesiano, Colin Atkinson, Joanne Atlee, Franck Barbier, Benoit Baudry, Nelly Bencomo, Jean Bezivin, James Bieman, Gordon Blair, Grady Booch, Francis Bordeleau, Ruth Breu, Lionel Briand, Ed Brinksmma, Manfred Broy, Jean-Michel Bruel, Didier Buchs, Loli Burgueño, Jordi Cabot, Alessandra Cavarra, Marsha Chechik, Betty Cheng, Antonio Cicchetti, Tony Clark, Peter Clarke, Philippe Collet, Steve Cook, Krzysztof Czarnecki, Juan De Lara, Davide di Ruscio, Jürgen Dingel, Heiko Dörr, Alexander Egyed, Gregor Engels, Andy Evans, Ulrich Frank, David Garlan, Sebastien Gerard, Sudipto Ghosh, Holger Giese, Martin Glinz, Martin Gogolla, Hassan Gomaa, Abel Gómez, Jeff Gray, Esther Guerra, David Harel, Alan Hartman, Oystein Haugen, Mats Heimdahl, Constance Heitmeyer, Brian Henderson-Sellers, Zhenjiang Hu, Heinrich Hußmann, Ludovico Iovino, Michael Jackson, Jean-Marc Jezequel, Gerti Kappel, Gabor Karsai, Stuart Kent, Jörg Kienzle, Cris Kobryn, Dimitris Kolovos, Ingolf Krüger, Thomas Kühne, Leen Lambers, Kevin Lano, Yves Le Traon, Gary Leavens, Timothy Lethbridge, Robyn Lutz, Tom Maibaum, Hong Mei, Jan Mendling, Judith Michael, Dragan Milicev, Ana Moreira, Pierre-Alain Muller, Oscar Nierstrasz, James Odell, Jefferson Offutt, Richard Paige, Dorina Petriu, Rob Pettit, Alfonso Pierantonio, Alexander Pretschner, Iris Reinhartz-Berger, Wolfgang Reisig, Jan Ringert, Adrian Rutle, Houari Sahraoui, Jesús Sánchez Cuadrado, Ina Schäfer, Bernhard Schätz, August Scheer, Andy Schürr, Bran Selic, Jim Steel, Perdita Stevens, Ketil Stølen, Gabriele Taentzer, Matthias Tichy, Massimo Tisi, Juha-Pekka Tolvanen, Antonio Vallecillo, Wil van der Aalst, Hans Vangheluwe, Dániel Varró, Jos Warmer, Andrzej Wasowski, Barbara Weber, Jon Whittle, Roel Wieringa, Manuel Wimmer, Alan Cameron Wills, Jim Woodcock, Andreas Wortmann, Tao Yue, and Steffen Zschaler.

We want to thank all involved people, and we are grateful that we have a stable yet active community of many long-lasting editors who regularly meet at annual conferences, exchanging new scientific developments and creating ideas. But we also want to give a special thanks to Jean Bezivin and Pierre-Alain Muller, who were instrumental in launching the journal back in 1999 and especially to Jean Bezivin who had the initial idea to create SoSyM.

Thanks to all!

We could talk a lot about the current and past organization of SoSyM:

- We could mention that we started with special issues and as the issues themselves grew, it was useful to map them to special sections.
- We could mention that articles in special sections are cited at least as well as regular papers.
- We could mention that we started a tool division to address the continuing problem that we had a lot of foundational insights that had little industrial impact because of lack of useful tools.
- We could enumerate the number of authors, publications, pages, etc., and would find a large corpus of SoSyM-related articles.

This is, however, already done in the annual state-of-the-art practice editorials, where this information is regularly integrated and the next one is coming up.

But we would like to let the community know that Vadim Zaytsev has created CfpBOK for SoSyM, a system that allows people to navigate through the SoSyM articles of the entire 25-year history (including links to the sources). It can be found at <https://cfpbok.github.io/sosym/> and contains a lot of interesting and also aggregated information; for example, we have currently 2885 authors in 1445 publications using 3468 distinct keywords (August 2026). Based on the tags we provide to classify articles, we have the following seven tags used the most: GPML (320), Verification (174), Model Transformation (151), BPM (134), Embedded (117), Metamodeling (106), and DSML (101).

