Foreword

MODELS, MODELS EVERYWHERE

Models are pervasive in engineering and serve multiple useful purposes. In general, a model, being an abstract representation of some system that highlights aspects of interest while omitting those that are not, provides a comprehensible view of a system whose complexity might otherwise overwhelm our cognitive abilities. This makes it suitable for reasoning about such systems, helping us identify its key characteristics. Through extrapolations based on such models, it may also be possible to predict the interesting properties of the represented systems even if they are not explicitly rendered in the models. For instance, we may use a mathematical model of a bridge design to compute its load-bearing capacity. Equally important is the ability of models based on some shared representational conventions to facilitate communication between stakeholders.

In addition to these prescriptive purposes, models can also serve a prescriptive role, that is, they can be used as specifications that guide designers and implementers; i.e., “models as blueprints.” In the case of software systems, this is a particularly interesting aspect, because the model and the modeled system share the same medium: the computer. Since computers are eminently flexible automation devices, this creates a number of rich possibilities including the potential to gradually evolve a model through a succession of refinements into the system, which it was modeling. This has the potential to mitigate or even eliminate the troublesome transition that is often the curse of traditional engineering: the discontinuities that stand between the idealized world of engineering design and the messy, complex, and unrelenting world of physical implementation.

The possibility of automatically generating implementations from high-level specifications that have evolved from abstract models is certainly one of the primary features of what is sometimes called Model-Based (Software) Engineering (MBSE). However, as discussed above, this is just icing on the cake, since models have many uses. For instance, computer-based specification models have a number of ancillary uses, notably as a basis for automatically generating various artifacts required in design and development: analysis models (e.g., performance, timing, security), test cases, design documents, etc.

The chapters in this volume provide an excellent sampling of the rich possibilities of models in software engineering. They cover the full range of roles that models play in the design of software and systems in general. However, in addition to capturing much of the state-of-the-art in MBE, these contributions provide valuable insight into the future directions of MBE. Researchers, doctoral students, industrial architects, and tool developers will undoubtedly find inspiration in the chapters included here.

The use of models as specifications, either as blueprints to guide implementers or as source artifacts for automated code generation, is represented in this volume by a number of contributions. Several of them focus on automating critical functions as a means of increasing productivity and quality. Essaidi et
Al. (“Model-Driven Data Warehouse Automation: A Dependent-Concept Learning Approach”) describe how machine learning can be used to automatically derive model transforms for data warehouse applications. Davis and Chang (“Variant Logic for Model-Driven Applications”) describe how metamodeling can be exploited to support the rapid evolution of software systems in a way that bypasses programming, thereby allowing end users to perform the necessary modifications. A similar theme is represented in the chapter by Chassels (“Object Model Development/Engineering: Proven New Approach to Enterprise Software Building”), which argues for separating business logic from implementation technology to allow applications to evolve without the need for code generation.

The use of metamodels is also the subject of the contributions by David and Gonczy (“Ontology-Supported Design of Domain-Specific Languages: A Complex Event Processing Case Study”), which deals with automated generation of custom domain-specific languages from an ontology model, and Pereira et al. (“A Rigorous Approach for Metamodel Evolution”), which addresses the problem of modeling language evolution using formal model-based specifications. Peck and Alexander (“Rosetta Composition Semantics”) describe Rosetta, a model-based system specification language that supports compositional system design.

Garg (“Aspect-Oriented Sculpting”) explains how model-based specifications of systems can be constructed by exploiting methods used in aspect-oriented programming. Techniques for specifying systems for the emerging “Internet of things” using a generic model-based API are presented by C. Garcia and Espada in their contribution (“Using Model-Driven Architecture Principles to Generate Applications Based on Interconnecting Objects and Sensors”).

The use of models for communications is a principal theme for several papers in the volume. Yu (“Model-Driven Applications”) describes how models can be used effectively as mediators to bridge the gaps between humans, computers, and the environment in which software systems operate. Fischer et al. (“Viewpoint-Based Modeling: A Stakeholder-Centered Approach for Model-Driven Engineering”) explain how viewpoints on models can be used to facilitate communication between various stakeholders. Model-based methods for eliciting system requirements from stakeholders are also discussed by Gibson (“The Human Role in Model Synthesis”). This chapter also explains how models can help improve productivity, which is also the theme of the contribution by Dubielewicz et al. (“Quality Assurance in Agile Software Development”).

Models as tools for analysis (reasoning) and prediction are the topic of several of the contributions. Thompson et al. (“Analyzing Mobile Application Software Power Consumption via Model-Driven Engineering”) demonstrate how models and simulation can help in predicting so-called non-functional properties of systems, while G. Garcia et al. focus on performance analysis using models (“Parameterized Transformation Schema for Non-Functional Properties Model in the Context of MDE”). The use of model-based simulation is also the subject of the chapter by Sapna et al. (“Consistency Checking of Specifications in UML”), which describes the use of a semi-formal variant of UML to ensure consistency between a design model and corresponding test cases.

In summary, we can see from these contributions that the future of model-based engineering is a very promising one. It seems that we are at the threshold of a new generation of software engineering methods—methods that are, at last, bringing the design and development of software in line with more established engineering disciplines in terms of dependability and quality. In time, we can expect that the model-based approach will be accepted as standard practice, with models and model-based technologies and methods recognized as an integral and essential part of any systems development. We will know that
this has been achieved when the term “model-based” (or “model-driven,” if you prefer)—chosen initially to draw attention to models and related technologies—becomes redundant because it is self-evident. The results presented in this book are solid steps that take us firmly in this direction.

Bran Selic
Malina Software Corp., Canada & Simula Research Laboratory, Norway & University of Toronto, Canada

Bran Selic is President and Founder of Malina Software Corp., a Canadian company that provides consulting services to clients worldwide. He is also a part-time Research Scientist at Simula Research Laboratory, in Oslo, Norway, as well as an adjunct at the University of Toronto. Bran has over 35 years of practical industrial experience in designing and implementing large-scale software systems and has pioneered the application of model-based methods in real-time and embedded applications. In the past, he has led several international standards efforts related to modeling technologies, including the widely used UML 2 modeling language. A frequently invited and keynote speaker at various technical events, he is on the editorial board of several mainline scientific journals and has been the general and technical program chair of a number of technical conferences. He has lectured widely on the topic of model-based engineering in various universities and research institutions around the world.