Composing Model-Based Analysis Tools

Robert Heinrich • Francisco Durán •
Carolyn Talcott • Steffen Zschaler
Editors

Composing Model-Based Analysis Tools

Springer

Editors
Robert Heinrich
Karlsruhe Institute of Technology
Karlsruhe, Germany

Francisco Durán
University of Málaga
Málaga, Spain

Carolyn Talcott
SRI International
Menlo Park, CA, USA

Steffen Zschaler
King's College London
London, UK

ISBN 978-3-030-81914-9 ISBN 978-3-030-81915-6 (eBook)
https://doi.org/10.1007/978-3-030-81915-6

This Springer imprint is published by the registered company Springer Nature Switzerland AG.
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Foreword by Jeff Gray

The composition of software tools to support a task-specific process is a need that often arises in projects of all sizes. Engineers, scientists, and others who have expertise in a certain domain often depend on the integration of a tool pathway to complete work-related tasks. For example, a business analyst may export data from a company-specific dashboard to conduct some analysis using a specialized secondary tool, with visualization of the results then handled by a third tool that best serves the particular requirements of a project.

The need to compose software tools is not new. Over a quarter-century ago, the "software component" wars between standardization efforts like OpenDoc, with direct competition from Microsoft's Object Linking and Embedding (OLE)/Component Object Model (COM), rivalled that of the fervor of a religious debate. The need to compose and analyze information from different sources of origin has been a common need in computer-based solutions.

Engineers use models to abstract properties of a system, which can then be analyzed by different tools for various purposes. For example, avionics engineers may have a model that one division of a company uses for reliability analysis, and a separate model and supporting tool used by colleagues on another team for failure modes effect analysis. These two separate models and tools represent the same targeted system, but are created and maintained by different groups, for different objectives. In such a common scenario, each tool is a highly specialized package that contributes to a critical step in the engineering process.

Although tool vendors offer what they advertise as all-encompassing tool suites, many of the tools used in practice are very rarely integrated across the engineering process. Often, the tools were not designed with composition in mind, resulting in isolated stovepipes. In such cases, engineers must force integration in a human intensive and error-prone manner. Solving the model composition problem across diverse tools is not easy and requires consideration of both *syntax* (e.g., file formats, APIs) and *semantics* (i.e., what is the meaning of a concept in each individual tool and how do concepts map across tools?). The different abstractions used in separate models may also lead to topics of uncertainty and ambiguity, such as when one model captures more detail than another model, leading to a lossy situation during a round-trip translation between models. Simple composition solutions are insufficient in the case of complex engineering tools and processes. The composition solution must also account for the evolution of the system, as tools and processes change over time. Scalability is also a concern. Adding a new tool to the analysis tools ecosystem should not break the process or require exponential effort.

The editors and authors of this book recognize that the same challenges exist for software and systems engineers who use different models and (often independent) tools to analyze desired system properties (e.g., functional correctness, performance, and reliability). They wrote, "The composition of systems, their models and analyses is a challenging but unavoidable issue for today's complex systems" (Chap. 1). At the "Composing Model-Based Analysis Tools" Dagstuhl seminar in November 2019, they assembled an impressive cohort of experts in both software engineering and formal methods to discuss the challenges and potential solutions for analysis tool composition. The various chapter authors cover a range of foundational topics (e.g., modeling language composition, tool integration, uncertainty, and ambiguity) from the perspectives of formal methods and software engineering, with representation from both industry and academia. Readers are also offered a series of case studies that concretize the most important challenges and issues faced in applying model-based composition to analysis tools in different domains and contexts.

This book is recommended to anyone who is involved in the important decisions that emerge when composing multiple analysis tools during software and systems modeling. The book is suitable for both practitioners and researchers, and may also serve as a textbook for a graduate course on model-based analysis tools.

Tuscaloosa, AL, USA Jeff Gray
May 2021

Foreword by Antonio Vallecillo

It is essential to have good tools, but it is also essential that the tools should be used in the right way

Wallace D. Wattles

Conceptual modeling aims at raising the level of abstraction at which systems are described to cope with their increasing complexity. To this end, precise languages are used to represent the elements of the system that are relevant to the purposes of the modelers, abstracting away those that are not. These high-level representations of a system are known as software models, and their role in software engineering has been gaining relevance as soon as they were considered, stored, and managed like any other software artifacts.

