AI-Driven Software Engineering – The Role of Conceptual Modeling

Hans-Georg Fill[*,a], Jordi Cabot[b], Wolfgang Maass[c], Marten van Sinderen[d]

[a] University of Fribourg, Research Group Digitalization and Information Systems, Boulevard de Pérolles 90, 1700 Fribourg, Switzerland
[b] Luxembourg Institute of Science and Technology and University of Luxembourg, Luxembourg
[c] Saarland University and German Research Center for Artificial Intelligence (DFKI), Germany
[d] University of Twente, Netherlands

Communicated by Peter Fettke. Received 2023-11-02. Accepted after 1 revision on 2023-12-05.

The following discussion paper summarizes the results of a panel discussion conducted on July 10, 2023 at the International Conference on Software Technologies (ICSOFT) in Rome, Italy. The panelists included Jordi Cabot from Luxembourg Institute of Science and Technology, Luxembourg, Wolfgang Maass from University of Saarland, Germany, and Marten van Sinderen from University of Twente, Netherlands. The panel was moderated by Hans-Georg Fill from University of Fribourg, Switzerland.

1 Introduction

Hans-Georg Fill

The application of artificial intelligence (AI) techniques to software engineering has a long history (Rich and Waters 1988). Researchers have long sought to provide programmers with human-like assistants to make their work more efficient and potentially automate the coding process. In the early stages of these attempts, conceptual modeling has been used for capturing and organizing knowledge in symbolic form, which could be ultimately fueled into knowledge bases and processed by machines (Mylopoulos et al. 1997). One of the main goals of conceptual modeling is however to enhance human understanding and communication in order to aid with the implementation of software systems (Mylopoulos 1992). While conceptual modeling is still being heavily used until today, for example in requirements engineering, software design and development, as well as in process and quality management and auditing (Brambilla et al. 2017; Dieste et al. 2001; Fettke 2009; Härer and Fill 2020; Michael et al. 2023; Muff et al. 2022; Sandkuhl et al. 2018), it may be questioned whether it is still necessary to manually create, organize and interpret such models in today's AI world.

The uprise of sub-symbolic knowledge representation approaches in the form of machine learning (ML) and deep learning (DL) in particular, led to a focus on data-driven AI approaches (LeCun et al. 2015). The reason behind these developments were the emergence of massive computing power, the collection of enormous data sets, as well as advances in algorithms (Kaynak 2021). Most recently, the invention of the Transformer neural network architecture and its application to language processing in the form of large language models (LLM) has sparked another surge in AI development (OpenAI 2023; Vaswani et al. 2017).

These developments led to numerous ideas and approaches on how data-based AI techniques could change the way how software is being created (Ma et al. 2023). Instead of using the traditional approach of expressing knowledge about software in the form of conceptual models, LLM-based approaches are currently being proposed

* Corresponding author.
E-mail. hans-georg.fill@unifr.ch

for all steps in the development life-cycle. This includes for example the use of natural language prompts for the elicitation and step-wise refinement of requirements, automated software design and code generation, or the improvement of code quality or refactoring (Ma et al. 2023; White et al. 2023). Even conceptual models themselves may be created or interpreted in this way without requiring an interaction with a traditional modeling tool (Cámara et al. 2023; Fill et al. 2023).

The question thus arises as to what can be expected from these novel AI methods for the field of software engineering and what limitations these methods may pose. Further, it needs to be investigated which role conceptual modeling will play in the future in these new processes and what developers, users, and contractors of future software systems should be trained on to achieve the maximum benefits.

2 AI, Data Models, and Data

Jordi Cabot

Jordi Cabot emphasized the importance of the data, and the associated data models, in an AI-driven world.

In the last decade, we have witnessed an explosion of research on new architectures, training methods, fine-tuning strategies, etc. for machine learning. But we are now entering a new phase where all these new approaches are becoming a commodity. Platforms like HuggingFace[1] do an outstanding job in making all the latest results accessible to everyone.

Therefore, using the latest ML architectures alone is not a competitive advantage anymore. Instead, companies need to turn their attention to the data used for training the ML models. Better data turns into better ML. This is why all companies are turning their attention to data. Data and data models are back in fashion and problems like data annotation, data mining, data composition, etc. now in an ML context, must be revisited.

