Extracted from:

Domain Modeling Made Functional

Tackle Software Complexity with Domain-Driven Design and F#

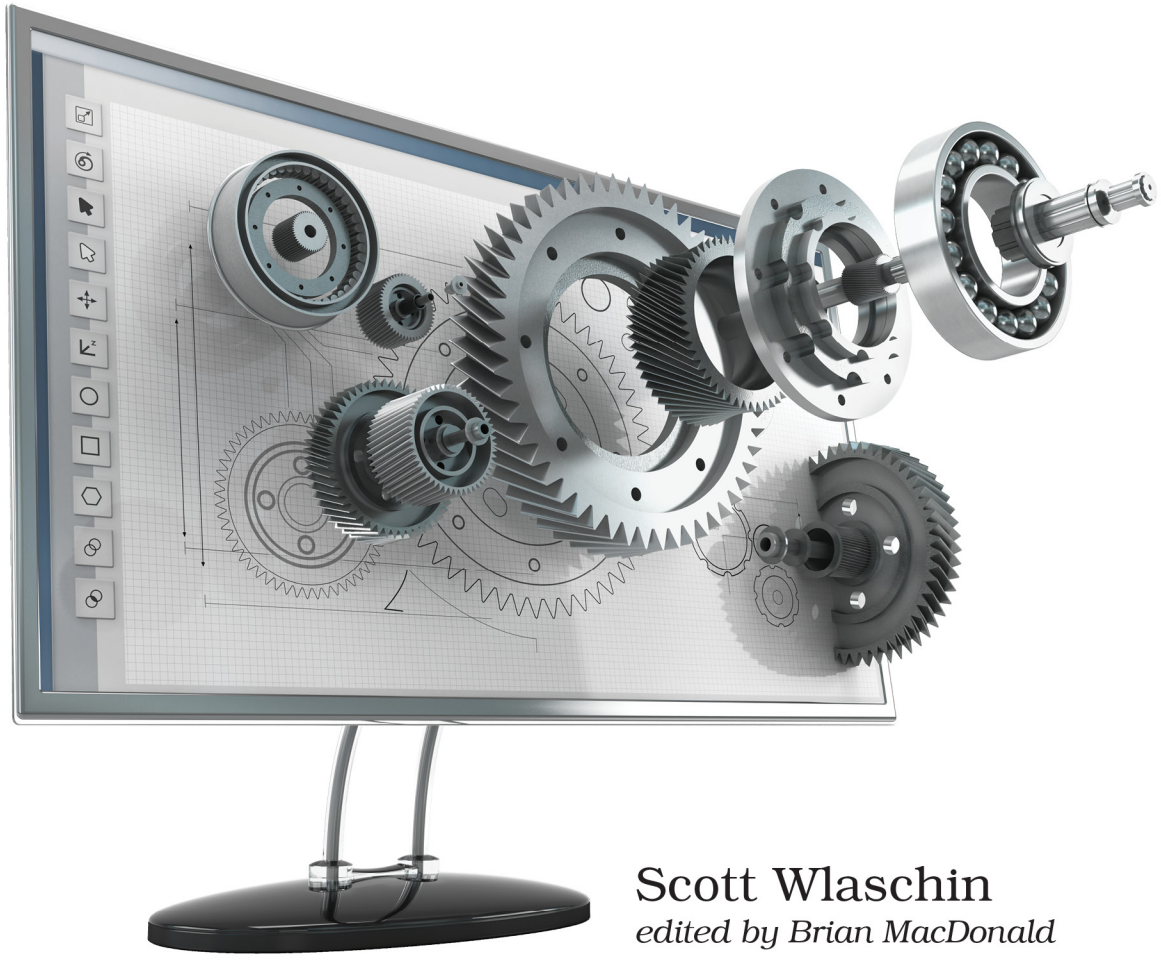
The Pragmatic Bookshelf

Raleigh, North Carolina

The Pragmatic Programmers

Domain Modeling Made Functional

Tackle Software Complexity with Domain-Driven Design and F#



Scott Wlaschin
edited by Brian MacDonald

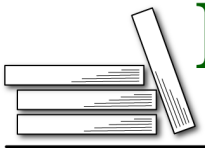
Domain Modeling Made Functional

Tackle Software Complexity with Domain-Driven Design and F#

Scott Wlaschin

The Pragmatic Bookshelf

Raleigh, North Carolina



Pragmatic Bookshelf

Our Pragmatic books, screencasts, and audio books can help you and your team create better software and have more fun. Visit us at *https://pragprog.com*.

The team that produced this book includes:

Publisher: Andy Hunt
VP of Operations: Janet Furlow
Managing Editor: Brian MacDonald
Supervising Editor: Jacquelyn Carter
Indexing: Potomac Indexing, LLC
Copy Editor: Molly McBeath
Layout: Gilson Graphics

For sales, volume licensing, and support, please contact *support@pragprog.com*.

For international rights, please contact *rights@pragprog.com*.

Printed in the United States of America.
ISBN-13: 978-1-68050-254-1
Encoded using the finest acid-free high-entropy binary digits.
Book version: P1.0—January 2018

Preface

Many people think of functional programming as being all about mathematical abstractions and incomprehensible code. In this book, I aim to show that functional programming is in fact an excellent choice for domain modeling, producing designs that are both clear and concise.

Who Is This Book For?

This book is for experienced software developers who want to add some new tools to their programming tool belt. You should read this book if:

- You are curious to see how you can model and implement a domain using only types and functions.
- You want a simple introduction to domain-driven design and want to learn how it is different from object-oriented design or database-first design.
- You are an experienced domain-driven design practitioner who wants to learn why DDD is a great fit with functional programming.
- You want to learn about functional programming, but have been put off by too much theory and abstraction.
- You want to see how F# and functional programming can be applied to real-world domains.

