Preface

This book is a software developer's guide to using the Microsoft Tools for Domain-Specific Languages ("DSL Tools"), which are included in the SDK (Software Development Kit) for Microsoft Visual Studio 2005.

The software industry is showing considerable interest in using "domain-specific languages," an approach to building software that promises to reduce software development costs, especially in large projects. A domain-specific language (DSL) is a language specially geared to working within a particular area of interest: a vertical domain such as telephone design, or a horizontal one like workflow. It may be a programming language or a specification or design language. It may be textual or graphical, or a mixture of both. The language is expressed in terms that are used in a particular domain, such as "connect," "ringtone," or "work item," uncluttered by the details of how those concepts are implemented. Software, configuration files, resources, and other documents can be generated from instances of the language—often many of those artifacts can be generated from one DSL—or the language may be interpreted directly. This makes it much easier to discuss the software at the requirements level, and to make changes in an agile way. In vertical domains, the accessibility of the language to business users helps when discussing requirements with them.

DSLs are not a new idea—HTML and SQL are well-known examples of DSLs. Less widespread, however, is the idea of creating your own DSL for your own project. The purpose of the Microsoft DSL Tools is to reduce the upfront cost of doing so. You can quickly create a range of diagrammatic languages, such as workflow, class, or entity diagrams, and you can create tools for generating artifacts from them.

Goals and Scope

This book is for you if you are a software developer or architect using, or thinking about using, the Microsoft DSL Tools. It explains how to create and use languages, how to tune them to your needs, and how to employ them within the context of your project. The book should also be of significant value to readers who are interested in the broader general topic of domain-specific languages, or who wish to compare and contrast different approaches to model-driven development, or tools that support model-driven development. Chapters 1 and 11 discuss the more general topic of domain-specific languages, and how you go about designing one. The middle chapters focus exclusively on providing a detailed yet readable reference on building DSLs and code generators using the DSL Tools.

The book's authors are the main designers of the Microsoft DSL Tools. They have worked together on the product since its inception, and are responsible for most of the key design decisions.

Why You Might Want to Use DSL Tools

If you (or your organization) are writing the same or similar code repeatedly, whether within a single large project or over the course of multiple projects, then such code can probably be generated. If this is the case, you should consider using the DSL Tools as a way to generate this code. This is especially the case if the code can be generated from structures that can easily be understood by domain specialists rather than software development specialists. After reading this book, you should be able to assess the capabilities of the DSL Tools to address problems of this kind, either directly or after some customization.

Organization of This Book

- Chapter 1, *Domain-Specific Development,* explains the DSL approach, compares it with similar techniques, and introduces typical scenarios in which a DSL is used.
- Chapter 2, *Creating and Using DSLs*, looks at the various parts of the DSL Tools system, shows how they fit together, and introduces the main examples that will be used through the remainder of the book.
- Chapter 3, *Domain Model Definition*, details how to define the concepts of the language.
- Chapter 4, *Presentation*, deals with defining the visual appearance of your language.
- Chapter 5, *Creation, Deletion, and Update Behavior*, covers these important aspects of the behavior of your language.
- Chapter 6, *Serialization*, deals with how models and diagrams in your language are represented in files.
- Chapter 7, *Constraints and Validation*, shows you how to ensure that the users of your language create valid statements.
- Chapter 8, *Generating Artifacts*, shows you how to use your language to drive or configure your system by creating configuration files, program code, resources, and other artifacts.
- Chapter 9, *Deploying a DSL,* explains how to create an installer that will install your finished language on multiple computers.
- Chapter 10, *Advanced DSL Customization*, shows you how to make specialized features of your language (or specialized behavior in the editor) in addition to those provided by the standard definition facilities.
- Chapter 11, *Designing a DSL,* provides a lightweight kit of principles and procedures for developing and evolving languages within the context of your project.

Updates and all of the main examples are available for download at the website www.domainspecificdevelopment.com.

What You Need to Use This Book

To get the full value of this book, you need to be reasonably familiar with the facilities that Visual Studio offers to developers of program code, including the code editor and XML editor. A basic knowledge of the C# programming language and the main aspects of the .NET class library are needed to understand the programming examples.

DSL Tools can be downloaded as part of the Visual Studio SDK and used with Visual Studio Professional Edition and later. Tools created using the DSL Tools can be deployed on Visual Studio Standard Edition and later. The website http://msdn.microsoft.com/vstudio/DSLTools/ is the entry point to information about the DSL Tools. There you can find links to where the SDK can be downloaded, a popular online forum with active discussions about the DSL Tools, weblogs containing discussions about the DSL Tools by the authors of this book and others, a tool for reporting bugs and making suggestions, white papers, chats, and other resources.

Acknowledgments

The authors would like to acknowledge the contributions of the following people who contributed materially to the design, development, documentation, and testing of the DSL Tools:

Annie Andrews, Steve Antoch, Austin Avrashow, Bhavin Badheka, Andy Bliven, Anthony Bloesch, Scott Chamberlin, Frank Fan, Jack Greenfield, Howie Hilliker, Ashish Kaila, Jesse Lim, George Mathew, Niall McDonnell, Blair McGlashan, Grayson Myers, Kirill Osenkov, Duncan Pocklington, Anatoly Ponomarev, Jochen Seemann, Keith Short, Pedro Silva, Patrick Tseng, Steven Tung, Dmitriy Vasyura, and Yu Xiao.

We would also like to acknowledge our community of early users, including participants in the DSL Tools Forum, who have stayed with us through a sequence of technology previews. The feedback of these early users has been immeasurably helpful in the process of getting the DSL Tools completed.

The following reviewers have given us invaluable detailed feedback on the contents of the book, which has improved it considerably:

Victor Garcia Aprea, Edward Bakker, Dan Cazzulino, Patrick Cooney, Dragos Manolescu, Jean-Marc Prieur, Jezz Santos, Gerben van Loon, and Markus Völter.

Joan Murray and her team at Addison-Wesley kept us going with patient moral support throughout the writing process.

We would also especially like to thank Bonnie Granat for her accuracy and responsiveness in making sense of and polishing our prose.

Finally, we thank our partners and families for putting up with the evenings and weekends when we have been working on the book instead of spending time with them.