Software Development and Modeling in the Age of Artificial Intelligence

Manifesto

Lukyanenko R., Samuel B. M., Tegarden D., Larsen K. R., Jabbari A., Abd El Aziz, R., Almeida J. P. A., Amrollahi, A., Beard, J.W., Bellatreche L., Bork, D., vom Brocke, J., Cabot, J., Castellanos, A., Gerber, A., Green, P., Grover, A., Guizzardi, G., Hertelendy, A., Kahlon, Y., Karlapalem, K., Khatri, V., Maass, W., Maurer, C., Mueller, R. M., Mylopoulos J., Nishant, R., Ogunseye, S., Paré, G., Parsons J., Pastor, O., Prester, J., Proper, H.A., Ralyte, J., Sadiq, S., Schuff, D., Siau, K., Snoeck M., Storey, V.C., Tremblay, M.C., Trujillo J., VanderMeer, D., Venkataraman, R., Wagner, G., Wiedemann, A., Woo, C., Yu, E., Yue, W. T., Zhao, L.

Abstract

Artificial intelligence (AI) is rapidly transforming software development. At the same time, approaches that align with the intellectual tradition of agile, including code-first methods, rapid prototyping, and vibe coding, have become more prominent, due to their speed and flexibility. The role of conceptual modeling has become less clear in this emerging environment. However, danger exists in discarding modeling, as it has traditionally shielded software development from risks and vulnerabilities of developing a solution without careful and systematic deliberation. These risks are now dramatically amplified due to the speed and scale of AI. This manifesto finds a new place for conceptual modeling in software development with AI, and as a result, proposes a new way to conduct software development more broadly. The manifesto comprises core principles, termed *SAFE-AI*, and the process of implementing the principles, *MADE with AI*. Together, they pave the way for a synergistic collaboration between AI and human developers with the aim of producing effective and responsible software.

Keywords: Artificial intelligence, software development, agile development, conceptual modeling, SAFE-AI, MADE with AI, model-driven development, vibe coding

Introduction

Artificial intelligence (AI) has revolutionized the way we approach software development. Generative AI, based on large language models (e.g., ChatGPT, Claude, Gemini, DeepSeek), excels at many content-interpreting and creating tasks, while Agentic AI (e.g., MakeAI, CrewAI) is increasingly capable of making decisions with minimal human supervision. Many AI systems, including general-purpose (e.g., ChatGPT, Claude) and more specialized tools (e.g., MakeAI, Bolt.new, Github Copilot, Microsoft AutoGen, Lovable) allow for the development and deployment of new code[1–7]. Tasks that used to take years of experience to master and days to complete, such as requirements specification, writing boilerplate code, debugging, refactoring, and even live deployment, can be done - *at least in principle* - within minutes with a few natural language prompts.

With the on-going shift toward increased coding automation, new forms of development are emerging, such as prompt-driven engineering (where AI generates code from user prompts)[8,9] or its specific approaches like vibe coding.[7,10] New metaphors are springing up to rationalize the "new kid on the development block." An emerging consensus treats AI as a coding hyper-assistant[9], a sort of sous-chef, following the lead of a human executive chef [7].

The use of AI for developing software is further catalyzed by the trend toward agile development, which has risen to become the most popular development methodology [11]. Agile methodologies such as Scrum, Kanban or eXtreme Programming prioritize speed of development and rapid iterations over longer planning and documentation[12,13]. With the pressure to deliver code faster, it is not a surprise that developers gravitate towards Generative and Agentic AI tools to keep up with the growing productivity expectations.

Yet, for all of agile's benefits, some have identified a new challenge - "Agile Burnout"[14–16] - caused by aggressive release cycles. At the individual level, Agile Burnout puts physical and mental strain on developers[15]. At the societal level, it can lead to harm through failed systems[17,18].

The debates around the place for AI and the role of humans in the new software development world so far have focused on creating, deploying, and maintaining code. However, code is only a means to an end. It remains unclear how we should approach the overall software development process in the wake of AI, including understanding what problem the software system seeks to solve and verifying that the developed system provides an effective, safe and responsible solution for everyone.

For problem-solution understanding and management, traditional development approaches, such as the Software Development Life Cycle (SDLC), emphasize the importance of humans engaging in conceptual modeling. Conceptual models are abstract representations that capture user requirements, provide blueprints of the problem at hand

and the pursued solution, and serve as documentation for the developed system and the process that led to it. Typical conceptual models used in software development are *entity-relationship diagrams* that model data requirements and database design, as well as *Business Process Model and Notation* (BPMN) that represent system processes. Modeling has additional ways to contribute to problem-solution understanding by providing techniques and methods to represent ontologies, goals, values, enterprises, strategies, and other models.[19]

The combination of agile development and capable AI seems to suggest that we should abandon conceptual modeling entirely. Indeed, traditional conceptual modeling is in decline across many contexts, especially those where agility takes priority[18,20–24]. Formal modeling is also not considered in the recent AI-human software development frameworks, such as vibe coding [7], and is relegated to an "afterthought" in related code-first development frameworks[25]. Not surprisingly, then, many ask[18,20,21,26–28] why we still need to design entity relationship diagrams to build databases, use UML activity flows to understand existing system behavior, reason through BPMN models to ascertain the process-to-be, or to populate Business Model Canvases with strategic priorities. Should we still learn, teach, and practice how to develop conceptual models?

In this Manifesto (following the spirit of the Agile Manifesto[12], a proclamation that was foundational for agile development), we contend that conceptual modeling is indispensable. It is a process that makes our AI-driven software development reliable, resilient, and aligned with human values. To create a safer, better world with AI and other technologies, we must rethink the value of modeling in the fast-paced AI and agile-powered world.[29] To do so, we re-imagine the cooperative roles of AI and human modelers based on the decision-making paradigm known as System 1 vs. System 2[30]. This new lens on modeling and AI permits us to rethink the benefits of modeling in the age of AI, advancing beyond the traditional and even more recent ideas about the benefits of modeling[28,31–33]. It further allows us to reconsider the entire process of software development and situate humans in a central place in the increasingly AI-dominated world.

We lay out a Manifesto for Software Development and Modeling in the Age of Artificial Intelligence. The manifesto comprises the core principles, termed *SAFE-AI* (Strategic deliberation; Attention; Freedom; and Empowerment). The Manifesto then articulates the process of implementing the SAFE-AI principles, *MADE with AI* (Model, Agree, Develop and Evaluate). We envision our work as paving the way for a more fruitful collaboration between AI and human developers and addressing the ongoing controversies regarding the role of conceptual modeling in the modern software development landscape.

Modeling in the Age of AI: A Perfect Storm and Opportunity

Conceptual modeling traditionally seeks to create systematic representations of domains, processes, and systems. Agile approaches, in contrast, prioritize minimal upfront design, rapid delivery, and adaptive responsiveness to change[12,13].

The conceptual modeling community has long investigated the benefits of modeling[19,34,34–46]. Distilled into broad categories, modeling provides cognitive support, communication, automation, and knowledge preservation support. These underlie specific use cases identified over the more than half a century of active use of conceptual models.

- **Cognitive support**. Conceptual models provide a foundation for reasoning, simulation, analysis, and design allowing stakeholders to evaluate alternatives, assess risks, and explore "what-if" scenarios before committing to costly implementations. They reduce mental overload by abstracting away the complexity of reality and focusing attention on the essential structures and relationships in a domain. This simplification allows both developers and stakeholders to reason effectively about complex socio-technical systems.
- **Communication support**. Conceptual models act as boundary objects[47] that bridge diverse perspectives among stakeholders, enabling communication across technical and non-technical communities. By providing a common language, they support negotiation, alignment, conflict resolution, and requirement completeness through an interactive elicitation process. Conceptual modeling enforces clarity in definitions, relationships, and constraints, reducing ambiguity in requirements and designs. It creates opportunities for conversations about the hidden assumptions underlying the structures and processes of a domain. By providing comprehensibility, models can thus be constructed and validated in a group setting. This rigor increases the likelihood that the resulting system will accurately reflect the stakeholder needs. The formality of models also drives an analyst to question-asking, thus increasing the likelihood of more complete requirements gathering.[48]
- **Automation support.** Even without AI, basic automation permits models to be turned into the components of software. For example, conceptual data models can be scripted as database objects, such as tables and indices. This idea underpins *model-driven development*, which emphasizes creating high-level visual or abstract models over traditional hand-coded implementations.[49,50]

- **Knowledge preservation support.** Models serve as durable documentation that records system design and domain knowledge in a structured form. This supports organizational and project memory and facilitates onboarding, auditing, and long-term maintenance.[34,46] Explicitly modeling the problem and solution increases the overall transparency of the development process.

As a result of these underlying benefits, conceptual modeling has seen a wide range of applications. Prominent use cases for conceptual modeling include [19,51–54] :

- Structuring and capturing software requirements.
- Developing and managing software in large organizations.
- Integrating siloed organizational knowledge into a unified, “global” view.
- Data modeling, for databases, integration, and interoperability.
- Business process engineering, simulation and management.
- Compliance, audit and governance.

Empirical research in conceptual modeling has demonstrated these benefits in various scenarios and for different types of conceptual modeling grammars[55–60]. These benefits, in various forms, have been suggested as reasons for modeling in agile development[20,26–28]. For example, in the Symboleo project, running at the University of Ottawa since 2018, Amyot and colleagues [61,62] demonstrated that for domain-specific software (e.g., smart contracts), it pays in human effort saved to generate a conceptual model/specification of the smart contract and from that model generate code automatically (instead of, for reasons of agility, forget modeling and develop code directly).

Nonetheless, even before Generative AI the role of conceptual modeling was contested as not being a value-added activity for software development, given the tension between the planning and up-front specifications and the iterative, adaptive ethos of agile development methods. Even practices such as model-driven development, remain relatively “uncommon”[63] in software development due to the complexity of managing model-to-code transformations and the lack of integration with agile and other adaptive methodologies.

As a result of a seeming misalignment between the values and needs of modeling and development, modeling is either limited or is nearly entirely absent in agile development.[1]

[1] Most commonly, modeling appears as user stories - concise natural-language expressions of functionality from the user’s perspective [64]. At the same time, while they capture some aspects of requirements, they lack

A cofounder of the Agile Manifesto, Ron Jeffries, summarized a popular view in the software development community as follows[27]:

> "Agile development, to me, is best done with as little specifications as possible. I teach and believe that the best results are obtained with a team of individuals who contribute whenever and however they can, without regard to who is architect, the modeler, the designer, the programmer, or the tester.[...] We should all come together and contribute as much of everything as we can.[...] Furthermore, software development (again, to me), is best done with as little modeling as possible. The point of software development is to develop software. Too much time is taken up in getting ready, leaving too little time for the important part – actually producing solid, well-designed, high-quality software."