We can see that the *golden triangle* resulting from the interweaving of data, conceptual models and AI reinforces each of them. For instance, let's see a relevant scenario as a representative of each combination:

- **Data + AI → Models**. We can use AI techniques to infer the data models representing the structure of the dataset and the behavioral models that could be used to Create/Read/Update/Delete instances of the dataset.
- **Data + Models → AI**. Properly annotated data – including ethical aspects (Giner-Miguelez et al. 2023) – can improve the quality of AI components trained on such data.
- **AI + Models → Data**. We can use AI techniques to synthesize new data compliant with a certain model structure.

As we will also discuss in Sect. 5, these scenarios impose new requirements for the conceptual modeling field. In this new AI age, models are not a static element in the development process, but they become dynamic as they often need to change and evolve to remain aligned with the data (and the data drifts). They are also partial (as they may represent only parts of the data) and uncertain (as we may not be completely sure of how accurate they are, e. g. when they are automatically inferred).

But despite these challenges, conceptual models remain a key asset. A good example of this is the promotion of the common European data spaces[2] to facilitate the data exchange among partners within a data domain. This exchange requires the partners to agree on a unified conceptual data model to ensure interoperability.

And let's not forget, ML models are also models.[3] This means that the conceptual modeling community has the chance to bring their expertise to the AI world, helping the AI community to improve the way they represent, transform, reuse and deploy ML artifacts. Looking forward to seeing how we can bring AI-based engineering to a whole new level thanks to our decades of expertise in conceptual modeling.

[1] https://huggingface.co/

[2] https://gaia-x.eu/what-is-gaia-x/deliverables/data-spaces/

[3] "Everything is a model" (Bézivin 2005)

3 AI for Humans vs. AI for AI

Wolfgang Maaß

Large software companies, such as Microsoft and Google, recognized early on that AI-based language models are excellent for supporting software developers. This is particularly significant against the backdrop of a global skills shortage. Powerful AI models, such as GPT4 and PaLM, show that AI models are increasingly becoming necessary tools for software developers. For example, Github Copilot is a fine-tuned GPT3 model trained on 54 million public Github repositories (Barke et al. 2023). The performance of DeepMind's AlphaCode system ranked in the top 54.3% among 5000 human programmers on the competitive programming platform Codeforces (Li et al. 2022).

The qualitative study by (Barke et al. 2023) suggests two distinct modes supported by the use of an LLM-based software development system (LLM-SE):

1. **Acceleration Mode**: the programmer knows what to do next and uses LLM-SE system to get to the goal faster.
2. **Exploration Mode**: the programmer is unsure how to proceed and uses LLM-SE system to explore options.

Feature-based explanatory approaches provide information on the importance of individual data attributes on predictions. Afterward, data attributes can be examined with respect to their importance, but also with respect to possible errors, such as biases in particular. However, this is mostly far away from the language of a domain expert, so that predictions of an AI model are very difficult to make.

Very large models, such as GPT4, Falcon 180B, Llama 2, and PaLM 2, span many domains and are particularly used for supporting software development. But these models are also used for conceptual designs, creation of visualizations, and providing explanations. Experiments show that the performance of software engineers who use LLM-based software development can increase significantly (Peng et al. 2023). This development gives reason to assume that very large ML models will be used for a wide area of software development, testing, and deployment tasks so that software engineers will more focus on the creative parts of software engineering including requirements engineering. With meta-learning capabilities (Schmidhuber 1993), ML models will also learn to self-refine and extend their software engineering capabilities.

Traditionally, conceptual modeling is one of the first phases of software development, when software engineers design and discuss potential solutions. Conceptual models are thereby developed as knowledge structuring artifacts to support the exchange of knowledge between involved persons and ultimately to develop a situational consensus. In contrast, AI models are created precisely with minimal use of a-priori representations and computational assumptions to avoid "hand-engineering" (Battaglia et al. 2018). At the same time, large AI models are black boxes, so results cannot be understood directly in terms of their origins and reasons. This leads to intensive research into explainable AI (Adadi and Berrada 2018). Since LLM models tend to fabricate facts, explanations are all the more important. Feature-based explanatory approaches provide information on the importance of individual data attributes on predictions. Afterward, data attributes can be examined with respect to their importance, but also with respect to possible errors, such as biases in particular. However, this is mostly far away from the language of a domain expert, so predictions of an AI model are very difficult to make from a domain expert's viewpoint.

