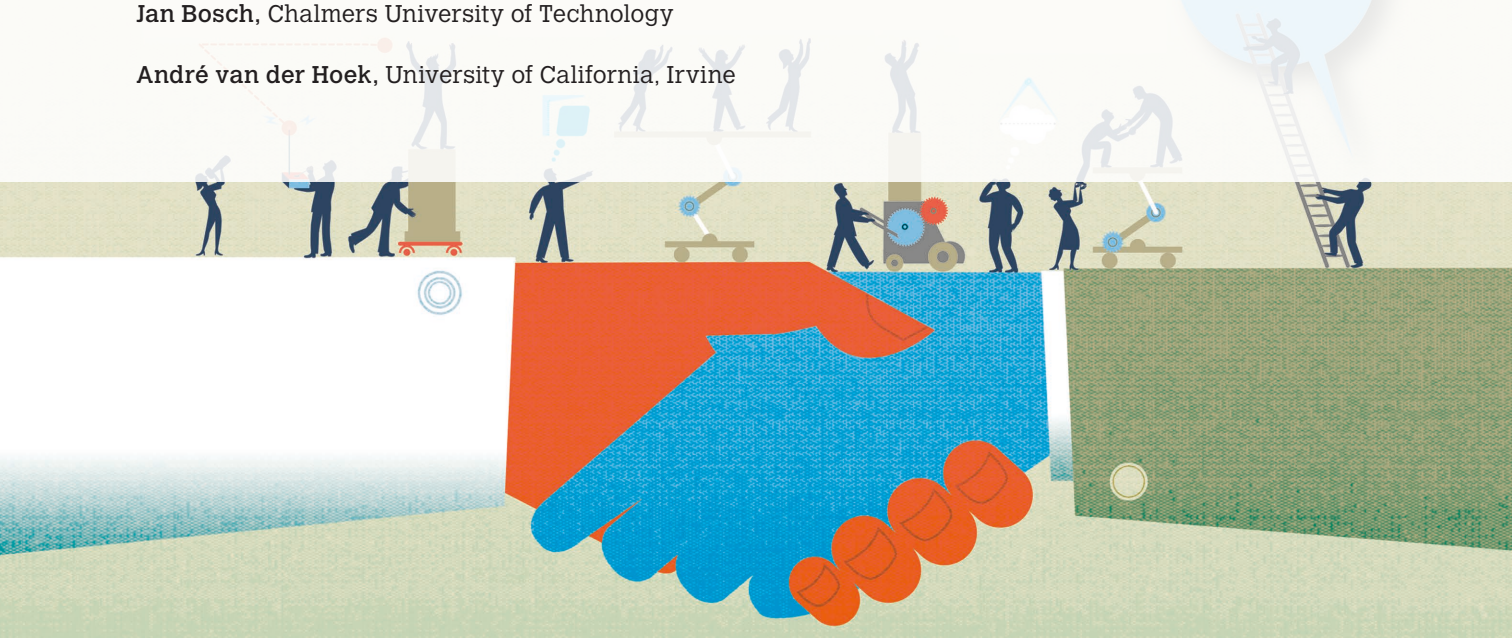
Collaborative Modeling in Software Engineering

Henry Muccini, University of L'Aquila

Jan Bosch, Chalmers University of Technology

André van der Hoek, University of California, Irvine



0740-7459/18/$33.00 © 2018 IEEE

FOR THOUSANDS OF years, humans have engaged in modeling. From conceptual models to predict animal movement during hunting expeditions, to graphical representations on cave walls to document and explain phenomena, humans have used models of all kinds to make sense of the world, predict the future, explore and document designs for making something new, and share knowledge with others.

Someone could argue that the Enlightenment in the 17th and 18th centuries and the adoption of the scientific method shortly thereafter were critically enabled by fundamental improvements in the models used to describe the world. Instead of relying on historical texts, religious beliefs, authoritarian figures, or individuals' opinions (however well reasoned), the use of models based on data and experiments allowed for a level of precision, comprehensiveness, and clarity that enabled new forms of inquiry, knowledge generation, and verification.

Today, models are at the heart of many an endeavor. Buildings are designed using models of all kinds, from paper blueprints to mockups to virtual-reality models—all of which help assess a building's different aspects. CPUs are designed using advanced models that feed into automated verification tools. Modern racing bicycles are designed using CAD/CAM and fluid-dynamics software, so that engineers can verify the bikes' stiffness, compliance, and aerodynamics before they go into production. Researchers in certain disciplines now almost exclusively conduct their explorations on models, instead of working in a lab with petri dishes or some set of organic compounds.