Agile practitioners readily acknowledge the benefits of better communication, stronger specification, and increased understanding of the problem and solution space through modeling, but point to the inevitable overhead and rigidity in the face of change that heavy specification brings, for example, the infamous analysis paralysis[13,27,65]. Furthermore, often the best solutions come from free exploration and trial and error during development. With the rise of code-capable AI, this sentiment is unlikely to change in favor of modeling.

Artificial Intelligence gives new impetus to the specification-agility tension as it introduces automation as a potential tool for a broad spectrum of development tasks. AI now can rapidly produce design artifacts, requirements specification, or even executable code, thereby challenging the perceived necessity of careful, human-driven conceptual modeling. Current AI tools can be given agency to extract information for constructing models from documents in the domain, and automatically develop code directly from this information[18]. Hence, the existence of AI may sway the argument even further away from conceptual modeling and more in favor of writing code and directly creating other components (e.g., database tables).

One emerging example of software development in the age of AI is "vibe coding". Vibe coding is a natural language-driven software development paradigm coined by Andrej Karpathy in 2025 [7], where developers interact with large language models (LLMs) to generate code by describing desired functionality rather than writing the code. This

the semantic precision leading to interpretive gaps. They are also not well-positioned to capture business rules (e.g., cardinality constraints on the relationships between relevant entities).

approach can evidently bypass traditional review and debugging processes [10]. While vibe coding democratizes programming and accelerates prototyping, recent studies have raised concerns about its reliability and security. For instance, the Databricks Red Team demonstrated how vibe coding led to a critical remote code execution vulnerability due to unsafe use of Python's pickle module, highlighting the risks of unvetted AI-generated code [66]. Further studies have demonstrated that AI-generated code samples often contain known vulnerabilities, such as SQL injection [67]. These findings indicate the need for practices that support abstraction, communication, durable documentation and developer education – hallmarks of conceptual modeling based approaches.

With the twin pressures of agile and AI, the practice of conceptual modeling faces a perfect storm. Its relevance has been too readily questioned by the proponents of agile development principles. Now the situation is even more dire, given the meteoric rise of Generative AI tools and the upcoming wave of Agentic AI.

However, much danger exists in discarding modeling, as it has traditionally shielded software development from risks and vulnerabilities of developing a solution without careful and systematic deliberation. These risks are now dramatically amplified due to the speed and scale of AI. Hence, rather than saving modeling in the age of AI, a bigger imperative is to save software development from the risks of inflicting harm and damage due to overreliance on methods that produce speed and scale.

This perfect storm around modeling is an opportunity to evaluate the role of conceptual modeling in the age of AI, and to reconsider the nature and goals of software development itself.

Software Development and Modeling in the Age of AI

To begin establishing new foundations of software development in the age of AI, we need to rethink some of the fundamental assumptions underlying modeling and offer a different value proposition for modeling in this brave new world. This new value proposition must complement agile and AI practices, acknowledging the relative strengths and weaknesses of agile and AI development versus human modeling. The key is finding the right balance that makes the overall combination stronger and more effective.

To realize the right balance, we need to establish a realistic view of the strengths and weaknesses of modeling in the age of AI.

Finding the new role of modeling

When speed of software creation is the priority, as is assumed in many modern development frameworks, modeling can be seen as consuming precious coding time. Furthermore, when developers immerse themselves in the act of coding, many argue that they find less need for modeling, as both the cognitive and communication processes are enacted by deep and intimate immersion in the coding process. This approach makes sense when the tasks are well-understood, the development team is extremely experienced, and the developed software has minor and predictable impact on people, other systems, and the environment[68]. In contrast, many of today's digital transformation initiatives are built on software that is integrally woven deeply into core organizational and personal processes, while also serving as platforms for launching new strategic initiatives that support profound and lasting change. Such complex situations call for a better integration of modeling into the development processes.

Given what is already possible, AI is poised to write more code. One can reasonably foresee a scenario in which much of the code is written by AI, with limited human involvement or oversight.

Ironically, AI promises to directly automate the core promise of agile – the code creation itself. This positions modeling as a natural complement to agile and AI. Since AI can speed up coding (or do the coding itself), the paradigm that "software development is to develop software" no longer holds because there is no software to develop (AI is doing that for us). However, what AI cannot do well right now is deep reasoning, domain understanding, and designing safe, sound solutions for everyone. Thus, in the age of AI, software development becomes a mix of domain understanding, some development and greater emphasis on solution verification.

In the new world of even more powerful AI, conceptual modeling is positioned to become the quintessential human task amidst rampant automated software development. Modelling enables a new partnership between humans and AI, combining the strengths of both to create a powerful symbiosis.

To gain a deeper insight into how AI-human partnerships can be accomplished, we turn to the System 1 vs. System 2 metaphor from cognitive psychology and decision-making theory.

Inspiration: System 1 and System 2 cognition

Modeling is a complex cognitive activity. It seeks to understand an existing situation and develop a novel solution. For humans, modeling involves the interpretation, abstraction, simplification, and representation of reality in a form that can be mentally understood, communicated, and manipulated. This process engages imagination, analytical reasoning, and prior knowledge.

The dynamics of human cognition can be understood through the System 1 vs. System 2 metaphor, as developed by Daniel Kahneman with his colleague Amos Tversky and others[30]. It is based on a Nobel prize-winning research program into the nature of decision making[30,69–71]. System 1 is referred to as automatic cognition and System 2 as deliberate cognition. System 1 runs involuntarily and continuously; it "cannot be turned off"[30]. System 1 can be thought of as your intuition. System 1 is fast and allows humans to quickly generate a response through pattern-matching information and heuristics in our memory that align with the current situation[30]. In contrast, System 2 is slower, rational, and effortful. System 2 is often invoked in unfamiliar, complex, or unusual situations. Similar to the distinction between fast intuition (System 1) and slow, effortful reasoning (System 2), the theoretical concepts in cognitive psychology distinguish between intrinsic load (the inherent complexity of the problem), extraneous load (unclear or fragmented information), and germane load (capturing the useful effort spent forming mental schemas). Studies show that high intrinsic or extraneous load strains working memory, whereas structured guidance that increases germane load improves the quality of complex analytical tasks.[72]

Humans need both System 1 and System 2 to function; each has its own strengths and limitations. Often System 1 dominates as it allows humans to follow learned patterns of the world instead of exerting deep cognitive energy to process every stimulus around them. While useful, System 1 is especially susceptible to biases. It performs best when learned responses match the problem at hand, but falters in the face of new evidence. This is where System 2 comes in. To cope with situational novelty, System 2 often requires additional information and support (e.g., more data, and transformations of data into new forms[73]). This slow, deliberate, and logical thinking helps us deal with challenges and opportunities that require deep concentration and situational judgment, particularly in the face of surprises or errors. Notably, in humans, System 1 and System 2 operate in parallel by mutually reinforcing and supporting each other.

The lesson of System 1 and System 2 for modeling and AI is that in many decision-making situations, there are going to be aspects that can be addressed rather quickly on the basis of recognizable patterns (consistent with System 1 processes), while others will require slower, logical, deliberate and careful determination (consistent with System 2 processes). These insights have already seeped into software development, in the form of neurosymbolic systems, which can outperform both expert systems and neural networks on certain tasks [74,75]. Further, the value of System 2 thinking for software development has already been suggested as a way to improve the quality and trustworthiness of AI applications 72. We extend these ideas to develop a broad vision of software development and modeling in the age of AI[76]. We extend these ideas to develop a broad vision of software development and modeling in the age of AI.

Speed is an unquestionable advantage of AI, even when System 1 in humans is involved. When used skillfully, AI can effectively streamline many of the System 1 processes in humans. For example, this may involve writing code when the code that needs to be written is easily described and performs familiar tasks.

Our position on AI as a good substitute for human System 1 in development is reinforced by the way AI functions today. Generative AI, in particular, largely acts as a non-deterministic pattern-matching engine that is trained on past data to present new output based on existing patterns[77,78]. However, it is generally not capable of considering the broader environment of the real world and what might actually be needed (e.g., a safe, ethical, transparent and responsible solution), even with good prompting techniques, especially when examples of solutions that meet these criteria are uncommon [79]. While there is research focused on moving AI into System 2 thinking (e.g., deep reasoning), the results so far have not been achieved at scale[80–82].

For now, contextual deliberation grounded in human values and needs remains largely the purview of humans, even as AI technology continues to make progress in deep reasoning. This kind of decision-making is presently enacted through a human's System 2 cognition.

Adapting this to the problem of modeling in the age of AI, we suggest that AI can be an invaluable partner in software development, with the present-day focus on System 1 processes. It also leaves humans with the luxury and the imperative to enact System 2. The *luxury of using System 2* comes from dramatic scaling and acceleration via AI's System 1-like capabilities. When the development process allows, this permits humans to engage in deeper thinking without the traditional pressure to deliver code quickly. The *imperative to enact System 2* in humans stems from the need to guide and control the dramatic speed and scale of AI, to ensure it delivers effective and safe results, and requires structuring development processes to provide room for this cognitive work.

Commonly, System 2 uses additional support such as charts or tables that facilitate careful and deliberate analysis. In software development, such support traditionally comes from modeling and models. As our summary of modeling benefits shows, modeling promises cognitive, communication, and knowledge preservation support. All these are critical in helping humans as they endeavor to take advantage of fast and scalable AI.

Hence, the partnership with AI allows humans not only to develop better software but also to realize the benefits of conceptual modeling without the shortcomings traditionally associated with the modeling process (e.g., being excessively time consuming).

Modeling and Software Development

We view the human-AI relationship as a sociotechnical system, with AI and humans as components. The partnership is flexible: both humans and AI can take on coding and other roles, under general human supervision while remaining open to the potential future where AI supervision is safe to permit.

To take advantage of differing capabilities, AI primarily realizes System 1 processes, giving humans the time to enact System 2 - conceptualize the domain and create models that guide and constrain the automated System 1 development. This ensures AI decisions are accurate and carefully attuned to the nuances of the complex socio-technical challenges and opportunities of modern life, while humans have the time and space to thrive in the AI-powered world.

The two components–humans and AI–work together to form a synergistic team [83]. This synergy is not the same as a mere act of outsourcing work to AI. It is co-development, where conceptual modeling provides the structure for AI contributions, and AI-generated insights can refine and evolve those models, much the same way as the System 1 and System 2 cognitive processes in a human mind interlace and mutually reinforce each other.

These ideas are captured in the four principles of our Manifesto, called ***SAFE-AI*** and in the process of realizing these principles, ***MADE with AI***. These further elaborate on SAFE-AI, provide specific guidance on how to adhere to the spirit and intent of these principles and suggest some open questions for future research and development.

SAFE-AI Principles

The four ***SAFE-AI*** Principles are: **Strategic deliberation** over speed and scale; **Attention** to critical requirements and impact; **Freedom** to develop software components as AI capabilities and human preferences evolve; and **Empowerment** of everyone, AI included, to challenge and improve the solution, in both development and deployment. The SAFE-AI Principles are summarized in Figure 1.

