Software & Systems Modeling (2018) 17:1053–1054
https://doi.org/10.1007/s10270-018-0694-1

EDITORIAL

Agile model-based system development

Jeff Gray[1] · **Bernhard Rumpe**[2]

Published online: 24 August 2018

Our journal is called "Software *and* Systems Modeling," but when we look at most of the papers that have been published in SoSyM recently, the focus is much more on *software* than on other kinds of *systems*. It is worth a fresh look at how other engineering disciplines use models.

In fact, modeling is at the heart of almost any engineering discipline. Thus, it is not surprising that our engineering colleagues have developed a detailed portfolio of modeling techniques to describe their systems in various perspectives, viewpoints, and abstractions. Furthermore, a plethora of tools has been developed to assist practitioners with each of the individual modeling languages. Colleagues from traditional engineering disciplines model many more aspects of their systems than software developers.

However, the state-of-the-art in systems modeling has several challenges, where each modeling aspect and view is often assisted by an individual modeling and analysis tool. Data exchange between the tools is complicated, even though mostly automated, but suffers from concerns about robustness, completeness of the mappings between the models, as well as regular version upgrades of tools. A second problem is that these mappings between models are not easy to standardize, because different projects use the same modeling languages in different forms (semantics), which enforces configurable mappings or individually developed translations per project. We have discussed these aspects on modeling partially in previous editorials (please see http://www.sosym.org/editorials/).

An important aspect of a modeling toolchain concerns the "length" of the toolchain—the longer the toolchain, the more information that may be added along the toolchain, potentially reducing the agility of the development process. Agility is a well-known software development technique that dominates smaller- and medium-sized software projects. Many projects have demonstrated that an agile, lean development process has merits in terms of cost reduction, quality, and time to market. Driven by the large software companies in Silicon Valley that have a development and innovation pace far beyond traditional engineering, there are currently several attempts to adapt agility to systems development, with examples including smart phones, autonomous and electrified cars, and also medical devices and home automation. Therefore, it is worthwhile to revisit the most important ingredients required by an agile process, which are:

- Lean processes with as little documentation overhead as possible,
- Immediate feedback as soon and early as possible,
- Feedback on all activities,
- Automation of many development tasks,
- Many, small iterations,
- Self-responsibility for developers to do whatever seems best at the current situation.

Traditional engineers use system models to design and understand the product through analysis and automation of engineering tasks. Therefore, models should be more than just documentation artifacts. Each model that captures a perspective of the project should either become a part of the product construction or should be used for automated validation and test quality management. Although synthesizing software products from models is a common practice in the software engineering domain, physical products can also be produced (e.g., with appropriate 3D printers). In systems modeling, simulations of the physical product in various abstractions and for various validation and testing purposes can be a core benefit of modeling. Simulations help to understand product sustainability (e.g., deterioration), usability of a product in its context (e.g., autonomous driving), and many additional aspects that are more difficult to explore as "what-if" analyses without modeling support. The benefit of models and simulation occurs when validation steps are available on very early versions of a model and not only on a complete

✉ 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

product specification. It is not evident that current systems modeling languages and tools are well prepared for this kind of progressive use of agility across model abstractions over the system development process.

Furthermore, mapping models from one tool into another and extending the model with additional information along the toolchain leads to a "one-shot" mapping that is similar to the traditional waterfall model. This is completely the opposite of agile, where iterations are important and re-doing work, such as model extension, must be avoided. In systems modeling, a reusable set of models should be free of redundancy and complement each other. It also seems that most systems modeling toolchains only work in one direction. That means feedback from an advanced analysis or simulation technique in one tool to other tools across the toolchain is either manual or is not possible, posing another important obstacle for agile modeling of traditional engineering systems.

From these observations, traditional systems development has not reached the maturity of agile software development. The challenges of integrating a sequential modeling toolchain provide little opportunity for feedback to earlier models. Instead, a "cocktail" of complementary tools is needed to focus on the domain-specific aspects of the product, control redundancy, and allow almost permanent and immediate feedback through all forms of consistency. Other high-level analyses and automated simulations are also desirable, such that systems engineers are freed from repeated and tedious tasks so that they can concentrate fully on inventions and designs of their systems.

Software and systems engineers need to collaborate closely on these topics to better understand each other's needs and strengths. We hope that future SoSyM authors can make significant contributions to these new forms of agile, model-based systems development.

1 Content of this issue

This volume contains the following 12 papers:

- "Encoding process discovery problems in SMT" by Marc Solé and Josep Carmona
- "Reusing metamodels and notation with Diagram Definition" by Conrad Bock and Maged Elaasar
- "On submodels and submetamodels with their relation: a uniform formalization through inclusion properties" by Bernard Carré, Gilles Vanwormhoudt, and Olivier Caron
- "VMTL: a language for end-user model transformation" by Vlad Acretoaie, Harald Störrle, and Daniel Strüber
- "A model-driven development approach for context-aware systems" by Imen Jaouadi, Raoudha Ben Djemaa, and Hanene Ben-Abdallah
- "Formalised EMFTVM bytecode language for sound verification of model transformations" by Zheng Cheng, Rosemary Monahan, and James Power
- "Scope in model transformations" by Maris Jukss, Clark Verbrugge, Maged Elaasar, and Hans Vangheluwe
- "Holistic security requirements analysis for socio-technical systems" by Tong Li, Jennifer Horkoff, and Jon Mylopoulos
- "An approach to clone detection in sequence diagrams and its application to security analysis" by Manar Alalfi, Elizabeth Antony, and Jim Cordy
- "On the automated translational execution of the action language for foundational UML" by Federico Ciccozzi
- "Efficient parallel reasoning on fuzzy goal models for run time requirements verification" by George Chatzikonstantinou and Kostas Kontogiannis
- "The Train Benchmark: cross-technology performance evaluation of continuous model queries" by Gabor Szarnyas, Benedek Izso, Istvan Rath, and Daniel Varro.