Software and Systems Modeling (2022) 21:2101–2107
https://doi.org/10.1007/s10270-022-01035-8

EXPERT VOICE

Modeling should be an independent scientific discipline

Jordi Cabot[1] · **Antonio Vallecillo**[2]

Received: 5 July 2022 / Revised: 21 July 2022 / Accepted: 22 July 2022 / Published online: 3 August 2022

Abstract
Software modeling started as a paradigm to help developers build better software faster by enabling them to specify, reason and manipulate software systems at a higher-abstraction level while ignoring irrelevant low-level technical details. But this same principle manifests in any other domain that has to deal with complex systems, software-based or not. We argue that bringing to other engineering and scientific fields, our modeling expertise is a win–win opportunity where we can all learn from each other as we all model, but in complementary ways. Nevertheless, to fully unleash the benefits of this collaboration, we must go beyond individual efforts trying to adapt single techniques from one field to another. It requires a deeper reformulation of modeling as a whole. It is time for modeling to become an independent discipline where all fields of knowledge can contribute and benefit from.

Keywords Modeling · Abstraction · Discipline · Transdisciplinarity · Science · Engineering

1 Introduction

The entire history of software engineering can be seen as one of raising levels of abstraction [6]. As such, we could claim that software modeling is as old as software engineering itself, as modeling is the framework that enables the precise definition and transformation of such abstractions. And with *software eating the world* [3], the relevance of software modeling keeps increasing.

Software modeling has developed significantly during the past 20 years, providing clear definitions of the main concepts and mechanisms related to modeling, as well as very useful tools for editing, building and managing models and their related artifacts. While its adoption by the software industry seems to be progressing differently than expected, its advances are still achieving outstanding results. This has been evidenced by approaches such as *low-code* (or *no-code*) that have lowered the barrier entry to application development by non-computer scientists. Similarly, the business process modeling community is helping companies to extract valuable information from very large datasets and manage it in a high-level and tractable manner. These are examples where the use of abstract models significantly pays off.

But modeling is not an activity unique to software engineering. Models have long been used to explain and predict the behavior of real objects or systems in a variety of scientific and engineering fields, including mathematics, physics, biology, chemistry, ecology, earth sciences, civil engineering, automotive or aerospace. These domains heavily use all kinds of models, and require better mechanisms and tools to manage them. They could certainly benefit from all the advances and tools we have in software modeling. In turn, this will also be good for us, the software modeling experts, who will learn about different requirements and needs, and will be able to develop and validate new modeling tools and techniques; and for society at large who will be able to benefit from the application of modeling in their fields as part of a combined effort.

In this article, we argue that to unleash the full potential of modeling we need to break free of our traditional positioning within software engineering and cooperate with scientists and engineers from other domains. And the best way to achieve this is for modeling to become an independent discipline that serves all the rest.

✉ Jordi Cabot
jordi.cabot@icrea.cat

Antonio Vallecillo
av@lcc.uma.es

[1] ICREA – Universitat Oberta de Catalunya, Barcelona, Spain

[2] ITIS Software – Universidad de Málaga, Málaga, Spain

Clearly, there are still plenty of technical and social challenges in software modeling itself [8] and, while improving, also plenty of work to do to push further the adoption of modeling and model-driven engineering in software companies [26,30]. Nevertheless, we believe limiting ourselves to the core software scope is a navel-gazing strategy. Instead, we propose to broaden our horizons and aim to realize the full potential of modeling. This will improve modeling as a whole, including but not limited to, software and systems modeling while opening the door to many new and exciting opportunities for the modeling community.

2 Models everywhere

2.1 Omnipresence of modeling

When you believe that everything is a model [5] you see models everywhere. Especially when having a broad view of what constitutes a model and the different types of models that exist [16], and not only in computer science.[1] In fact, modeling is as old as mankind itself[2] because it has always allowed humans to represent reality in order to communicate, understand how something works, and devise and build more complex artifacts and systems.