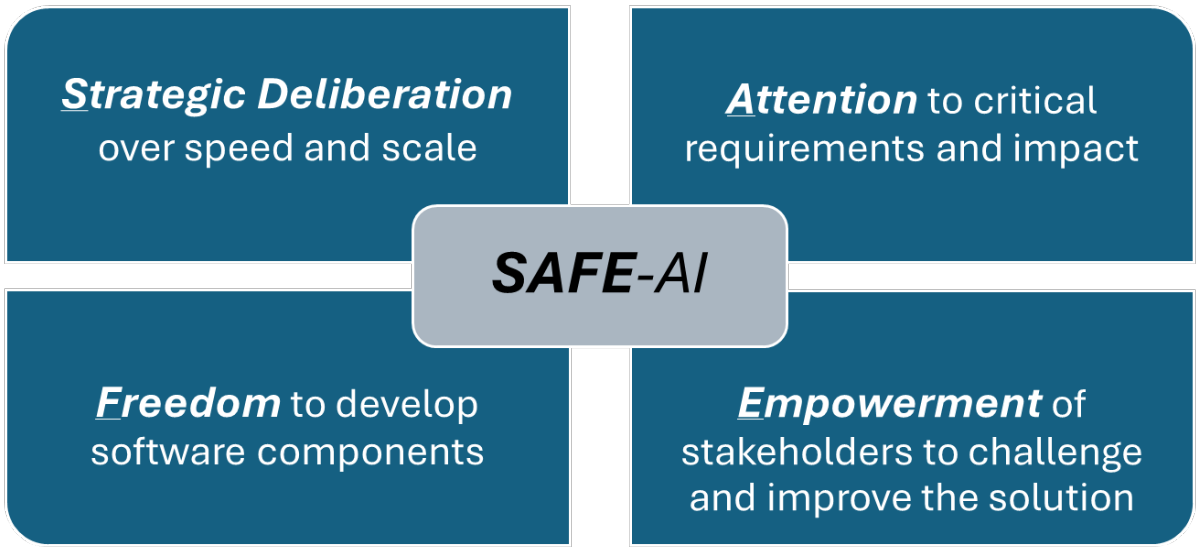
Strategic Deliberation over speed and scale
Attention to critical requirements and impact
SAFE-AI
Freedom to develop software components
Empowerment of stakeholders to challenge and improve the solution

Figure 1. SAFE-AI principles

These principles embody our aspiration to create a symbiotic partnership between humans and smart machines, resulting in the development of *safe for everyone*, responsible, and effective software systems.

Principle 1: Strategic deliberation over speed and scale

There are multiple ways to solve a given problem. AI may suggest several solutions, but not all are equally effective, beneficial, or safe. Strategic deliberation involves practicing slower System 2 thinking before leveraging AI's speed and scale. It safeguards against costly, if not catastrophic failures due to the careless use of the awesome powers of AI.

Despite impressive progress, powerful AI algorithms, including Generative AI, often exhibit biases and can also provide unsafe recommendations [32,84], such as triggering suicidal thoughts and actions [85]. Furthermore, Generative AI can hallucinate, that is, produce nonsensical or factually incorrect responses [3,86]. Researchers have also shown how Generative AI can be convinced to provide information on topics that are undesirable for the greater good of society [87]. Furthermore, the quality of Generative and Agentic AI outputs varies, affected by the quality of prompts, parameter settings (e.g., results can be intentionally non-deterministic based on the *temperature* of large language models), difficulties in reasoning about uncommon or "long-tail" patterns, among other factors [1,3,32,88]. In contrast, developing software components commonly requires consistency, predictability and precision. For software development in particular, modern AI has been trained extensively on low-level technical implementations (the final code of software systems) rather than on deliberation before the software was created, and requirements of those systems. As a result, AI struggles to generate reliable abstractions as humans understand them and tends to remain anchored to implementation decisions rather than conceptual design [89].

Finally, Generative AIs lack inherent transparency, that is the ability to explain their decision logic [90]. Transparency of the process and the outcomes (the software itself) is another desirable and often critical requirement in developing software [28,64,91]. Although Generative AIs can now provide some post hoc rationalizations of their outputs (when prompted), this is a far cry from the true transparency and explainability as the underlying model (e.g., the large language model with its trillions of parameters along with the data used for training) remains inaccessible and opaque to the human user.

Accepting and proceeding with whatever code is suggested by AI amounts to letting System 1 take control. As a result, seemingly minor errors could lead to large financial losses and compliance violations.[92] What is worse, if allowed to scale uncontrolled, AI-generated software has the capacity to cause great harm. This becomes even more problematic as we assign AI more agency, i.e., more freedom to make its own decisions and act directly with little or no supervision.

Strategic deliberation over speed and scale were famously practiced by Sun Tzu, a Chinese general, the author of *The Art of War*. Sun Tzu proclaimed that “strategy without tactics is the slowest route to victory. Tactics without strategy is the noise before defeat.” These ideas remain all too relevant in the world of AI.

In the age of powerful AI, human developers must understand the underlying problems that they are building software to solve. A simple modeling task may take a few extra minutes, but it forces a developer to think before acting. Unlike code that can cause real harm if incorrect, a model is a safe place to test ideas, identify flaws, and fix them before they become significant.

Principle 2: Attention to critical requirements and critical impact

With AI taking on many time-consuming tasks, human effort is theoretically freed to focus on what matters the most in terms of requirements. It also allows expanding the consideration of the system impact, including on the people, organizations, and the environment (e.g., markets, broader social and political institutions, plants and animals, natural resources), that might be less directly affected or impacted after some delay.

A new opportunity emerges to rebalance the workload between a human and AI, where AI can gather and structure requirements generally, leaving the human designer with the space and time to capture what really matters and to consider the bigger picture and broader impacts of the system-to-be. By 'what really matters' we mean those *critical requirements* that the system absolutely must get right. These are typically the requirements that deal with health, safety, legal, and ethical implications of the system, along with essential business rules that must be correctly executed by the system.

We also suggest leveraging System 2 thinking in considering *critical impact* [84] - the degree to which a possible failure in system behavior could produce irreversible harm, cascading systemic consequences, or severe degradation of stakeholder trust and institutional legitimacy. In other words, a requirement is of critical impact if getting it wrong would materially compromise the system's purpose, undermine its core value proposition, or generate substantial negative outcomes for users, organizations, society or the environment.

The human capture of mission-critical requirements contrasts with the traditional approach to modeling. In the past, conceptual models were built to "accurately and completely" and formally capture *all the requirements* for the systems to be built [28,93]. Clearly such an approach runs into challenges as it requires much time and dedication (the problem pointed out by the proponents of agile development). The shift from all requirements to critical ones addresses this issue. In the AI era, the human-created "critical models" serve as abstractions that highlight what humans believe is essential for domain understanding to guide AI in automating details hidden beneath those abstractions. These mission-critical models can sometimes be informal, but clear and unambiguous, as long as they effectively address the mission-critical requirements. AI can then be unleashed to capture as many requirements as possible - non-mission-critical and critical, the latter - to check against what humans deem essential.

The reclaimed space and time to focus on the critical requirements further permits human developers to deliberate more carefully about identifying the relevant stakeholders who can articulate these requirements and impacts. Traditionally, requirements are elicited from the likely potential users of the system, their managers, as well as some constituents assumed to be affected by the system [94–96]. However, given the typical time constraints, those affected less directly, or after some delay, are frequently missed or ignored [97–103]. The resulting systems result in unforeseen, yet often significant, negative impacts on different people, communities, and the environment [98,104–110,110].

By rebalancing the work load between humans and AI in requirements gathering, greater consideration of the full impact of the system should result, paving the way to more effective and responsible systems.

Principle 3: Freedom to develop software components as AI capabilities and human preferences evolve

There are different considerations in how to prioritize the distribution of effort between humans and AI in the age of smart machines. The most obvious are related to cost, effort, and quality. Some have argued that certain work is inherently human, such as creative, compassionate, or caring activities [111,112]. AI can thus focus on rote and repetitive tasks[112,113]. While such division of labor seems reasonable, it misses two important considerations. First, AI research is seeking to imbue AI with characteristics that humans may find compassionate and caring [114]. This may be desirable as human care is scarce in many situations. Second, some programmers find joy and relaxation in doing tasks that appear dull and unexciting to others. Indeed, what is 'rote' is in the eye of the beholder – it varies not only from person to person, but a given human may change their attitude toward what is exciting over time. Humans should have the freedom to choose the development contribution. This underscores the need for AI systems to be ethical toward human developers, remaining aligned with their goals and mindful of associated risks.

Freedom in development is not only a moral imperative or a matter of personal preference, but is necessary to ensure creativity and development of truly innovative solutions. Without freedom, we are risking stifling innovation and preserving the status quo.

Principle 4: Empowerment of everyone, AI included, to challenge and improve the solution in development and deployment

It is unlikely that the best design will emerge from a single code run. This is why it is important to empower everyone to challenge and evaluate an emerging solution. For humans, empowerment means having the authority and capabilities to question assumptions, review alternatives, reinforce AI learning and influence on decisions. For AI, empowerment refers to having the capability to propose diverse solutions, identify potential risks, and provide evidence-based recommendations rather than just executing instructions. The automation and scaling powers of AI, especially agentic AI based on multi-agent frameworks [6,115,116], permit AI to perform *artifact sampling* - developing and evaluating multiple possible solutions to help find the better one [117–120].

Different contributors to the solution—human, AI, and broad stakeholders—should be encouraged, with AI's facilitation, to challenge the emerging solution and its impact, permitting further iterations and refinements. This means creating opportunities for AI to suggest alternatives and for humans and stakeholders to question and review the suggestions to ensure continued improvement.

MADE with AI: Process of Realizing SAFE-AI

The Principles of SAFE-AI are realized through the process MADE with AI, aimed at leveraging the advantages and mitigating the limitations of System 1 and System 2 in humans and AI. The four steps of the MADE with AI process are: **Model** to (re)Activate System 2 of the AI-Human partnership into motion; **Agree** on critical requirements and critical impact; **Develop** the solution; and **Evaluate** developed and deployed solutions.

The MADE with AI process is general, allowing for a variety of scenarios, while simultaneously instantiating specific embodiments of the SAFE-AI principles.

The MADE with AI steps are iterative and can be repeated as many times as needed until a definitive solution emerges that passes the validation checks of Step 4. These steps are shown in Figure 2.

