Model-Driven Software Engineering in Practice

Second Edition

Synthesis Lectures on Software Engineering

Model-Driven Software Engineering in Practice: Second Edition
Marco Brambilla, Jordi Cabot, and Manuel Wimmer

ISBN: 978-3-031-01421-5 paperback
ISBN: 978-3-031-02549-5 ebook
ISBN: 978-3-031-03677-4 epub

DOI 10.1007/978-3-031-02549-5

A Publication in the Springer series
SYNTHESIS LECTURES ON SOFTWARE ENGINEERING

Lecture #4
Series Editor: Luciano Baresi, *Politecnico di Milano*
Series ISSN
Print 2328-3319 Electronic 2328-3327

Model-Driven Software Engineering in Practice

Second Edition

Marco Brambilla
Politecnico di Milano, Italy

Jordi Cabot
ICREA and Open University of Catalonia (UOC), Spain

Manuel Wimmer
TU Wien, Austria

SYNTHESIS LECTURES ON SOFTWARE ENGINEERING #4

ABSTRACT

This book discusses how model-based approaches can improve the daily practice of software professionals. This is known as Model-Driven Software Engineering (MDSE) or, simply, Model-Driven Engineering (MDE).

MDSE practices have proved to increase efficiency and effectiveness in software development, as demonstrated by various quantitative and qualitative studies. MDSE adoption in the software industry is foreseen to grow exponentially in the near future, e.g., due to the convergence of software development and business analysis.

The aim of this book is to provide you with an agile and flexible tool to introduce you to the MDSE world, thus allowing you to quickly understand its basic principles and techniques and to choose the right set of MDSE instruments for your needs so that you can start to benefit from MDSE right away.

The book is organized into two main parts.

- The first part discusses the *foundations of MDSE* in terms of basic concepts (i.e., models and transformations), driving principles, application scenarios, and current standards, like the well-known MDA initiative proposed by OMG (Object Management Group) as well as the practices on how to integrate MDSE in existing development processes.
- The second part deals with the *technical aspects of MDSE*, spanning from the basics on when and how to build a domain-specific modeling language, to the description of Model-to-Text and Model-to-Model transformations, and the tools that support the management of MDSE projects.

The second edition of the book features:

- a set of completely new topics, including: full example of the creation of a new modeling language (IFML), discussion of modeling issues and approaches in specific domains, like business process modeling, user interaction modeling, and enterprise architecture
- complete revision of examples, figures, and text, for improving readability, understandability, and coherence
- better formulation of definitions, dependencies between concepts and ideas
- addition of a complete index of book content

KEYWORDS

modeling, software engineering, UML, domain-specific language, model-driven engineering, code generation, reverse engineering, model transformation, MDD, MDA, MDE, MDSE, OMG, DSL, EMF, Eclipse

Contents

Foreword

Technology takes forever to transition from academia to industry. At least it seems like forever. I had the honor to work with some of the original Multics operating system development team in the 1970s (some of them had been at it since the early 1960s). It seems almost comical to point out that Honeywell only ever sold a few dozen Multics mainframes, but they were advanced, really advanced—many of Multics' innovations (segmented memory, hardware security and privacy, multi-level security, etc.) took literally decades to find their way into other commercial products. I have a very distinct memory of looking at the original Intel 386 chip, impressed that the engineers had finally put Multics-style ring security right in the hardware, and less impressed when I discovered that they had done it exactly backward, with highly secure users unable to access low-security areas, but low-security users able to access the kernel. Technology transfer is a difficult and delicate task!

When I had the opportunity to help introduce a new technology and manage hype around that technology, I took it. At loose ends in 1989, I agreed to join the founding team of the Object Management Group (OMG), to help define commercial uptake for Object Technology (called object-oriented systems in the academic world, at least since Simula in 1967), and equally to help control the hype around the Object Technology marketplace. Having participated in the Artificial Intelligence (AI, or expert systems) world in the 1980s, I really didn't want to see another market meltdown as we'd experienced in AI: from the cover of *Time* magazine to a dead market in only five years!