To illustrate this omnipresence of modeling, we give examples of the use of modeling in different fields, starting with our own (software) and broadening more and more the scope of modeling applications.

The key role of abstraction and modeling in software engineering and computer science in general has been widely acknowledged [2,20,28]. As a recent example of this, it is worth mentioning the growing number of AI solutions that are becoming model-based.[3] As in any other field, once the core components of the AI technology have been established, the effort moves to facilitate their use and adoption by a larger user base, and this always involves moving up the abstraction ladder and creating model-based solutions that require less technical expertise to reach the same result.

This key role of modeling also holds if we go beyond computer science. This is obvious when we look at systems engineering, where models are massively used (probably even more than in software engineering itself) for verification and simulation purposes. The growing importance of digital twins, which are model-based by definition [19], is a clear example. It is also easy to find examples in other engineering disciplines, e.g., plant-wide models for water resource recovery facilities [27], or of a moving crane [32].

Models are also widely used in science. From climate modeling [17] to epidemiology modeling [9] and going through farming modeling,[4] healthcare[5] or core cellular biological function [14], we can easily find plenty of examples of models of different shape, type and function in the scientific literature.

Last but not least, citizens are also exposed to numerous models in their daily life. City and transport maps, workflow models and hierarchical models commonly used in any company are three examples of that.

All these models provide representations of systems. It is important to realize that all models, not only the biological, mathematical or engineering models, but also those commonly used in our daily lives, are of different natures but share a set of common characteristics. First, a conceptual model consists of *concepts* used to help people know, understand, or simulate the system that the model represents. Second, models are expressed using a concrete and precise *notation*, let it be mathematical equations, abstract symbols or a dedicated textual language (concrete syntax). Third, a model has a *semantics* that permits interpreting its symbols in a precise and unambiguous manner. Finally, models count on an *inference system* that allows reasoning about the estimated or desired behavior of the system being represented.

2.2 The software modeling contribution

Even if modeling is everywhere, we believe software modeling has some particularities that make it especially useful as one of the pillars of the new modeling discipline we propose in the next sections.

Software engineering models have been very good at identifying and formally defining all model characteristics, and providing valuable tool support for them. For example, metamodels are used to define the abstract syntax of models at the appropriate level of abstraction, in terms of their concepts, the relationships between them, and the corresponding well-formed constraints of the modeled domain. Then, concrete syntaxes define the notations used to represent the concepts, and we already count on useful tools to generate powerful model editors and other model management operations: comparison, instance generation, satisfiability, etc. Semantics is commonly defined in software engineering in terms of mappings to domains with well-defined semantics, whose inference systems are used to reason about the source models, and where the reasoning tools available in the target domains can be used by the source models.

[1] https://en.wikipedia.org/wiki/Conceptual_model.

[2] For example, cave paintings were used to represent hunting scenes that served to teach hunters about dangers they knew they had to face but had not yet encountered, or to model how to hunt some types of preys [7].

[3] https://modeling-languages.com/tools-modeling-artificial-intelligence-code/.

[4] https://github.com/gemoc/farmingmodeling.

[5] https://www.hl7.org/.

Compared to the models used in other domains, we can see how software engineering has been able to provide a formal background and mature tools to define, develop, manage and reason about models at the right level of abstraction. Other valuable contributions of software engineering to conceptual modeling are the automated processing of models, as well as the interoperability of tools and the seamless exchange and sharing of models.

Going back to the examples mentioned above, some engineering or biological models and tools are very powerful for simulating physical systems, but their definition mechanisms remain at a very low level of abstraction and do not support high-level abstraction mechanisms such as inheritance, polymorphism or modularity. In addition, most of these models and tools live in silos that do not allow for seamless model interchange or tool combination. These are precisely the types of support that model-based software engineering can provide. In turn, the application of software modeling mechanisms and tools to these domains can help to validate and significantly improve these mechanisms and tools, uncover some of their limitations, broaden their scope and open up new challenges.