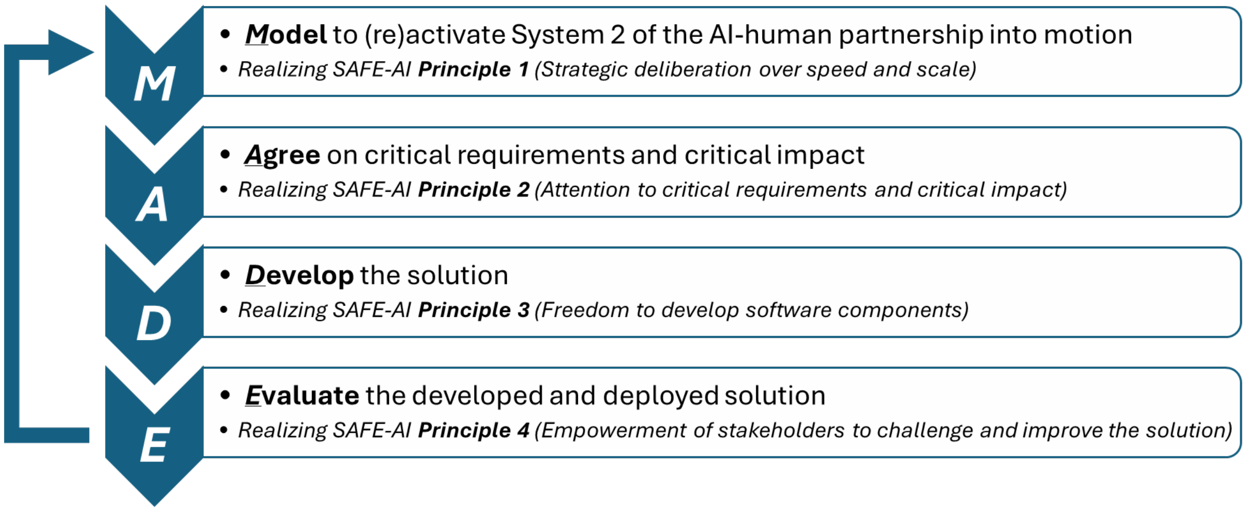
M
A
D
E
Model to (re)activate System 2 of the AI-human partnership into motion
Realizing SAFE-AI Principle 1 (Strategic deliberation over speed and scale)
Agree on critical requirements and critical impact
Realizing SAFE-AI Principle 2 (Attention to critical requirements and critical impact)
Develop the solution
Realizing SAFE-AI Principle 3 (Freedom to develop software components)
Evaluate the developed and deployed solution
Realizing SAFE-AI Principle 4 (Empowerment of stakeholders to challenge and improve the solution)

Figure 2. MADE with AI Steps

We elaborate each step of MADE with AI below.

Step 1: Model to (re)Activate System 2

Enacting strategic deliberation would require starting software development System 2 thinking. To take advantage of the power of System 2, it needs to be set in motion and be periodically re-activated when development hits roadblocks or when new contradictory evidence (e.g., on critical requirements) emerges between iterations. Engaging in any modeling achieves this, as it requires externalizing thoughts.

Presently, we assume it is a human who will be enacting System 2 via modeling, but we remain open to the possibility of this being done by AI, especially if deep machine reasoning capabilities of Generative AI become reliable. Even at current capability levels, AI can complement human modeling efforts by generating multiple alternative models, “seed” solutions that serve as starting points for human reflection which allows humans to identify gaps in their own models and consider perspectives they might have overlooked.[121] Still, danger exists in over-relying on AI models, as they can frame the problem prematurely and anchor the human into wrong solutions. Hence, we recommend humans to practice independent strategic deliberation, and always remain critical of their own as well as the AI-generated models.

The mere act of modeling forces one to pause and think. Drawing diagrams on paper, a whiteboard, a napkin, or a computer, requires translating human mental models and ideas into another medium. This process is especially valuable when the individual does not know much about the domain, in which case they can engage in a back-and-forth conversation with AI to identify general patterns and incorporate suggestions from other models used in similar contexts. This would enable a baseline to build a model.

Even if the initial models do not end up being implemented, the practice of slowing down and creating models at the very beginning or at the start of new iterations ensures that due deliberation is given to the problem and the solution. Dwight Eisenhower, the 34th President of the United States, captured this benefit of modeling when he famously proclaimed: “Plans are worthless, but planning is everything” [122]. Perhaps Eisenhower was inspired by another famous American, Benjamin Franklin, who had made this warning: “If you fail to plan, you are planning to fail!”

Strategically slowing down and forcing deliberation avoids costly mistakes. When AI can churn out working code on demand, the temptation to deploy quickly without pausing to reflect on the tradeoffs or other options in implementation can lead to suboptimal or outright problematic outcomes. Modeling before engaging AI to write code can help identify potential defects and enable humans to rein in the scaling powers of AI, ensuring they are used responsibly.

Step 2: Agree on critical requirements and critical impact

Once System 2 thinking is engaged, humans and AI can begin working closer together toward a solution. At this stage, an important human task is focusing on the critical requirements and anticipating potential system impacts. Humans no longer need to completely and accurately capture all the requirements. Rather, they need to determine what is critical to get right as the system is being built. Artificial Intelligence, then, can be tasked with structuring and understanding requirements in general.

With expanding AI capabilities to understand and model requirements, AI is becoming especially adept at extracting relevant information from documents or even engaging users in conversation to elicit their requirements.

Although not without issues and challenges[123], many general purpose and specialized tools automatically generate diagrams from prompts, external documents, and user stories. They can check model consistency and keep models updated as requirements change. As model creation capabilities of AI improve, AI will be better positioned to capture the non-mission critical requirements. For example, requirements dealing with user preferences for interface layout or adaptive content presentation can be structured and modeled by AI. These aspects can benefit from AI's ability to learn from many sources without jeopardizing the system's core integrity or compliance with critical constraints. Less critical requirements like these can then be then checked for compliance with mission-critical requirements.

Critical requirements are impossible to specify if the expected impacts of the system are unknown. Therefore, humans should expect to spend more time deliberating on the system impacts, which requires developing skills in ethical thinking and speculation. This is especially challenging, as predicting all the possible ways a particular system will be used, be coupled with other systems, and have an impact on the broader environment, is nearly impossible [124]. However, the cognitive and mental resources saved by AI can now be used to engage human System 2 thinking in trying to predict likely impacts (with some AI support). Guidance and support can come from the emerging areas of disciplined imagination and speculation about the future[125,126] and foundations in human values and ethics.[127,128]

Engaging broader stakeholders is one important avenue in foreseeing the possible human impacts, and AI and modeling can facilitate this process. In addition, long-horizon planning, emergent behavior modeling, feedback loops, multiple-dependency analysis, causal chains, and other complexity modeling approaches [43,129–134], along with the analytical and predictive capabilities of AI itself, can be used to better anticipate the direct, indirect, and delayed system impacts.

Building “critical conceptual models” - models of critical requirements and impact - can be done using simpler, lightweight modeling languages (see, e.g., [40,135–141]). Such languages abstract from nuanced semantics, permitting the essential rules to be captured clearly and efficiently. Still, simple languages should also strive to be accessible to broadest audiences, while also being unambiguous and precise - motivating future research on modeling efficiency (e.g., return on modeling effort) [18,34,142], language accessibility and universality [40,53,101,109], and semantic clarity (e.g., via foundational ontologies) [39,143–156].

Step 3: Develop the solution

AI is increasingly useful for developing various components of software. As these capabilities evolve and improve, the future will witness a continuous shift in the work typically performed by human developers versus AI.

The flexibility and increased versatility of modern AI tools facilitate the ability of humans to make independent and highly individual choices in what components of software to create. Models can further facilitate human development freedom by providing additional support to both humans and AI.

First, modeling can support prompt-driven engineering (or specific frameworks like vibe coding) [157]. Prompt-driven engineering permits humans to write code and create other components indirectly, via AI, using natural language prompts[8,158]. However, natural language is imprecise and ambiguous. Conceptual models can support prompt-driven engineering by increasing clarity and precision of natural language prompts[159]. Furthermore, formal conceptual models are typically superior to natural language for communicating spatial, sequential, and causal information[160]. Conceptual models drawn by humans can be sent to AI to provide greater precision of what a human wishes AI to do, leading to model-driven or model augmented prompt-driven engineering.

Second, models can be converted into software components directly. This idea traces back to CASE tools that emerged in the 1980s and permitted the development of software code based on the semantics captured in conceptual modeling representations[50,161–163]. It is also the core aspect of model-driven development[50,163] , underpinning component automation support, a key function of conceptual models. Modern AI is becoming even more capable of realizing the software component automation support function of conceptual modeling by building code and other components from conceptual models.

Model-driven development with AI support has strong potential in promoting development freedom. Some humans may find it more enjoyable, natural, or expedient to produce models, rather than code or issue AI prompts. We see evidence of this in the practice of model-driven development[50,163], and even more evidence in the wide popularity of drag-and-drop and no-code programming[164,165]. By holding development freedom as an immutable principle, we are open to any mode of development humans may choose to engage in. Given the value of modeling to humans, we especially call on academia and industry to continue improving model-driven development methods and tools.

Finally, models can support AI and human development by increasing the transparency and traceability of development processes. Modern AI continues to struggle with the problem of explainability, i.e., explaining the logic of its outputs, to diverse stakeholders[166,167]. The explainability problem has worsened with the creation and application of trillions of parameters that make up advanced large language models[3,168]. While efforts to make the decision logic of AI understandable (e.g., explainable AI) continue, human developers can improve the explainability of AI by making AI-driven development more transparent through modeling. For example, an AI's parameters and learned coefficients can be superimposed on or mapped to, an underlying enterprise model (e.g., entity-relationship diagram) to provide a visual representation of the data used in AI training and its impact on the AI model's performance [169,170]. This can reveal any gaps in the "AI developer" capabilities and world-view[171] that may not be evident from traditional approaches to AI explainability[166,172], and may precipitate fine-tuning or retraining the AI. Using modeling in this way would empower humans to work with AI on developing better, safer solutions.

Step 4. Evaluate developed and deployed solutions with AI and stakeholders

The efficiencies gained by working with AI permit continuous and comprehensive evaluation of the emerging solutions with broader stakeholders. First, AI can be used to continuously monitor development and evaluate emerging solutions. A new era of model-driven evaluation has arrived, where evaluation is guided by modeling and performed by AI - a combination of symbolic AI (e.g., theorem proving, model checking), statistical and machine learning methods, and generative AI techniques. Conceptual models such as process, data, value, legal, and causal diagrams – previously constructed by humans and AI – can be consulted by AI agents to ensure the solution's continued alignment with domain logic and intended outcome[173,174]. Here, the use of conceptual models, as opposed to natural language prompts, is especially important, as validation requires precision. By supporting these *validity checks* [173], models now act as both representational and evaluation tools.

Second, the ability of AI to scale and communicate with humans gains new uses, since AI could directly consult diverse stakeholders for verifying the emerging solution. The natural language protocols now perfected by Generative AI can turn the verification process into an engaging dialogue. These could be the same stakeholders from previous MADE steps or new ones, thereby increasing ecological and external validity [173] of the emerging software. This means the likelihood of the new solution performing well under a variety of circumstances increase[976]. The results of these verifications can be further structured into human-accessible models and be compared against the models in previous steps for inconsistencies. Automated assessment of models and solutions with traditional symbolic validation and verification techniques such as model-based testing [175] , model finding and simulation[176], model checking[177] can further improve the evaluation of critical MADE with AI systems.