Thus, on the one hand, a conceptual phase is omitted, but on the other hand, a knowledge and understanding gap in the use of AI models arises. This results in a new role for conceptual modeling in the usage phase of AI-based information systems as part of an AI explanation component that transforms the predictive behavior of ML models into conceptual structures in the language of domain experts (Maass et al. 2022a,b).

Future applications of conceptual modeling include:

1. Developing a domain understanding of the predictive behavior of ML (Machine Learning) models
2. Identifying potentials for optimizing ML models
3. Reviewing biases and legal frameworks
4. Increasing the re-usability of ML models
5. Identifying potentials for further development
6. Accelerating and optimizing development processes of AI (Artificial Intelligence)-based information systems.

In summary, the traditional use of conceptual models will be temporarily maintained to develop classic components such as user interfaces, middleware, and database functions. However, the implementation of business logic in software is already being replaced by AI models. Against this backdrop, the use of conceptual models is sustainably changing from a ***designing function*** to an ***explanatory function***.

4 Patterns for AI

Marten van Sinderen

We consider AI-driven software engineering as a subset of data-driven software engineering (DDSE), and we consider (traditional) conceptual modeling as the cornerstone of model-driven software engineering (MDSE). In this section, we discuss the characteristics of DDSE and MDSE and the relevance of both approaches with the aim of uncovering patterns that may prove useful in answering the question of what role conceptual modeling could play in AI-driven software engineering, and related questions formulated at the end of Sect. 1.

DDSE emphasizes the use of data to generate prediction models that are used in data-driven decision-making to improve efficiency of software development processes.

Technology advances related to networking and sensing started the explosion of digital data, often referred to as the big data movement (McAfee and Brynjolfsson 2012). Companies readily recognized the potential of data-driven decision-making that could turn them into data-driven organizations with improved performance. However, it was the rapid progress in parallel computing and machine learning (ML) that enabled practical applications and gave a powerful push to automated analysis and decision-making based on big data (Fradkov 2020).

With ML, complex prediction models can be learned from training data, without the need for programming based on a premeditated design. Applying these prediction models to input data allows for fast and accurate decision-making within the area of application for which the ML algorithm and model were optimized. That said, it is important to realize that ML has important application prerequisites and severe limitations when applied outside the boundaries of the training set (Carbone 2022). Specifically, the training data should be representative for the domain of application and should be sufficient in quantity and quality. The interpolation function that is used to fit the training data and generate the prediction model should be appropriate for the application at hand, which requires good prior knowledge of the application domain. Furthermore, the input data to which the model is applied should be within the interpolation window, representing phenomena that are within the range of expectations based on prior knowledge of the domain. If these conditions are not met, problems can arise that lead to unreliable predictions and risky decision-making. Especially in the social domain, such conditions are hard to control and guarantee, and therefore mistakes are easily made (Kleinberg et al. 2016). As can be observed from these characterizations, DDSE rests on generating descriptions of regularities found in training data in order to make predictions driven by unseen data, without providing explanations of why these regularities exist and why predictions

Data-driven Software Engineering	Model-driven Software Engineering
Data as starting point	Concepts as starting point
Complex prediction models can be learned automatically	Complex prescriptive models can be built with domain specialists
Models are as good as the data	Models are as good as the specialists' understanding of the domain
Models are weak on rare situations, due to scarcity of data	Models are abstractions, do not necessarily account for conditions of practice
Models are interpolations of what happened, without explanation why something happened and why interpolation makes sense	Models are conceptualizations of our understanding, and have to be proven with future data

Table 1: Data-driven Software Engineering (DDSE) and Model-Driven Software Engineering (MDSE) side-by-side

can be trusted (Guizzardi et al. 2023).

MDSE emphasizes the use of concepts to create explanatory and prescriptive models that are central artifacts to improve the effectiveness of software development processes.

