Software and Systems Modeling (2020) 19:1045–1053
https://doi.org/10.1007/s10270-020-00814-5

EXPERT VOICE

Opportunities in intelligent modeling assistance

Gunter Mussbacher[1] · Benoit Combemale[2] · Jörg Kienzle[1] · Silvia Abrahão[3] · Hyacinth Ali[1] · Nelly Bencomo[4] · Márton Búr[1] · Loli Burgueño[5,6] · Gregor Engels[7] · Pierre Jeanjean[8] · Jean-Marc Jézéquel[8] · Thomas Kühn[9] · Sébastien Mosser[10] · Houari Sahraoui[11] · Eugene Syriani[11] · Dániel Varró[1] · Martin Weyssow[11]

Received: 3 June 2020 / Accepted: 5 June 2020 / Published online: 17 July 2020

Abstract
Modeling is requiring increasingly larger efforts while becoming indispensable given the complexity of the problems we are solving. Modelers face high cognitive load to understand a multitude of complex abstractions and their relationships. There is an urgent need to better support tool builders to ultimately provide modelers with intelligent modeling assistance that learns from previous modeling experiences, automatically derives modeling knowledge, and provides context-aware assistance. However, current intelligent modeling assistants (IMAs) lack adaptability and flexibility for tool builders, and do not facilitate understanding the differences and commonalities of IMAs for modelers. Such a patchwork of limited IMAs is a lost opportunity to provide modelers with better support for the creative and rigorous aspects of software engineering. In this expert voice, we present a conceptual reference framework (RF-IMA) and its properties to identify the foundations for intelligent modeling assistance. For tool builders, RF-IMA aims to help build IMAs more systematically. For modelers, RF-IMA aims to facilitate comprehension, comparison, and integration of IMAs, and ultimately to provide more intelligent support. We envision a momentum in the modeling community that leads to the implementation of RF-IMA and consequently future IMAs. We identify open challenges that need to be addressed to realize the opportunities provided by intelligent modeling assistance.

Keywords Model-based software engineering · Intelligent modeling assistance · Integrated development environment · Artificial intelligence · Development data · Feedback

1 Introduction

Over the last decades, modeling activities have been applied across the whole life-cycle of complex software-intensive systems to support all stakeholders involved in software development, mostly thanks to the use of abstractions—provided by general-purpose and domain-specific modeling languages—and separation of concerns. Modeling is continually requiring larger efforts and it is often indispensable in tackling the constantly increasing intrinsic complexity of such systems [11,13]. Owing to (i) the ever-increasing complexity of the problems that modelers are trying to solve, (ii) the growing number of stakeholders whose needs have to be addressed, and (iii) the increase in domain-specific modeling abstractions used by these stakeholders, modelers face higher and higher cognitive load. Modelers need to handle a multitude of specific abstractions, their use across the software life-cycle, and their relationships with other abstractions to ensure global consistency of the modeling activities and resulting modeling artifacts [44]. There is an urgent need to better support tool builders to ultimately provide modelers with more intelligent modeling assistance.

✉ Benoit Combemale
benoit.combemale@irisa.fr

1 McGill University, Montreal, Canada
2 Université de Toulouse and Inria, Toulouse, France
3 Universitat Politècnica de València, Valencia, Spain
4 Aston University, Birmingham, UK
5 Universitat Oberta de Catalunya, Barcelona, Spain
6 CEA LIST, Palaiseau, France
7 Paderborn University, Paderborn, Germany
8 Inria, CNRS, IRISA, Université de Rennes, Rennes, France
9 Karlsruher Institut für Technologie, Karlsruhe, Germany
10 Université du Québec à Montréal, Montreal, Canada
11 Université de Montréal, Montreal, Canada

In parallel, in our modern era of information technology, data has become increasingly prevalent in model-driven practices thanks to the explosion of data available and observed from different modeling activities. There is an opportunity to provide intelligent assistance to the modeler that learns from, e.g., previous modeling experiences, historical information contained in model repositories, surrounding context of the modeler, and domain-specific knowledge. Progress in data-driven techniques [9], such as data mining and machine learning [15], enables the automatic derivation of modeling knowledge and the provision of context-aware assistance. The dichotomy and complementary nature of data and models are increasingly apparent in all phases of the software development life-cycle [26], i.e., not only during system use but also during system development, maintenance, and posterior evolution. Other contributing factors are (i) the development of modern data-driven software systems, which potentially involve different interdisciplinary perspectives [20,29], and (ii) the role of recent black-box techniques, such as machine learning, on understandability [34].

Inevitably, all the above have increased both the complexity of the problems where modeling techniques are used and the consequent demand for support for working with large models [7]. Supporting modelers with a certain level of assistance is of outermost importance to optimally leverage available data with the power of abstraction for modeling activities.

