Software & Systems Modeling (2019) 18:191–193
https://doi.org/10.1007/s10270-018-0686-1

GUEST EDITORIAL

Introduction to the STAF 2015 special section

Jasmin Blanchette[1] · Francis Bordeleau[2] · Alfonso Pierantonio[3] · Nikolai Kosmatov[4] · Gabriele Taentzer[5] · Manuel Wimmer[6]

Received: 7 June 2018 / Accepted: 10 June 2018 / Published online: 7 July 2018

Software Technologies: Applications and Foundations (STAF) is a federation of a number of leading conferences on software technologies. It provides a loose umbrella organization for practical software technologies conferences. In 2015 the STAF federated event has been held in L'Aquila (Italy) with a special focus on practical and foundational aspects of software technology, from object-oriented design, testing, mathematical approaches to modeling and verification, model transformation, graph transformation, model-driven engineering, aspect-oriented development, and tools. Besides the main and satellite events (14 overall), three conferences:

- The International Conference on Model Transformation (ICMT 2015)
- The European Conference on Modelling Foundations and Applications (ECMFA 2015)
- The International Conference on Tests & Proofs (TAP 2015)

✉ Alfonso Pierantonio
alfonso.pierantonio@univaq.it

Jasmin Blanchette
j.c.blanchette@vu.nl

Francis Bordeleau
francis.bordeleau@cmind.io

Nikolai Kosmatov
nikolai.kosmatov@cea.fr

Gabriele Taentzer
taentzer@informatik.uni-marburg.de

Manuel Wimmer
wimmer@big.tuwien.ac.at

1 Vrije Universiteit Amsterdam, Amsterdam, The Netherlands

2 Cmind, Ottawa, Canada

3 Università degli Studi dell'Aquila, L'Aquila, Italy

4 CEA LIST, Gif-sur-Yvette, France

5 Philipps-Universität Marburg, Marburg, Germany

6 Techniche Universität Wien, Wien, Austria

collected papers on relevant topics of software and system modeling and verification techniques; among them, 10 papers have been selected and extended to be part of this special section. In particular:

The International Conference on Model Transformation (ICMT) is the premier forum for researchers and practitioners from all areas of model transformation. Model transformation encompasses a variety of technical spaces, including modelware, grammarware, dataware, and ontoware, a variety of model representations, e.g., based on different types of graphs, and a variety of transformation paradigms including rule-based transformations, term rewriting, and manipulations of objects in general-purpose programming languages, to mention just a few. The study of model transformation includes foundations, structuring mechanisms, and properties, such as modularity, composability, and parameterization of transformations, transformation languages, techniques, and tools. The extended versions of five ICMT 2015 papers are published in this special section.

The paper *Toward live domain-specific languages—From text differencing to adapting models at run time* by Riemer van Rozen and Tijs van der Storm proposes a new paradigm to interact with executable models. In particular, they adapt ideas from live programming which allows to modify a running program by changing its source code and receiving immediate execution feedback without recompiling and restarting the program. In this respect, van Rozen and van der Storm propose a novel model differencing algorithm. The differencing algorithm produces model deltas which are applicable to adapt running model executions without requiring a restart of the execution. The authors illustrate their approach in a live programming environment implemented in Rascal for simultaneously defining and executing state machines.

The second contribution *F-Alloy: a relational model transformation language based on Alloy* by Loïc Gammaitoni and Pierre Kelsen proposes a sublanguage of Alloy called F-Alloy specifically designed to implement model transformations. Two main design principles have been fol-

lowed for the development of F-Alloy. The syntax of F-Alloy is built in a way to allow the efficient execution of the specified transformations. The semantics of F-Alloy is given as a direct translation to Alloy. Thus, all automatic analysis features of Alloy are directly applicable for model transformations implemented in F-Alloy.

The third contribution *Change propagation and bidirectionality in internal transformation DSLs* by Georg Hinkel and Erik Burger deals with the realization of model transformation languages in host languages which allow for the so-called internal DSL extensions. While in the past, such internal transformation languages have been already defined, they typically lack change propagation or bidirectional transformation support. To tackle this shortcoming, Hinkel and Burger introduce a novel formalism describing incremental, bidirectional model synchronizations using synchronization blocks. They implemented this formalism within a C#-based transformation language which allows for unidirectional as well as for bidirectional execution capabilities. In their validation they show a potential speedup of up to multiple orders of magnitude by using bidirectional executions compared to classical batch transformation executions.