3 Modeling as a transdisciplinary discipline

We have just seen how pervasive is modeling, even if sometimes it is not called as such or it is used in a rather ad hoc or informal way. But this pervasiveness is right now not fairly retributed with the level of attention, recognition and funding that modeling deserves. And this hampers the evolution of the modeling field and, as a consequence, limits the impact and benefits it could bring to all other fields that depend on it.

A way to give modeling the recognition it deserves, increase its visibility, and attract the talent and resources it needs to become a key instrument in the scientific advancement of all other knowledge fields, is to elevate modeling into a new independent academic discipline.

In short, an academic (or scientific) discipline can be defined as academic studies that focus on a self-imposed limited field of knowledge [15]. Given that this definition is rather generic and ambiguous, Krishnan [21] identifies six characteristics of an academic discipline. Let us explore below whether modeling as a discipline does fit the criteria.

1. *Disciplines have a particular object of research, although the object of research may be shared with another discipline*. Our object of research is to provide languages, operations and tools to create and manipulate abstractions that facilitate the comprehension, reasoning and manipulation of complex technical, social, biological and natural systems, and not only software-based ones. We partner together with other disciplines to reach these goals.
2. *Disciplines have a body of accumulated specialist knowledge referring to their object of research, which is specific to them and not generally shared with another discipline*. Software modeling has a well-defined body of knowledge [10] that characterizes the specific contributions of modeling as a knowledge field.
3. *Disciplines have theories and concepts that can organize the accumulated specialist knowledge effectively*. Modeling comprises a good number of theories (some grounded on formal methods, some on more empirical evidence) to compare, merge and organize the growing number of modeling concepts and techniques being developed.
4. *Disciplines use specific terminologies or a specific technical language adjusted to their research object*. Over the years, modeling has precisely defined a set of core terminology (model, metamodel, DSL, transformation, etc.). New terms are then contextualized in terms of the existing ones (e.g., the positioning of low-code [11,25]).
5. *Disciplines have developed specific research methods according to their specific research requirements*. Research methods in modeling are mostly derived from research methods in software engineering adapted to the specificities of modeling. More work on developing more specific methods that can be employed when applying modeling in other fields will be needed.
6. *Disciplines must have some institutional manifestation in the form of subjects taught at universities or colleges*. Modeling courses are part of the syllabus in most computing curricula, and discussions on how to better teach modeling are an important part of every modeling conference (e.g., see the annual Educators Symposium at the Models conference). There was even an attempt to create a full postgraduate course on model-driven engineering [13]. Even if short-lived, this experience showed that modeling is rich enough to be the focus of a full teaching specialization.

Based on these criteria, we claim that modeling does satisfy the requirements to be considered as a new academic discipline. The aspects where modeling may still be somehow lacking will be quickly fixed by the same process (and corresponding visibility and experience) of *migrating* from a mostly software-focused field to a more general applicability.

In particular, modeling should be regarded as a transdiscipline as its goal is to provide holistic solutions that cross disciplinary silos [15].

4 Roadmap

A full characterization of this new modeling discipline is far beyond the goal of this proposal. However, this section highlights a few first important aspects that we should keep in mind when doing so. Note that some aspects to be discussed are not just technical but involve social and economical dimensions of modeling as well.

4.1 Keep a broad perspective of what modeling is

If you ask software practitioners, they will typically equate modeling to UML. Or, even worse, "UML from which to generate some partial code." In the new modeling discipline, models are much more than software models for forward engineering. Models can be partial, used just for sketching or as blueprints,[6] and focus on rather different functions of the modeled system [16,23]. Similarly, if you ask systems engineers, they will think of SysML models to represent the structure of their systems and Simulink or Modelica models to describe the equations of their behavior; these models provide partial views of the system, which must be 'somehow' merged together and combined with other models, e.g., battery power consumption models. A civil architect can think in terms of BIM (Building Information Models), combined with strength models of the construction materials. In biology, different models can recreate aspects of the functioning of human tissues or diseases, or represent strands of DNA. In short, every community tends to have a very narrow view of what modeling is. We all need to make an explicit effort to evolve toward a more inclusive perspective.

