PREFACE

This book is for experienced developers who want to improve the productivity of their software development teams. Productivity is not just about speed, but also about the quality of the systems produced, both intrinsically and as perceived by the people who have to use them. Our focus is on the power and tools we can put in the hands of our software developers, and how to make the best use of the experience and skills already present in our teams.

We are not talking here about squeezing another 20% out of our existing developers, programming languages, or development environments. Our industry is long overdue for a major increase in productivity: the last such advance was over 30 years ago, when the move from assemblers to compilers raised productivity by around 500%. Our experiences, and those of our customers, colleagues, and competitors, have shown that a similar or even larger increase is now possible, through what we call Domain-Specific Modeling. Indeed, the early adopters of DSM have been enjoying productivity increases of 500–1000% in production for over 10 years now.

WHAT IS DOMAIN-SPECIFIC MODELING?

Domain-Specific Modeling requires an experienced developer to create three things, which together form the development environment for other developers in that domain. A domain here is generally a highly focused problem domain, typically worked on by 5–500 developers in the same organization. The three things are as follows:

- A domain-specific modeling language

- A domain-specific code generator
- A domain framework

With these three parts of a DSM solution in place, the developers need only create models in the DSM language, and the applications are automatically generated as code running on top of the domain framework. The generated code need not be edited or even looked at, thus completing the analogy with the move from assemblers to compilers: with each major leap of our industry, developers need no longer look at the previous generation's source format.

The changes wrought by Domain-Specific Modeling may seem radical, but at its heart are three simple practices that any experienced software engineer will recognize:

- Don't repeat yourself
- Automate after three occurrences
- Customized solutions fit better than generic ones

Other books have discussed these principles, the basic ideas of modeling, and how to move modeling to be more central to the development process. In this book, we will explain what Domain-Specific Modeling is, why it works, and how to create and use a DSM solution to improve the productivity of your software development.

BACKGROUND OF THE AUTHORS

Both authors have been working in DSM for over 15 years at the time of publication. In that time we have seen DSM successfully applied in a vast array of different problem domains, to create applications in a similarly broad collection of programming languages and platforms. Across the board, DSM has consistently increased productivity by a factor of 5–10. We take no personal credit for those results: the approach itself simply works.

Similar results have been achieved by our customers creating DSM solutions on their own, and by people using different tools. However, there have also been plenty of failures in those two situations: the approach and its tooling are not so self-evident that anybody can create them anew *in vacuo*. We too have made plenty of mistakes along the way and have learned from and with our customers. In particular, by teaching our customers and others, we have been forced to put our experience into words and try various ways of modularizing and presenting it. By writing this book, we hope to be able to pass on our experience in an easily digested form, to help you to have a smoother path to success.

An important part of smoothing the way to success in creating your own DSM solution is good tooling. It is possible to create your own modeling tool from scratch using graphics frameworks and so on, but for all but the largest groups of developers such an approach will be prohibitively expensive and time consuming. There are currently several DSM tools available that will allow you to simply

specify your modeling language, and which offer in return a modeling tool for that language.

The authors have played a central role in the development of one such DSM tool, MetaEdit+. The earlier MetaEdit was created as a research prototype in 1991, and released commercially in 1993, being the first tool to allow people to define their modeling languages graphically. As is common with such first versions, the original architecture was found to be too limiting for large-scale commercial use, lacking support for multiple simultaneous modelers, multiple modeling languages, and multiple integrated models. These and other requirements were met in MetaEdit+, released commercially in 1995. MetaEdit+ was created from a clean slate, but building on the experience gained with MetaEdit: a prime case of ''build one to throw away.''

This book, however, like DSM itself, is not limited to or focused on any particular tool. As far as possible, we have steered clear of tool-specific details. The principles presented here can be applied in any mature tool, and the benefits of DSM can be obtained—albeit at a higher cost—even with immature tools. That at least was our experience and that of our customers with the first version of MetaEdit+, which was definitely in that category in terms of its user interface!

HOW TO READ THIS BOOK

The book is divided into four main parts.

- Part I explains what DSM is (Chapter 1) and what value it has to offer (Chapter 2).
- Part II defines DSM, both with respect to current practices (Chapter 3) and in terms of its constituent parts (Chapter 4).
- Part III presents five in-depth examples of DSM in increasingly complex domains (Chapters 5–9).
- Part IV teaches how to create the various parts of a DSM solution (Chapters 10–12), discusses the processes and tools for creating and using the DSM solution (Chapters 13–15), and wraps up with a summary and conclusions in Chapter 16.

In Parts I and II after Chapter 1, readers will find material directed toward those of a more technical, business minded, or academic bent, and should feel free to skip sections, returning to them later if necessary. The examples in Part III build on each other and are also often used in explaining the principles in Part IV, so readers would be advised to at least skim all the examples. In Part IV, the various parts of a DSM solution may all be the responsibility of one person, or then they may be split between two or more people. Chapters 11 and 12, and to a slightly lesser extent Chapters 14 and 15, will make most sense to experienced programmers. Chapters 10 and 13 may interest those in more of an architect or project management role.

The book web site at http://dsmbook.com contains updates, the modeling languages from Part III, and a free MetaEdit+ demo.

ACKNOWLEDGMENTS

With the solid base we had to build on, subsequent years of work with colleagues and customers, and great interactions with our peers in this community, it is clear that we are indebted to far too many people to mention even a representative sample of them here. Even just the OOPSLA workshops on DSM account for several hundreds of authors over the past seven years: seeing so many others coming to similar conclusions and achieving similar successes played a major role in encouraging us on this path.

Some people however simply must be mentioned, so we shall first return to the beginnings of DSM to thank Richard Welke, Kalle Lyytinen, and Kari Smolander for the research that we built on, and for an environment where what was right mattered more than who was right. From that MetaPHOR project to the present day Pentti Marttiin and Matti Rossi have played a central role in keeping us sane, amused, and at least tolerably fit, not to mention an innumerable amount of discussions ranging over all possible metalevels.

All our colleagues at MetaCase have contributed to pushing DSM forward, but we would particularly like to thank Janne Luoma and Risto Pohjonen for their work, discussions and support, as well as their major roles in some of the example cases in Part III. Many, many customers and prospects have shaped and stretched our understanding of DSM as it has been applied to ever new areas. Most are happier keeping their competitive advantage to themselves, but we simply must mention Jyrki Okkonen for his pioneering work with DSM at Nokia. We would also like to thank our esteemed competitors, in particular, Microsoft's Alan Cameron Wills and Aali Alikoski, for their genuine and unprecedented levels of cooperation.

For the book itself, we would first like to thank Stan Wakefield for suggesting we write it, and for playing matchmaker between the publishers and us. Rachel Witmer at Wiley has earned our gratitude for her patience in the face of implausibly long deadline overruns, as DSM took off as a hot topic at just the right and the wrong time. Dave Thomas has fought our corner in North America for a long time, and we are deeply honored to have him write our foreword. In addition to many of those already mentioned, we would like to thank the following for their contribution to or feedback on the book: Jeff Gray of the University of Alabama at Birmingham, Angelo Hulshout of ICT NoviQ, Juha Pärssinen of VTT, Laurent Safa of Matsushita Electric Works, and Jos Warmer of Ordina.