Software and Systems Modeling (2020) 19:261–262
https://doi.org/10.1007/s10270-020-00784-8

EDITORIAL

# Compositional model analysis

Jeff Gray[1] · Bernhard Rumpe[2]

Published online: 3 March 2020

Recently, one of the editors attended a Dagstuhl seminar about "Composing Model-Based Analysis Tools" organized by Francisco Durán, Robert Heinrich, Diego Pérez-Palacín, Carolyn L. Talcott, and Steffen Zschaler [1].

Never heard about Dagstuhl? Schloss Dagstuhl is a wonderful venue that offers invitation-only Computer Science seminars every week of the year (except Christmas). It regularly manages to bring together experienced scientists and young aspiring researchers with interest in a specific research theme to foster the discussion of exciting topics. Dagstuhl is located in a rural area of Germany, which helps to remove distractions from other concerns. Attendees are compelled to spend time together while focused on scientific and social discussions that are detailed, interesting, and fruitful. The local organizers and staff do everything to help participants focus on collaborative scientific activities. The food is exceptional, and the organization is always professional and attentive. An invitation to Dagstuhl is an honor that should be deeply considered when offered!

SoSyM regularly benefits from Dagstuhl seminars through papers that originate from these discussions.

The "Composing Model-Based Analysis Tools" seminar had approximately 50 attendees who explored various topics related to model analysis; in particular, how model analysis techniques can be modularized in such a way that they can be composed and built upon each other. There are various examples where simple analysis techniques (e.g., typing of expressions, completeness and reachability algorithms) are the foundation of analysis, with other desired characteristics (e.g., reliability, efficiency, and security metrics) built from the foundational concepts.

The participants at this seminar represented diverse backgrounds. They introduced many different ways *models* are used to describe *views* on *systems* and their (potentially hierarchically decomposed) *components*. One model typically describes a particular view on a part of the system (e.g., subsystem, component). Thus, many models are often used to describe a larger system. The multiple-models systems view can be contrasted with the Big-Blob-Gozilla-Model that covers everything, with limited potential for abstraction and thus reducing the benefits of modeling. A single large model also prevents reuse of any independently developed model and has some other drawbacks (e.g., less potential for collaborative design). The idea of compositional model analysis can benefit from much of the earlier research that intersected programming languages and software engineering to promote a notion of composition together with the idea of encapsulation (e.g., the works of Parnas, such as "On the criteria to be used in decomposing systems into modules" and "Modular structure of complex systems").

However, as models are being composed, it is an interesting question to ask whether the model composition structure and system composition structure are orthogonal, or exactly the same. Several examples have been discussed that support both variants (e.g., several models can be used to describe different aspects of the same system component, such as interface, internal data structure, behavioral constraints, or implementation; but, e.g., typically one architecture diagram describes many system components). The situation becomes more wicked as various viewpoints of the system are modeled using different modeling techniques, which enforces that not only models, but also the modeling languages need to be composed (e.g., usually quality analyses depend on very specific quality attributes evenly spread over many models, but defined within one sub-language; on the contrary, interaction analysis can be spread over many forms of interactions, imports, includes, and other forms of uses spear across many sub-languages).

Modeling is not a simple and easy task. Deep skills and expertise are needed to create models that are useful, well-structured, understandable, and correct. Models offer many benefits, from the sketching of new ideas to the full formal specification of a dynamic system. Many industry applica-

✉ 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

tions of modeling also focus on the generation of code from the higher abstractions provided in a model. There is a second benefit to modeling, namely *understanding the system under development* by *applying analysis techniques* to the representative model. This benefit of model usage has not been consistently present in commercial modeling tools, at least not for the currently available UML and SysML tooling. It can be speculated why this is the case. To some extent, we believe that the commercial pressures often drive vendors to market their tooling solutions to a broad range of customers. This may require the same notation to be used in very different forms and styles, which in turn restricts the expressive capability, resulting in the analysis algorithms of the tools reporting some error in their model to the user. This is a similar effect with strongly typed programming languages, where an experienced software developer may appreciate this kind of help, while in certain domains a developer may feel too constrained by the typing system and prefer to explore untyped code.

However, in the many situations when analysis is helpful, a larger part of the community will adopt compositional model analysis and become more efficient in developing reliable and reusable systems. Analysis techniques are needed for all of our modeling languages. In order to support reusability of analysis techniques, we must investigate and develop techniques that allow us to parameterize, refine, adapt, and especially compose analysis techniques for different variations of modeling languages, such as specializations of UML, SysML, and also our own customized DSLs.

We also need deeper understanding on whether the composition of analysis techniques is orthogonal to the composition of models and their languages, and how that relates to the orthogonality of the composed systems. Each of the variants may actually apply. It will be beneficial to see more investigation and results on compositional analysis techniques in the future.

Content of this Issue

1. **Expert Voice**
   - "The Software Language Extension Problem" by Manuel Leduc, Thomas Degueule, Eric Van Wyk, and Benoit Combemale
2. **EMMSAD 2018 Special Section**
   Guest Editors: Iris Reinhartz-Berger and Sérgio Guerreiro
3. **ICMT 2018 Special Section**
   Guest Editors: Jesús Sánchez Cuadrado and Arend Rensink
4. **Regular Papers**
   - "Modelling the interplay of security, privacy and trust in sociotechnical systems: a computer-aided design approach" by Mattia Salnitri, Konstantinos Angelopoulos, Michalis Pavlidis, Vasiliki Diamantopoulou, Haralambos Mouratidis, and Paolo Giorgini
   - "ExpRunA: a domain-specific approach for technology-oriented experiments" by Eneias Silva, Alessandro Leite, Vander Alves, and Sven Apel

We hope that you develop new insights and observations when reading the articles in this issue.

## Reference

1. Dagstuhl Seminar on "Composing Model-Based Analysis Tools" https://www.dagstuhl.de/19481, organized by Francisco Durán, Robert Heinrich, Diego Pérez-Palacín, Carolyn L. Talcott, and Steffen Zschaler, Nov. 2019