Foreword

Modern model-driven development in its current, widely recognized form was born 10 years ago when the UML 1.0 proposal was submitted to the OMG. As a unification of the leading object-oriented methods existent at that time, UML 1.0 set aside trivial notation debates and galvanized the software engineering community into exploring the true potential of modeling. In the early years, models were primarily seen as an aid to analysis and design activities. By providing compact and easy-to-understand depictions of system properties, models were found to be useful for communicating ideas to customers and fellow developers, and for exploring design ideas. However, they were not regarded as central artifacts in the development process, but rather as imprecise, auxiliary visualizations of the "true" product – the code. As a result, there was little point in worrying about the quality of models – it was the quality of the code that mattered.

With the advent of true model-driven development during the last few years, however, all this has changed. Not only has the UML gone through several revisions, giving it a much more precise and well-documented abstract syntax and semantics, but a new generation of tools and transformation languages have emerged, which largely automate the translation of models into code. UML models therefore have a much tighter and well-defined relationship to code and are no longer regarded as unimportant, supplemental artifacts. Indeed, the day is not far off when models will become the primary development artifacts and traditional source code will be regarded as supplemental. But as the role of models becomes more central and more important, so does their impact on the overall quality of the software product. This means that instead of being of marginal interest, in the years to come the quality of models will play an ever more central role for the success of software projects and products.

Assuring quality in model-driven development is much more challenging and multi-faceted than quality assurance at the code level, however. First, model-driven development involves a lot more views and diagram types than code-level representations of software, and keeping all these different views consistent and optimized is much more difficult than maintaining a single, textual view of a software product. Second, since model-driven development regards the definition of model transformations as a normal part of software engineering, these are primary development artifacts in their own right, and must be subject to the same defect detection and quality assurance activities as other human-defined documents. Third, since the abstraction gap between primary (i.e., human-developed) artifacts and execution platforms is much greater when models are used to describe software rather than source code, the relationships between model properties and product properties is much more tenuous and ill-defined. A whole new genre of quality metrics therefore needs to be defined and their value as quality indictors needs to be established and experimentally confirmed. And last but not least, there is the issue of the expressive power and representation fidelity of the modeling notations themselves. Although the UML was a significant step forward over previous notations, it is certainly not the last word in visual repre-

sentation languages, and a great deal of work still needs to be done to evaluate which notations best convey different types of information and are the least prone to errors.

Although the field is in its infancy, this book demonstrates that a lot of work has already been done and there is an active and vibrant research community studying the quality aspects of model-driven development. With the creation of this book, the editors and authors have compiled one of the most comprehensive and authoritative overviews of the state of the art in model-driven quality assurance available to date. The book therefore represents an important step in the evolution of model-driven development and helps turn it into a mature engineering discipline. There is currently no better or more extensive body of knowledge on quality assurance in model-driven development, and I hope you will be able to learn from the book as I have.

Colin Atkinson
University of Mannheim
May 2008