That worked. OMG named, and helped define, the middleware marketplace that flourished in the 1990s, and continues today. Middleware ranges from: TCP socket-based, hand-defined protocols (generally an awful idea); to object-based, request-broker style stacks with automatically defined protocols from interface specifications (like OMG's own CORBA); to similarly automatically-defined, but publish-and-subscribe based protocols (like OMG's own DDS); to semantic integrate middleware with high-end built-in inference engines; to commercial everything-but-the-kitchen-sink "enterprise service bus" collections of request brokers, publish-and-subscribe, expert-system based, automatic-routing, voice-and-audio streaming lollapaloozas. Middleware abounds, and although there's still innovation, it's a very mature marketplace.

By the late 1990s, it was clear that the rapid rise of standardization focused on vertical markets (like healthcare IT, telecommunications, manufacturing, and financial services, OMG's initial range of so-called "domain" standards) would need something stronger than interface definition languages; to be more useful, standards in vertical markets (and arguably, all standards) should be defined using high-level, precise but abstract "modeling" languages. This class of languages should be much closer to user requirements, more readable by non-technical people, more focused on capturing process and semantics; in general, they should be more expressive. The natural choice for OMG and its members was of course OMG's own Unified Modeling Language (UML), standardized in 1997 as an experimental use of OMG's standards process that had heretofore focused on middleware. Even better, the UML standardization effort had produced a little-known but critical modeling language called the Meta-Object Facility (MOF) for defining modeling languages. This core of MOF plus extensible, profileable UML would easily be the foundation for a revolution in software development—and beyond.

As the millennium approached, OMG's senior staff met to consider how we could nudge the OMG membership in this valuable new direction. We came up with a name (Model-Driven Architecture, or MDA); a picture (which hasn't been universally adopted, but still helped make the transition); and a well-received white paper that explained why MDA would be the next logical step in the evolution of software engineering (and by extension, how it matches modeling in other engineering disciplines, though generally with other names, like "blueprints.") OMG's senior staff then spent a few months pitching this idea to our leading members, to a very mixed review. Some had been thinking this way for years and welcomed the approach; while some thought that it would be seen as an abandonment of our traditional middleware space (which by the way, we have never abandoned; the latest OMG middleware standards are weeks old at this writing and many more are to come). The CEO of one of our key member companies found the concept laughable, and in a memorable phrase, asked "Where's the sparkle?"

I truly believe, however, that organizations which resist change are the least stable. OMG therefore carried on and in 2001 introduced Model-Driven Architecture to a waiting world with a series of one-day events around the world. David Frankel's eponymous book, written and edited as he and I flew around the world to introduce the MDA concept, came out shortly thereafter; key influencers joined us in the campaign to add major new OMG standardization efforts in the modeling space. We would continue to create, extend, and support existing standards and new standards in the middleware and vertical-market spaces, but we would add significant new activities. It occurred to me that we actually had already been in the modeling space from the beginning; one can think of the Interface Definition Language of CORBA and DDS as simply a poor-man's modeling language, with limited expression of semantics.

For a while, the "sparkle" our members looked for was in academia primarily. As an avid participant in several academic conferences a year, I can tell you that uptake of MDA concepts (and terminology, like "platform-specific model" and "platform-independent model") took off like a rocket in universities. It took time, but the next step was a technology "pull" from engineering organizations that needed to perform better than the past (many of whom had already been using MDA techniques, and now had a name to point to); the creation of the Eclipse Foundation, starting in 2002, and its early embrace of modeling technology, also helped greatly. By 2010, modeling was firmly embedded in the world's software engineering psyche, and Gartner and Forrester were reporting that more than 71 UML tools were available on the market and adopted at some level. That's some serious "sparkle," and OMG members reveled in the success.

