Software and Systems Modeling (2026) 25:315–317
https://doi.org/10.1007/s10270-026-01362-0

EDITORIAL

Agentic AI in the next frontier of model-based software engineering: the arrival of AI-hyper-agile software engineering methods?

Marsha Chechik[1] · Benoit Combemale[2] · Jeff Gray[3] · Bernhard Rumpe[4]

Published online: 24 February 2026

Model-Based Software Engineering (MBSE) has matured into a rich ecosystem of methods and tools that support the systematic design, analysis, and evolution of complex systems. Model-Based Systems Engineering (MBSysE) is now following a similar trajectory, building on the advances of MBSE. Over the past decades, we have developed ways to encode our understanding in models, to formalize designs, and to automate transformation and validation steps.

In contrast, the recently emerged Agentic AI enables the development of autonomous AI systems that can perceive, reason, plan, and act independently. Such systems can pursue complex goals with minimal human oversight, moving beyond simple command–response interactions to proactively leverage external tools. In this setting, AI agents exhibit agency: They demonstrate goal-driven behavior and can execute multi-step tasks.

In software, and more broadly, systems development projects, such AI agents can be viewed as junior, or even senior, developers capable of executing substantial tasks. This creates both a challenge and an unprecedented opportunity for MBSE. At present, it remains unclear whether agentic AI can reliably translate informal, high-level requirements into correctly designed, implemented, and validated systems. The end-to-end process is long, and the risk of hallucinations or other incorrect actions remains significant.

✉ Bernhard Rumpe
bernhard.rumpe@sosym.org

Marsha Chechik
marsha.chechik@sosym.org

Benoit Combemale
benoit.combemale@sosym.org

Jeff Gray
jeff.gray@sosym.org

[1] University of Toronto, Toronto, ON, Canada

[2] University of Rennes, Rennes, France

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

[4] RWTH Aachen University, Aachen, Germany

Agentic AI promises not only to generate code or suggest designs but also to actively participate in modeling processes. An agentic assistant could negotiate trade-offs among conflicting design constraints, propose architectural alternatives, or maintain traceability links as a project evolves. In many domains, established (non-AI) tools already exist to rigorously evaluate designs and check consistency, completeness, and optimality. Rather than producing opaque, black-box solutions, an AI agent could follow a systematic internal methodology to derive correct, optimized results. This entails decomposing complex implementation tasks into smaller, well-defined steps within the agent itself, enabling the documentation and externalization of intermediate reasoning. Such an approach would support the construction and review of traceable artifacts that explain how the agent arrived at its conclusions.

If AI agents were to follow an explicit internal development methodology, it becomes plausible to train them to enact established MBSE processes for the systematic development and evolution of software-intensive systems. Such processes may span multiple abstraction levels and viewpoints, following lifecycle frameworks such as the V-Model, the Rational Unified Process, or agile variants, as well as fine-grained modeling patterns including refinement, abstraction and concretization, model transformation, model repair, synthesis, and model co-evolution. More generally, these practices support principled decomposition and composition of system models, separation of concerns through decoupled artifacts, and the systematic application of architectural and design patterns. When adopted by agentic AI, MBSE-aligned methodologies can improve model fidelity and consistency, strengthen traceability across abstraction levels, and provide a foundation for rigorous verification, validation, and testing, ultimately supporting more robust, resilient, secure, and explainable system development.

Much of this methodological guidance, together with its supporting frameworks and tooling, originates in the software engineering body of knowledge and is deeply rooted in UML- and SysML-style structural, behavioral, and

interaction models (see, e.g., SWEBOK v4). This raises a fundamental question for the SoSyM community: Should AI development agents be trained to enact established model-based development methods? Should entirely new, AI-native methods be devised, or should hybrid approaches emerge that combine existing MBSE principles with agentic capabilities?

As agentic AI increasingly shifts the engineer's role from manual model construction toward supervision, orchestration, and methodological governance, new layers of methodological reflection are required. In particular:

1. How can AI-based modeling assistants augment core MBSE activities such as abstraction, refinement, and model transformation?
2. Which elements of modeling methodology, such as lifecycle structures, viewpoints, consistency rules, and refinement strategies, should be explicitly embedded within AI agents?
3. What patterns of interaction and responsibility sharing between human modelers and AI agents best support effective collaboration, traceability, and trust?