Although models can easily be seen as conceptual, modeling has an important empirical dimension. Experimentation with, as well as the engineering and construction of, real-world systems not only is based on models but also equally informs models and the modeling process. The vocabulary of our models is based on practitioners' empirical needs, and collaboration is driven largely by educated practitioners' ability to read, interpret, use, and change each other's models. In fact, a common modeling language provides the basis for collaboration. So, it's no surprise that a plethora of modeling languages exist and new ones continue to be invented and explored.

Collaboration

Critically, models facilitate collaboration. Behind every architect is a team of specialists contributing to what eventually become the blueprint and planning documents governing the building's construction. Different engineers design different parts of CPUs. Bicycles result from teams working to refine a conceptual first shape into a complete design including the frame, wheels, handlebars, saddle, gears, shifters, brakes, and more—often by incorporating designs from external manufacturers for off-the-shelf (and in some cases custom) components.

And, in science, despite the myth of the lone researcher slaving away to achieve unfathomable intellectual breakthroughs, the reality is that behind every Nobel Prize scientist is a research community that for decades has worked on models and theories. Collaboration, then, can be both direct—as in people working together at the same time on a model—and indirect—as in leveraging a model through communication among parties that otherwise might never interact.

Software engineering is inherently collaborative. Ranging from Amazon's "two pizza" teams to the thousands of software engineers building telecom network equipment at companies such as Ericsson or autonomous-driving solutions at companies such as Zenuity, the creation of large, complex software systems is highly collaborative. Any software project with more than one person is created through *collaborative software engineering*.[1] In this process, collaboration occurs on all sorts of artifacts, including documents detailing a client's workflows, architectural diagrams, source code, and project-planning timelines.

It's no surprise, then, that modeling is central to software development. Models enable the representation, approximation, and view of complex software, letting us understand and constructively contribute to the creation of novel software-intensive systems. (Christof Ebert and Capers Jones reported that the size of software systems grows an order of magnitude—10 times the size—every five to 10 years.[2])

Sometimes, modeling is more formal, with detailed representations developed in dedicated environments with advanced features for creating and analyzing those representations. Other times, modeling is more informal, involving, for instance, whiteboard sketching or working through an architectural decision through pseudocode. Regardless, most often, modeling is highly collaborative: multiple developers work together to create, analyze, and understand a model capturing some aspect of the software on which they're working.

Collaborative Modeling

The topic of collaborative modeling has long been relevant in software. It has been a subject of extensive research, collaborative facilities are part and parcel of modeling tools, and every day hundreds of thousands of developers engage in collaborative modeling of some sort. Yet, a theme issue on the topic is particularly timely, given the following four phenomena.

First, in practice, we're witnessing the emergence of a large number of Web 3.0 collaborative-modeling environments. Anyone can start a collaborative-modeling session nearly instantly with tools such as GenMyModel (https://www.genmymodel.com), emfCollab (http://qgears.com/products/emfcollab), WebGME (https://webgme.org), or Visual Paradigm (https://www.visual-paradigm.com/features/collaborative-modeling). Although some of these are more successful than others, the steady uptick in these tools' use indicates that a significant need exists.

Second, a number of projects are focusing on enabling large teams of modelers to construct and refine large models collaboratively. Examples include

- the Dawn Eclipse project (http://wiki.eclipse.org/Dawn), which investigates collaborative UIs to provide collaborative access for GMF (Graphical Modeling Framework) diagrams;
- the Morsa NoSQL-based collaborative model repository;[3]
- the Modeling Team Framework (MTF; http://www.eclipse.org/proposals/mtf) open source project under the Eclipse Modeling Framework Technology Project (EMFT); and
- UNICASE (https://marketplace.eclipse.org/content/unicase).

These tools aim to go beyond what current tools provide, enabling easier, more powerful collaboration.

Third, we're witnessing a surge in academic publications discussing collaborative modeling from a range of angles. These publications include theoretical contributions offering conceptual frameworks,[4,5] newly proposed tools and frameworks supporting collaborative modeling,[6,7] and new modeling languages designed to be friendlier for collaborative modeling.[8] As with the new tools stemming from the practice community, these academic tools aim to build more powerful and more usable tools.