Many model-based approaches for the design of software systems have been proposed in the literature, particularly aimed at establishing a shared understanding of the system under design and allowing control over the design project (Da Silva 2015; Mylopoulos 1992). Among these were also approaches that emphasized the distinction between a design model conceived through its domain concepts and a model representation in terms of modeling language elements (Van Sinderen et al. 1992). However, the term model-driven software engineering came in fashion after OMG introduced the Model Driven Architecture approach as a method to get more value from models in the software engineering process, using defined levels of abstraction, model transformations and metamodeling (Kent 2002; Sendall and Kozaczynski 2003). Nowadays, the term is used more loosely to indicate any software engineering approach that uses models as central artifacts in a controlled step-wise software engineering process.

MDSE starts with the creation of a conceptual model that captures goal-relevant features of a software system in terms of domain concepts, such that stakeholders can understand, communicate, and discuss the system under design. This model is typically the result of a laborious process in which real-world aspects are conceptualized by a team of professionals. It presents a consolidated view, constrained by human cognitive abilities and language expressivity. The model is abstract (ignoring details not deemed relevant so far), explanatory (the interaction of features can be understood in relation to goals), and prescriptive (serves as a specification for more detailed models) (Robinson et al. 2015). In the remaining design process, more detailed models are derived in orderly steps, possibly using semi-automated transformations, until a straightforward mapping onto software code is within reach. The data that is generated by testing and evaluating the realized software system can subsequently be used to validate and evolve the (reusable) models. In short, MDSE is based on handcrafting a model of desirable features in terms of domain concepts, as explicit descriptions of regularities in the real world, in order to develop software systems that can be explained in terms of domain concepts and justified with reference to stakeholder goals and decisions. Tab. 1 summarizes the characteristics of DDSE and MDSE side-by-side.

In order to further illustrate the different nature of the data-driven and model-driven approaches, we can use the following example of human knowledge acquisition as an analogy: Suppose a person

Consequences DDSE	Consequences MDSE
Achieves impressive results when interpolating between historical facts (represented by training data), but cannot generalize beyond these facts	Provides a general grounding in real-world semantics, but goal-driven design choices have to be confirmed by future facts (represented by data from testing, validation, and evaluation)
Software generation for prediction and decision-making based on training data and generic prediction models	Software development requires considerable human expertise and effort
Prediction models provide no explanation, results cannot connected to domain goals	Conceptual models allow understanding in terms of domain concepts
Applications can be problematic in ethical/responsible software systems	Applications allow consideration of human and societal values

Table 2: Consequences of Data-driven Software Engineering (DDSE) and Model-Driven Software Engineering (MDSE) Characteristics

wants to start driving a car, and therefore has to acquire the knowledge for making decisions in traffic situations, including hazard recognition, traffic rules and understanding the traffic. Two extreme approaches can be distinguished:

- Learn from observing car traffic behavior, and
- Learn from theory lessons and instructions.

With the first approach, numerous observations need to be done and properly memorized, each relating to how a car behaves in a certain situation. Given a sufficient number of observations, certain patterns will emerge, such as in which situations a car gives priority and in which situations it is given priority. Once these patterns are internalized by our learner, she can quickly act in situations to which the patterns apply. This means that she will act correctly and efficiently in common situations since these are well represented by the observations. However, there may be many rare situations, for which no or few observations exist. In these cases, there is a high risk of incorrect or lacking actions, with possible negative effects.

With the second approach, the lessons and instructions will give insights in which traffic rules exist and why they are necessary for safe driving. The rules collectively define how to behave in any situation, given realistic assumptions on the domain. Once the pupil has learned the rules and passed the tests, she can act correctly in all situations, including rare situations. This means that the actions are effective, producing the intended result, but not necessarily efficient, because of lack of experience.

Of course, we want both: act quickly (efficient) and act correctly (effective), so both approaches are needed, combining experience-based knowledge and theoretical knowledge. Luckily this is the case in real life, where one must pass a theory test for obtaining a driving license and can improve skills by driving in practice. The analogy between the first approach and SSDE, and between the second approach and MDSE, though possibly flawed in many respects, does highlight some key points of DDSE and MDSE:

- DDSE helps to improve the efficiency of the software development process by automatic generation of prediction models based on training data, which can have impressive results, but results may be unreliable if used in situations for which the training data is not representative;
- MDSE builds on domain knowledge and helps to develop software that effectively takes goals (including human values) into account, traceable to the knowledge base and stakeholder decisions, but development and maintenance requires considerable human expertise and effort.

In line with this, DDSE and MDSE are respectively viewed as fast and slow strategies for software development (Guizzardi et al. 2023), bearing similarity to the thinking systems distinguished by Daniel Kahneman (Kahneman 2011).