Based on these principles, model-based engineering (MBE) is the software engineering discipline that advocates the use of these software models as primary artifacts for the software engineering process. In addition to the initial goals of being useful to capture user requirements and architectural concerns, and to generate code from them, software models are proving to be effective for many other engineering tasks. Model-based engineering approaches, such as model-driven modernization, models-at-runtime, or model-based testing, already provide useful concepts, mechanisms, and tools for the engineering of complex systems at the right level of abstraction. Software engineers have also realized the extensive possibilities that models offer when treated as actual software artifacts, and how they enable, for example, the development of powerful software engineering tools.

It was more than a decade ago, when MBE was starting to gain acceptance as a software engineering discipline, that Jean Bézivin contacted me because he was happy to see the remarkable developments, artifacts, and tools produced by the modeling community, but worried about the lack of interoperability between them. At that moment in time I was working on tools for the RM-ODP framework, and I was facing similar problems for integrating the separate analysis tools available for each viewpoint language. As a visionary, Jean thought that such interoperability was key to the successful development and adoption of MBE, and that models were again the essential elements to achieve it. Therefore, he coined the term "model-driven interoperability" to refer to this approach.

We were fully aware that interoperability implies much more than simply defining a common serialization format, for example, XMI. This would just resolve the syntactic (or "plumbing") issues between models and modeling tools. Interoperability should also involve further aspects, including the integration of different behavioral specifications, reaching agreements on names and concepts (ontologies), overcoming the differences between separate models of operation (e.g., discrete vs. continuous), or handling other semantic issues such as inconsistency management or exception handling. Furthermore, interoperability not only means being able to exchange information, but also to exchange services and functions to effectively operate together.

We soon realized that the best way to proceed was to set up a forum for the modeling community to discuss all these issues, because it was not a one-person effort (or even two). So, we contacted Richard Mark Soley from OMG, who at the time was also concerned about the same issues, and the three of us decided to organize a workshop at the MODELS 2010 conference in Oslo, on "Model-Driven Interoperability",[1] where the community could meet to exchange ideas and problems about these topics. The workshop was a great success, and more than 30 people participated by presenting their contributions, proposing problems and challenges, and exploring possible solutions. We all discovered there that the subject entailed more complexity than expected, that the relevance to industry was higher than anticipated, but at the same time that successful solutions could be possible if MBE concepts and techniques were used.

Unfortunately, and despite the interest raised by the first edition, the workshop was not continued and no other dedicated forum took its place to allow the software engineering community discuss about making tools interoperate using MBE technologies and artifacts. This is why I was so glad to learn about the Dagstuhl seminar on "Composing Model-based Analysis Tools" and, even more, about this book!

I believe that this initiative fills an existing gap in current research on this fundamental topic by compiling the main concepts and issues related to the composition of model-based analysis tools and, more importantly, by describing a set of concrete case studies that illustrate successful implementations of the book's central ideas. I am sure that the software engineering community, both researchers and practitioners, will truly appreciate the efforts made by the editors and authors to put together such a useful and valuable compilation of concepts, results, and case studies into a coherent body of work.

Finally, I am very grateful to the editors for inviting me to write the foreword to this book, especially when the authors are all the world's best-known experts on model-based concepts and tools. This is undoubtedly the best book that could have been written on this topic, and I look forward to the next Dagstuhl seminar,

[1] Bézivin J., Soley R.M., Vallecillo A. (2011) Model-Driven Interoperability: MDI 2010. In: Dingel J., Solberg A. (eds) Models in Software Engineering, MODELS 2010. Lecture Notes in Computer Science, vol 6627. Springer. https://doi.org/10.1007/978-3-642-21210-9_14.

which I hope will be held soon to further discuss about this fascinating subject and to produce the continuation of this excellent book!

Málaga, Spain Antonio Vallecillo
May 2021

Preface

Modelling and analysis are key to managing the increasing complexity and heterogeneity of today's software-intensive systems. Historically, different research communities have studied the modelling and analysis of different software quality properties (e.g., performance or security) for different types of systems. As a result, the tools available for designing and maintaining software that meets such properties are also distinct, using different languages and techniques, making interaction difficult. This leads to a significant amount of unnecessary development work when building modern applications that must meet combinations of these properties—for example, it may be necessary to construct redundant models in different formalisms and using different tools to support analyses for different quality properties.