Indeed, each engineering or scientific domain requires different types of models and their combinations. So far, we have a solid understanding of those combinations more typical of software projects. We will need to develop appropriate bridges between these models, as well as new modeling techniques for other relevant combinations, often involving going all the way down from exploratory models to simulation ones. Multi-paradigm modeling [4,29] should also play a key role in this context.

4.2 Community

Even if we have the most inclusive perspective of modeling, we will soon realize that other communities do not call what they do modeling, or assign a different meaning to the word model. Or maybe they are just not aware that what they do could be regarded as part of a more global modeling initiative.

Before setting on a precise characterization of the new modeling discipline, we will need to conduct a systematic search of the many uses of modeling in different knowledge fields and invite people from each field to join the initiative. The former is more difficult than the latter. Once we identify the right people, they will likely be happy to participate, as many of them are already convinced of the benefits a proper modeling strategy could bring to their field—they just lack the expertise of doing it alone and the time to learn on their own how their "modeling" complements other "modeling" approaches.

4.3 Teaching modeling

As shown in Sect. 3, disciplines should be linked to teaching initiatives. We can probably take MBEBOK [10] as starting point, with slight modifications to broaden the scope.

Then, on top of this core content, we will need to see what specializations are needed for those domains with specific particularities. Probably, also depending on the profile of the student, since now not all students will come with a computer science background, so modeling for non-technical people will be an important point here. The systematic review above should shed some light on the number and shape of such specializations. In any case, variations of data and process modeling initiatives will probably show up in all domains, as they all need to store and process data. In particular, data interchange formats are the only typical fully specified domain description models we see in scientific fields.

Another important aspect is to have a clear distinction between "modeling users" and "modeling developers." The first group needs to understand the modeling concepts, when and where modeling can help, how to model using a given DSL, or the tools to use, for example. The second group is the one that will create the DSLs, transformations and all the tooling infrastructure that the first group needs and will be using in their daily practice. In current modeling courses, both profiles are mixed, which is a mistake [12]. Many modeling users have zero needs (and interest) in being also modeling developers and will only get confused if we try to explain these advanced concepts to them.

4.4 User-driven DSLs

A key role of the *modeling developers* profile will be the definition of DSLs for all the new domains we will be targeting. Definitely, there is no general modeling language (like UML) for the whole scientific field. While we can envision some DSL hierarchy, there is not much in common between a DSL for water treatment plans [27] and one for internal cell functions [14].

Defining new DSLs is a must as most domains use informal models (beyond low-level mathematical models for simulation) right now, not backed by any formal structure definition, i.e., a metamodel or grammar. The DSL is implicit (and ambiguous). Thus, the first step to evolve toward a

[6] https://martinfowler.com/bliki/UmlMode.html.

proper modeling approach is making the DSLs explicit to clarify the important knowledge of the domain that should be represented and exchanged.

At the same time, we need to keep in mind that DSLs will now need to involve, more than ever, the *modeling users* of the DSL. Firstly, because we will be further than ever from the domain of the DSL. Until now, we could often rely on some partial knowledge of the domain when creating the DSL as many DSLs were semantically closer to our own software or systems background. This may not be the case here. Secondly, because we should not be the only ones creating the DSLs. With modeling becoming an independent discipline, people with different backgrounds may decide to take the role of *modeling developer*s. And this may be in fact needed for scalability purposes as there may not be enough modeling experts to cover all the modeling developer needs. Issues such as the types of language workbenches that will be needed for this type of users, how to gather language requirements from this variety of domains, or the notations that are most effective for them remains to be seen.

