Preface

The success of the Unified Modeling Language (UML) reflects a growing consensus in the software industry that modeling is a key ingredient in the development of large and ultra-large software systems. Recent developments to industrialize the software development process are also using technologies such as components, model-driven architectures (MDA), and product lines. These technologies drastically alter the software development process, which is characterized by a high degree of innovation and productivity. MDA and model-driven software development (MDSD) focus on the idea of constructing software systems not by programming in a specific programming language but by designing models that are translated into executable software systems by generators. These characteristics enable designers to deliver product releases within much shorter periods of time compared to the traditional methods. In theory, this process makes it unnecessary to worry about for an executable system's quality, as it is "optimized" by the generators.

However, proponents of MDSD must provide convincing answers to questions such as "What is the quality of the models and software produced?" The designed models are a work product that requires a minimal set of quality aspects (e.g., the maintainability of models over a longer life-time). Furthermore, models created in the earlier phases of development (e.g. analysis and design) are often only used as an abstract template for the software and typically are of little value, unless they can be readily mapped to correct and efficient executable forms, which means high-level object-oriented programming languages. Any problem in the transformation path from requirements via models to code not only has a negative impact on the quality of the delivered software system, but also obstructs its future maintenance and/or reuse.

Quality assurance techniques such as testing, inspections, software analysis, model checking, or software measurement are well researched for programming languages, but their application in the domain of software models and model-driven software development is still in an embryonic phase. In general, quality assurance is related to all phases of the software lifecycle, is needed within all application domains, and comes in many different flavors, ranging from reviews and inspections via metrics and quality models to holistic approaches for the quality-driven development of software systems. The goals of quality assurance for model-driven software development are diverse and include the improvement of quality aspects such as maintainability, reusability, security, or performance. Quality assurance for model-driven software development will play an important role for the future wide-spread usage of model-driven architectures in general, as well as in specific application domains.

In order to foster the development of quality assurance research in MDSD and to give a solid overview of the field, we have brought together research and practice in this book.

MODEL-DRIVEN SOFTWARE DEVELOPMENT

Model-driven software development methods aim at supporting software engineers in producing large and ultra-large software systems that are very flexible, portable, and of high value to their customers. Basically, programmers are freed from the burden of tedious standard tasks, which are also a source of errors. It is envisioned that by systematically applying MDSD, the quality of software systems, the degree of reuse and thus, implicitly, the development efficiency will improve.

The core idea of MDSD is that models are becoming the "source code" of a system from which the executables are simply generated. Thus, models cover different abstraction layers, ranging from conceptual diagrams in the problem space to detailed low-level models adapted to a specific platform. In general, model-driven software development is the process of generating executable software systems from formal models, starting with computational independent models (CIMs) that are extended to platform independent models (PIMs) to be adapted into platform specific models (PSMs) and finally result in source code (e.g., Java). In other words, models now bridge the traditional gap between, human-readable requirements and source code. Contemporary approaches of MDSD also create platform specific skeletons (PSS), which have to be completed by programmers.

Typically, models have had a long tradition in software engineering and are used in many software projects. However, there is not one commonly used language for models used in software development. Software models may range from sketches on a whiteboard via UML diagrams to mathematical specifications.

In order to enable the automatic generation of executable models, these models have to follow a precisely defined syntax and semantic. One widespread language for depicting such models is the Unified Modeling Language (UML), but model-driven development is not necessarily bound to the UML. Other modeling languages (e.g., Petri-nets, MathLab, Modelica, etc.) are successfully applied and have their value. And while the UML provides a large selection of diagram types and a more formal specification language (e.g., OCL), the information contained within a model is often not sufficiently concise and precise (i.e., UML models often have to be enriched by textual specifications).

Nevertheless, by using the full power of the UML diagrams on different abstraction layers with accompanying textual specifications, we can model complete systems today. However, as with other work products such as source code, this opens the question of how to assure quality within model-driven development. Quality assurance in MDSD has to address quality issues at different abstraction levels and has to face the challenge of the very richness (and complexity) of the UML. Today, quality assurance is often used as a synonym for testing, but in reality it is a much wider discipline – it includes other techniques such as inspections or metrics. Even though model-based testing is a well-known way to use models for quality assurance, modeling has much more potential in this regard: Models can be verified before code is generated, requirements can be modeled and checked against design models, etc.

OVERALL OBJECTIVE OF THE BOOK

This book aims at providing an in-depth coverage of important concepts, issues, trends, methodologies, and technologies in quality assurance for MDSD. It focuses on non-testing approaches for quality assurance and discusses quality in the context of MDSD. This premier reference source presents original academic work and experience reports from industry that can be used for developing and implementing high-quality model-based software.

It is a comprehensive guide that helps researchers and practitioners in the model-driven software development area to avoid risks and project failures that are frequently encountered in traditional and agile software projects. The whole development process and the developed products (i.e., CIMs, PIMs, PSM, etc.) must be analyzed, measured, and validated from the quality point of view.

TARGET AUDIENCE

The topic of integrating quality assurance into model-driven software development is broad and comes in many different flavors. However, when applying model-driven development and quality assurance, the basic principles and concepts have to be known to all participants of such a project. This book provides a comprehensive overview to those who are interested in studying the field of quality assurance for model-driven software development. However, this book is not meant to be a textbook that supports lectures or self-studies for novices. This book is aimed at researchers, project managers, and developers who are interested in promoting quality assurance for model-driven software development, in further educating themselves, and in getting insights into the latest achievements.

VOLUME OVERVIEW

The following chapters provide significant details about the topics outlined in this introduction. All chapters describe innovative research and, where possible, experience collected in industrial settings. Therefore, this book provides significant contributions to both the research and practice of assuring quality in model-based development. Several case studies are presented as a means for illustrating approaches, methods, and techniques in order to provide evidence. Most authors use or refer to the Unified Modeling Language (UML) in their chapters as a means for modeling the problem or solution domain. However, it appears that the latest version of the UML (version 2.1.1) is not used consistently. Thus, it is important to note that all references to the UML cover versions 1.x to 2.x. In summary, the book is organized as follows:

- Section I gives an introduction to quality in model-driven software development, which is presented in four chapter (Chapters I to IV). The chapters cover quality in general, quality aspects, and quality models for quality assurance in model-driven software development.
- Section II presents three chapters (Chapters V to VII) that are concerned with the evaluation of software models. They cover techniques for obtaining objective information from models that support the measurement, evaluation, and assessment of the model's quality.
- Section III covers the improvement of a model's quality. Six chapters (Chapters VIII to XIII) present different techniques such as refactoring, inspections, and constraint checking that help to improve the quality of a model. The chapters address approaches from the viewpoint of quality criteria and describe how model-driven development might become quality-driven model-based development.
- Section IV presents four chapters (Chapters XIV to XVIII) on using quality assurance techniques for model-driven development in specific application domains. Most papers are devoted to the domain of embedded systems (esp. in the automotive domain) and report about experience collected in specific industrial environments.

In summary, this book provides an overview of state of the art approaches to quality assurance in model-driven software development and presents the main challenges surrounding the subject. Each of the following chapters presents a set of issues and problems commonly encountered when performing research on or applying model-driven development. All authors share their vision about the importance of quality issues and agree that quality has a strong impact on system development and deployment. We hope that the insights and experiences described in this book will provide readers with new research directions and valuable guidelines.

Jörg Rech and Christian Bunse
Editors