ChatGPT in software modeling

Benoit Combemale, Jeff Gray, Bernhard Rumpe

Software and Systems Modeling (2023) 22:777–779
https://doi.org/10.1007/s10270-023-01106-4

EDITORIAL

ChatGPT in software modeling

Benoit Combemale[1] · Jeff Gray[2] · Bernhard Rumpe[3]

Published online: 11 May 2023

We wonder—how many SoSyM readers have tried using OpenAI's ChatGPT to formulate prompts to explore the interesting results that may be produced?

It is really fascinating to consider the results when the query is defined appropriately. Of course, we abstained from using ChatGPT to write any part of this editorial, but if we actually would generate portions of our text, we wonder how difficult it would be for readers to identify the generated parts. ChatGPT is rather chatty, but even the produced text can be adapted by asking it for a specific answering style. The options and future extensions seem to be endless. In the context of this editorial, we do not consider legal or ethical issues of ChatGPT—these have to be sorted out eventually. We also do not discuss the problems of having ChatGPT text, or text generated by other Large Language Models (LLMs), in scientific papers. Rather, in this editorial, we focus more positively and ask the main question:

How will ChatGPT be able to help us in a development process, especially in the tasks of developing or using models for analysis, production or understanding of software and systems?

ChatGPT can provide useful pieces of programs that are sometimes correct and perform the task that the user asked to be implemented. ChatGPT also knows several textual modeling languages. This even seems to hold for DSLs. The example of GraphGPT suggests that a language designer can tell ChatGPT what the desired modeling language looks like and it will produce pieces within that language. GraphGPT uses the trick of asking for a JSON-encoding of the graph that is rendered into a diagram. The options seem endless and we do not have the prophetic ability to predict precisely how generative AI will change the business and activities of modeling in the future.

But where will the changes occur? We see several options:

(1) Modeling as we know may go completely out of business, because a helpful improved version of ChatGPT will directly map the informal description of a system's requirements into executable code. In this case, end-user programming will have arrived and will be in place in many simpler cases for user-defined languages for everyday use (e.g. customized languages for controlling our lighting or heating systems at home).
(2) Another possibility is that models as an intermediate step between informal requirements and the executable solution will still exist, but the transformation between requirements to the models are done by ChatGPT. The transformation between the models and the executable solution are then either automated by a standard code generator or ChatGPT.

Less bold alternatives could be that ChatGPT is leveraged partially as a kind of recommender or modeling assistant, but still lets the modeler be responsible and the final owner of the produced models. It may also be that this is only an intermediate step to more automated alternatives. The modeler's expertise could be reduced to fine tuning of requirements in such a way that the LLM in use will produce desired, executable results, but is not able to understand, review and assess the generated models.

Using models as intermediate steps may have some advantages over the direct mapping of requirements to code. First, humans may look at generated models and use them as a review point to better ensure the software's quality. Second, the results may become improved, when ChatGPT does not have to make the big step from requirements to implementation, but may go in several small and smooth transformations, using the idea of platform-independent, platform-dependent and finally computational models. These intermediate models may still be models of the standard languages, like UML or SysML or appropriate DSLs, allowing the already existing analysis techniques to be used in between. In critical

✉ Bernhard Rumpe
bernhard.rumpe@sosym.org

Benoit Combemale
benoit.combemale@sosym.org

Jeff Gray
jeff.gray@sosym.org

[1] University of Rennes, Rennes, France

[2] University of Alabama, Tuscaloosa, AL, USA

[3] RWTH Aachen University, Aachen, Germany

domains, it may be the case that generated models are then used for certification, verification and other kinds of quality assurance techniques. It may be beneficial to have ChatGPT generate recommendations and explanations, or even verifiable proofs, on why the code fits to a model. In less critical domains, we may still use all of these intermediate steps and only ask ChatGPT for explanations of the tracing between models and code.

ChatGPT also can be applied for identifying potential test cases or test data sets that can then be used as parts of traditional test-driven development. However, it is currently unclear whether or how intensively ChatGPT could be able to produce reliable oracles for for black-box tests of code.