Third, the validation efficiencies permitted by AI enable humans to focus on ensuring that the emerging solution complies with the mission-critical requirements. These efficiencies also allow humans to continue considering the broader impacts of the system, hopefully better anticipating some of the more indirect and delayed outcomes.

The co-developed software components may be deployed continuously as they pass validation or as one system after all validation checks have been passed. This choice depends on the nature of the project, such as its scale and time-to-market requirements. Additionally, post implementation evaluations are advisable to verify the system is behaving as expected.

As initial human-AI-made solutions take shape, they can spark new ways of thinking about the original problem, precipitating novel solutions. Therefore, repeating the first three steps of MADE with AI will be advantageous. This will involve pondering on the outcomes of the evaluation, coming to consensus with the relevant stakeholders on updated requirements and expected impacts, and developing new components together with AI.

Unleashing the Human-AI Symbiosis

In a world of increasingly powerful AI, new thinking is needed on how to develop software in an effective, transparent, responsible, and *safe* way according to social values, norms, and goals.

To build a better world with AI, we must fundamentally rethink the partnership between human modelers and AI. Following the inspiration from System 1 and System 2 thinking in humans, we present the Manifesto for Software Development and Modeling in the Age of Artificial Intelligence. At its heart are the guiding principles, SAFE, and a pathway for enacting them, MADE with AI.

Through SAFE-AI and MADE with AI, conceptual modeling emerges as a trusted bedrock of software development. It gives humans the space to reflect, a way to capture what matters, a tool for communication, and a method of supervising AI. AI can help us build models faster, but humans remain essential for interpreting them, grounding them in context, and ensuring they align with real-world needs. Far from being outdated, in the age of AI, modeling is not an agility obstacle – it is a safeguard.

Although we provided a roadmap with SAFE-AI and MADE with AI, more needs to be done to make this vision central to software development. We thus close this Manifesto with a call for action. It is a call to everyone concerned with the risks and opportunities unleashed by AI and fast-paced software development approaches. We propose the following areas of attention for the educator, research and practitioner communities to unite towards further research and development.

1. Education and Capacity Building

To prepare the next generation of software engineers and information systems professionals, education must evolve to reflect the dual demands of speed and deliberation in AI-augmented development. The skillset of software engineers is becoming versatile and generalistic. Coding remains an essential foundation, but it is no longer sufficient on its own. Software engineers must learn new competencies such as augmented development, which involves using AI-assisted tools to improve efficiency and decision-making. Critical thinking and quality assurance skills remain vital to ensure reliability and maintainability. Modeling should remain central in computing curricula as a means of cultivating reflective, ethical, and systematic thinking (what the Manifesto calls System 2 cognition). Students should learn not only to create models but also to evaluate, critique, and refine AI-generated artifacts using modeling as a form of reasoning and oversight. Educational programs should also foster cognitive diversity and creativity, encouraging learners to choose development and modeling methods that best align with their individual styles and reinforce their capacity to engage in deliberation and impact assessment.

In short:

1. Integrate modeling into AI-age curricula: Next-generation programmers and software engineers should be trained to understand the value of modeling as a safeguard against AI-generated risks.

2. Teach model evaluation and critical reasoning: Curricula should emphasize evaluating, validating, and refining human - and AI-generated models, not only creating them.
3. Embed modeling in AI-assisted workflows: Students should learn to incorporate conceptual modeling checkpoints within agile and AI-driven projects.
4. Promote cognitive diversity in modeling education: Encourage creativity and individual freedom in how developers use modeling to express, reason, and design alongside AI.
5. Train reflective practitioners: Develop courses that strengthen System 2 thinking, focusing on deliberation, ethical reflection, and impact critical assessment.

2. Research and Innovation

The research community must advance new theories, tools, and empirical evidence to guide human–AI collaboration in modeling and software development. Priority should be given to developing hybrid modeling frameworks that combine formal representations with natural-language prompting, and to supporting deliberate reasoning within AI-driven workflows. Conceptual modeling can also serve as the foundation for explainable AI, helping to visualize, trace, and justify AI decisions. Open and FAIR (Findable, Accessible, Interoperable, and Reusable) repositories of shared models [178], benchmarks, and case studies are needed to evaluate how modeling contributes to safety, transparency, and quality in AI-assisted software engineering. Moreover, the community should revive systems theory and complexity modeling to better anticipate indirect, delayed, and emergent consequences of software deployments. Finally, researchers should investigate how humans and AI can co-create and co-evaluate software, exploring the cognitive, social, and technical dynamics that underpin effective symbiosis between deliberate human reasoning and rapid AI generation.

In short:

1. Advance hybrid modeling frameworks: Researchers should explore AI-supported tools that activate System 2 thinking and allow formal models to guide natural-language prompts and AI code generation.
2. Develop model-driven explainable AI: Integrate conceptual models into AI pipelines to visualize, trace, and explain AI decision-making processes.
3. Create open repositories and benchmarks: Build shared datasets and reference models to assess how modeling enhances safety, transparency, and quality in AI-assisted software development.

4. Develop new and refine existing conceptual modeling languages, with particular attention to modeling efficiency, semantic clarity and accessibility.
5. Reinvestigate complexity and impact modeling: Revive and modernize systems theory and complexity modeling to better predict indirect, delayed, and emergent effects of software systems.
6. Study human-AI modeling collaboration: Examine how humans and AI can co-create software effectively, balancing cognitive strengths and limitations of both.

3. Professional and Industry Practice

Practitioners and industry leaders are urged to reestablish modeling as a vital component of fast-paced software development in the age of AI. To do so, we urge developers to adopt the SAFE-AI philosophy and build safer software using the MADE with AI processes.

Organizations should introduce conceptual model reviews alongside traditional code reviews to ensure deliberate reflection on system purpose, safety, and critical requirements. Participatory modeling approaches should be practiced more often, leveraging AI to facilitate the inclusion of diverse stakeholders in requirements elicitation and system design. Continuous, model-driven validation loops can be established where AI agents automatically assess emerging solutions against domain models and mission-critical constraints. Models should also serve as enduring repositories of organizational knowledge, providing a transparent record of system rationale, design evolution, and AI–human decision-making across development cycles.

In short:

1. Adopt the SAFE-AI philosophy and build safer software using the MADE with AI processes.
2. Institutionalize conceptual model reviews: Introduce formal model reviews alongside code reviews in agile and AI-augmented workflows to ensure deliberate reflection and alignment with critical requirements.
3. Adopt participatory modeling practices: Use AI as a facilitator to involve diverse stakeholders in modeling requirements and exploring potential system impacts.
4. Implement model-driven validation loops: Encourage continuous evaluation of systems by AI agents referencing human-created domain models to ensure alignment with intended goals.
5. Build integrated modeling environments: Develop seamless environments that combine modeling, prompting, simulation, and documentation to support iterative human-AI collaboration.

6. Preserve organizational knowledge through models: Promote models as living documentation that retain organizational memory in rapidly evolving AI-powered software ecosystems.

Finally, the global community would benefit from forming a cross-disciplinary consortium on "SAFE AI Development", uniting experts in software engineering, conceptual modeling, information systems, ethics, cognitive science and other related areas of sciences and humanities to continue refining the ideas laid out in this Manifesto for a safer and happier world for everyone!

References

1. Bornet, P. *et al*. *Agentic Artificial Intelligence: Harnessing AI Agents to Reinvent Business, Work and Life*. (Irreplaceable Publishing, NA, 2025).
2. Krishna, M., Gaur, B., Verma, A. & Jalote, P. Using llms in software requirements specifications: An empirical evaluation. in *2024 IEEE 32nd International Requirements Engineering Conference (RE)* 475–483 (IEEE, 2024).
3. Storey, V. C., Yue, W. T., Zhao, L. J. & Lukyanenko, R. Generative Artificial Intelligence: Evolving Technology, Growing Societal Impact, and Opportunities for Research. *Inf. Syst. Front.* (2025).
4. Yuan, D., Yang, G. & Zhang, T. UI2HTML: utilizing LLM agents with chain of thought to convert UI into HTML code. *Autom. Softw. Eng.* **32**, 1–24 (2025).
5. Nguyen, N. Microworkers Crowdsourcing Approach, Challenges and Solutions. in 1–1 (ACM, 2014).
6. Wu, Q. *et al.* Autogen: Enabling next-gen LLM applications via multi-agent conversations. in *First Conference on Language Modeling* (2024).
7. Kim, G. & Yegge, S. *Vibe Coding*. (IT Revolution, Portland, Oregon, 2025).
8. Liu, P. *et al.* Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.* **55**, 1–35 (2023).
9. Qiu, K., Puccinelli, N., Ciniselli, M. & Di Grazia, L. From today's code to tomorrow's symphony: The AI transformation of developer's routine by 2030. *ACM Trans. Softw. Eng. Methodol.* **34**, 1–17 (2025).
10. Pimenova, V., Fakhoury, S., Bird, C., Storey, M.-A. & Endres, M. Good Vibrations? A Qualitative Study of Co-Creation, Communication, Flow, and Trust in Vibe Coding. in *arXiv:2509.12491* (2025).
11. Johnson, J. *CHAOS 2020: Beyond Infinity*. https://www.standishgroup.com/news/49 (2020).
12. Beck, K. *et al.* Manifesto for agile software development. *Agile Alliance* https://agilemanifesto.org (2001).
13. Cockburn, A. *Agile Software Development: The Cooperative Game*. (Pearson Education, Hoboken, NJ, 2006).
14. Abou-Gabal, M. How to Reduce Lead-Times in long Devops Agile Projects. in 43–47 (Atlantis Press, 2022).