As further motivation, consider the following future scenario for intelligent modeling assistance where an *intelligent modeling assistant* (IMA) supports modelers in their trade-off analysis with the help of a *multi-disciplinary pull request*. Often highly trained specialists have to work together to find appropriate engineering solutions using domain-specific models. However, modeling choices made by specialists in their domain-specific models may impose constraints on other specialists or impact which assumptions remain valid. In many cases, a specialist cannot be aware of such dependencies and cannot perform required trade-off analyses, because of the deep knowledge required from the other specialists' domains. An abundance of data about this modeling context is available to the IMA in the form of, e.g., historical project data, model coherence rules, patterns observed in sets of multi-disciplinary models, the role of domain experts in the engineering process and their associated profile data (e.g., digital twins [17]), and operational system data relevant for the trade-off, such as resource consumption. Going beyond existing collaborative tools, version control systems, as well as traceability and change impact analyses, the IMA derives contextualized knowledge from the available data to help identify the appropriate domain experts that need to be contacted based on the type of change and then make the trade-off options explicit to all, while individually adapting to the vocabulary, skill level, and interaction style of each expert.

However, IMAs are complex tools to develop, and can be rather heterogeneous at first glance depending on different aspects, such as varying purposes, the supported abstractions, required and/or available domain-specific knowledge, the modeling activity addressed, the expected means of interaction, or the IMAs' integration in the modeling process. Unfortunately, most existing IMAs lack transparency to understand their scope as well as adaptability and flexibility to be customized, compared, and combined beyond their original intended use. Furthermore, a common understanding of the differences and commonalities of IMAs does not exist. A standardized view of the common protocols, interfaces, and interactions of IMAs would address these issues.

This expert voice results from the week-long *Data and Models*[1] workshop attended by experts in model-driven engineering, software language engineering, artificial intelligence, and human-centric adaptive systems. We advocate that common foundations and main components can be shared across the many possible IMAs, thus opening exciting research opportunities for the modeling community to address the needs of both tool builders and modelers. We identify these components and their interactions in a conceptual *reference framework for intelligent modeling assistance* (RF-IMA) and describe how such a framework would help comprehend, develop, and ultimately compare IMAs.

In Sect. 2, we present the current landscape of intelligent modeling assistance. In Sect. 3, we propose our conceptual reference framework in support of modern modeling activities and discuss its properties as well as potential sources of data, information, and knowledge. RF-IMA addresses the needs of two major stakeholder groups. First, RF-IMA aims to help the tool building community engineer IMAs more systematically. Second, RF-IMA aims to facilitate comprehension and comparison of IMAs, and ultimately to provide more intelligent support for modelers. We conclude this call for action with a list of open challenges to integrate intelligent modeling assistants into modern modeling environments in Sect. 4 and envision a momentum in the modeling community that leads to the implementation of RF-IMA and consequently future IMAs.

2 Current landscape of intelligent modeling assistance

Assistants have a longstanding history in software engineering, either to help with specific activities (e.g., design, development) or to support the enactment of the overall process. Since the '80s, several assistants have been pro-

[1] http://www.bellairs2020.ece.mcgill.ca/.

posed, e.g., to drive the development according to a particular methodology [19], or capture the current context and recommend some actions [18,43]. However, those assistants embed only fixed, pre-determined knowledge (e.g., a specific development process), and the collected context is rather limited to the session initiated by the developer and the local workspace.

With the advent of large amounts of data and APIs to conveniently access such data, and the identification of more complex contexts [10], specific assistants in the form of sophisticated recommenders [32] have been brought to the attention of software engineering [33]. New opportunities have been explored in the field of recommenders in IDEs for specific programming languages or platforms, e.g., for code completion [6,40], for using external libraries [41], contributing to new projects [25], or solving standard programming tasks [38].

In the field of *Software and System Modeling*, assistants provide important support to the whole life-cycle of ever more complex systems. For example, widely used languages like UML have been supported with large efforts, e.g., to support a particular modeling process [39], and to help build diagrams from natural language [16] or through recommendations from similarities [12] or established patterns [23]. However, the diversity of domain-specific abstractions and heterogeneity of stakeholders are important challenges that prevent the ability to scale the development of all needed IMAs, because it is very time-consuming to build individual IMAs for each of those targets. The community, hence, has to move away from from-scratch solutions and adopt engineered solutions based on standardized protocols (e.g., plug-and-play).