Currently, we observe an increasing interest in explainable AI (XAI), fueled by the growing awareness that many applications of AI/data-driven software engineering require explanations in order to avoid unfair outcomes. We expect that ontology-based conceptual models will play an important role here, in order to provide meaning and clarification of the data and processes (e. g., of the ML pipeline) and to understand whether outcomes can be trusted and decisions would be fair and unbiased (Maass and Storey 2021).

We conclude that DDSE and MDSE are complementary rather than alternative approaches to software engineering: DDSE does not replace MDSE, nor does it make traditional conceptual modeling useless or redundant.The Tab. 2 summarizes some of the major consequences of the characteristics of DDSE and MDSE that underline this conclusion.

5 Future Challenges and Opportunities for Conceptual Modeling in AI

Following the summaries of the panel contributions from above, we can derive a number of challenges and opportunities for conceptual modeling in times of AI-driven software engineering. First, it is our shared belief that conceptual modeling could play a key role in all the aspects of AI-driven software engineering mentioned above - however, it yet has to find its role here – see also the considerations expressed in (Fettke 2020) on the relation of AI and modeling. In essence, modeling has the potential to help in any discipline, beyond software itself (Cabot and Vallecillo 2022; Fill et al. 2021). However, the use of conceptual modeling is often impeded with several challenges, including for example the large effort in creating models by hand and thus the scalability of modeling activities, the involvement of non-experts in modeling, or a range of technical challenges when using models in collaborative settings (Bucchiarone et al. 2020; Sandkuhl et al. 2018).

The emergence of large language models (LLMs) at a quality level of GPT-3 and GPT-4 will offer new opportunities for all aspects of developing software. In contrast to previous approaches that regarded for example the use of data and artificial intelligence techniques for intelligent modeling assistance (Burgueño et al. 2022; Mussbacher et al. 2020), or AI in different stages of software engineering (Barenkamp et al. 2020), LLMs have the potential to enable more natural interactions and can at the same time be flexibly adapted to multiple scenarios.

To play this key role, we believe that there are a number of open challenges that our community should work on in the future. We describe some of them:

- Finding the ***right level of abstraction for AI concepts***, such as different types of learning and reasoning approaches, involved data sets, representation formalisms, etc. Grady Booch famously said that the entire history of software engineering is one of rising levels of abstraction (Booch 2018). We need to agree on the right domain-specific languages, notations, model transformations, etc. to specify and manipulate the representations of AI concepts of this new breed of AI-enhanced software systems.
- ***Modeling beyond data aspects***. In AI systems, there are other dimensions that are key to understanding the limitations of the system. In particular, we believe that *uncertainty modeling* (Troya et al. 2021) should be considered a first-level concern – as AI systems are full of uncertainty at all levels, i. e. the data level and the ML model level. And so, this uncertainty propagates to the rest of the system. Conceptual modeling languages should include uncertainty primitives to properly model this aspect.
- ***Modeling for explainability of black-box AI systems***. Conceptual modeling and in particular ontologies are expected to play a major

role in contributing to the explainability of AI systems - see also the ongoing discussions in the context of the AAAI-MAKE symposia (Proceedings of the AAAI 2021 Spring Symposium on Combining Machine Learning and Knowledge Engineering (AAAI-MAKE 2021), Stanford University 2021; Proceedings of the AAAI 2022 Spring Symposium on Machine Learning and Knowledge Engineering for Hybrid Intelligence (AAAI-MAKE 2022), Stanford University 2022; Proceedings of the AAAI 2023 Spring Symposium on Challenges Requiring the Combination of Machine Learning and Knowledge Engineering (AAAI-MAKE 2023) 2023). Thereby, conceptual models can contribute the domain knowledge to facilitate the user-specific, human understanding of explanations of black-box models (Confalonieri et al. 2021). Especially the assumed performance-explainability trade-off of current learning techniques – often the most effective AI techniques are most difficult to explain – has to be considered (Gunning and Aha 2019).

