Proposal for:
"Domain Driven Design: a Functional Approach using F#"
by Scott Wlaschin

Overview

A developer's job is to *solve a problem through software,* and coding is just one aspect of software development. Good design and clear requirements are just as important, if not more so.

If you think of software development as a pipeline with an input (requirements) and an output (the final deliverable), then the "garbage in, garbage out" rule applies. No amount of coding can fix unclear requirements or a bad design.

This book is a pragmatic, down-to-earth guide to minimizing the “garbage-in” by using a design approach focused on clear communication and shared domain knowledge: *Domain-Driven Design*.

It shows how applying the core principles of functional programming in conjunction with Domain-Driven Design can result in software designs that model real-world requirements in an elegant and concise way. This in turn means increased customer satisfaction, faster development cycles, and less wasted work.

Using examples from familiar business domains, the book explains and demonstrates the principles of DDD through a series of practical examples in F#. You’ll learn how to apply these techniques to your own software projects, and create business-focused designs that are precise, maintainable, and testable.

What is covered

This book will:

- Introduce you to the key concepts of DDD through real-world examples.
- Show you how the principles of functional programming (such as composability and immutability) can be applied to domain modeling and high-level design.
- Show you how to encode business rules in the design so that you have "compile-time unit tests".
- Explain why the combination of functional programming and DDD leads naturally to service-oriented and hexagonal architectures.
- Show you how to keep your projects free from ugly error-handling code and at the same time reducing the number of tests you have to write.
- Show you how to use these functional domain models in conjunction with traditional databases, NoSQL, and event stores.

- Show you how to safely expose your domain via a website or API, and how functional programming and RESTful HATEOAS go together like peanut butter and jelly.

Audience

This book is suitable for experienced developers who love designing software and want to learn more about DDD and functional programming in a no-nonsense way.

F# experience is not required -- you can learn the language as you work through the book. No jargon! No mathematics!

Why is your book different?

This is the only book which covers the intersection of functional programming and high-level software design.

Most functional programming books on the market have an academic focus or use off-putting jargon. This book avoids jargon and introduces you to F# and type-driven design in a series of easy-to-understand examples.

Similarly, many of the DDD books focus on the DDD terminology first, which can be intimidating. This book introduces the concepts through concrete examples, and shows how many of these ideas arise naturally from business-focused design.

What will readers be able to do that they couldn't do before?

- They will be able to use F# to create concise but readable domain models that can be shared with non-technical team members and customers.
- They will be able to have better design-related communications, with more rapid feedback, without needing complicated processes and tools.
- They will know how to use F# to create working software which is also comprehensive documentation – no more UML diagrams!
- They will be able to create functional designs that ensure fully decoupled, fully testable code.
- They will be able to encode business rules into the type system, so that many runtime errors are avoided, and fewer tests need to be written.
- They will able to apply functional programming concepts such as composability and immutability to their own projects.
- They will be able to write APIs that cannot be used incorrectly.

Why we should get excited about your topic

Functional programming is rapidly becoming mainstream, and yet there is very little material available to help with design, as opposed to coding.

This book aims to fill this gap. Most developers just want to understand the business requirements, simplify their code, and reduce the number of tests that they have to write. They don’t want to know about the theory of monads.

The goal of this book is to make those developers happy!

Market analysis and promotion strategy

Competing books

There is no direct competition, as this is the only book which covers the intersection of functional programming and business-focused domain modeling.

DDD books

All the well-known DDD books are object-oriented and slightly out-of-date with respect to the latest trends such as immutability, microservices, and lambda architectures.

This book does not go into detail on things like microservices, but the reader will be shown how they evolve naturally from the kinds of functional approaches demonstrated in this book.

The most popular domain-driven design books are:

- *Domain-Driven Design: Tackling Complexity in the Heart of Software* by Eric Evans (560 pages). First published in 2003, it is classic but can be quite dense to read. It uses Java as the programming language.
- *Implementing Domain-Driven Design* by Vaughn Vernon (656 pages). Published in 2013, this focuses on the implementation of DDD. Examples in Java.
- *Applying Domain-Driven Design and Patterns: With Examples in C# and .NET* by Jimmy Nilsson. An older book (2006) with examples in (old!) C#.
- *Patterns, Principles and Practices of Domain-Driven Design* by Scott Millett (576 pages). Published in May 2015, this combines the high-level of the "blue book" (Eric Evans) and some implementation notes like the "red book" (Vaughn Vernon)

As you can see they are all quite heavyweight -- none are under 500 pages!

Vaughn Vernon has also published a condensed version of his book this year: *Domain-Driven Design Distilled* (176 pages).

Functional programming books

On the other hand all the well-known functional programming books tend to be quite low-level and focused on computer-science oriented topics (algorithms and abstract data structures) rather than on enterprise development.

