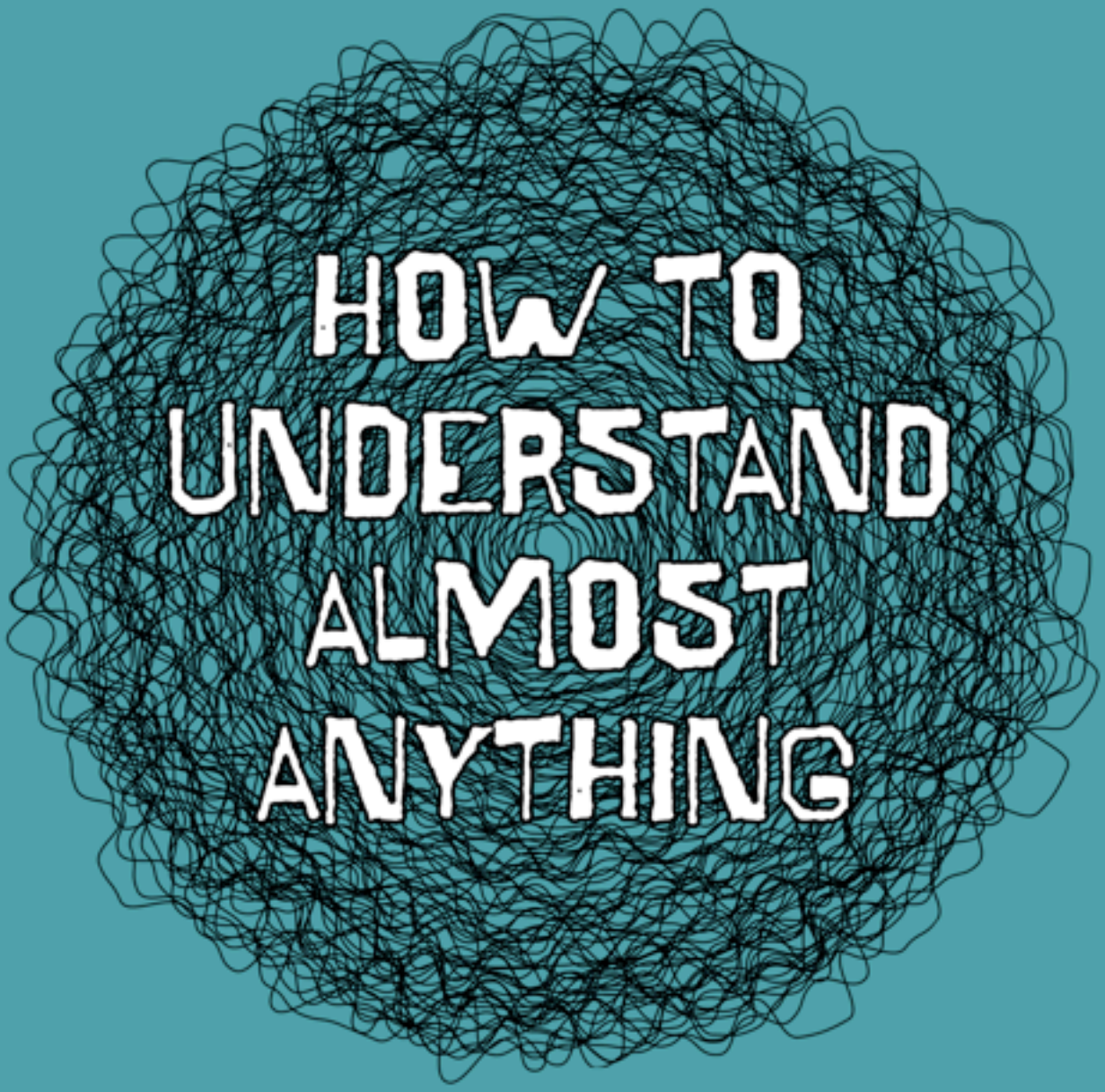
MARKUS VOELTER



A PRACTITIONER'S GUIDE TO DOMAIN ANALYSIS

V1.0

VOELTER.DE/HTUAA

Markus Voelter

How to Understand Almost Anything

A Practitioner's Guide to Domain Analysis

Version 1.0, April 2023

See http://voelter.de/HTUAA for details.

Domain analysis is about understanding a body of knowledge, the subject matter of a domain. You want to understand the concepts, vocabulary, rules and constraints clearly enough so you can define a set of abstractions and their relationships, a formal model, a "language" that can be used to completely and unambiguously describe subject matter in the domain, and ultimately, to execute the subject matter. This book explains how to do this. After the introduction, the core of the book is three sections that cover the three aspects of domain analysis: collecting, thinking and validating. Chapter five zooms out beyond the core domain analysis activities and discusses related issues such as agility, business strategy and process change. The last chapter describes recurring ideas and concepts when defining abstractions. You can use those as reminders – or a checklist – of what to think about when designing your abstractions. The book is based on more than a decade of analysing domain and designing languages in a wide variety of domains from science to finance to public administration to healthcare.