Specific efforts to apply recommendation in the context of domain-specific modeling languages have been limited (i) to narrow contexts, such as the current state of the model and its possible extension with regards to the metamodel [36], (ii) to specific interactions, such as the use of chatbots to help modelers build and query domain-specific models using natural language [30,31], and (iii) to specific activities, such as system requirements (e.g., focus on variability [3] or behavior [14]), domain modeling [2], or model transformations [4,8]. Finally, leveraging a collective knowledge for a recommender system for the whole modeling process is envisioned to not only infer model transformations but also recommend model repair or refactoring [22].

Commercial tools in the area of low-code platforms also recently introduced AI-powered assistants to support citizen programmers developing their own applications, e.g., Mendix Assist [28] and ServiceStudio [37].

While the current landscape of intelligent modeling assistance is as broad as modeling itself, all existing approaches discussed above have been specifically tailored from scratch for a particular purpose, which makes them time-consuming to develop (e.g., little reuse), compare, and integrate. Therefore, we introduce in the next section a conceptual reference framework to support a disciplined approach for the development of new IMAs to ultimately provide intelligent modeling assistance to modelers.

3 Toward a reference framework for intelligent modeling assistance

To realize the opportunities provided by intelligent modeling assistance, this section introduces our proposed conceptual reference framework RF-IMA that lays the foundation for modeling activities supported by an IMA. RF-IMA serves a twofold purpose. It is expected to (i) help tool builders when engineering future assistants, which will (ii) eventually provide more intelligence and situational awareness about modeling activities to various stakeholders involved in modeling activities. Assistance to these stakeholders (e.g., domain modeler, transformation engineer, language engineer) covers a broad range, from existing basic model autocompletion tools and modeling chatbots to more advanced modeling environments that are inspired by, e.g., literate programming [21], exploratory programming [5], and live programming [27], and fueled by data-driven techniques.

From a conceptual point of view, RF-IMA gives a holistic perspective of how a *modeler's context is connected with external sources* of data, information, and knowledge to realize the opportunities for intelligent modeling assistance. From a modeler's point of view, an IMA's goal is to improve the modeler's *user experience* and to increase both the *quality of the models* produced and the *quality of the modeling process*. Consider the future scenario from the introduction. Engaging in a conversation with an IMA in a vocabulary tailored to the modelers expertise and at the right skill level will improve the modeler's user experience, while enabling complex trade-off reasoning across many problem domains will improve the quality of the modeling process and the quality of the resulting modeling artifacts. IMAs help with respect to the lack of resources, such as the modeler's potential unawareness of knowledge for the activity at hand or time constraints to access such knowledge. The conceptual reference framework illustrated in Fig. 1 considers (1) the *socio-technical modeling system* (STMS) that provides context, (2) an IMA, (3) the connecting communication between both, (4) external sources of data, information, knowledge, and experts, and (5) crosscutting quality properties.

(1) The **context in an STMS** consists of a modeler realizing a modeling activity in a modeling environment. It is hence composed of three parts (top left of Fig. 1): (i) the *actor* that can be a modeler, a team, or another software; (ii) the *technical infrastructure*, such as a computer and

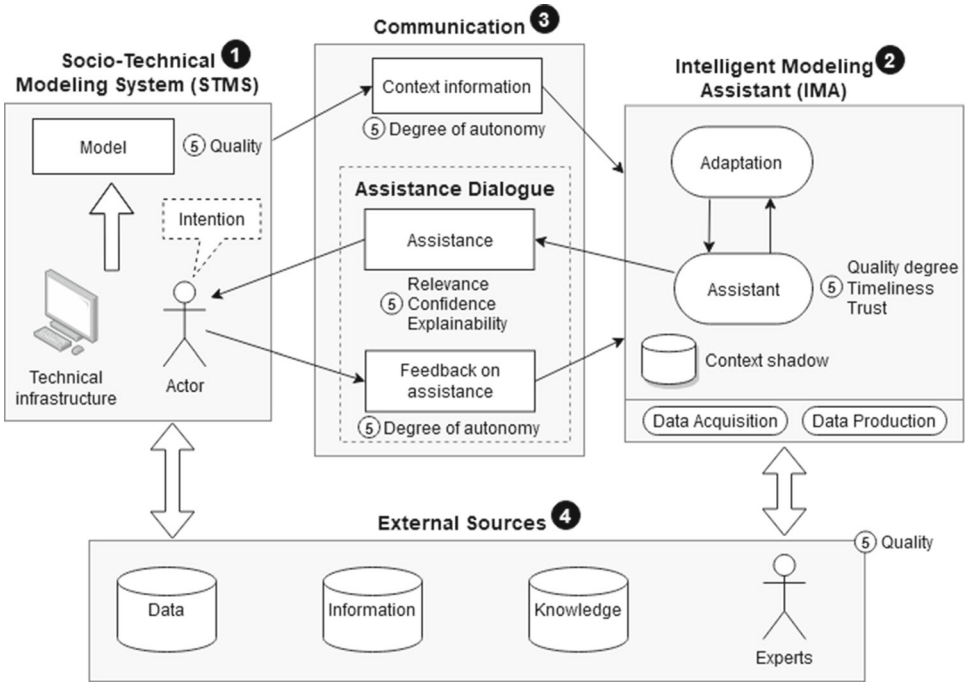


Fig. 1 Conceptual reference framework for intelligent modeling assistance (RF-IMA), highlighting four main components (1–4) and the crosscutting concern related to quality properties (5)