2 The future of software and systems modeling

Future is hard to predict. And 25 years is a long time. It is easier and potentially more helpful to only look into the next ten years. Of course, it is also helpful to look at other predictions that have been made, potentially by us within the SoSyM journal, over the last few years and how well they have been aging. In particular, we did a number of editorials where we discussed the current state, what needs to be done, and what would be an appropriate vision in various

sub-domains. We especially identify the following 20 editorials (out of a longer list that we originally wanted to mention because we still think they are highly relevant):

[CCGR26b] M. Chechik, B. Combemale, J. Gray, and B. Rumpe: [How UML models and ontologies can complement each other](#) In: Journal Software and Systems Modeling (SoSyM). Springer Berlin / Heidelberg. ISSN 1619–1366. Volume 25(3), Pages 635–637, 2026.

[CCGR26] M. Chechik, B. Combemale, J. Gray, and B. Rumpe: [Agentic AI in the next frontier of model-based software engineering: the arrival of AI-hyper-agile software engineering methods?](#) In: Journal Software and Systems Modeling (SoSyM). Springer Berlin / Heidelberg. ISSN 1619–1366. Volume 25(2), Pages 315–317, 2026.

[CCGR25d] M. Chechik, B. Combemale, J. Gray, and B. Rumpe: [Pragmatic specification of software behavior, configuration, and orchestration: the precision and usability of domain-specific modeling.](#) In: Journal Software and Systems Modeling (SoSyM). Springer Berlin / Heidelberg. ISSN 1619–1366. Volume 24(6), Pages 1621–1622, 2025.

[CCGR25c] M. Chechik, B. Combemale, J. Gray, and B. Rumpe: [Standards in software development and modeling.](#) In: Journal Software and Systems Modeling (SoSyM). Springer Berlin / Heidelberg. ISSN 1619–1366. Volume 24(5), Pages 1315–1316, 2025.

[CCGR25b] M. Chechik, B. Combemale, J. Gray, and B. Rumpe: [On theory and management of dependencies between models.](#) In: Journal Software and Systems Modeling (SoSyM). Springer Berlin / Heidelberg. ISSN 1619–1366. Volume 24(4), Pages 975–976, 2025.

[CGJR24] B. Combemale, J. Gray, J. Jézéquel, and B. Rumpe: [How does your model represent the system? A note on model fidelity, underspecification, and uncertainty.](#) In: Journal Software and Systems Modeling (SoSyM). Springer Berlin / Heidelberg. ISSN 1619–1366. Volume 23(5), Pages 1053–1054, 2024.

[CGR24b] B. Combemale, J. Gray, and B. Rumpe: [Model modularity for reuse, libraries and composition: symbol management is key.](#) In: Journal Software and Systems Modeling (SoSyM). Springer Berlin / Heidelberg. ISSN 1619–1366. Volume 23(3), Pages 525–526, 2024.

[CGR24a] B. Combemale, J. Gray, and B. Rumpe: [Model-based code generation works: But how far does it go?—on the role of the generator.](#) In: Journal Software and Systems Modeling (SoSyM). Springer Berlin / Heidelberg. ISSN 1619–1366. Volume 23(2), Pages 267–268, 2024.

[CGR24e] B. Combemale, J. Gray, and B. Rumpe: [Model hybridization: towards a unifying theory for inductive and deductive reasoning.](#) In: Journal Software and Systems Modeling (SoSyM). Springer Berlin/Heidelberg. ISSN 1619–1366. Volume 23(6), Pages 1307–1308, 2024.

[CGR23a] B. Combemale, J. Gray, and B. Rumpe: [How to define modeling languages?](#) In: Journal Software and Systems Modeling (SoSyM). Springer Berlin / Heidelberg. ISSN 1619–1366. Volume 22(2), Pages 449–451, 2023.

[GR21c] J. Gray and B. Rumpe: [The wild-west of modeling \(Revisited\).](#) In: Journal Software and Systems Modeling (SoSyM). Springer Berlin / Heidelberg. ISSN 1619–1366. Volume 20(5), Pages 1333–1334, 2021.

[GR20b] J. Gray and B. Rumpe: [Modeling dynamic structures.](#) In: Journal Software and Systems Modeling (SoSyM). Springer Berlin / Heidelberg. ISSN 1619–1366. Volume 19(3), Pages 527–528, 2020.

[GR19b] J. Gray and B. Rumpe: [Models as the subject of research.](#) In: Journal Software and Systems Modeling (SoSyM). Springer Berlin / Heidelberg. ISSN 1619–1366. Volume 18(6), Pages 3189–3191, 2019.