4.5 Strong focus on usability

To successfully integrate new domains in our new modeling discipline, it must be easy to onboard new users. Usability is already an issue with technical people [1,31], we will need to make an extra effort for users with different backgrounds.

Improving existing modeling tools may not be enough. Instead, we should explore how to bring modeling to the tools they already use so that they are not forced to learn new tools. Or at least not new tools in a completely new environment or ecosystem.[7]

Even better if models do not need to be created from scratch. A partial model could be automatically derived from the data the user has already available or any low-level mathematical definitions it has already formulated. Or we could extend our perennial goal of model repositories [18,24] to cover also non-software models. If none of this is possible, modeling assistants could also be very helpful [22].

4.6 Economics of modeling

Following up on the previous point, a modeling discipline needs to cover also the economic aspects of modeling. Economic models to calculate the ROI (return on investment) of modeling and other relevant economic metrics, depending on a number of domain and application parameters, should also be a core element of the discipline.

This is already an important point when discussing the adoption of modeling in the software industry, where the ROI should be easier to calculate as it is a more quantifiable domain. This will be even more problematic to compute in less engineering and more scientific domains.

While modeling experts may appreciate the beauty or elegance of a modeling solution, any other user will be only driven by the perceived benefits of the modeling tool implementing such solution. Modeling will need to embrace this pragmatic perspective.

4.7 Publishing on modeling topics

The foundation of a modeling discipline will need to involve both researchers and practitioners. Like it or not, researchers' participation will be linked to the possibility of publishing their contributions to the discipline in well-respected conferences and journals.

We know publishing interdisciplinary papers is difficult as they do not present a core contribution to any single specific area but to their interrelationship and many venues do not appreciate this type of research contribution. As a community, we must show there exists a viable career path for researchers in the modeling discipline.

5 Conclusions

Modeling is a key activity in any engineering and scientific or engineering field and therefore has the potential to be a *force multiplier* in all of them. However, this will only happen if we are able to adapt our current modeling techniques (so far, mainly oriented toward software and systems engineering challenges) to the needs of those other communities with the goal to enhance the kind of modeling they are already doing with our own modeling expertise and toolsets.

To achieve this goal, we advocate in this paper the elevation of modeling to an independent academic discipline to facilitate this transformation and expansion of modeling, and to encourage the participation of new actors from other knowledge fields.

We see this as a strong and exciting opportunity as well for our community. As modeling experts, we have the chance to influence and learn from all other communities—for example, what better way to validate our modeling assumptions and tools than to try to use them in scenarios we never envisaged?

All models are wrong but some are useful.[8] Let us lay the foundations for a new modeling discipline that can help everyone to obtain and benefit from all these useful models.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

[7] Not to point fingers, but I do see it difficult to convince anybody that our usual Eclipse-based environments are ideal for newcomers.

[8] Aphorism generally attributed to the statistician George Box.