its software, including one or more modeling tools; and (iii) the *models* of interest. The actor has *intention* which defines what the actor wants to accomplish and why. Intention may be expressed explicitly or implicitly (as indicated by the dashed outline).

(2) An **IMA** consists of four components (top right of Fig. 1): (i) the *assistant* with algorithms to compute the assistance (e.g., contextualized information, recommendations); (ii) the *data acquisition/production layer* that gives access to different external sources (data, information, knowledge) and has the ability to connect a modeler with domain experts; (iii) the *context shadow*, which contains information about the context that captures the intention of the modeling activity, any relevant information for the activity or specific to the current situation, personalized information about the actor like an internal representation of a modeler (e.g., a digital twin), and retains the local historical data related to the modeling activity, and (iv) an optional *adaptation* component which adapts to changing context based on the feedback by the actor (e.g., by assessing the modeler's skill level from provided feedback).

(3) In general, there might be a federation of IMAs that process a context with various information and each actor may interact with multiple IMAs. Individual IMAs may interact with each other, e.g., to collaborate and coordinate changes across actors. The **communication** between STMS and IMA (top middle of Fig. 1) is bidirectional and characterized by different degrees of autonomy [35,42]. STMS to IMA communication is used to deliver context information and actor feedback. IMA to STMS communication feeds the assistance to the actor in a way that is specific to the modeling activity. In turn, this communication may affect all parts of the context (not only the intention and the specific actor, but others as well, as context may be shared among actors). IMA to STMS communication may also provide an explanation about the choice of the assistance.

(4) An IMA may use a variety of **external sources** (bottom of Fig. 1), which may be categorized into data, information, and knowledge according to the data, information, knowledge, and wisdom pyramid [24]. These external sources may be used for specific recommendations. For example, technical project information may lead to feedback regarding modeling patterns, information about the technical infrastructure to feedback on energy models, and development process information to improved resource usage models. Furthermore, information from social networks and technical background from résumés may lead to feedback on which modeling expert to contact, business information to improved development processes, information from the execution of a system or simulation to more accurate metrics about model elements, and information about the environment to cheaper locations to run computationally intensive model analyses. Information about laws and regulations may lead to warnings of possible certification issues, scientific information to suggestions on novel modeling solutions, and information about stakeholders and culture to feedback on the real values and intentions of

stakeholders. Last but not least, metadata about an IMA itself and interactive feedback from actors may improve an IMA's performance over time.

(5) **Quality properties** are highlighted across all four main components of the reference framework. First for IMAs, the assistant's *quality degree* and *timeliness* (how often is adequate assistance provided in a timely manner?) as well as the *trust* of modelers (as in the perception of the quality of IMAs from the modeler's perspective) are issues. At a high level, one of the factors that has a big impact on the quality degree relates to the level of sophistication required from IMAs (e.g., syntactic auto-completion vs. design trade-off analysis).

Second for Assistance, the *relevance* of and *confidence* in the provided information are important (how adequate is it for the current modeling context and how sure is the IMA that it is adequate?). Furthermore, *explainability* is important, characterized as the degree to which an actor understands why a particular assistance is provided to her. Without adequate assistance at a high confidence level, the usefulness of an IMA suffers. However, even with high relevance and high confidence, an IMA's help may be of limited use, if it is not possible to explain why the provided help is relevant.

Third for Feedback on assistance and fourth for Context information, the *degree of autonomy* needs to be considered (e.g., existing levels of autonomy [1], such as offering no assistance and the modeler making all decisions, suggesting an action and executing that action if approved by the modeler, automatic execution of improvements, to an IMA acting fully autonomously).

Last but not least, the *quality* of the model is also an issue in terms of well-formedness, absence of anti-patterns, etc. as is the *quality* of the modeling activity (i.e., process-related quality indicators). Similarly, the *quality* of external sources also must be considered (e.g., transparency, accessibility, data curation concerns).