[GR18b] J. Gray and B. Rumpe: [UML customization versus domain-specific languages.](#) In: Journal Software and Systems Modeling (SoSyM). Springer Berlin / Heidelberg. ISSN 1619–1366. Volume 17(3), Pages 713–714, 2018.

[GR17b] J. Gray and B. Rumpe: [Logic formulas in models.](#) In: Journal Software and Systems Modeling (SoSyM). Springer Berlin/Heidelberg. ISSN 1619–1366. Volume 16(3), Pages 625–626, 2017.

[FRS13] R. France, B. Rumpe, and M. Schindler: [Why it is so hard to use models in software development: observations.](#) In: Journal Software and Systems Modeling (SoSyM). Springer Berlin / Heidelberg. ISSN 1619–1366. Volume 12(4), Pages 665–668, 2013.

[FR12b] R. France and B. Rumpe: [Data warehouse concepts for model artifacts?](#) In: Journal Software and Systems Modeling (SoSyM). Springer Berlin / Heidelberg. ISSN 1619–1366. Volume 11(2), Pages 151–152, 2012.

[FR11b] R. France and B. Rumpe: [Variability in UML language and semantics.](#) In: Journal Software and Systems Modeling (SoSyM). Springer Berlin / Heidelberg. ISSN 1619–1366. Volume 10(4), Pages 439–440, 2011.

[Fra09] R. France: [Why Johnny can't model.](#) In: Journal Software and Systems Modeling (SoSyM). Springer Berlin / Heidelberg. ISSN 1619–1366. Volume 8(2), Pages 163–164, 2009.

[FR05b] R. France and B. Rumpe: [Model aesthetics.](#) In: Journal Software and Systems Modeling (SoSyM). Springer Berlin/Heidelberg. ISSN 1619–1366. Volume 4(2), Pages 107–108, 2005.

Together with our viewpoints based on current research, here are our predictions:

Looking ahead to the coming decade, modeling will remain a central intellectual and technological instrument of software and systems engineering. Depending on the type of projects, its relevance may, however, appear less visible on the surface. Models will increasingly be created, maintained, and used by intelligent tools rather than built

manually via conventional editors. They will be derived from requirements, source code, operational data, simulations, and natural-language descriptions and will continuously evolve together with the systems they represent. Modeling will therefore not disappear in the age of artificial intelligence. Instead, it will become more pervasive, more automated, and more closely integrated with development and operations. Maybe the most relevant way of using models in the future will be that automated smart tools would use certain forms of models to explain what the system intends to do.

The fundamental reason for the continued relevance of modeling is unchanged: Complex systems cannot be understood or engineered solely at the level of their implementation details. Abstraction, decomposition, viewpoints, interfaces, and explicit assumptions remain indispensable. Indeed, autonomous, adaptive, and highly interconnected systems make these capabilities even more important. Generative AI can produce programs, configurations, and designs quickly, but generated artifacts still need to satisfy architectural constraints, domain rules, safety requirements, and legal obligations. Models can provide the explicit structures, concise behavioral definitions, and thus the semantics needed to guide, constrain, explain, and validate such system synthesis.

In software engineering, models will increasingly become part of integrated, AI-supported development environments. Architecture models, behavioral specifications, data models, deployment descriptions, and test models will be connected with source code and runtime observations. Whether these will be UML models, models in domain-specific languages, or maybe in standards that will be developed in the future is, of course, an interesting discussion. Iterative evolution will enforce automated synchronization between these artifacts and will replace isolated modeling activities. This is especially necessary along the development activities starting from requirements down to quality assurance. Models will help developers explore design alternatives through smart and potentially compositional analyses, to assess the consequences of changes, generate implementations and tests, and prevent architectural erosion. An important research objective will be to establish models as continuously maintained knowledge structures rather than only transient documentation.

Cyber-physical and especially (mechanical/physical) systems engineering is only starting to use explicit modeling techniques such as SysML and other domain-specific languages. This will definitely increase because, other than programs, the targets of systems engineering are physical gadgets, which are less agile in their actual physical development. So non-physical models will become even more agile than these development processes are today. Vehicles, robots, production systems, energy networks, medical devices, and intelligent infrastructure combine software, physical processes, communication, and increasingly

autonomous decision-making. Holistic modeling during their development requires multiple modeling formalisms, for software, for physical gadgets, for processes, for the environment and for the interaction between all of those. From a software perspective, there are new elements to address, such as physics, uncertainty, robustness, and, to some extent, geometry. Research must enable the composition and coordinated analysis of these heterogeneous models. Particular attention will need to be paid for safety, security, robustness, and the identification of adversarial situations that were not anticipated during development.

