Preface

The success of software modeling languages and especially that of the Unified Modeling Language (UML) demonstrates a growing consensus in software industry that the general principles of abstraction, divide-and-conquer, and stepwise-refinement, enabled by modeling, are key in the development of large and complex software systems. This is also indicated by the recent advent of technologies such as component-based software systems, software product-lines, et cetera.

However, using technologies such as modeling, components, et cetera drastically alters the software development process and process organization. Today a software model tends to exist for a long time and will not be discarded when shipping the system. Thus, the maintenance, evolution, re-engineering, as well as reuse of software models gains importance for the practical application of model-based development. Furthermore, as technologies and tools mature, we face the challenge of shifting from short model-based sketches to complex models with a long lifespan that will likely appear in several versions of a system or even across multiple systems.

In the long-term, regarding those fine-grained software models, we have to address emerging questions regarding their maintenance, evolution, and re-engineering. This includes partitioning and maintaining of models, as well as managing their change. Systematic methods need to support the quality of software models, ways to define model libraries, techniques for impact analysis, and integration for varying domains.

EVOLUTION AND MAINTENANCE OF SOFTWARE MODELS

In general, the term “evolution” describes the process of change of groups or elements over generations and puts a specific focus on mutations, drift, and “natural selection.” Following the ideas of Richta and Bloomfield, technology, as all elements, evolves. When used in the domain of software, and more specifically in software maintenance, evolution refers to the process of initially developing a software system and evolving/changing by various generations and reasons. Any (software) system with a significant complexity has to face changing conditions during its lifetime. Thus, modern software systems have to become flexible to “embrace” changes by either static or dynamic adaptation. Software evolution addresses this by putting the focus on adaptation and migration.

While substantial research from both industry and academia has been devoted to software evolution and maintenance, there has been only a limited number projects that studied software evolution in model-driven software development. Getting a better understanding of these concepts is important especially when putting the focus on evolution-in-the-large. The evolution of large-scale software systems is of increasing importance and often has an impact onto many technology areas. However, following Favre, regardless the technology used, recognizing the concepts of model and meta-models is important in this context.
OBJECTIVE OF THE BOOK

In this book, we have collected and connected the latest views from research and practice, in order to stimulate and support the development of research in the evolution, maintenance, and re-engineering of software models. In general, modeling is a means for mastering complexity by abstraction. A scientific model can provide a way to read elements easily which have been broken down to a simpler form. Therefore science uses a significant number of methods, techniques, and theory about all kinds of specialized scientific modeling. Within software development, specific modeling languages and diagram types are used throughout all development phases to express information/knowledge or systems in a structure that is defined by a consistent set of rules. Within this book the use of models, mainly based on UML, are used to support the maintenance and evolution of software systems. This covers meta-models to define or specify language extensions and semantics (e.g., of the UML), models that describe how systems or -parts (i.e., components) can be tested, or even test themselves when reused, and models that demonstrate the structure and semantics of systems that were designed for evolution.

In summary, this book provides an in-depth coverage of important issues, concepts, trends, tools, methodologies, and technology in the field of model evolution and maintenance and is targeted to become a premier reference resource that presents original academic work as well as experience reports from industry. The book therefore supports the installment of successful project organization structures that are based on model-driven development and reuse across projects. It therefore helps to broaden the view from single projects to project organizations. By understanding and applying the knowledge contained within this book, typical risks and failures - as found in many projects - can be avoided.

TARGET AUDIENCE

Model-driven development and reuse has already reached the mass adoption phase of technology transfer and is widely used throughout software industry. This not only has made the topic broad, but also created a constantly growing demand for technology support regarding the evolution and maintenance of software models. Unfortunately, model maintenance is not a simple question of tools and techniques. Its basic principles and concepts have to be known by all involved developers by heart.

This book provides a comprehensive overview to those who are interested in studying and learning in the field of model evolution and maintenance. However, the book is not meant to be a textbook that supports lectures or self-studies for the inexperienced. It is aimed at researchers, project managers, and developers with a sufficient background in model-driven development who want to learn about the current research trends and latest achievements.

In order to transfer as much knowledge as possible, we gave the authors the room to explain and document their research in a larger space and in more detail than workshops, conferences, and even journals allow.

ORGANIZATION OF THE BOOK / VOLUME OVERVIEW

The chapters of this book provide a comprehensive overview on the current state-of-the-art of model evolution and maintenance for the development of software. All chapters provide descriptions of innova-
tive research and, if existing, practical or empirical evidence. As such, the book supports both research and practice. In detail, the book is organized as follows:

- Section 1 gives an overview to the emerging technologies for the evolution and maintenance of software models, which is presented in six chapters. The approaches described include quality-driven development methods, the usage semantics and ontologies, or change impact analysis techniques in order to support MER activities.
- Section 2 presents two techniques for the re-engineering or modernization of software systems using a model-driven approach.
- Section 3 covers the use of testing techniques to improve the quality and development of software models as well as the model-based testing of software systems.
- Section 4 presents two interesting approaches about the deployment of MDA artifacts and the versioning of software models.

CONCLUSION / WHAT WAS LEFT OUT

A book, even of the size of the one you currently hold in your hands is not large enough to cover all aspects of model-driven software development, evolution, and maintenance. We tried to collect the current state of the art in the field of evolution and maintenance of software models, however, much has to be left out and could not be covered.

For more detail, readers may want to refer to additional material that is given by the references of single chapters. Furthermore, the editors ask all interested readers to ask for more information by contacting the authors and/or editors.

Jörg Rech
SAP Research Center Karlsruhe, Germany

Christian Bunse
University of Applied Sciences Stralsund, Germany