Software and Systems Modeling (2022) 21:435–436
https://doi.org/10.1007/s10270-022-00999-x

EDITORIAL

Modeling in advanced systems engineering

Jeff Gray[1] · Bernhard Rumpe[2]

Published online: 30 March 2022

Previous editorials highlighted the use of models and the activity of modeling as an integral part of almost all development activities for complex systems. Complexity is an omnipresent challenge, for software, for mechanical systems with intelligence (e.g., autonomous cars, robots, healthcare gadgets or airplanes), for production plants, and also for biological systems (e.g., advanced medicine, human cells, organs, and even full organisms), that will be engineered with increased frequency in the future. "Systems of systems" are typically confederated and collaborating ensembles of systems that were originally designed individually, but through new opportunities that were not originally envisioned, are now required to tightly cooperate. Examples of such systems include telecommunications services, multiple integrated web services across the internet, energy networks and also the city networks of collaborating smart buildings and transportation infrastructure.

All of the examples listed previously need models. During development of these initially independent systems, models are often defined before the system exists, requiring explicit modeling activities to explicate requirements on the system to document multiple design decisions and alternatives. This is an entirely different process from how models come to life when using data science techniques to extract models (e.g., of behavior or typical dynamic configurations) from observations of existing systems. These other kinds of extracted models are helpful when optimizing existing systems and processes, but are not intended for the original design (which may have occurred by nature). However, it is our belief that even though the process for model creation and use is different across various domains, the same underlying modeling paradigms and concepts can often be incorporated. As a consequence, these models could also be adopted in similar modeling languages.

This brings us to the interesting question of whether and how far a modeling language from UML or SysML (e.g., Statecharts or Activity Diagrams) is already fit for this purpose? For example, how much extension is needed to add concepts such as modeling uncertainty and statistical distribution of possible behaviors? Even though there is much research in these domains that has produced compelling results, how long will it take to see these results made available in broadly used tools? We believe that much progress still needs to be made in order to realize the full potential.

Currently, there is an ongoing effort to inject more explicit modeling techniques into the Systems Engineering discipline, which we define as including the intersection of the mechanical, electrical and software subdomains. Software Engineering already provides many modeling techniques to address complexity and architectural structure. This is also true for Electrical Engineering, with Mechanical Engineering supporting the use of many modeling ideas in different contexts. The concepts of modeling across these disciplines can expand the ability of a systems engineer to address safety, security, robustness and operability concerns, as well as many other desirable properties of the systems to be developed and produced in the future.

This observation brings us to the well-known challenge that most of these modeling techniques do not integrate very well. As a consequence, it is still frustratingly difficult to define an appropriate tool chain that adequately addresses a new project's needs (e.g., systems analyses and synthesis activities).

From a Software Engineering point of view, the existence of components developed using Artificial Intelligence does not really change the game. Software Engineering has established the foundational understanding that software is never completely correct and the unpredictable behavioral errors must be detected and absorbed internally to prevent full system failure. The integration of AI components becomes just another type of unreliable component that must be considered. Specific precautions must be taken to address the

✉ Bernhard Rumpe
bernhard.rumpe@sosym.org

Jeff Gray
jeff.gray@sosym.org

1 University of Alabama, Tuscaloosa, AL, USA

2 RWTH Aachen University, Aachen, Germany

possible failures of AI-based components. Even the idea of self-explainability is not an entirely new concept. A common challenge of Software Engineering is the need to explain to a user what the software is doing or why it cannot do a specific requested action. The existence of explicitly defined models can be helpful in explaining behavior to users (e.g., using state model as underlying concepts).

The idea of "Advanced Systems Engineering" is an approach that tries to address many of the current challenges, often by adding AI-based development techniques for the software needs of the system. However, many of the existing modeling techniques, methods and practices of Software Engineering are often ignored because of an assumption that software need not be developed in the future as in the past. This approach places an emphasis on training software through condensing available data to directly embeddable models. Yet, this idea will only work in very very narrow cases and it is not clear what to do with the majority of the complex software that is not addressed by an "Advanced Systems Engineering" approach. Innovative systems seem to derive their innovations to a larger extent from the software and software services around the mechanical part of the system.

Computer Science (in general) and Software Engineering (in particular) have not been able to transfer all of their knowledge, techniques and methods to Systems Engineering. There is much to do and the use of an integrated set of modeling techniques, defined in explicit modeling languages and assisted by an integrated toolchain, is still far away. Software development methods, such as Agile Methods or DevOps, amidst the general use of models is steadily increasing in the integrated Systems Engineering domain.

It will be interesting to see which forms of model modularity will be more successful and beneficial for Systems Engineers in the long-term and how all of these forms and their use of models will collaborate during design and operation of advanced, intelligent systems.

1 Content of this Issue

1. **Expert Voice**
   - "Low-code development and model-driven engineering: Two sides of the same coin?" by Davide Di Ruscio, Dimitris Kolovos, Juan de Lara, Alfonso Pierantonio, Massimo Tisi, and Manuel Wimmer
2. **Theme Section on Multi-Level Modeling**
   Guest Editors: Adrian Rutle and Manuel Wimmer
3. **Regular Papers**
   - "Probabilistic modelling and verification using RoboChart and PRISM" by Kangfeng Ye, Ana Cavalcanti, Simon Foster, Alvaro Miyazawa, and Jim Woodcock
   - "Model-based test case generation and prioritization: a systematic literature review" by Muhammad Luqman Mohd-Shafie, Wan Wan Kadir, Horst Lichter, Muhammad Khatibsyarbini, and Mohd Adham Isa
   - "A cross-technology benchmark for incremental graph queries" by Georg Hinkel, Antonio García-Domínguez, Rene Schöne, Artur Boronat, Massimo Tisi, Théo Le Calvar, Frédéric Jouault, József Marton, Tamás Nyíri, János Benjamin Antal, Márton Elekes, and Gabor Szarnyas
   - "Contrasting dedicated model transformation languages versus general purpose languages: a historical perspective on ATL versus Java based on complexity and size" by Stefan Höppner, Timo Kehrer, and Matthias Tichy

Funding Open Access funding enabled and organized by Projekt DEAL.