Preface

I have written several books related to software engineering. All of them covered technical content: software architecture, middleware, modelling and code generation, as well as domain-specific language (DSL) design and implementation. Why, you might ask, is this guy writing about 'soft' topics now, things like analysis, writing, communication and processes. Has he become one of these lots-of-talk-but-no-walk consultants?

I hope not. However, the DSL[1] projects I have been involved in have grown in scope, size and importance (for the client), and my role has shifted away from purely technical advice and language implementation. I spend more of my time with domain analysis, architecting systems in which DSLs and 'executable subject matter' play a central role, and I also consult on how processes should change to take full advantage of this approach. So while I've been writing code probably every day over the last couple of years, a significant portion of my time was spent on these 'softer' topics.

Something else has also changed for me. When I started out with modelling, DSLs and MPS around 15 years ago I was totally fascinated by the technology. I didn't care too much what people used it for as long as they paid me to 'play' with it. Now that I am used to the technologies and their potential benefits, I'm more interested in reaping these benefits for my customers. In other words, I am much more interested in solving *real* problems using DSLs and more generally, tools that focus on capturing subject matter. This means that all the 'soft' topics I mention above have to be taken into account. Hence my personal evolution in this direction.

Several of my buddies, colleagues and customers asked me how I go about this domain analysis thing. How do I find out what should go into a language? How do I extract all that domain knowledge from people's heads and capture it in a set of consistent abstractions? Is there a methodology? (Spoiler: no.) So I decided to record my experience in this little book. Note the word 'experience'. This is not a scientific book – I am not citing 250 other people who analyse domains. It's my approach, and as experience can be both

[1] This is not a DSL book. Please continue reading :-)

positive and negative, you'll read suggestions on what to do and what not to do. So what I write in this book is not the one and only way to approach domain analysis, but it is one that has worked in several projects of various sizes and in domains ranging from science through tax calculation to healthcare. That has to count for something.

This is also not a DSL book, because domain analysis is important for non-DSL projects as well. Domain analysis is relevant whenever you need a consistent set of abstractions – a language – that allows users to express subject matter concisely and correctly in a way that can be analysed and executed by a machine.

Zooming all the way out, there's also a 'bigger picture' perspective for this book. With all the changes going on in the world, (Western) industrial countries have to become more competitive to be able to retain our current standard of living. One way of achieving this is with breakthrough ideas: if we can get fusion to work, lots of problems will be solved (I'm only half-joking here). Another way of increasing competitiveness is to do keep doing what we do now, but do it more efficiently. We can make the the existing workforce more effective, so that we can build better software faster, optimise how we capture subject matter and de-crappify processes – we all know that there is lots to optimise in this space in our corporate world! This seems especially relevant with the skills shortages caused by an ageing population. To do things more efficiently, organisations first have to understand clearly what they are actually doing. They have to understand their domain precisely. Understanding how a domain works, what an organisation does now and how things should work going forwards is genuinely non-trivial. With this book I hope to help with making this knowledge-discovery process more efficient.

The book is relatively short: you should be able to read it in about five hours. It could have been longer, with more examples and maybe a complete case study. But after writing several 'large' books in my career I didn't have the energy to do another one, which is why this one is shorter. A side effect is that it is less of a commitment for you to read. I chose a very colloquial style; I hope this makes the book more fun to read.

Before I let you move on to the core of the book, please join me in thanking my reviewers – it is hard to overstate how important their feedback was. I wrote a lot of this stuff almost as therapy for some of my frustrating experiences in recent projects. In the original draft some things sounded even more frustrated than they do now, while other things were too black-and-white, or too unsystematic. In Chapter Six things were way to brief. Without the honest, constructive and plentiful feedback – I counted around 300 comments, excluding typos and other trivial stuff – this book would not be what it is now. So thank you very much to (in alphabetical order): Boris Holzer, Bruce Trask, Federico Tomassetti, Kresimir Vukorepa, Lisa Hartl, Mike Vogel, Sascha Lißon and Sergej Koscejev.

You know what my first book Server Component Patterns and indeed all my other English-language books have in common with this one? The copyeditor Steve Rickaby! Thank you very much Steve for once again cleaning up my writing, geeking out about details of how to best express things, and the occasional anecdote in the review comments. By the way: if the final version of the book still contains mistakes, it's likely my fault, not Steve's, because I have made a few changes to the text after I got the draft back from the bottom-left corner of the UK :-)

I also want to thank my customers over the last 15 years or so who gave me the opportunity to learn and grow professionally and eventually condense my experience into this book. Although it might come across different because my anecdotes often focus on the things that didn't go well, most of my work is productive and enjoyable.

Finally, I'm also looking forward to *your* feedback! Let me know whether you like what you're reading, whether you disagree, or whether you have additional input. This will go into future versions of the book. Contact me via http://voelter.de/ping or voelter@acm.org . You can also use this email address to register the book so I can notify you when there's a new version.[2]

Enjoy the ride!

Markus Voelter, April 2023

[2] Needless to say I won't spam you or try to sell you my services through this channel.

Table of Contents