An interesting parallel world began to appear around MOF and UML, recognizing that modeling languages didn't have to be limited to modeling software systems (or "software intensive systems," as many called them); that, in fact, most complex software systems have to interact with other complex engineered systems, from building architecture to complex devices like mobile phones and aircraft carriers. We decided to roll out an entire fleet of MOF-defined languages to address the needs of many different modelers and marketplaces:

- UML System on a Chip: for microchip hardware/firmware/software definition;
- SoaML: for service-oriented architectures;
- BPMN: for business process modelers;
- BMM: for modeling the motivations and structure of a business;
- SysML: for modeling large, complex systems of software, hardware, facilities, people, and processes;
- UPDM: for modeling enterprise architectures;
- CWM: for data warehouses.

Each of these have been successful in a well-defined marketplace, often replacing a mix of many other languages and techniques that have fragmented a market and market opportunity. Along the way, our terminology morphed, changed, and extended, with arguments about the difference between "model-driven" and "model-based;" one of my favorite memories is of a keynote speech I gave just a couple of years ago in Oslo, after which an attendee came up to argue with me about my definition of the phrase "model-driven architecture." He wasn't particularly impressed that I had made up the term; it reminded me of a classic (and possibly apocryphal) story about the brilliant pianist Glenn Gould, who when accosted by a composer for his interpretation of the composer's work, yelled, "You don't understand your own composition!"

Over the past decade many new phrases have appeared around MDA, and one of the ones I consider most apt is Model-Driven Software Engineering (MDSE). This history lesson brings us to the work of this book, to help the neophyte understand and succeed with the technologies that make up MDSE. What are these mystical "modeling languages," how do we transform (compile) from one to another, and most importantly, how does this approach bring down the cost of development, maintenance, and integration of these systems? These aren't mysteries at all, and this book does a great job enlightening us on the techniques to get the most from a model-driven approach.

I'd like to leave you, dear reader, with one more thought. Recently, I had the opportunity to create, with dear friends Ivar Jacobson and Bertrand Meyer, an international community dedicated to better formalizing the software development process, and moving software development out of the fragmented "stone age" of insufficient theory chasing overwhelming need, to the ordered, structured engineering world on which other engineering disciplines depend. The Software Engineering Method and Theory (Semat) project brings together like-minded people worldwide to help bring software development into the 21st century, much as building architecture was driven into modernism by growing size and failures a millennium ago, and the shipbuilding industry had to formalize into ship blueprints some four centuries ago. My dream is that software engineering becomes engineering, and the huge stack of should-be-integrated engineering disciplines (civil, materials, software, hardware, etc.) be integrated into Model-Driven Engineering.

In your hands is part of the first step.

Richard Mark Soley, Ph.D.
Chairman and Chief Executive Officer
Object Management Group, Inc.
June 2012

10,000 meters over the central United States

Acknowledgments

This book wouldn't be the same without all the enriching discussions we have had with many other MDSE fans (and detractors!) during the last years—in person or within online forums. It would be almost impossible to list all of them here and therefore we wish to thank them all and to acknowledge their direct or indirect contribution to this book and to the MDE field at large, especially our current and previous colleagues.

An explicit mention must go to the ones who concretely helped us in the writing of this book. First of all, thanks to Diane Cerra, our Managing Editor at Morgan & Claypool, who believed in our project since the beginning and followed us with infinite patience throughout the whole book production process.

Secondly, thanks to Richard Soley, Chairman and CEO of OMG, who graciously agreed to introduce our work with his authoritative foreword.

And finally, last but not least, thanks to all the people that helped review the book: Ed Seidewitz (Model Driven Solutions), Davide di Ruscio (L'Aquila University), Juan Carlos Molina (Integranova), Vicente Pelechano (Polytechnic University of Valencia), and a bunch of our own colleagues and friends who carefully read and commented on what we were writing.

Marco Brambilla, Jordi Cabot, and Manuel Wimmer
January 2017