In the future, predictive capabilities in the context of continuous software engineering will be leveraged for dynamically evolving ecosystems to address ecologic and social challenges such as sustainability at a much more complex scale. To this aim, techniques for the coordinated use of heterogeneous descriptive, predictive, and prescriptive models need to be elaborated, and the propagation of uncertainty investigated, leading toward the definition of a unifying theory for inductive and deductive reasoning.

Digital twins will further extend modeling into the operational phase. In manufacturing, mobility, energy, construction, and smart cities, digital twins will be continuously linked with their physical counterparts through data. And those digital twins may systematically be derived from the engineering models originally used to develop the physical counterpart. They will support monitoring, diagnosis, prediction, optimization, and certain forms of controlled adaptation. Research will need to address model federation or composition, synchronization, uncertainty, provenance, and the controlled evolution of twins throughout long system lifecycles.

In enterprise information systems and public administration, modeling will continue to provide an efficient bridge between organizational intentions and executable software processes. Process, decision, information, organizational, and regulatory models can make complex procedures transparent and amenable to systematic improvement. For example, a UML class diagram is meant to describe the real world (in a way that ontologies usually do). Yet, migrating from analysis to design, the very same notation is used to describe software structures that embody relevant data about the real world. We expect that soon (and certainly 10 years from now), AI assistants will allow domain experts to easily describe processes and rules in natural language, and modeling technologies will translate these descriptions into analyzable and executable representations. Important research questions concern the preservation of organizational and legal intentions (respectively, regulations) and the ability to explain a process from both the institutional and the individual user's perspective, for example, to guide Individual users through the more complex workflows.

The healthcare and life-science domains will require models that deal with heterogeneous forms of biological, medical, and clinic-organizational knowledge. These models will have to integrate and modularize clinical processes, medical devices, and treat patient data in highly specific computational analyses.

And the list could go on. In domains such as financial and insurance, models will be used to detect security breaches as well as unusual financial transactions, risks, etc. Scientific research itself will become an increasingly important domain for modeling. Currently, models are mostly focused on simulation in this domain, but we expect that other types of models will occur as problems become more complex.

Yet just technical correctness of the models in use will not be sufficient to allow them to reach their full potential. Future modeling techniques must represent uncertainty, evidence, ethical constraints, and the limits of available knowledge. Models must become capable of expressing not only what is known, but also how confidently it is known and under which assumptions this knowledge remains valid. All too often this is currently just encoded into the data structure that the model is talking about but not in the modeling language itself. It means, for example, that in most of the modeling languages today there is no modeling concept to describe fidelity (uncertainty, robustness, timing, etc.) of some measurement directly. Consequently, modeling tools usually do not embed algorithms to analyze and aggregate fidelity (uncertainty, robustness, timing, etc.) constraints to understand whether relevant constraints are preserved.

Across the above-mentioned domains, several research topics will probably shape the next decade. A first major topic is the relationship between modeling and generative AI. Humans have always tried to reduce their workload by automating as much as possible. Now LLMs and their embedding into multi-agent orchestrations will assist developers in the creation, completion, translation, and explanation of models. The most promising approaches will probably combine the flexibility of generative AI with the repeatable precision of modeling languages and their traditional transformation and generation tooling.

However, as the generative AI tools have become pretty good and will probably become even better over the next few years. It may well be that, for certain kinds of projects, especially if they are only medium-sized, it will not be necessary to use explicit modeling techniques at all. Instead, developers would be able to directly map informal textual descriptions down to code. Currently, the “vibe coding” approach is assisting that idea, even though it breaks down for larger projects. It may, however, also be that generative AI agents themselves use modeling techniques to structure their internal steps, e.g., to break down their development steps using explicit models similar to what we do today. That would also improve

explainability. As usual, the future is difficult to predict, especially in current highly uncertain and transformative times. It may be a relatively safe prediction to argue that all of these approaches will have their merits and will be used in the future (more or less).

A second topic is the development of modeling languages and tools that are usable by a much broader community. Domain experts should be able to express knowledge without first becoming modeling-language specialists. Textual, graphical, tabular, conversational, and immersive representations will then coexist and offer different views of the same underlying development artifacts. Research must investigate how these representations can be synchronized, and how intelligent assistants can support users without obscuring relevant design decisions. With this idea of community, modeling tools have specific challenges to manage: Usability, comfort, and ease of understanding in an individual user as well as collaborative modeling will then become as important as the expressive power of the language.