These questions lead to a broader meta-question: When developing systems in collaboration with AI agents, how traditional should the underlying model-based method remain? Rather than merely adapting existing methods, it may be necessary to co-design development processes in which AI agents actively participate in shaping problem-appropriate methods, guided by MBSE principles. Such a division of labor would allow AI agents to carry out well defined, automatable modeling and transformation tasks, while human engineers retain responsibility for validation, assurance, and accountability.

We propose to refer to this emerging class of model-centric, AI-enabled development approaches as *AI-hyper-agile software engineering (AHASE)*.

The agentic paradigm challenges traditional distinctions between *tools* and *developers*. If AI agents are treated as co-developers, they can be integrated into existing development teams as regular, albeit exceptionally fast, contributors. These agents can be tasked with producing standard development artifacts, such as models at various abstraction levels, that are amenable to review, validation, testing, and other forms of analysis, regardless of whether they are generated by AI or authored by humans. Crucially, such artifacts can be evolved incrementally throughout development, since they remain intelligible and manipulable by both human engineers and AI agents.

This, in turn, requires training AI agents to reason in ways analogous to human developers, enabling them to create, adapt, and apply modeling techniques as intermediate artifacts when mapping high-level requirements to concrete designs and implementations. AI agents must be grounded in development methodologies, modeling languages, best practices, and systematic techniques for refinement, decomposition, and evolution. They must also be able to operate existing toolchains to correctly modify and co-evolve models, code, tests, and related artifacts. This includes using established analysis and transformation algorithms, such as checking semantic equivalence and identifying semantic differences between successive versions of evolving models—and we can teach them these skills!

This perspective also has implications for how development tools are used more broadly. If an AI agent understands both the development methodology and the associated toolchain, it can be delegated concrete tasks without requiring the human engineer to master the detailed mechanics of each tool. This has two important consequences. First, the AI agent can act as an intelligent interface to existing tools, simplifying access to inherently complex modeling environments, mediating interactions, and even teaching effective tool usage. Second, humans and AI agents can jointly execute complex development tasks in which the human specifies goals and desired effects, while the agent manages the technical execution within the toolchain.

This approach can be taken even further. A methodology-aware AI agent can also guide the development process itself, suggesting appropriate next steps, recommending relevant modeling or analysis activities (and even pushing the buttons on tool interfaces!), and adapting the process to the current development context. In doing so, the agent supports a shift from manual tool operation toward goal-oriented, method-driven collaboration, placing us squarely in an AI-hyper-agile software engineering (AHASE) setting.

1 Summary and outlook

We contend that AI agents in the near future of MBSE will move beyond automating recurring tasks, such as model consistency checking, code synthesis, and test generation. Instead, supported by rigorous methodological scaffolding, they will systematically derive consistent, high-quality artifacts. By intelligently composing established modeling, analysis, and transformation activities, agentic systems can help ensure correctness, completeness, and explainability of development outcomes. Investigating AI-hyper-agile software engineering (AHASE) as a research direction is therefore both timely and necessary.

This vision immediately raises a practical question for the community: *How agent-ready are our current modeling tools and infrastructures?*

2 A call for systematic exploration

The SoSyM community is exceptionally well positioned to guide this transition. Its long-standing strengths in formal foundations, domain-specific modeling, verification and validation, empirical evaluation, and tool engineering provide the essential foundations for advancing agentic MBSE in a responsible and principled manner. We therefore invite methodological, empirical, and technical contributions that rigorously examine the challenges and opportunities outlined above.

Agentic AI is unlikely to replace MBSE; rather, it can amplify its core strengths by making modeling more scalable, interactive, and continuously evolvable. Realizing this potential will require careful experimentation, transparent reporting, and a willingness to rethink established workflows. As earlier generations of model-driven methods reshaped software engineering, agentic AI, if integrated with strong methodological foundations, may once again redefine the field.

Funding Open Access funding enabled and organized by Projekt DEAL.