Some examples of popular functional programming books for statically typed functional programming languages (like F#) are:

- *Real-World Functional Programming* by Tomas Petricek & Jon Skeet. Manning (500 pages). Published in 2010, it is still the most popular book on F# for people coming from a C# background.
- *Functional Programming in Scala* by Paul Chiusano & Rúnar Bjarnason. Manning (320 pages). A recent (2014) and well-reviewed book, it is still quite intense for the average enterprise developer. It does not mention business-focused design at all.
- *Learn You a Haskell for Great Good!* by Miran Lipovaca. No Starch Press (400 pages). Published in 2011, it is the goto book for learning Haskell (because there is a free version). There is a lot about monads, but nothing about business-focused design.
- Coming soon (at the end of 2016) is *Functional and Reactive Domain Modeling* by Debasish Ghosh. Manning (325 pages). Examples in Scala. I have read the sample chapters and watched his talk on the topic, and although it is interesting, I don't think it comes across as accessible to most mainstream developers.

Market information

This is a book which crosses two markets, those interested in learning DDD and those interested in practical applications of functional programming using F#.

Market for DDD books

The nice thing about DDD books is that they don't go out of date quickly, and so they stay viable for many years.

Based on their Amazon rankings, I guestimate that the Eric Evans book is currently selling at about 300 copies/month (13 years later!) and the Vaughn Vernon book at 100 copies/month (3 years after publication).

I also have some of my own numbers for people interested in the functional approach to domain modeling, based on analytics from my own SlideShare decks.

My two slide decks (one and two) on this functional DDD have been viewed almost 200,000 times and downloaded 3000 times. That's with zero promotion. It's not clear what the conversion rate would be though.

Market for F# books

I have some numbers for people interested in F# in general, based on the analytics from my site fsharpforfunandprofit.com.

- In the first nine months of 2016, 30,000 people visited the site 50 or more times, and 12,000 people visited the site 100 or more times.
- I made the content of the site available as a free eBook download at the end of July 2016. Since then it has been downloaded over 15,000 times (in two months). Obviously, the content is free, but I expect that we could convert maybe 10% of the book downloaders.

Going back to the Amazon sales for functional programming books, I estimate that the *Real-World Functional Programming* book is currently selling about 20 copies/month (after 5 years) and the *Functional Programming in Scala* book is currently selling about 200 copies/month (after 2 years).

Promotional ideas

I am well-known in the functional community, and it's a small world, so I think that a word-of-mouth social media campaign would be most effective.

- **Promotion by me on social media**. As you can see above, I have a large number of repeat visitors to my site, and tens of thousands of people have watched my talks on video, so I would expect to get many conversions just from my own blogging, tweeting and interactions on forums like Hacker News, Reddit, Quora, etc.
- **Promotion by me at conferences**. I am speaking at "Domain-Driven Design Europe 2017" next February and would promote it there and at the other conferences and meetups I attend such as NDC, F# exchange, etc.
- **Promotion by influencers**. I have quite a few connections to influencers like Eric Evans and Greg Young. I think they would promote the book if they thought it was useful to their followers.
- **"Recommended books" lists**. I would endeavor to get the book added to all places where F# and DDD books are mentioned, such as fsharp.org, DDD Community, and the Wikipedia entry on Domain-driven design. Also, "recommended programming books" lists (example) are always popular on social media, and I would reach out to creators of those lists to add this book.
- **Drip marketing**. I have not been actively maintaining an email list, but that is something I am planning on doing now, and I would start a drip-marketing campaign for the book.

The channels above match my own experience of how I have learned about new programming books.

- **Press-based PR**. Based on the analytics for my site, the technical press (what's left of it) does not drive nearly as much traffic as Hacker News, and Reddit and so I feel that standard press-based PR would not be the most effective approach.

My bio

I have been a programmer for 25 years, always working in the mainstream or enterprise world. I am not an academic -- I like to develop software that is useful and practical.

I spent many years programming in OO languages (including C# and Smalltalk) before becoming interested in functional programming, so I am very sensitive to the effort involved in learning a new paradigm. In my writing, I always try to keep my examples grounded in everyday problems, and I avoid jargon as much as possible -- I even have a list of banned words (like "monad") on my website!

This approach seems to have been successful, as judged by the popularity of my site. Many people have complimented me on having a friendly and unintimidating style and of doing a great job of explaining advanced concepts (examples). For example, I created the "railway-oriented programming" metaphor for functional error handling which has gained some traction.

As to writing, I enjoy it a lot, and I have been writing for as long as I have been a programmer. For example, I self-published 3 Smalltalk books back in the 1990's, and I estimate that there are currently about 500,000 words on my fsharpforfunandprofit.com site.