We, the editors of this book, have been working on modelling and analysing software-intensive systems for a long time. In our work, we faced the need for more flexibility in model-driven engineering and for decomposing and composing models and analyses in several areas. Addressing this need provokes fundamental questions—for example, on validity, uncertainties, behaviour and property preservation, and termination of analyses. Traditionally, research on these topics has been conducted in different communities isolated from each other. This is why we organised the Dagstuhl seminar 19481 on "Composing Model-based Analysis Tools", held in 24–29 November 2019, at Schloss Dagstuhl, Leibniz Center for Informatics, Germany, to bring together researchers and industry experts from the software engineering and formal methods communities to leverage synergy effects and make progress towards establishing the foundations for a common understanding on composing model-based analysis tools.

This book is an outcome of this Dagstuhl seminar. As such, it presents current challenges, background on those challenges, and concepts to address those challenges in the broad area of the composition of model-based analysis tools, based on the discussions initiated during the seminar. The book also illustrates and underpins the challenges and concepts by discussing case studies.

We are grateful to the participants of Dagstuhl seminar 19481, who were kind enough to accept the challenge of participating in the seminar and later writing this book. We hope that the seminar and the book will make a small contribution towards bringing these communities together into this joint endeavour.

Karlsruhe, Germany	Robert Heinrich
Málaga, Spain	Francisco Durán
Menlo Park, CA, USA	Carolyn Talcott
London, UK	Steffen Zschaler
May 2021	

Contents

Contributors

Sofia Ananieva FZI Research Center for Information Technology, Karlsruhe, Germany

Kyungmin Bae POSTECH, Gyeongbuk, South Korea

Simona Bernardi University of Zaragoza, Zaragoza, Spain

Erwan Bousse University of Nantes, Nantes, France

Arvid Butting RWTH Aachen, Aachen, Germany

Benoit Combemale IRISA – University of Rennes, Rennes, France

Jesús Sánchez Cuadrado Universidad de Murcia, Murcia, Spain

Juan De Lara Autonomous University of Madrid, Madrid, Spain

Francisco Durán University of Málaga, Málaga, Spain

Michalis Famelis Université de Montréal, Montréal, QC, Canada

Anas Fattouh Mälardalen University, Västerås, Sweden

Martin Gogolla Universität Bremen, Bremen, Germany

Esther Guerra Autonomous University of Madrid, Madrid, Spain

Jörg Henß FZI Research Center for Information Technology, Karlsruhe, Germany

Robert Heinrich Karlsruhe Institute of Technology, Karlsruhe, Germany

Mark Hills East Carolina University, Greenville, NC, USA

Katrin Hölldobler RWTH Aachen, Aachen, Germany

Jean-Marc Jézéquel IRISA – University of Rennes, Rennes, France

Kenneth Johnson Auckland University of Technology, Auckland, New Zealand

Narges Khakpour Linnaeus University, Växjö, Sweden

Ehsan Khamespanah University of Tehran, Tehran, Iran

Sandro Koch Karlsruhe Institute of Technology, Karlsruhe, Germany

Raffaela Mirandola Polytechnic University of Milan, Milan, Italy

Pavle Mrvaljevic Mälardalen University, Västerås, Sweden

Diego Perez-Palacin Linnaeus University, Växjö, Sweden

Fiona A. C. Polack Keele University, Newcastle, UK

Daniel Ratiu CARIAD, Wolfsburg, Germany

Arend Rensink University of Twente, Enschede, Netherlands

Ralf Reussner Karlsruhe Institute of Technology, Karlsruhe, Germany

Elvinia Riccobene Università degli Studi di Milano, Milano, Italy

Bernhard Rumpe RWTH Aachen, Aachen, Germany

Houari Sahraoui Université de Montréal, Montréal, QC, Canada

Patrizia Scandurra University of Bergamo, Bergamo, Italy

Marjan Sirjani Mälardalen University, Västerås, Sweden

Carolyn Talcott SRI International, Menlo Park, CA, USA

Catia Trubiani Gran Sasso Science Institute, L'Aquila, Italy

Hans Vangheluwe University of Antwerp, Antwerp, Belgium

Dániel Varró McGill University, Montreal, QC, Canada

Arthur Vetter Dr. Ing. h.c. F. Porsche AG, Stuttgart, Germany

Andreas Wortmann Universität Stuttgart, Stuttgart, Germany

Marc Zeller Siemens AG, Munich, Germany

Steffen Zschaler King's College London, London, UK