4 Opportunities for the modeling community

Beyond the identification of the core foundations and main components for IMAs, the proposed conceptual reference framework RF-IMA also helps identify open challenges (based on the outcomes of the week-long workshop). In this section, we go through the main components of RF-IMA (i.e., STMS, IMA, the connecting communication, and the external sources) and discuss the major challenges that need to be addressed to realize the opportunities provided by intelligent modeling assistance, while building on existing work, such as data mining, AI techniques, repository management, and adaptive systems.

(1) **For STMS**, understanding the modeling context (incl. the actor and its intention) is of utmost importance. Hence, context monitoring of the modeler (e.g., to predict the actor's intentions) requires *non-invasive, privacy-preserving methods*. *Mining of modeling intention* may be needed instead of relying on explicit expressions of modeling objectives, and *quality criteria* for modeling activities and artifacts need to be established considering the domain of interest. For instance, interpreting actor's activities and predicting intentions correctly may require to build a model, to determine which variables influence each other, to learn about model's parameters, and to complete the model in case of missing data (e.g., an actor's preferences).

(2) **For an IMA**, learning opportunities exist to provide the most relevant assistance, such as (i) *understanding behavior and skill of modelers*, as well as understanding the specific modeling need for assistance or (ii) *transfer learning* between different, yet similar, domain-specific modeling languages to cope with the possible lack of models in a particular domain. There is also a need to *determine the most appropriate algorithms* to learn from existing data and infer relevant new knowledge in the context of a given a model or modeling activity. In addition, there are questions on how to *analyze the impact* for recommended model changes for both short-term and long-term effects, on *how to adapt to the evolving needs* of the modeler, and on how to *select, filter, structure, and manage relevant context information* in the context shadow. As the multi-disciplinary pull request example explained earlier has shown, context may involve vocabulary from many different domains, which needs to be understood to be able to provide assistance across domains and possibly across IMAs.

(3) **For communication**, there is a need to study *user interactions* (among the very heterogeneous stakeholders and modeling activities) and *user experience for assistance* in terms of (i) degrees of automation, (ii) the emotional state of the modeler as well as her positive and negative modeling experiences, (iii) content and form of presenting assistance, and (iv) dialog structure with the modeler to facilitate explainability.

(4) **For external sources**, one crucial challenge is related to the availability of required *repositories for models and their metamodels*. What is the corresponding *modeling knowledge* for an IMA that needs to be made explicit to support modeler's decision-making, comprehension, etc.? How does the *modeling knowledge base* evolve and how is it maintained, and at what granular-

ity of classifications (by domain, ontologically, etc.)? Which *data schemas* are required to manage modeling artifacts? Another challenge involves the *mining of modeling knowledge and professional information of modelers* to determine best practices and identify domain experts.

From a global perspective, a key challenge is to *identify modeling activities* where intelligent assistance is relevant, together with required models and data. An immediate concern for the modeling community is how to *determine the, possibly evolving, requirements for an IMA* to provide timely assistance based on confidence and relevance as well as modeler's skill and trust. To identify such requirements, further work is needed to support comparison of IMAs based on more fine-grained properties of RF-IMA (e.g., through an assessment grid). As with any other reference framework, RF-IMA helps define the boundaries between components. An overarching challenge is now to *define appropriate, generic, domain-independent modeling interfaces and protocols* (e.g., similar to the language server protocol for programming languages), allowing seamless integration of various IMAs and modeling environments. Such interfaces between STMS and IMA (in both directions) and protocols for communication have to (i) provide understanding of context as well as the feedback on the offered assistance and (ii) convey available knowledge from external sources that is useful for an IMA. The grand challenge is to *make functionality for modeling assistance reusable* across a wide range of domains.

The authors of this expert voice hope the proposed RF-IMA will serve as a call for action to create a research momentum and address the open challenges to realize the opportunities provided by intelligent modeling assistance.