Fourth, new settings are giving rise to new challenges in the domain. For example, the emergence of software ecosystems requires collaborative-modeling techniques and tools that function over organizational boundaries and across different roles in a company. This, in turn, brings up sharing and intellectual-property issues, because not all of a collaborative model can be or needs to be shared with all parties.

In short, collaborative modeling is at an interesting point. On one hand, it represents one of the more challenging activities in software engineering. As we stated earlier, hundreds of thousands of developers engage in it every day. However, the effectiveness of their efforts, what might be considered best practices, tooling needs, and how to adjust for different settings are not as understood as they should be. On the other hand, we observe high levels of activity in both the research community and in practice, with experimental techniques and novel tools emerging regularly. If we're to continue to develop increasingly complex software, these new approaches must successfully address the many practical challenges.

The Theme Articles

The four articles (selected from a rich set of 24 submissions) in this theme issue elucidate some of these challenges, introduce new techniques, and shine a light on collaborative modeling as an area rich with opportunities for improvement.

"Collaborative-Design Conflicts: Costs and Solutions," by Jae young Bang, Yuriy Brun, and Nenad Medvidović, looks at the problem of conflicts that might arise from multiple developers working on the same model at the same time. Developers might make conflicting changes that are difficult to resolve. The authors

- motivate the need for proactive, early conflict detection;
- identify requirements for conflict detection tools; and
- introduce FLAME (Framework for Logging and Analyzing Modeling Events).

"Secure Views for Collaborative Modeling," by Csaba Debreceni and his colleagues, addresses the problem of sharing only parts of a model with different parties, so as to protect intellectual property. The authors

- identify key challenges in collaborative modeling, particularly the need for secure views to enable role-based access to parts of a model;
- introduce the MONDO collaboration framework, which supports such sharing at a fine-grained level; and
- summarize this approach's benefits and limitations.

"Does Distance Still Matter? Revisiting Collaborative Distributed Software Design," by Rodi Jolak

and his colleagues, revisits the classic 2000 "Distance Matters" paper by Gary Olson and Judith Olson.[9] Nearly two decades later, with a host of powerful collaboration tools available, the question arises: Does distance still matter? Jolak and his colleagues answered this question through a carefully constructed experiment in which they analyzed how distance affects design decisions, collaborative communication, and the technical and social challenges. They found that, yes, distance very much still matters, particularly because of a lack of social awareness and trust.

Finally, "Collaborative Modeling and Group Decision Making Using Chatbots in Social Networks," by Sara Pérez-Soler, Esther Guerra, and Juan de Lara, explores a radically different way for developers to collaborate in building a model. SOCIO, an experimental chatbot,

- listens to developers' conversations,
- interprets specific sentences,
- automatically builds a diagrammatic representation of the concepts being discussed, and
- enables developers to come to a consensus about the discussed model.

These four articles provide only a glimpse of what's happening in collaborative modeling. Many other tools are being researched and developed, other studies are being performed, and many interesting experience reports exist. To learn more about the state of the art and research, see the websites of the International Workshop on Collaborative Modeling in Model-Driven Engineering (http://cs.gssi.it/commitmde2018) and the Workshop on Modeling in Software Engineering (https://sselab.de/lab2/public/wiki/MiSE/index.php?title=Main_Page), and Jonas Sorgalla and his colleagues' survey involving a classification framework and research roadmap.[4]

We thank the authors of all the submitted papers. This field of research wouldn't exist without the relentless work of many who are addressing the challenges of increased software system complexity and team development practices through collaborative modeling. We hope you readers enjoy the authors' insights as much as we enjoyed preparing this issue for you.

ABOUT THE AUTHORS

HENRY MUCCINI is an associate professor of computer science at the University of L'Aquila. His research interests include the role of software architectures, model-driven engineering, and software verification and validation in producing higher-quality systems. Muccini received a PhD in computer science from the University of Rome—La Sapienza. He's on the *IEEE Software* editorial board. Contact him at henry.muccini@univaq.it.

JAN BOSCH is a professor of software engineering and the Software Center director at Chalmers University of Technology. His research interests include software ecosystems, compositional software engineering, software architecture, and software product lines. Bosch received a PhD in computer science from Lund University. He's on the *IEEE Software* advisory board. Contact him at jan@janbosch.com.