As ChatGPT improves at explaining its results, we might also be tempted to ask it to explain pieces of the code or models to us. This would offer the possibility of using ChatGPT as a kind of replacement for the traditional forms of documentation.

There are other forms of usages; for example, using ChatGPT to produce the correct configuration for a complex software stack or to produce GUI elements or web pages for visual representation of collected data and information.

In personal discussions, opinions have been expressed that it should be an algorithm that performs the modeling, because humans are too expensive. Of course, if automation from generative AI is possible and the results can be truly trusted, then we should consider it. Software engineers always seek higher degrees of automation and the use of generative AI is a viable possibility.

We should not forget that ChatGPT actually draws its knowledge from already existing information found in many sources (e.g., scientific papers, Stack Overflow, arxiv, and archived GitHub projects). Thus, innovations in generative AI presumably do not come from fresh insights, but from re-combinations of existing solutions and knowledge (and sometimes only our expectations of a presupposed solution, which only had to be found). In the case of models, we continue to be constrained by a problem that has existed for many years—we do not have large and open libraries of models, in the same magnitude as open source code already exists. Modeling has suffered from not being able to create large model repositories for various reasons. The lack of open libraries of models has prevented us from reusing elaborate predefined models, and now the absence of such model repositories limits the ways that ChatGPT can learn about modeling theory and practice. Maybe it is time to change this?

We wish you a fruitful time reading the papers in this issue and thinking about your future experiments with ChatGPT. We solicit future submission on this interesting topic! To start this discussion, we have included in this issue an expert voice from Javier Cámara, Javier Troya, Lola Burgueño, and Antonio Vallecillo where they address ChatGPT for modeling in a detailed experience report.

Content of this Issue

1. **Expert Voice**
   - "On the assessment of generative AI in modeling tasks: an experience report with ChatGPT and UML" by Javier Cámara, Javier Troya, Lola Burgueño, and Antonio Vallecillo
2. **Theme Section on Modeling Language Engineering**
   Guest editors: Benoit Combemale, Romina Eramo, and Juan de Lara
3. **Regular Papers**
   - "Tracing security requirements in industrial control systems using graph databases" by Awais Tanveer, Chandan Sharma, Roopak Sinha, and Matthew Kuo
   - "MoDMaCAO: a model-driven framework for the design, validation and configuration management of cloud applications based on OCCI" by Faiez Zalila, Fabian Korte, Johannes Erbel, Stéphanie Challita, Jens Grabowski, and Philippe Merle
   - "Reference architectures modelling and compliance checking" by Alessio Bucaioni, Amleto Di Salle, Ludovico Iovino, Ivano Malavolta, and Patrizio Pelliccione
   - "A data-driven approach for constructing multilayer network-based service ecosystem models" by Mingyi Liu, Zhiying Tu, Xiaofei Xu, Zhongjie Wang, and Yan Wang
   - "Formal translation of YAWL workflow models to the Alloy formal specifications: a testing application" by Mehran Rivadeh and Seyed-Hassan Mirian-Hosseinabadi
   - "A systematic literature review on IoT-aware business process modeling views, requirements and notations" by Ivan Compagnucci, Flavio Corradini, Fabrizio Fornari, Andrea Polini, Barbara Re, and Francesco Tiezzi
   - "Modeling difficulties in creating conceptual data models: Multimodal studies on individual modeling processes" by Kristina Rosenthal, Stefan Stecker, and Monique Snoeck
   - "Conflict management techniques for model merging: a systematic mapping review" by Mohammadreza Sharbaf, Bahman Zamani, and Gerson Sunyé

A note in our rules of conduct: An Editor-in-Chief, like Benoit, usually does not lead theme sections or publish papers in SoSyM. This rule is useful to address concerns related to conflicts of interest and an appearance of bias for all SoSyM publications. The theme section presented in this issue, however, was organized by Benoit before he knew that he would

become SoSyM Editor-in-Chief and was supervised before his appointment by Bernhard and Jeff.

Funding Open Access funding enabled and organized by Projekt DEAL.