References

1. AI4AUI: Workshop on AI Methods for Adaptive User Interfaces (2020). https://doi.org/10.1145/3306307.3328180
2. Agt-Rickauer, H., Kutsche, R., Sack, H.: Domore—a recommender system for domain modeling. In: Proceedings of the 6th International Conference on Model-Driven Engineering and Software Development (MODELSWARD'18), pp. 71–82 (2018). https://doi.org/10.5220/0006555700710082
3. Bakar, N.H., Kasirun, Z.M., Salleh, N.: Feature extraction approaches from natural language requirements for reuse in software product lines: a systematic literature review. J. Syst. Softw. **106**(C), 132–149 (2015). https://doi.org/10.1016/j.jss.2015.05.006
4. Baki, I., Sahraoui, H.A.: Multi-step learning and adaptive search for learning complex model transformations from examples. ACM Trans. Softw. Eng. Methodol. **25**(3), 20:1–20:37 (2016)
5. Beth Kery, M., Myers, B.A.: Exploring exploratory programming. In: 2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), pp. 25–29 (2017)
6. Bruch, M., Monperrus, M., Mezini, M.: Learning from examples to improve code completion systems. In: van Vliet, H., Issarny, V. (eds.) Proceedings of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2009, Amsterdam, The Netherlands, August 24–28, 2009, pp. 213–222. ACM (2009). https://doi.org/10.1145/1595696.1595728
7. Bucchiarone, A., Cabot, J., Paige, R., Pierantonio, A.: Grand challenges in model-driven engineering: an analysis of the state of the research. Softw. Syst. Model. **19**, 1–9 (2020). https://doi.org/10.1007/s10270-019-00773-6
8. Burgueño, L., Cabot, J., Gérard, S.: An LSTM-based neural network architecture for model transformations. In: 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS), pp. 294–299. IEEE (2019)
9. Combemale, B., Kienzle, J., Mussbacher, G., Ali, H., Amyot, D., et al.: A Hitchhiker's guide to model-driven engineering for data-centric systems. IEEE Softw. (2020). https://doi.org/10.1109/MS.2020.2995125. https://hal.inria.fr/hal-02612087
10. Danylenko, A., Löwe, W.: Context-aware recommender systems for non-functional requirements. In: 2012 Third International Workshop on Recommendation Systems for Software Engineering (RSSE), pp. 80–84 (2012). https://doi.org/10.1109/RSSE.2012.6233417
11. Derakhshanmanesh, M., Ebert, J., Grieger, M., Engels, G.: Model-integrating development of software systems: a flexible component-based approach. Softw. Syst. Model. **18**(4), 2557–2586 (2019)
12. Elkamel, A., Gzara, M., Ben-Abdallah, H.: An UML class recommender system for software design. In: 2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA), pp. 1–8 (2016). https://doi.org/10.1109/AICCSA.2016.7945659
13. France, R., Rumpe, B.: Model-driven development of complex software: a research roadmap. In: Future of Software Engineering (FOSE '07), pp. 37–54 (2007)
14. Friedrich, F., Mendling, J., Puhlmann, F.: Process model generation from natural language text. In: International Conference on Advanced Information Systems Engineering (CAISE), pp. 482–496. Springer, Berlin (2011)
15. Hartmann, T., Moawad, A., Fouquet, F., Le Traon, Y.: The next evolution of MDE: a seamless integration of machine learning into domain modeling. In: 2017 ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS), pp. 180–180 (2017). https://doi.org/10.1109/MODELS.2017.32
16. Ibrahim, M., Ahmad, R.: Class diagram extraction from textual requirements using natural language processing (NLP) techniques. In: Proceedings of the 2nd International Conference on Computer Research and Development, pp. 200–204 (2010). https://doi.org/10.1109/ICCRD.2010.71
17. Josifovska, K., Yigitbas, E., Engels, G.: A digital twin-based multimodal UI adaptation framework for assistance systems in industry 4.0. In: HCI (3), Lecture Notes in Computer Science, vol. 11568, pp. 398–409. Springer, Berlin (2019)
18. Kaiser, G.E., Feiler, P.H., Popovich, S.S.: Intelligent assistance for software development and maintenance. IEEE Softw. **5**(3), 40–49 (1988). https://doi.org/10.1109/52.2023
19. Karimi, J., Konsynsky, B.R.: An automated software design assistant. IEEE Trans. Softw. Eng. **14**(2), 194–210 (1988). https://doi.org/10.1109/32.4638
20. Kienzle, J., Mussbacher, G., Combemale, B., Bastin, L., Bencomo, N., et al.: Towards model-driven sustainability evaluation. Commun. ACM **63**(3), 80–91 (2020). https://doi.org/10.1145/3371906
21. Knuth, D.E.: Literate programming. Comput. J. **27**(2), 97–111 (1984). https://doi.org/10.1093/comjnl/27.2.97