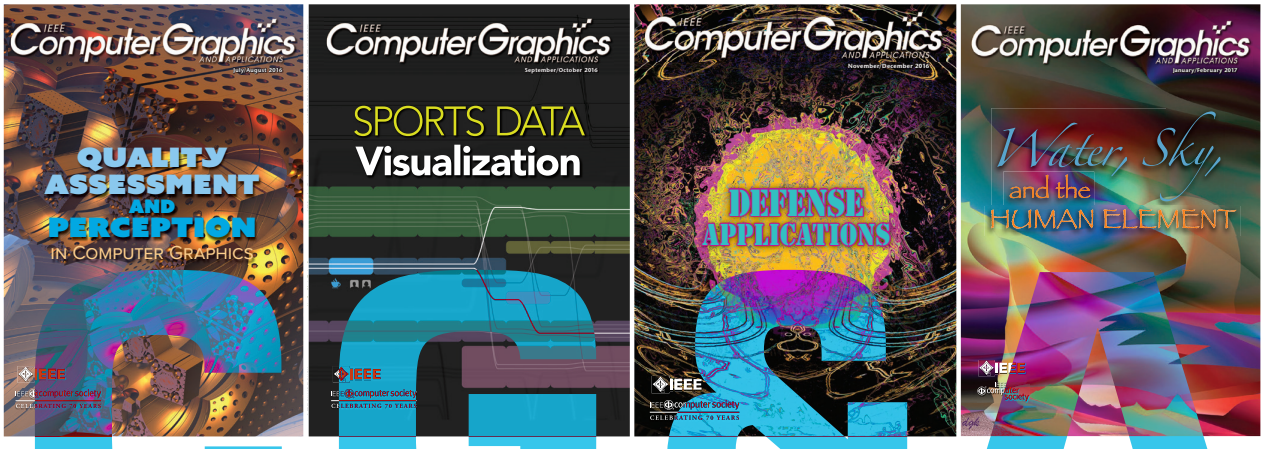
ANDRÉ VAN DER HOEK is a professor in, and the chair of, the Department of Informatics at the University of California, Irvine. His research interests are design, collaboration, and education in software engineering. Van der Hoek received a PhD in computer science from the University of Colorado Boulder. Contact him at andre@ics.uci.edu.

References

1. I. Mistrık et al., eds., *Collaborative Software Engineering*, Springer, 2010.
2. C. Ebert and C. Jones, "Embedded Software: Facts, Figures, and Future," *Computer*, vol. 42, no. 4, 2009, pp. 42–52.
3. J.E. Pagán and J.G. Molina, "Querying Large Models Efficiently," *Information and Software Technology*, vol. 56, no. 6, 2014, pp. 586–622.
4. M. Franzago et al., "Collaborative Model-Driven Software Engineering: A Classification Framework and a Research Map," to be published in *IEEE Trans. Software Eng.*; doi:10.1109/TSE.2017.2755039.

5. C. Masson, J. Corley, and E. Syriani, "Feature Model for Collaborative Modeling Environments," 2017; http://ceur-ws.org/Vol-2019/commitmde_5.pdf.
6. P. Nicolaescu et al., "Near Real-Time Collaborative Modeling for View-Based Web Information Systems Engineering," *Information Systems*, May 2018, pp. 23–39.
7. N. Rußkamp and A. Nicolaescu, "Tool Support for Collaborative UML Modelling," *Continuous Software Engineering and Full-Scale Software Engineering*, 2018, pp. 43–48; https://www2.swc.rwth-aachen.de/docs/teaching/seminar2018/CSE%20FsSE%202018.pdf.
8. J. Sorgalla et al., "On Collaborative Model-Driven Development of Microservices," 2018; https://arxiv.org/abs/1805.01176.
9. G.M. Olson and J.S. Olson, "Distance Matters," *Human–Computer Interaction*, vol. 15, nos. 2–3, 2000, pp. 139–178; http://dx.doi.org/10.1207/S15327051HCI1523_4.

myCS Read your subscriptions through the myCS publications portal at http://mycs.computer.org



IEEE Computer Graphics and Applications bridges the theory and practice of computer graphics. Subscribe to *CG&A* and

- stay current on the latest tools and applications and gain invaluable practical and research knowledge,
- discover cutting-edge applications and learn more about the latest techniques, and
- benefit from *CG&A*'s active and connected editorial board.