You don't need to have prior knowledge of domain-driven design or functional programming in order to read this book. This is an introductory book and all the important concepts will be explained as we need them.

What's in This Book?

This book is divided into three parts:

- Understanding the domain
- Modeling the domain
- Implementing the model

Each part builds on the previous one, so it's best if you read them in order.

In the first part, Understanding the Domain, we'll look at the ideas behind domain-driven design and the importance of having a shared understanding of a domain. We'll have a brief look at techniques that help to build this shared understanding, such as Event Storming, and then we'll look at decomposing a large domain into smaller components that we can implement and evolve independently.

To be clear, this book is not meant to be a thorough exploration of domain-driven design. That's a large topic that many excellent books and websites cover in detail. Instead, the goal of this book is to introduce you to domain-driven design as a partner to functional domain modeling. We will cover the most important concepts of domain-driven design, of course, but rather than diving deeply into the subject, we'll stay at a high level and stress two things: (a) the importance of communication with domain experts and other non-technical team members and (b) the value of a shared domain model based on real-world concepts.

In the second part, Modeling the Domain, we'll take one workflow from the domain and model it in a functional way. We'll see how the functional decomposition of a workflow differs from an object-oriented approach, and we'll learn how to use types to capture requirements. By the end, we'll have written concise code that does double-duty: first as readable documentation of the domain but also as a compilable framework that the rest of the implementation can build upon.

In the third part, Implementing the Model, we'll take that same modeled workflow and implement it. In the process of doing that, we'll learn how to use common functional programming techniques such as composition, partial application, and the scary-sounding "monad."

This book is not intended to be a complete guide to functional programming. We'll cover just what we need in order to model and implement the domain, and we won't cover more advanced techniques. Nevertheless, by the end of Part III, you'll be familiar with all the most important functional programming concepts and you'll have acquired a toolkit of skills that you can apply to most programming situations.

As sure as the sun rises, requirements will change, so in the final chapter we'll look at some common directions in which the domain might evolve and how our design can adapt in response.

Other Approaches to Domain Modeling

This book focuses on the "mainstream" way of doing domain modeling, by defining data structures and the functions that act on them, but other approaches might be more applicable in some situations. I'll mention two of them here in case you want to explore them further.

- If the domain revolves around semistructured data, then the kinds of rigid models discussed in this book are not suitable and a better approach would be to use flexible structures such as maps (also known as dictionaries) to store key-value pairs. The Clojure community has many good practices here.
- If the emphasis of the domain is on combining elements together to make other elements, then it's often useful to focus on what these composition rules are (the so-called "algebra") before focusing on the data. Domains like this are widespread, from financial contracts to graphic design tools, and the principle of "composition everywhere" makes them especially suitable for being modeled with a functional approach. Unfortunately, due to space limitations, we will not be covering these kinds of domains here.

Working with the Code in This Book

This book will use the F# programming language to demonstrate the concepts and techniques of functional programming. The code has been tested with the latest version of F# as of June 2017, which is F# 4.1 (available in Visual Studio 2017 or installable separately). All the code will work with earlier versions of F# as well, and any important differences will be pointed out in the text.

One of the great features of F# is that it can be used like a scripting language. If you are playing with the example code in conjunction with reading this book, I suggest that you type it into a file and evaluate it interactively rather than compiling it. For how to do this, search the Internet for "F# scripting tips."

All the code in this book is available on this book's page on the Pragmatic Programmers website.[1]

1. https://pragprog.com/titles/swdddf/source_code

Getting Started with F#

If you are new to F#, here's some helpful information:

- F# is an open-source, cross-platform language. Details of how to download and install it are available at fsharp.org.[2]
- Many free development environments are available for F#. The most popular are Visual Studio[3] (for Windows and Mac) and Visual Studio Code with the Ionide plugin.[4] (all platforms)
- For help learning F#, there is StackOverflow (using the "F#" tag) as well as the Slack forums run by the F# Software Foundation.[5] The F# community is very friendly and will be happy to help if you have questions.
- For F# news, follow the "#fsharp" tag on Twitter and read the F# Weekly newsletter.[6]

This book uses only a small set of features from F#, and most of the syntax will be explained as we go. If you need a fuller overview of F# syntax, I suggest searching the Internet for "F# cheat sheet" or "F# syntax."

Questions or Suggestions?

I would love to get your feedback, so if you have questions or suggestions, please participate in the PragProg community forum for this book.[7] And if you find any specific problems with the text, please use the errata submission form there.

Credits

All diagrams were created by the author using Inkscape. The clipart is from openclipart.org. The script typeface ("KitType") used in the diagrams was created by Kenneth Lamug.[8]

2. http://fsharp.org/
3. https://code.visualstudio.com/
4. http://ionide.io/
5. http://fsharp.org/guides/slack/
6. https://sergeytihon.com/category/f-weekly/
7. https://forums.pragprog.com/forums/457
8. https://www.dafont.com/kittype.font

Acknowledgments

I'd like to thank the reviewers of this book for their very helpful comments and feedback: Gien Verschatse, Mathias Brandewinder, Jérémie Chassaing, Clément Boudereau, Brian Schau, Nick McGinness, Tibor Simic, Vikas Manchanda, Stephen Wolff, Colin Yates, Gabor Hajba, Jacob Chae, Nouran Mhmoud and the early access commenters on the book's forum page.

I'd also like to thank my editor, Brian MacDonald, for his editorial feedback and for keeping me on track, and the rest of the PragProg team for making the publishing process so smooth.

Finally, I'd like to thank you, dear reader, for devoting some of your precious time to this book. I hope you find it useful.