22. Kögel, S.: Recommender system for model driven software development. In: 11th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2017, pp. 1026–1029. Association for Computing Machinery, New York (2017). https://doi.org/10.1145/3106237.3119874
23. Kuschke, T., Mäder, P.: Pattern-based auto-completion of uml modeling activities. In: Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering, ASE '14, pp. 551–556. Association for Computing Machinery, New York (2014). https://doi.org/10.1145/2642937.2642949
24. Liew, A.: DIKIW: data, information, knowledge, intelligence, wisdom and their interrelationships. Bus. Manag. Dyn. **2**, 49 (2013)
25. Liu, C., Yang, D., Zhang, X., Ray, B., Rahman, M.M.: Recommending github projects for developer onboarding. IEEE Access **6**, 52082–52094 (2018). https://doi.org/10.1109/ACCESS.2018.2869207
26. Mazak, A., Wolny, S., Wimmer, M.: On the need for data-based model-driven engineering, pp. 103–127 (2019). https://doi.org/10.1007/978-3-030-25312-7_5
27. McDirmid, S.: Living it up with a live programming language. In: Proceedings of the 22nd Annual ACM SIGPLAN Conference on Object-Oriented Programming Systems and Applications, OOPSLA '07, pp. 623–638. Association for Computing Machinery, New York (2007). https://doi.org/10.1145/1297027.1297073
28. Mendix Assist: https://www.mendix.com/blog/introducing-ai-assisted-development-to-elevate-low-code-platforms-to-the-next-level
29. Mussbacher, G., Amyot, D., Breu, R., Bruel, J.M., Cheng, B., et al.: The relevance of model-driven engineering 30 years from now (2014). https://doi.org/10.1007/978-3-319-11653-2_12
30. Pérez-Soler, S., Daniel, G., Cabot, J., Guerra, E., de Lara, J.: Towards automating the synthesis of chatbots for conversational model query. In: Proceedings of the International Conference on Exploring Modeling Methods for Systems Analysis and Development (2020). **(To appear)**
31. Pérez-Soler, S., Guerra, E., de Lara, J.: Collaborative modeling and group decision making using chatbots in social networks. IEEE Softw. **35**(6), 48–54 (2018). https://doi.org/10.1109/MS.2018.290101511
32. Ricci, F., Rokach, L., Shapira, B.: Introduction to Recommender Systems Handbook, pp. 1–35. Springer US, Boston (2011). https://doi.org/10.1007/978-0-387-85820-3_1
33. Robillard, M.P., Maalej, W., Walker, R.J., Zimmermann, T.: Recommendation Systems in Software Engineering. Springer Publishing Company, Incorporated, Berlin (2014)
34. Rocha, A., Papa, J.P., Meira, L.A.A.: How far you can get using machine learning black-boxes. In: 2010 23rd SIBGRAPI Conference on Graphics, Patterns and Images, pp. 193–200 (2010). https://doi.org/10.1109/SIBGRAPI.2010.34
35. Rovatsos, M., Weiss, G.: Autonomous Software, pp. 63–84. https://doi.org/10.1142/9789812775245_0003
36. Sen, S., Baudry, B., Vangheluwe, H.: Towards domain-specific model editors with automatic model completion. Simulation **86**(2), 109–126 (2010). https://doi.org/10.1177/0037549709340530
37. ServiceStudio from OutSystems: https://www.outsystems.com/ai
38. Silva, R., Roy, C., Rahman, M., Schneider, K., Paixao, K., Maia, M.: Recommending comprehensive solutions for programming tasks by mining crowd knowledge. In: 27th International Conference on Program Comprehension (ICPC), pp. 358–368. IEEE, Association for Computing Machinery (2019)
39. Subramaniam, K., Liu, D., Far, B.H., Eberlein, A.: UCDA: use case driven development assistant tool for class model generation. In: SEKE (2004)
40. Svyatkovskiy, A., Zhao, Y., Fu, S., Sundaresan, N.: Pythia: AI-assisted code completion system. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '19, pp. 2727–2735. Association for Computing Machinery (2019). https://doi.org/10.1145/3292500.3330699
41. Thummalapenta, S., Xie, T.: Parseweb: a programmer assistant for reusing open source code on the web. In: Proceedings of the Twenty-Second IEEE/ACM International Conference on Automated Software Engineering, ASE '07, pp. 204–213. Association for Computing Machinery, New York (2007). https://doi.org/10.1145/1321631.1321663
42. Timm, I.J., Knirsch, P., Kreowski, H.J., Timm-Giel, A.: Autonomy in Software Systems, pp. 255–273. Springer, Berlin (2007). https://doi.org/10.1007/978-3-540-47450-0_17
43. Tran, Q., Chung, L.: NFR-assistant: tool support for achieving quality. In: Proceedings 1999 IEEE Symposium on Application-Specific Systems and Software Engineering and Technology. ASSET'99 (Cat. No.PR00122), pp. 284–289 (1999). https://doi.org/10.1109/ASSET.1999.756782
44. Whittle, J., Hutchinson, J., Rouncefield, M.: The state of practice in model-driven engineering. IEEE Softw. **31**(3), 79–85 (2014)

Gunter Mussbacher is an Associate Professor at McGill University. His research interests include Model-Driven Requirements Engineering, Next-Generation Reuse Frameworks, Sustainability, and Human Values. Contact him at gunter.mussbacher@mcgill.ca.