References

1. Abrahão, S., Bordeleau, F., Cheng, B.H.C., Kokaly, S., Paige, R.F., Störrle, H., Whittle, J.: User experience for model-driven engineering: Challenges and future directions. In: Proceedings of MODELS'17, pp. 229–236. IEEE Computer Society (2017). https://doi.org/10.1109/MODELS.2017.5
2. Aho, A.V., Ullman, J.D.: Abstractions, their algorithms, and their compilers. Commun. ACM **65**(2), 76–91 (2022). https://doi.org/10.1145/3490685
3. Andreessen, M.: Why software is eating the world. Wall Street J. (2011). https://a16z.com/2011/08/20/why-software-is-eating-the-world/
4. Barisic, A., Ruchkin, I., Savic, D., Mohamed, M.A., Ali, R.A., Li, L.W., Mkaouar, H., Eslampanah, R., Challenger, M., Blouin, D., Nikiforova, O., Cicchetti, A.: Multi-paradigm modeling for cyber-physical systems: a systematic mapping review. J. Syst. Softw. **183**, 111,081 (2022). https://doi.org/10.1016/j.jss.2021.111081
5. Bézivin, J.: On the unification power of models. Softw. Syst. Model. **4**(2), 171–188 (2005). https://doi.org/10.1007/s10270-005-0079-0
6. Booch, G.: The history of software engineering. IEEE Softw. **35**(5), 108–114 (2018). https://doi.org/10.1109/MS.2018.3571234
7. Bronowski, J.: The Ascent of Man. BBC Books, London (1973)
8. Bucchiarone, A., Cabot, J., Paige, R.F., Pierantonio, A.: Grand challenges in model-driven engineering: an analysis of the state of the research. Softw. Syst. Model. **19**(1), 5–13 (2020). https://doi.org/10.1007/s10270-019-00773-6
9. Bui, T.M.A., Papoulias, N., Stinckwich, S., Ziane, M., Roche, B., et al.: The Kendrick modelling platform: language abstractions and tools for epidemiology. BMC Bioinform. **20**(1), 1–13 (2019). https://doi.org/10.1186/s12859-019-2843-0
10. Burgueño, L., Ciccozzi, F., Famelis, M., Kappel, G., Lambers, L., Mosser, S., Paige, R.F., Pierantonio, A., Rensink, A., Salay, R., Taentzer, G., Vallecillo, A., Wimmer, M.: Contents for a model-based software engineering body of knowledge. Softw. Syst. Model. **18**(6), 3193–3205 (2019). https://doi.org/10.1007/s10270-019-00746-9
11. Cabot, J.: Positioning of the low-code movement within the field of model-driven engineering. In: Companion Proceedings of MODELS'20, pp. 76:1–76:3. ACM (2020). https://doi.org/10.1145/3417990.3420210
12. Cabot, J., Kolovos, D.S.: Human factors in the adoption of model-driven engineering: an educator's perspective. In: Proceedings of ER'16 Workshops, LNCS, vol. 9975, pp. 207–217 (2016). https://doi.org/10.1007/978-3-319-47717-6_18
13. Cabot, J., Tisi, M.: The MDE diploma: first international postgraduate specialization in model-driven engineering. Comput. Sci. Educ. **21**(4), 389–402 (2011). https://doi.org/10.1080/08993408.2011.630131
14. Clerx, M., Cooling, M.T., Cooper, J., Garny, A., Moyle, K., Nickerson, D.P., Nielsen, P.M., Sorby, H.: Cellml 2.0. J. Integr. Bioinform. (2020). https://doi.org/10.1515/jib-2020-0021
15. Cohen, E.B., Lloyd, S.J.: Disciplinary evolution and the rise of the transdiscipline. Inf. Sci. **17**, 189–215 (2014). https://doi.org/10.28945/2045
16. Combemale, B., Kienzle, J., Mussbacher, G., Ali, H., Amyot, D., Bagherzadeh, M., Batot, E., Bencomo, N., Benni, B., Bruel, J., Cabot, J., Cheng, B.H.C., Collet, P., Engels, G., Heinrich, R., Jézéquel, J., Koziolek, A., Mosser, S., Reussner, R.H., Sahraoui, H.A., Saini, R., Sallou, J., Stinckwich, S., Syriani, E., Wimmer, M.: A Hitchhiker's guide to model-driven engineering for data-centric systems. IEEE Softw. **38**(4), 71–84 (2021). https://doi.org/10.1109/MS.2020.2995125
17. Easterbrook, S.: Modelling the climate system: Is model-based science like model-based engineering? (keynote). In: Proceedings of MODELS'15, p. 1. IEEE Computer Society (2015). https://doi.org/10.1109/MODELS.2015.7338227