15. Kaufman, R. Understanding and Gauging Burnout in Agile Information Technology Teams. (2024).
16. Dovleac, R. The rise of DevQualOps and implications on software quality. *Int. J. Comput.* **8**, (2023).
17. Siddiqui, Z. CrowdStrike update that caused global outage likely skipped checks, experts say. *Reuters* (2024).
18. Lukyanenko, R. *et al.* Intelligent Agile: Conceptual Modeling and AI for Next-Generation Agile Software Development. in *HICSS 2026* 1–10 (Hawaii, 2026).
19. Storey, V. C., Lukyanenko, R. & Castellanos, A. Conceptual Modeling: Topics, Themes, and Technology Trends. *ACM Comput. Surv.* **55**, 1–38 (2023).
20. Beard, J. W. *et al.* Agile Development: The Promise, the Reality, the Opportunity. in *CAiSE 2024* (Cypress, 2024).
21. Romeo, J., Raglianti, M., Nagy, C. & Lanza, M. UML is Back. Or is it? Investigating the Past, Present, and Future of UML in Open Source Software. in *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)* 692–692 (IEEE Computer Society, 2025).
22. Atzeni, P. *et al.* The relational model is dead, SQL is dead, and I don't feel so good myself. *ACM SIGMOD Rec.* **42**, 64–68 (2013).
23. Tratt, L. Laurence Tratt: UML: My Part in its Downfall. https://tratt.net/laurie/blog/2022/uml_my_part_in_its_downfall.html?utm_source=chatgpt.com (2022).
24. Google. Google Trends: UML Trending. *Google Trends* https://trends.google.com/trends/explore?date=all&geo=US&q=UML&hl=en (2025).
25. Boronat, A. Code-first model-driven engineering: On the agile adoption of mde tooling. in *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)* 874–886 (IEEE, 2019).
26. Gupta, A., Poels, G. & Bera, P. Using Conceptual Models in Agile Software Development: A Possible Solution to Requirements Engineering Challenges in Agile Projects. *IEEE Access* **10**, 119745–119766 (2022).
27. Ambler, S. *Agile Modeling: Effective Practices for Extreme Programming and the Unified Process*. (John Wiley & Sons, Hoboken, NJ, 2002).
28. Tegarden, D., Samuel, B. M., Lukyanenko, R., Dennis, A. & Wixom, B. *Systems Analysis and Design: An Object-Oriented Approach with UML*. (Wiley, Hoboken, NJ, 2025).
29. Cito, J. & Bork, D. Lost in Code Generation: Reimagining the Role of Software Models in AI-driven Software Engineering. (2025).
30. Kahneman, D. *Thinking, Fast and Slow*. (Farrar, Straus and Giroux, New York, NY, USA, 2013).
31. Bork, D. Conceptual Modeling and Artificial Intelligence: Challenges and Opportunities for Enterprise Engineering. in *Enterprise Engineering Working Conference* 3–9 (Springer, 2022).
32. Storey, V. C. *et al.* Domain Knowledge in Artificial Intelligence: Using Conceptual Modeling to Increase Machine Learning Accuracy and Explainability. *Data Knowl. Eng.* 102482 (2025).
33. Storey, V. C. *et al.* Large language models for conceptual modeling: Assessment and application potential. *Data Knowl. Eng.* 102480 (2025).
34. Guizzardi, G. & Proper, H. A. On Understanding the Value of Domain Modeling. in *Proceedings of 15th International Workshop on Value Modelling and Business Ontologies (VMBO 2021)* (2021).
35. Moody, D. L. The "physics" of notations: toward a scientific basis for constructing visual notations in software engineering. *Softw. Eng. IEEE Trans. On* **35**, 756–779 (2009).

36. Siau, K. & Rossi, M. Evaluation techniques for systems analysis and design modelling methods – a review and comparative analysis. *Inf. Syst. J.* **21**, 249–268 (2011).
37. Thalheim, B. The theory of conceptual models, the theory of conceptual modelling and foundations of conceptual modelling. in *Handbook of Conceptual Modeling: Theory, Practice, and Research Challenges* 543–577 (Springer, 2011).
38. Thalheim, B. Modelology—The New Science, Life and Practice Discipline. in *Information Modelling and Knowledge Bases XXXV* 1–19 (IOS Press, Netherlands, 2024).
39. Wand, Y. & Weber, R. Research commentary: Information systems and conceptual modeling - A research agenda. *Inf. Syst. Res.* **13**, 363–376 (2002).
40. Lukyanenko, R. *et al.* Universal Conceptual Modeling: Principles, Benefits, and an Agenda for Conceptual Modeling Research. *Softw. Syst. Model.* 1–35 (2024).
41. Hills, T. *NoSQL and SQL Data Modeling: Bringing Together Data, Semantics, and Software*. (Technics Publications, 2016).
42. Chua, C. E. H., Indulska, M., Lukyanenko, R., Maass, W. & Storey, V. C. Data Management. *MIS Q. Online* 1–10 (2022).
43. Lukyanenko, R., Storey, V. C. & Pastor, O. System: A Core Conceptual Modeling Construct for Capturing Complexity. *Data Knowl. Eng.* **141**, 1–29 (2022).
44. Woo, C. The role of conceptual modeling in managing and changing the business. in *International Conference on Conceptual Modeling* 1–12 (Springer, 2011).
45. Pohl, K. The three dimensions of requirements engineering: a framework and its applications. *Inf. Syst.* **19**, 243–258 (1994).
46. Valle Sousa, I., Prince Sales, T., Guerra, E., Olavo Bonino da Silva Santos, L. & Guizzardi, G. What do I get from modeling? An empirical study on using structural conceptual models. in 21–38 (Springer, 2023).
47. Star, S. L. & Griesemer, J. R. Institutional ecology,translations' and boundary objects: Amateurs and professionals in Berkeley's Museum of Vertebrate Zoology, 1907-39. *Soc. Stud. Sci.* **19**, 387–420 (1989).
48. Verbruggen, C. & Snoeck, M. Practitioners' experiences with model-driven engineering: a meta-review. *Softw. Syst. Model.* **22**, 111–129 (2023).
49. Frankel, D. *Model Driven Architecture: Applying MDA to Enterprise Computing*. (John Wiley & Sons, Hoboken, NJ, 2003).
50. Pastor, O. & Molina, J. C. *Model-Driven Architecture in Practice: A Software Production Environment Based on Conceptual Modeling*. (Springer Science & Business Media, 2007).
51. Fettke, P. How conceptual modeling is used. *Commun. Assoc. Inf. Syst.* **25**, 43 (2009).
52. Dobing, B. & Parsons, J. How UML is used. *Commun. ACM* **49**, 109–113 (2006).
53. Michael, J., Bork, D., Wimmer, M. & Mayr, H. C. Quo vadis modeling. *Softw. Syst. Model.* (2023).
54. Recker, J., Rosemann, M., Indulska, M. & Green, P. Business process modeling-a comparative analysis. *J. Assoc. Inf. Syst.* **10**, 1 (2009).
55. Batra, D. Cognitive complexity in data modeling: causes and recommendations. *Requir. Eng.* **12**, 231–244 (2007).
56. Guizzardi, G. *et al.* Endurant types in ontology-driven conceptual modeling: Towards OntoUML 2.0. in 136–150 (Springer, 2018).
57. Parsons, J. Effects of Local Versus Global Schema Diagrams on Verification and Communication in Conceptual Data Modeling. *J. Manag. Inf. Syst.* **19**, 155–184 (2003).

58. Recker, J., Rosemann, M., Green, P. & Indulska, M. Do ontological deficiencies in modeling grammars matter? *MIS Q.* **35**, 57–79 (2011).
59. Samuel, B. M., Khatri, V. & Ramesh, V. Exploring the Effects of Extensional Versus Intentional Representations on Domain Understanding. *MIS Q.* **42**, 1187–1209 (2018).
60. Siau, K. & Rossi, M. Evaluation of information modeling methods-a review. in vol. 5 314–322 (IEEE, 1998).
61. Rasti, A. *et al.* Automated generation of smart contract code from legal contract specifications with Symboleo2SC. *Softw. Syst. Model.* **24**, (2025).
62. Sharifi, S., Parvizimosaed, A., Amyot, D., Logrippo, L. & Mylopoulos, J. Symboleo: Towards a specification language for legal contracts. in 364–369 (IEEE, 2020).
63. Onunga, J. & Mbugua, S. Model Driven Architecture: A Review of Current Trends. *Int. J. Comput. Sci. Mob. Appl.* (2023).
64. Shore, J. & Warden, S. *The Art of Agile Development*. (OReilly Media, Inc., Sebastopol, CA, 2021).
65. McDonald, K. J. *Beyond Requirements: Analysis with an Agile Mindset*. (Addison-Wesley Professional, 2015).
66. Archibald, N. & Kaplan, C. Passing the Security Vibe Check: The Dangers of Vibe Coding. *Databricks* https://www.databricks.com/blog/passing-security-vibe-check-dangers-vibe-coding (2025).
67. Rana, S. & Chicone, R. *Generative AI Security: Defense, Threats, and Vulnerabilities*. (John Wiley & Sons, Hoboken, NJ, 2025).
68. Zuboff, S. *In The Age Of The Smart Machine: The Future Of Work And Power*. (Basic Books, New York, NY, 1988).
69. Kahneman, D. & Tversky, A. Prospect Theory - Analysis of Decision under Risk. *Econometrica* **47**, 263–291 (1979).
70. Gilovich, T., Griffin, D. & Kahneman, D. *Heuristics and Biases: The Psychology of Intuitive Judgment*. (Cambridge University Press, Cambridge, UK, 2002).
71. Tversky, A. & Kahneman, D. Judgment under Uncertainty: Heuristics and Biases. *Science* **185**, 1124–1131 (1974).
72. Cheng, X. *et al.* Idea convergence quality in open innovation crowdsourcing: A cognitive load perspective. *J. Manag. Inf. Syst.* **37**, 349–376 (2020).
73. Parsons, J., Lukyanenko, R., Greenwood, B. N. & Cooper, C. B. Understanding and Improving Data Repurposing. *MIS Q.* **Forthcoming**, 1–40 (2025).
74. Bhuyan, B. P., Ramdane-Cherif, A., Tomar, R. & Singh, T. Neuro-symbolic artificial intelligence: a survey. *Neural Comput. Appl.* **36**, 12809–12844 (2024).
75. Sheth, A., Roy, K. & Gaur, M. Neurosymbolic AI--Why, What, and How. *ArXiv Prepr. ArXiv230500813* (2023).
76. Guizzardi, G., Pastor, O. & Storey, V. C. Thinking fast and slow in software engineering. *IEEE Softw.* **40**, 139–142 (2023).
77. Bender, E. M., Gebru, T., McMillan-Major, A. & Shmitchell, S. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? in 610–623 (2021).
78. Vaswani, A. *et al.* Attention is all you need. *Adv. Neural Inf. Process. Syst.* **30**, (2017).
79. Thoring, K., Huettemann, S. & Mueller, R. M. The augmented designer: a research agenda for generative AI-enabled design. *Proc. Des. Soc.* **3**, 3345–3354 (2023).
80. Xu, F. *et al.* Toward large reasoning models: A survey of reinforced reasoning with large language models. *Patterns* **6**, (2025).