- ***Economic aspects of modeling***. We know that the adoption of modeling in companies and organizations is a complex sociotechnical problem (Hutchinson et al. 2014; Sandkuhl et al. 2018). This is aggravated when looking at the types of multidisciplinary teams behind the development of AI systems. Therefore, we need to propose new theories to emphasize and calculate the value of modeling, especially the ROI (return of investment) of modeling practices and investigate how novel AI-driven approaches may help in this regard (Fill et al. 2023).
- ***Intellectual property, ethical, and legal aspects***. Data-driven approaches demand large amounts of high-quality data for their training. In this context, questions of ethics and intellectual property in regard to the correct use of data need to be investigated (Strowel 2023). This applies in particular to potentially used models (Martinez et al. 2019) and the traceability of configurations of machine learning pipelines in terms of *transparency of the provenance of data*, e. g. (Fill and Härer 2020), and the clarification of results in terms of domain concepts, i. e. the correct *interpretability*.
- ***Teaching data-driven and model-driven software engineering***. As both directions are not disjoint but rather complement each other in various ways, it becomes important to acknowledge for this in teaching and thus adapt existing computer science curricula (Rosenthal et al. 2023). This includes for example teaching objectives such as the efficient and effective use of both techniques, the knowledge about trade-offs in precision vs. human effort, or the limitations in terms of ethical aspects.

References

Adadi A., Berrada M. (2018) Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). In: IEEE access 6, pp. 52138–52160

Barenkamp M., Rebstadt J., Thomas O. (2020) Applications of AI in classical software engineering. In: AI Perspectives 2(1), pp. 1–15

Barke S., James M. B., Polikarpova N. (2023) Grounded copilot: How programmers interact with code-generating models. In: Proceedings of the ACM on Programming Languages 7(OOPSLA1), pp. 85–111

Battaglia P. W., Hamrick J. B., Bapst V., Sanchez-Gonzalez A., Zambaldi V., Malinowski M., Tacchetti A., Raposo D., Santoro A., Faulkner R., et al. (2018) Relational inductive biases, deep learning, and graph networks. In: arXiv preprint arXiv:1806.01261

Bézivin J. (2005) On the unification power of models. In: Softw. Syst. Model. 4(2), pp. 171–188

Booch G. (2018) The history of software engineering. In: IEEE Software 35(5), pp. 108–114

Brambilla M., Cabot J., Wimmer M. (2017) Model-driven software engineering in practice. Morgan & Claypool Publishers

Bucchiarone A., Cabot J., Paige R. F., Pierantonio A. (2020) Grand challenges in model-driven engineering: an analysis of the state of the research. In: Softw. Syst. Model. 19(1), pp. 5–13

Burgueño L., Cabot J., Wimmer M., Zschaler S. (2022) Guest editorial to the theme section on AI-enhanced model-driven engineering. In: Softw. Syst. Model. 21(3), pp. 963–965

Cabot J., Vallecillo A. (2022) Modeling should be an independent scientific discipline. In: Softw. Syst. Model. 21(6), pp. 2101–2107

Cámara J., Troya J., Burgueño L., Vallecillo A. (2023) On the assessment of generative AI in modeling tasks: an experience report with ChatGPT and UML. In: Softw. Syst. Model. 22(3), pp. 781–793

Carbone M. (2022) When not to use machine learning: a perspective on potential and limitations. In: MRS Bulletin 47, pp. 968–974

Confalonieri R., Weyde T., Besold T. R., del Prado Martín F. M. (2021) Using ontologies to enhance human understandability of global post-hoc explanations of black-box models. In: Artif. Intell. 296, p. 103471

Da Silva A. R. (2015) Model-driven engineering; a survey supported by the unified conceptual model. In: Computer language, Systems & Structures 43, pp. 139–155

Dieste O., Juristo N., Moreno A. M., Pazos J., Sierra A. (2001) Conceptual modeling in software engineering and knowledge engineering: Concepts, Techniques and trends. In: Handbook of Software Engineering and Knowledge Engineering: Volume I: Fundamentals. World Scientific, pp. 733–766

Fettke P. (2009) How conceptual modeling is used. In: Communications of the Association for Information Systems 25(1), p. 43

Fettke P. (2020) Conceptual Modelling and Artificial Intelligence: Overview and research challenges from the perspective of predictive business process management. In: Modellierung 2020, Vienna, Austria Vol. 2542, pp. 157–164