The fourth contribution *A systematic approach to constructing families of incremental topology control algorithms using graph transformation* by Roland Kluge, Michael Stein, Gergely Varró, Andy Schürr, Matthias Hollick, and Max Mühlhäuser is concerned with the application of model transformation in the communication system domain, in particular, for constructing and maintaining network topologies via topology control algorithms. Kluge et al. generalize their previously published constructive approach for topology control algorithms to support the specification of families of such algorithms. Feasibility is demonstrated by reengineering six existing topology control algorithms and by developing a novel energy-efficient variant of an already existing algorithm. Also the automated evaluation of topology control algorithms is provided by integrating the graph transformation tool eMoflon and the Simonstrator network simulator.

Finally, the paper *Translation of ATL to AGT and application to a code generator for Simulink* by Elie Richa, Etienne Borde, and Laurent Pautet is bridging the gap between two well-known but independently developed model transformation formalisms to exploit the benefits of both worlds. On the one hand, ATL is a powerful model transformation language, but lacks formal analyses. On the other hand, algebraic graph transformation (AGT) is an approach with strong theoretical foundations allowing for formal analyses. Richa et al. propose a translation of ATL to AGT in order to provide the theoretical analyses capabilities of AGT, e.g., supported by the Henshin framework, also for ATL transformations. The resulting higher-order transformation covers a large subset of ATL which is demonstrated with an industrial case study about a Simulink-based source code generator.

The ECMFA conference series is dedicated to advancing the state of knowledge and fostering the industrial application of model-based software engineering and related approaches. Its focus is on engaging the key figures of research and industry in a dialog which will result in stronger and more effective practical application of model-based software engineering, hence producing more reliable software based on state-of-the-art research results. The extended versions of three ECMFA 2015 papers are published in this special section.

The paper *Type inference in flexible model-driven engineering using classification algorithms* by Athanasios Zolotas, Nicholas Matragkas, Sam Devlin, Dimitris Kolovos and Richard Paige contributes to flexible or bottom-up model-driven engineering (MDE). Domain experts, who have detailed domain knowledge, typically lack the technical expertise to transfer this knowledge using traditional MDE tools. In such approaches, type information is not given upfront, but has to be inferred from example models. This paper presents classification algorithms to help with the inference of untyped model elements.

The second ECMFA-related paper in this issue is concerned with *Incremental execution of model-to-text transformations using property access traces* written by Babajide Ogunyomi, Louis Rose, and Dimitris Kolovos. Languages for model-to-text transformations as they are used for automatic code and document generation lack support for developing transformations that scale with the size of the input model. This lack of scalability hinders industrial adoption. This paper presents a runtime analysis called property access traces to perform efficient source incremental transformations. Experiments show an average reduction of 60% in transformation execution time compared to non-incremental (batch) transformations.

The third contribution to the ECMFA-related issue part is the paper *Advanced and efficient execution trace management for executable domain-specific modeling languages* by Erwan Bousse, Tanja Mayerhofer, Benoit Combemale, and Benoit Baudry. Executable Domain-Specific Modeling Languages (xDSMLs) enable the application of early dynamic verification and validation techniques for behavioral models. At the core of such techniques, execution traces are used to represent the evolution of models during their execution. In order to construct execution traces for any xDSML, generic trace metamodels can be used. Since this generic approach lacks efficiency and usability, the authors developed a generative approach that defines a multidimensional and domain-specific trace metamodel specifically for a given xDSML. Evaluations show a significant performance improvement and simplification of the semantic differencing rules compared to the generic approach.

The International Conference on Tests and Proofs (TAP) is devoted to the synergy of proofs and tests, and to the application of techniques from both sides and their combination for the advancement of software quality. The TAP conference series aims to bring together researchers and practitioners working in the converging fields of testing and proving by offering a generous forum for the presentation of ongoing research, for tutorials on established technologies and for informal discussions. The extended versions of two TAP 2015 papers are published in the special section.

The paper *Experimental evaluation of a novel equivalence class partition testing strategy* by Felix Hübner, Wen-ling Huang, and Jan Peleska presents an experimental evaluation of a model-based equivalence class testing strategy. The authors investigate the question how this strategy performs for systems whose behaviors lie outside the fault domain and present an extension of the strategy , based on randomized data selection from input equivalence classes. Its evaluation relies on mutation coverage achieved by the original and the extended strategy with the coverage obtained by random testing.

The second extended version is the article *Fast test suite-driven model-based fault localisation with application to pinpointing defects in student programs* by Geoff Birch, Bernd Fischer, and Michael Poppleton. It describes a model-based fault localization algorithm based on symbolic execution. The proposed technique iterates over failing test cases and collects locations where an exhibited faulty behavior can be repaired by an assignment change. This technique is shown to reduce the effort for the symbolic execution of the models, leading to significant speedups.