Benoit Combemale is a Full Professor of Software Engineering at University of Toulouse and a Research Scientist at Inria. His research interests in Software Engineering include Software Language Engineering, Model-Driven Engineering, and Software Validation & Verification. Contact him at benoit.combemale@inria.fr.

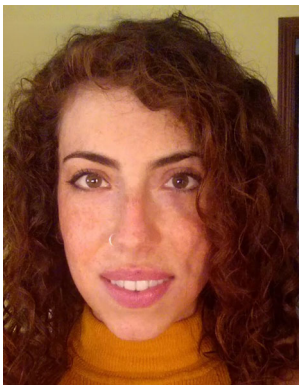
Jörg Kienzle is an Associate Professor at McGill University. His research interests include Model-Driven Software Development, Separation of Concerns, Reuse, Software Composition, and Modularity. Contact him at joerg.kienzle@mcgill.ca.

Silvia Abrahão is an Associate Professor at Universitat Politècnica de València, Spain. Her research interests include Quality Assurance in Model-Driven Engineering, Empirical Assessment of Software Modeling Approaches, Model-Driven Cloud Services Development and Monitoring, and the Integration of Usability into Software Development. Contact her at sabrahao@dsic.upv.es.

Hyacinth Ali is a Ph.D. candidate at McGill University. His research interests include Model-Driven Engineering, Concern-Oriented Reuse, Combination and Reuse of Software Languages. Contact him at hyacinth.ali@mail.mcgill.ca.

Nelly Bencomo is a Senior Lecturer at Aston University in the UK. Her research interests in Software Engineering include Decision-making under Uncertainty, Model-Driven Engineering, Autonomous Systems, Artificial Intelligence, and Requirements Engineering. Contact her at nelly@acm.org.

Márton Búr is a Ph.D. candidate in Software Engineering at McGill University. His research focuses on Model-Driven Software Engineering, with a special interest in model-based runtime assurance techniques for smart and safe embedded systems. Contact him at marton.bur@mail.mcgill.ca.



Loli Burgueño is a Postdoctoral Researcher and Lecturer at the Open University of Catalonia (Spain) and CEA LIST (France). Her research interests in Model-Driven Engineering include the performance, scalability and testing of model transformations, the modeling of uncertainty in software models for its use in the Industry 4.0 and the integration of Artificial Intelligence methods into modeling tools and processes. Contact her at lburguenoc@uoc.edu.

Gregor Engels is Full Professor of Database and Information Systems at Paderborn University, Germany. His research interests include Model-based, Human-Centric Software Engineering, Self-Adaptive Assistance Systems, Future of Work. Contact him at engels@upb.de.

Pierre Jeanjean is a PhD candidate at Inria (France), currently working in the DiverSE team in Rennes. His research interests include Software Language Engineering and Domain-Specific Languages, and more specifically interactive language interpreters (REPLs) and IDE features. Contact him at pierre.jeanjean@inria.fr.

Jean-Marc Jézéquel is a Professor at the University of Rennes. His research interests include Model-Driven Software Engineering for Software Product Lines. Contact him at jezequel@irisa.fr.

Thomas Kühn is a Postdoctoral Researcher at the Karlsruhe Institute of Technology (KIT). His research interests in Software Engineering include Software Product Line Engineering with the particular focus on tool support for families of domain-specific and general-purpose programming languages. Contact him at thomas.kuehn@kit.edu.

Sebastien Mosser is Professor of Software Engineering at Université du Québec à Montréal. His research interests are related to Domain-Specific Modelling and Software Composition, from the scalability point of view. Contact him at mosser.sebastien@uqam.ca.

Houari Sahraoui is a Professor at Université de Montréal. His research interests include AI Techniques Applied to Software Engineering, Search-based Software Engineering, Model-Driven Engineering, and Software Visualization. Contact him at sahraouh@iro.umontreal.ca.

Eugene Syriani is an Associate Professor at the University of Montreal. His main research interests fall in Software Design based on the Model-Driven Engineering approach, the Engineering of Domain-Specific Languages, Model Transformation and Code Generation, Simulation-based Design, Collaborative Modeling, and User Experience. Contact him at syriani@iro.umontreal.ca.

Dániel Varró is a Professor at McGill University and at Budapest University of Technology and Economics. He holds a research chair of the MTA-BME Lendület Cyber-Physical Systems Research Group in Hungary. His research interests include Cyber-Physical Systems, Model Based Systems Engineering, and Software Verification and Validation. Contact him at daniel.varro@mcgill.ca.

Martin Weyssow is a PhD candidate at the University of Montreal and University of Namur. His research interests include Machine Learning for Software Engineering, Neural Language Models, Program Comprehension and Recommender Systems. Contact him at martin.weyssow@umontreal.ca.