Fill H., Fettke P., Köpke J. (2023) Conceptual Modeling and Large Language Models: Impressions From First Experiments With ChatGPT. In: Enterp. Model. Inf. Syst. Archit. Int. J. Concept. Model. 18, p. 3

Fill H., Härer F. (2020) Supporting Trust in Hybrid Intelligence Systems Using Blockchains. In: Proceedings of the AAAI 2020 Spring Symposium on Combining Machine Learning and Knowledge Engineering in Practice, AAAI-MAKE 2020. CEUR-WS.org

Fill H., Härer F., Muff F., Curty S. (2021) Towards Augmented Enterprise Models as Low-Code Interfaces to Digital Systems. In: Business Modeling and Software Design - 11th International Symposium Vol. 422. Springer, pp. 343–352

Fradkov A. (2020) Early history of machine learning. In: IFAC PapersOnLine 53(2), pp. 1385–1390

Giner-Miguelez J., Gómez A., Cabot J. (2023) A domain-specific language for describing machine learning datasets. In: Journal of Computer Languages 76, p. 101209

Guizzardi G., Pastor O., Storey V. (2023) Thinking fast and slow in software engineering. In: IEEE Software 40(6) forthcoming, pp. 1–3

Gunning D., Aha D. W. (2019) DARPA's Explainable Artificial Intelligence (XAI) Program. In: AI Mag. 40(2), pp. 44–58

Härer F., Fill H. (2020) Past Trends and Future Prospects in Conceptual Modeling - A Bibliometric Analysis. In: Conceptual Modeling - 39th International Conference, ER 2020, Vienna, Austria, November 3-6, 2020, Proceedings. Lecture Notes in Computer Science Vol. 12400. Springer, pp. 34–47

Hutchinson J. E., Whittle J., Rouncefield M. (2014) Model-driven engineering practices in industry: Social, organizational and managerial factors that lead to success or failure. In: Sci. Comput. Program. 89, pp. 144–161

Kahneman D. (2011) Thinking, fast and slow. Farrar, Straus and Giroux

Kaynak O. (2021) The golden age of Artificial Intelligence: Inaugural Editorial. In: Discover Artificial Intelligence 1, pp. 1–7

Kent S. (2002) Model driven engineering. In: Integrated Formal Methods. Lecture Notes in Computer Science Vol. 2335. Springer, pp. 286–298

Kleinberg J., Ludwig J., Mullainathan S. (2016) A guide to solving social problems with machine learning. In: Harvard Business Review Available online: https://hbr.org/2016/12/a-guide-to-solving-social-problems-with-machine-learning (accessed on 13 September 2023)

LeCun Y., Bengio Y., Hinton G. (2015) Deep learning. In: Nature 521(7553), pp. 436–444

Li Y., Choi D., Chung J., Kushman N., Schrittwieser J., Leblond R., Eccles T., Keeling J., Gimeno F., Dal Lago A., et al. (2022) Competition-level code generation with alphacode. In: Science 378(6624), pp. 1092–1097

Ma W., Liu S., Wang W., Hu Q., Liu Y., Zhang C., Nie L., Liu Y. (2023) The Scope of ChatGPT in Software Engineering: A Thorough Investigation. In: (arXiv:2305.12138) arXiv:2305.12138 [cs]

Maass W., Castellanos A., Tremblay M. C., Lukyanenko R., Storey V. C. (2022a) AI Explainability: Embedding Conceptual Models. In: Proceedings of the 43rd International Conference on Information Systems, ICIS 2022

Maass W., Castellanos A., Tremblay M. C., Lukyanenko R., Storey V. C. (2022b) Concept Superimposition: Using Conceptual Modeling Method for Explainable AI.. In: AAAI Spring Symposium: MAKE, pp. 1–6

Maass W., Storey V. C. (2021) Pairing conceptual modeling with machine learning. In: Data & Knowledge Engineering 134, p. 101909

Martinez S., Gerard S., Cabot J. (2019) On the Need for Intellectual Property Protection in Model-Driven Co-Engineering Processes. In: 24th International Conference, EMMSAD 2019. Springer, pp. 169–177

McAfee A., Brynjolfsson E. (2012) Big data: the management revolution. In: Harvard Business Review 90(10), pp. 60–68

Michael J., Bork D., Wimmer M., Mayr H. C. (2023) Quo Vadis modeling? In: Software and Systems Modeling