18. France, R.B., Bieman, J.M., Mandalaparty, S.P., Cheng, B.H.C., Jensen, A.C.: Repository for model driven development (remodd). In: Proceedings of ICSE'12, pp. 1471–1472. IEEE Computer Society (2012). https://doi.org/10.1109/ICSE.2012.6227059
19. Kirchhof, J.C., Michael, J., Rumpe, B., Varga, S., Wortmann, A.: Model-driven digital twin construction: synthesizing the integration of cyber-physical systems with their information systems. In: Proceedings of MODELS'20, pp. 90–101. ACM (2020). https://doi.org/10.1145/3365438.3410941
20. Kramer, J.: Is abstraction the key to computing? Commun. ACM **50**(4), 36–42 (2007). https://doi.org/10.1145/1232743.1232745
21. Krishnan, A.: What are academic disciplines? Some observations on the disciplinarity vs. interdisciplinarity debate (2009). https://eprints.ncrm.ac.uk/id/eprint/783/
22. 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.: Opportunities in intelligent modeling assistance. Softw. Syst. Model. **19**(5), 1045–1053 (2020). https://doi.org/10.1007/s10270-020-00814-5
23. Olivé, A.: Conceptual Modeling of Information Systems. Springer, Berlin (2007). https://doi.org/10.1007/978-3-540-39390-0
24. Rocco, J.D., Ruscio, D.D., Iovino, L., Pierantonio, A.: Collaborative repositories in model-driven engineering. IEEE Softw. **32**(3), 28–34 (2015). https://doi.org/10.1109/MS.2015.61
25. Ruscio, D.D., Kolovos, D.S., de Lara, J., Pierantonio, A., Tisi, M., Wimmer, M.: Low-code development and model-driven engineering: two sides of the same coin? Softw. Syst. Model. **21**(2), 437–446 (2022). https://doi.org/10.1007/s10270-021-00970-2
26. Selic, B.: What will it take? A view on adoption of model-based methods in practice. Softw. Syst. Model. **11**(4), 513–526 (2012). https://doi.org/10.1007/s10270-012-0261-0
27. Solís, B., Guisasola, A., Flores-Alsina, X., Jeppsson, U., Baeza, J.A.: A plant-wide model describing GHG emissions and nutrient recovery options for water resource recovery facilities. Water Res. **215**, 118,223 (2022). https://doi.org/10.1016/j.watres.2022.118223
28. Thalheim, B.: Models: the fourth dimension of computer science. Softw. Syst. Model. **21**(1), 9–18 (2022). https://doi.org/10.1007/s10270-021-00954-2
29. Vangheluwe, H., De Lara, J., Mosterman, P.J.: An introduction to multi-paradigm modelling and simulation. In: Proceedings of AIS'02 (AI, Simulation and Planning in High Autonomy Systems), pp. 9–20 (2002)
30. Whittle, J., Hutchinson, J.E., Rouncefield, M.: The state of practice in model-driven engineering. IEEE Softw. **31**(3), 79–85 (2014). https://doi.org/10.1109/MS.2013.65

31. Whittle, J., Hutchinson, J.E., Rouncefield, M., Burden, H., Heldal, R.: A taxonomy of tool-related issues affecting the adoption of model-driven engineering. Softw. Syst. Model. **16**(2), 313–331 (2017). https://doi.org/10.1007/s10270-015-0487-8
32. Zhidchenko, V., Malysheva, I., Handroos, H., Kovartsev, A.: Faster than real-time simulation of mobile crane dynamics using digital twin concept. J. Phys. (2018). https://doi.org/10.1088/1742-6596/1096/1/012071

Jordi Cabot is an ICREA Research Professor at the Open University of Catalonia, Barcelona, Spain, where he leads the SOM lab. His research interests include software and systems modeling, pragmatic formal model verification techniques, analysis of software communities and the role AI can play in software development (and vice versa). For more information, visit https://jordicabot.com

Antonio Vallecillo is full professor of software engineering at the University of Malaga, Spain, where he leads the Atenea research group dedicated to systems modeling and analysis. His research interests include open distributed processing, model-based software engineering, and software quality. For more information, please visit http://www.lcc.uma.es/~av