A third topic concerns semantics, interoperability, and composition. This has been a regular topic in the modeling community, but lots of problems remain. Future systems will rarely be described by one modeling language or their development managed by one tool. Models must be compatible and integrable across organizational, technological, and disciplinary boundaries. Common semantic foundations, explicit and clearly defined model interfaces, and compositional reasoning will therefore be essential. Here again we have this dichotomy between seeking one universal language in a “one fits all” approach versus the well-defined cooperation among specialized languages and viewpoints.

A fourth topic is the management of models at scale. We have seen development projects for industrial systems, such as in the automotive domain, that, when taken seriously from high-level holistic viewpoints down to individual system components, involve millions of model elements, many variants and versions. The creation of models of that size is only economical when these models have long lifecycles and can be used in various activities of the system development, production, and during operation. Efficient storage, querying, differencing, merging, migration, and incremental analysis will then be crucial. Furthermore, that means that models must also be treated as governed engineering assets with known provenance, ownership, quality, and validity with clearly known development states. This requires scalable infrastructures as well as empirical research into the organizational conditions under which modeling creates sustainable value.

There is also a good chance that verification and validation will gain renewed importance. Models can serve as the basis for simulation, testing, formal verification, runtime monitoring, and certifications of assurance cases. Again, this may

well happen together with generative AI. Currently, generative AI is not yet in shape for doing formal verification, especially for complex semantic properties. But if there is a chance that the same high-level AI agents that can find rare security errors in complex pieces of code can also be used for formal verification, then these techniques will have a strong revival. It may well be that interactive verification tools will then be equipped with agentic AI orchestrators that proactively try to find proofs or counter-examples (i.e., security breaches), and we will finally be able to prove the correctness of our software. And if verification during development time fails, certain parts of models will become part of continuous assurance processes that connect design-time reasoning with runtime evidence. If we remember the early days, some of us may have actually dreamt of that from the very beginning of our scientific life.

The next decade will thus be characterized by trying to come up with synergetic approaches between modeling, coding, and artificial intelligence as intelligent helpers. AI is already changing how models are created and used; models will make AI-supported engineering more systematic, controllable, and trustworthy. Modeling will remain relevant wherever complexity must be reduced without losing essential meaning, where alternatives must be explored before costly decisions are made, and where the behavior of consequential systems must be understood and justified. The field's central challenge will be to transform modeling from a specialized development activity into an intelligent and continuously available engineering capability.

If we have forgotten relevant and important considerations, if you disagree, or are able to refine the above considerations, you are very welcome to express those. Did you know that SoSyM has the concept of a letter to the editor, e.g., by writing us a substantiated letter via email? This would also allow you to make your considerations public.

3 Bran Selic

This issue also marks a very special moment for our community, as we celebrate Bran Selic on the occasion of his retirement. Bran has been a major figure in software and systems modeling for several decades, contributing profoundly to the foundations and evolution of the field. From the early days of Real-Time Object-Oriented Modeling (ROOM) and the modeling of distributed, software-intensive systems interacting with the physical world, his work and ideas have helped shape many of the concepts and technologies that later emerged through UML and its profiles, MARTE, and SysML (see also pUML discussions of that time). His scientific contributions, long-standing engagement with the community, and service to this journal have had a lasting impact on generations of researchers and practitioners. It is therefore

particularly fitting that this issue includes a special section dedicated to Bran and his remarkable career.

Beyond his scientific achievements, Bran has also been a role model for many of us through his kindness, generosity, and consistently thoughtful and constructive advice. We still remember with great pride the moment when Bran received the first MODELS Career Award in Saint-Malo in 2016; it was a privilege to share that celebration with him. His influence extends far beyond his publications and technical contributions: It is also reflected in the people he has inspired, supported, and encouraged throughout his career. On behalf of the journal and its Editorial Board, we would like to express our deepest gratitude to Bran for everything he has brought to our community, and to wish him and his family all the very best for this new chapter and the many adventures ahead.

Marsha Chechik, Benoit Combemale, Bernhard Rumpe, and Martin Schindler

Editors-in-Chief, *Software and Systems Modeling*.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

Funding Open Access funding enabled and organized by Projekt DEAL.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.