Muff F., Härer F., Fill H. (2022) Trends in Academic and Industrial Research on Business Process Management - A Computational Literature Analysis. In: 55th Hawaii International Conference on System Sciences, HICSS 2022, pp. 1–10

Mussbacher G., Combemale B., Kienzle J., Abrahão S., Ali H., Bencomo N., Búr M., Burgueño L., Engels G., Jeanjean P., Jézéquel J., Kühn T., Mosser S., Sahraoui H. A., Syriani E., Varró D., Weyssow M. (2020) Opportunities in intelligent modeling assistance. In: Softw. Syst. Model. 19(5), pp. 1045–1053

Mylopoulos J. (1992) Conceptual modelling and Telos In: Conceptual Modeling, Databases, and CASE: An Integrated View of Information Systems Development John Wiley & Sons, Inc., pp. 49–68

Mylopoulos J., Borgida A., Yu E. (1997) Representing software engineering knowledge. In: Automated Software Engineering 4, pp. 291–317

OpenAI (2023) GPT-4 Technical Report. arXiv preprint arXiv:2303.08774 [cs.CL]

Peng S., Kalliamvakou E., Cihon P., Demirer M. (2023) The impact of ai on developer productivity: Evidence from github copilot. In: arXiv preprint arXiv:2302.06590

Proceedings of the AAAI 2021 Spring Symposium on Combining Machine Learning and Knowledge Engineering (AAAI-MAKE 2021), Stanford University. CEUR Workshop Proceedings Vol. 2846. CEUR-WS.org

Proceedings of the AAAI 2022 Spring Symposium on Machine Learning and Knowledge Engineering for Hybrid Intelligence (AAAI-MAKE 2022), Stanford University. CEUR Workshop Proceedings Vol. 3121. CEUR-WS.org

Proceedings of the AAAI 2023 Spring Symposium on Challenges Requiring the Combination of Machine Learning and Knowledge Engineering (AAAI-MAKE 2023). CEUR Workshop Proceedings Vol. 3433. CEUR-WS.org

Rich C., Waters R. C. (1988) The programmer's apprentice: A research overview. In: Computer 21(11), pp. 10–25

Robinson S., Arbez G., Birta L., Tolk A., Wagner G. (2015) Conceptual modelling: definition, purpose and benefits. In: 2015 Winter Simulation Conference. IEEE, pp. 2812–2826

Rosenthal K., Strecker S., Asensio E. S., Snoeck M. (2023) Guest Editorial to the Special Issue on Teaching and Learning Conceptual Modeling. In: Enterprise Modelling and Information Systems Architectures (EMISAJ) – International Journal of Conceptual Modeling 18

Sandkuhl K., Fill H., Hoppenbrouwers S., Krogstie J., Matthes F., Opdahl A. L., Schwabe G., Uludag Ö., Winter R. (2018) From Expert Discipline to Common Practice: A Vision and Research Agenda for Extending the Reach of Enterprise Modeling. In: Bus. Inf. Syst. Eng. 60(1), pp. 69–80

Schmidhuber J. (1993) A neural network that embeds its own meta-levels. In: IEEE International Conference on Neural Networks. IEEE, pp. 407–412

Sendall S., Kozaczynski W. (2003) Model driven transformation: the heart and soul of model-driven development. In: IEEE Software 20(5), pp. 42–45

Strowel A. (2023) ChatGPT and Generative AI Tools: Theft of Intellectual Labor? In: IIC - International Review of Intellectual Property and Competition Law 54(4), pp. 491–494

Troya J., Moreno N., Bertoa M. F., Vallecillo A. (2021) Uncertainty representation in software models: a survey. In: Softw. Syst. Model. 20(4), pp. 1183–1213

Van Sinderen M., Ferreira Pires L., Vissers C. (1992) Protocol design and implementation using formal methods. In: The Computer Journal 35(5), pp. 478–491

Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser Ł., Polosukhin I. (2017) Attention is All you Need. In: Advances in Neural Information Processing Systems Vol. 30. Curran

White J., Hays S., Fu Q., Spencer-Smith J., Schmidt D. C. (Mar. 2023) ChatGPT Prompt Patterns for Improving Code Quality, Refactoring, Requirements Elicitation, and Software Design. In: (arXiv:2303.07839) arXiv:2303.07839 [cs]