81. Jaeger, H. Deep neural reasoning. *Nature* **538**, 467–468 (2016).
82. Li, Z.-Z. *et al.* From system 1 to system 2: A survey of reasoning large language models. *ArXiv Prepr. ArXiv250217419* (2025).
83. Licklider, J. C. Man-computer symbiosis. *IRE Trans. Hum. Factors Electron.* 4–11 (2008).
84. Guizzardi, R., Amaral, G., Guizzardi, G. & Mylopoulos, J. An ontology-based approach to engineering ethicality requirements. *Softw. Syst. Model.* **22**, 1897–1923 (2023).
85. CBC News. Judge allows lawsuit alleging AI chatbot pushed Florida teen to kill himself to proceed. *CBC News* (2025).
86. Ji, Z. *et al.* Survey of hallucination in natural language generation. *ACM Comput. Surv.* **55**, 1–38 (2023).
87. Meincke, L. *et al.* Call Me A Jerk: Persuading AI to Comply with Objectionable Requests. *Available SSRN* (2025).
88. Kaplan, J. *Generative Artificial Intelligence: What Everyone Needs to Know*. (Oxford University Press, 2024).
89. Bencomo, N. *et al.* Abstraction engineering. *ArXiv Prepr. ArXiv240814074* (2024).
90. Mumuni, F. & Mumuni, A. Explainable artificial intelligence (XAI): from inherent explainability to large language models. *ArXiv Prepr. ArXiv250109967* (2025).
91. Hevner, A. *et al.* Transparency in Design Science Research. *Decis. Support Syst.* **182**, 1–19 (2024).
92. Vaidya, J. & Asif, H. A critical look at ai-generate software: Coding with the new ai tools is both irresistible and dangerous. *Ieee Spectr.* **60**, 34–39 (2023).
93. Olivé, A. *Conceptual Modeling of Information Systems*. (Springer Science & Business Media, Berlin, Germany, 2007).
94. Davis, A., Dieste, O., Hickey, A., Juristo, N. & Moreno, A. M. Effectiveness of requirements elicitation techniques: Empirical results derived from a systematic review. in 179–188 (IEEE, 2006).
95. Davis, C. J., Fuller, R. M., Tremblay, M. C. & Berndt, D. J. Communication challenges in requirements elicitation and the use of the repertory grid technique. *J. Comput. Inf. Syst.* **46**, 78–86 (2006).
96. Pacheco, C., García, I. & Reyes, M. Requirements elicitation techniques: a systematic literature review based on the maturity of the techniques. *IET Softw.* **12**, 365–378 (2018).
97. Lukyanenko, R., Parsons, J., Wiersma, Y., Sieber, R. & Maddah, M. Participatory Design for User-generated Content: Understanding the challenges and moving forward. *Scand. J. Inf. Syst.* **28**, 37–70 (2016).
98. Bratteteig, T. & Wagner, I. *Disentangling Participation: Power and Decision-Making in Participatory Design*. (Springer International Publishing, New York, NY, 2014).
99. Björgvinsson, E., Ehn, P. & Hillgren, P.-A. Agonistic participatory design: working with marginalised social movements. *CoDesign* **8**, 127–144 (2012).
100. Hussain, S., Sanders, E. B.-N. & Steinert, M. Participatory design with marginalized people in developing countries: Challenges and opportunities experienced in a field study in Cambodia. *Int. J. Des.* **6**, (2012).
101. Lukyanenko, R., Bork, D., Storey, V. C., Parsons, J. & Pastor, O. Inclusive Conceptual Modeling: Diversity, Equity, Involvement, and Belonging in Conceptual Modeling. in *ER Forum 2023* 1–4 (Springer, Lisbon, Portugal, 2023).
102. Lukyanenko, R. *et al.* Representing Crowd Knowledge: Guidelines for Conceptual Modeling of User-generated Content. *J. Assoc. Inf. Syst.* **18**, 297–339 (2017).

103. Faik, I., Sengupta, A. & Deng, Y. Inclusion by design: Requirements elicitation with digitally marginalized communities. *MIS Q.* **48**, 219–244 (2024).
104. Lukyanenko, R., Maass, W. & Storey, V. C. Trust in artificial intelligence: From a Foundational Trust Framework to emerging research opportunities. *Electron. Mark.* **32**, 1993–2020 (2022).
105. Ethayarajh, K. & Jurafsky, D. Utility is in the eye of the user: A critique of NLP leaderboards. *ArXiv Prepr. ArXiv200913888* (2020).
106. Storey, V. C., Lukyanenko, R., Parsons, J. & Maass, W. Explainable AI: Opening the Black Box or Pandora's Box? *Commun. ACM* **66**, 1–6 (2022).
107. Harrison, M. I., Koppel, R. & Bar-Lev, S. Unintended consequences of information technologies in health care—an interactive sociotechnical analysis. *J. Am. Med. Inform. Assoc.* **14**, 542–549 (2007).
108. Parmentola, A. & Tutore, I. Industry 4.0 Technologies for Environmental Sustainability. *CSR Sustain. Ethics Gov.* (2023).
109. Bork, D., Klikovits, S., Michael, J., Netz, L. & Rumpe, B. Inclusive Model-Driven Engineering for Accessible Software. in *MODELS 2025 NIER - Int. Conf. on Model Driven Engineering Languages and Systems* (2025).
110. Carroll, S. *et al.* The CARE principles for indigenous data governance. *Data Sci. J.* **19**, (2020).
111. de Zulueta, P. C. Developing compassionate leadership in health care: an integrative review. *J. Healthc. Leadersh.* 1–10 (2015).
112. Stokes, F. & Palmer, A. Artificial intelligence and robotics in nursing: ethics of caring as a guide to dividing tasks between AI and humans. *Nurs. Philos.* **21**, e12306 (2020).
113. Jia, N., Luo, X., Fang, Z. & Liao, C. When and how artificial intelligence augments employee creativity. *Acad. Manage. J.* **67**, 5–32 (2024).
114. DeFalco, A. Towards a theory of posthuman care: Real humans and caring robots. *Body Soc.* **26**, 31–60 (2020).
115. Cao, Y., Yu, W., Ren, W. & Chen, G. An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Trans. Ind. Inform.* **9**, 427–438 (2012).
116. Jennings, N. R. Commitments and conventions: The foundation of coordination in multi-agent systems. *Knowl. Eng. Rev.* **8**, 223–250 (1993).
117. Lukyanenko, R., Parsons, J. & Samuel, B. M. Artifact Sampling: Using Multiple Information Technology Artifacts to Increase Research Rigor. in *Proceedings of the 51st Hawaii International Conference on System Sciences (HICSS 2018)* 1–12 (Big Island, Hawaii, 2018).
118. Lukyanenko, R., Parsons, J. & Samuel, B. M. Artifact Sampling in Experimental Conceptual Modeling Research. in *EmpER'18 - First International Workshop on Empirical Methods in Conceptual Modeling* (Xi'an, China, 2018).
119. Kohavi, R. & Thomke, S. The surprising power of online experiments. *Harv. Bus. Rev.* **95**, 74–87 (2017).
120. Kohavi, R. *et al.* Online controlled experiments at large scale. in 1168–1176 (ACM, ACM SIGKDD, 2013).
121. Cabot, J. Vibe Modeling: Challenges and Opportunities. in *International Conference on Conceptual Modeling* 105–118 (Springer, 2025).
122. Eisenhower, D. Remarks at the National Defense Executive Reserve Conference | The American Presidency Project. https://www.presidency.ucsb.edu/documents/remarks-the-national-defense-executive-reserve-conference (1957).

123. Snoeck, M. & Pastor, O. Teaching conceptual modelling in the age of LLMs: shifting from model creation to model evaluation skills. *Softw. Syst. Model.* 1–11 (2025) doi:10.1007/s10270-025-01307-z.
124. Bar-Yam, Y. Why complexity is different. *N. Engl. Complex Syst. Inst.* (2017).
125. Hovorka, D. & Mueller, B. Speculation: Forms and Functions. in *HICSS 2024* 6447–6457 (Hawaii, 2024).
126. Weick, K. E. Theory construction as disciplined imagination. *Acad. Manage. Rev.* 516–531 (1989).
127. Anscombe, G. E. M. Modern Moral Philosophy. *Philosophy* **33**, 1–19 (1958).
128. Rachels, J. & Rachels, S. *The Elements of Moral Philosophy 7e*. (McGraw Hill, New York NY, 2012).
129. Hoover, K. D. The discovery of long-run causal order: A preliminary investigation. *Econometrics* **8**, 31 (2020).
130. Johnson, S. *Emergence: The Connected Lives of Ants, Brains, Cities, and Software*. (Simon and Schuster, New York NY, 2002).
131. Bedau, M. A. & Humphreys, P. E. *Emergence: Contemporary Readings in Philosophy and Science*. (MIT Press, Cambridge, MA, 2008).
132. Bunge, M. A. *Emergence and Convergence: Qualitative Novelty and the Unity of Knowledge*. (University of Toronto Press, Toronto, ON, 2003).
133. von Bertalanffy, L. General system theory: Foundations, development, applications. *Braz. N. Y.* (1968).
134. Skyttner, L. *General Systems Theory: Ideas & Applications*. (World Scientific, New York, NY, 2001).
135. Lukyanenko, R., Samuel, B. M., Parsons, J., Storey, V. C. & Pastor, O. Datish: A Universal Conceptual Modeling Language to Model Anything by Anyone. in *ER Forum 2023* 1–14 (Springer, Lisbon, Portugal, 2023).
136. Amyot, D., Horkoff, J., Gross, D. & Mussbacher, G. A lightweight GRL profile for i* modeling. in 254–264 (Springer, 2009).
137. Bieliková, M. & Rástočný, K. Lightweight semantics over web information systems content employing knowledge tags. in 327–336 (Springer, 2012).
138. Lukyanenko, R. & Parsons, J. Lightweight Conceptual Modeling for Crowdsourcing. in *Conceptual Modeling* 508–511 (Springer, 2013).
139. Wüest, D., Seyff, N. & Glinz, M. FlexiSketch: a lightweight sketching and metamodeling approach for end-users. *Softw. Syst. Model.* **18**, 1513–1541 (2019).
140. Lukyanenko, R., Parsons, J. & Samuel, B. M. Representing Instances: The Case for Reengineering Conceptual Modeling Grammars. *Eur. J. Inf. Syst.* **28**, 68–90 (2019).
141. Castellanos, A., Tremblay, M., Lukyanenko, R. & Samuel, B. M. Basic Classes in Conceptual Modeling: Theory and Practical Guidelines. *J. Assoc. Inf. Syst.* **21**, 1001–1044 (2020).
142. Farias, K., Garcia, A., Whittle, J., Chavez, C. von F. G. & Lucena, C. Evaluating the effort of composing design models: a controlled experiment. *Softw. Syst. Model.* **14**, 1349–1365 (2015).
143. Almeida, J. P. A., Falbo, R. A. & Guizzardi, G. Events as entities in ontology-driven conceptual modeling. in *International Conference on Conceptual Modeling* 469–483 (Springer, 2019).
144. Guizzardi, G. *Ontological Foundations for Structural Conceptual Models*. (Telematics Instituut Fundamental Research Series, Enschede, The Netherlands, 2005).

145. Guizzardi, G. & Halpin, T. Ontological foundations for conceptual modelling. *Appl Ontol* **3**, 1–12 (2008).
146. Guarino, N., Guizzardi, G. & Mylopoulos, J. On the philosophical foundations of conceptual models. *Inf. Model. Knowl. Bases* **31**, 1 (2020).
147. Recker, J., Rosemann, M., Green, P. & Indulska, M. Do ontological deficiencies in modeling grammars matter? *MIS Q.* **35**, 57–79 (2011).
148. Lukyanenko, R., Storey, V. C. & Pastor, O. Foundations of information technology based on Bunge's systemist philosophy of reality. *Softw. Syst. Model.* **20**, 921–938 (2021).
149. Wand, Y. & Weber, R. Thirty Years Later: Some Reflections on Ontological Analysis in Conceptual Modeling. *J. Database Manag. JDM* **28**, 1–17 (2017).
150. Wand, Y. & Weber, R. On the ontological expressiveness of information systems analysis and design grammars. *Inf. Syst. J.* **3**, 217–237 (1993).
151. Smith, B. Beyond concepts: ontology as reality representation. in *International conference on formal ontology in information systems* 73–84 (Izmir, Turkey, 2004).
152. Mark, D. M., Smith, B. & Tversky, B. Ontology and Geographic Objects: An Empirical Study of Cognitive Categorization. in *Proceedings of the International Conference on Spatial Information Theory: Cognitive and Computational Foundations of Geographic Information Science* 283–298 (Springer-Verlag, London, UK, UK, 1999).
153. Burton-Jones, A. & Weber, R. Building conceptual modeling on the foundation of ontology. in *Computing handbook: information systems and information technology* 15.1-15.24 (Boca Raton, FL, United States, 2014).
154. Eriksson, O. & Agerfalk, P. J. Speaking things into existence: ontological foundations of identity representation and management. *Inf. Syst. J.* **35**, 1–30 (2021).
155. Herre, H. General Formal Ontology (GFO): A foundational ontology for conceptual modelling. in *Theory and applications of ontology: computer applications* 297–345 (Springer, 2010).
156. Guizzardi, G. *et al.* UFO: Unified foundational ontology. *Appl. Ontol.* **17**, 167–210 (2022).
157. Dijkstra, E. W. On the foolishness of "natural language programming". in *Program Construction: International Summer School* 51–53 (Springer, 2005).
158. Tony, C., Díaz Ferreyra, N. E., Mutas, M., Dhif, S. & Scandariato, R. Prompting techniques for secure code generation: A systematic investigation. *ACM Trans. Softw. Eng. Methodol.* **34**, 1–53 (2025).
159. Morales, S., Clarisó, R. & Cabot, J. Impromptu: A framework for model-driven prompt engineering. *Softw. Syst. Model.* 1–19 (2025).
160. Tversky, B. Spatial mental models. *Psychol. Learn. Motiv.* **27**, 109–145 (1991).
161. Hoppenbrouwers, S. J., Proper, H. E. & van der Weide, T. P. A fundamental view on the process of conceptual modeling. in 128–143 (Springer, 2005).
162. Davies, I., Green, P., Rosemann, M., Indulska, M. & Gallo, S. How do practitioners use conceptual modeling in practice? *Data Knowl. Eng.* **58**, 358–380 (2006).
163. France, R. & Rumpe, B. Model-driven development of complex software: A research roadmap. in *Future of Software Engineering (FOSE'07)* 37–54 (IEEE, 2007).
164. Elshan, E., Ebel, P., Söllner, M. & Leimeister, J. M. Leveraging Low Code Development of Smart Personal Assistants: An Integrated Design Approach with the SPADE Method. *J. Manag. Inf. Syst.* **40**, 96–129 (2023).
165. Albert, M., Muñoz, J., Pelechano, V. & Pastor, Ó. Model to text transformation in practice: generating code from rich associations specifications. in 63–72 (Springer, 2006).

166. Adadi, A. & Berrada, M. Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). *IEEE Access* **6**, 52138–52160 (2018).
167. Gunning, D. & Aha, D. W. DARPA's explainable artificial intelligence program. *AI Mag.* **40**, 44–58 (2019).
168. Zhao, H. *et al.* Explainability for large language models: A survey. *ACM Trans. Intell. Syst. Technol.* **15**, 1–38 (2024).
169. Lukyanenko, R. *et al.* Superimposition: Augmenting Machine Learning Outputs with Conceptual Models for Explainable AI. in *1st International Workshop on Conceptual Modeling Meets Artificial Intelligence and Data-Driven Decision Making* 1–12 (Springer, Vienna, Austria, 2020).
170. Maass, W., Castellanos, A., Tremblay, M. C., Lukyanenko, R. & Storey, V. C. ConceptSuperimposition: Using Conceptual Modeling Method for Explainable AI. in *AAAI Spring Symposium: MAKE 2022* 1–6 (Palm Springs, California, 2022).
171. Marcus, G. Generative AI's crippling and widespread failure to induce robust models of the world. *Marcus on AI* https://garymarcus.substack.com/p/generative-ais-crippling-and-widespread (2025).
172. Guidotti, R. *et al.* A survey of methods for explaining black box models. *ACM Comput. Surv. CSUR* **51**, 93 (2019).
173. Larsen, K. R. *et al.* Validity in Design Science. *MIS Q.* 1–40 (2025).
174. Whitman, L. & Huff, B. L. A living enterprise model. in (1997).
175. Dalal, S. R. *et al.* Model-based testing in practice. in *Proceedings of the 21st international conference on Software engineering* 285–294 (1999).
176. Braga, B. F., Almeida, J. P. A., Guizzardi, G. & Benevides, A. B. Transforming OntoUML into Alloy: towards conceptual model validation using a lightweight formal method. *Innov. Syst. Softw. Eng.* **6**, 55–63 (2010).
177. Chan, W. *et al.* Model checking large software specifications. *IEEE Trans. Softw. Eng.* **24**, 498–520 (1998).
178. Sales, T. P. *et al.* A FAIR catalog of ontology-driven conceptual models. *Data Knowl. Eng.* **147**, 102210 (2023).

Manifesto authors and their affiliations*

1. Roman Lukyanenko, University of Virginia, United States, romanl@virginia.edu
2. Binny M. Samuel, University of Cincinnati, United States, samuelby@uc.edu
3. David Tegarden, Virginia Tech, United States, david.tegarden@vt.edu
4. Kai R. Larsen, University of Colorado, United States, kai.larsen@colorado.edu
5. Araz Jabbari, Université Laval, Canada, araz.jabbari@fsa.ulaval.ca
6. Rasha Abd El Aziz, Arab Academy for Science, Technology & Maritime Transport (AASTMT), Alexandria, Egypt, rasha_a@aast.edu
7. João Paulo A. Almeida, Federal University of Espírito Santo, Brazil, jpalmeida@ieee.org

8. Alireza Amrollahi, Macquarie University, Australia, ali.amrollahi@mq.edu.au
9. Jon W. Beard, Iowa State University, United States, jwbeard@iastate.edu
10. Ladjel Bellatreche, LIAS/ISAE-ENSMA, Poitiers, France, ladjel.bellatreche@ensma.fr
11. Dominik Bork, TU Wien, Austria, dominik.bork@tuwien.ac.at
12. Jan vom Brocke, University of Münster, Germany, jan.vom.brocke@uni-muenster.de
13. Jordi Cabot, Luxembourg Institute of Science and Technology, Luxembourg, jordi.cabot@list.lu
14. Arturo Castellanos, William & Mary, United States, arturo.castellanosbueso@mason.wm.edu
15. Aurona Gerber, Stellenbosch University, South Africa, auronagerber@sun.ac.za
16. Peter Green, Queensland University of Technology, Australia, p.green@qut.edu.au
17. Andrea Grover, University of Nebraska at Omaha, United States, andreagrover@unomaha.edu
18. Giancarlo Guizzardi, University of Twente, Netherlands, g.guizzardi@utwente.nl
19. Attila Hertelendy, Florida International University, United States, ahertele@fiu.edu
20. Yuval Kahlon, Tokyo Institute of Technology, Japan, kahlon.y.aa@m.titech.ac.jp
21. Kamal Karlapalem, IIIT-Hyderabad, India, kamal@iiit.ac.in
22. Vijay Khatri, University of Colorado, United States, vijay.khatri@colorado.edu
23. Wolfgang Maass, Saarland University, Germany, wolfgang.maass@iss.uni-saarland.de
24. Chris Maurer, University of Virginia, United States, maurer@virginia.edu
25. Roland M. Mueller, Berlin School of Economics and Law, Germany, roland.mueller@hwr-berlin.de
26. John Mylopoulos, University of Toronto, Canada, jm@cs.toronto.edu
27. Rohit Nishant, Queen’s University Belfast, Northern Ireland, r.nishant@qub.ac.uk
28. Shawn Ogunseye, Bentley University, United States, sogunseye@bentley.edu
29. Guy Paré, HEC Montréal, Canada, guy.pare@hec.ca
30. Jeffrey Parsons, Memorial University of Newfoundland, Canada, jeffreyp@mun.ca
31. Oscar Pastor, Universitat Politècnica de València, Spain, opastor@dsic.upv.es
32. Julian Prester, The University of Sydney, Australia, julian.prester@sydney.edu.au
33. Henderik A. Proper, TU Wien, Vienna, Austria, e.proper@acm.org
34. Jolita Ralyte, University of Geneva, Switzerland, jolita.ralyte@unige.ch
35. Shazia Sadiq, University of Queensland, Australia, shazia@eecs.uq.edu.au
36. David Schuff, University of Virginia, United States, david.schuff@virginia.edu

37. Keng Siau, Singapore Management University, Singapore, klsiau@smu.edu.sg
38. Monique Snoeck, KU Leuven, Belgium, monique.snoeck@kuleuven.be
39. Veda C. Storey, Georgia State University, United States, vstorey@gsu.edu
40. Monica Tremblay, William & Mary, United States, monica.tremblay@mason.wm.edu
41. Juan Carlos Trujillo Mondéjar, Lucentia Research Group - University of Alicante, Spain, jtrujillo@dlsi.ua.es
42. Debra VanderMeer, Florida International University, United States, vanderd@fiu.edu
43. Ramesh Venkataraman, Indiana University, United States, venkat@iu.edu
44. Gerit Wagner, Otto-Friedrich-Universität Bamberg, Germany, gerit.wagner@uni-bamberg.de
45. Anna Wiedemann, Bern University of Applied Sciences, Switzerland, anna.wiedemann@bfh.ch
46. Carson Woo, The University of British Columbia, Canada, Carson.Woo@sauder.ubc.ca
47. Eric Yu, University of Toronto, Canada, eric@cs.toronto.edu
48. Wei Thoo Yue,City University of Hong Kong, Hong Kong, China, Wei.T.Yue@cityu.edu.hk
49. Leon Zhao, Chinese University of Hong Kong, China leonzhao@cuhk.edu.cn

*** *Notes*:**

1. After the first five authors, the authors are listed alphabetically;
2. The Manifesto upholds an inclusive view of authorship, the listed co-authors contributed to the Manifesto in one or more of the following ways:
    a. conceptualization of ideas,
    b. content writing,
    c. content editing,
    d. debating critical issues,
    e. presentation of Manifesto ideas to colleagues, students and practitioners and garnering their feedback,
    f. critical review and validation and endorsement of the claims and arguments.