Extracted from:

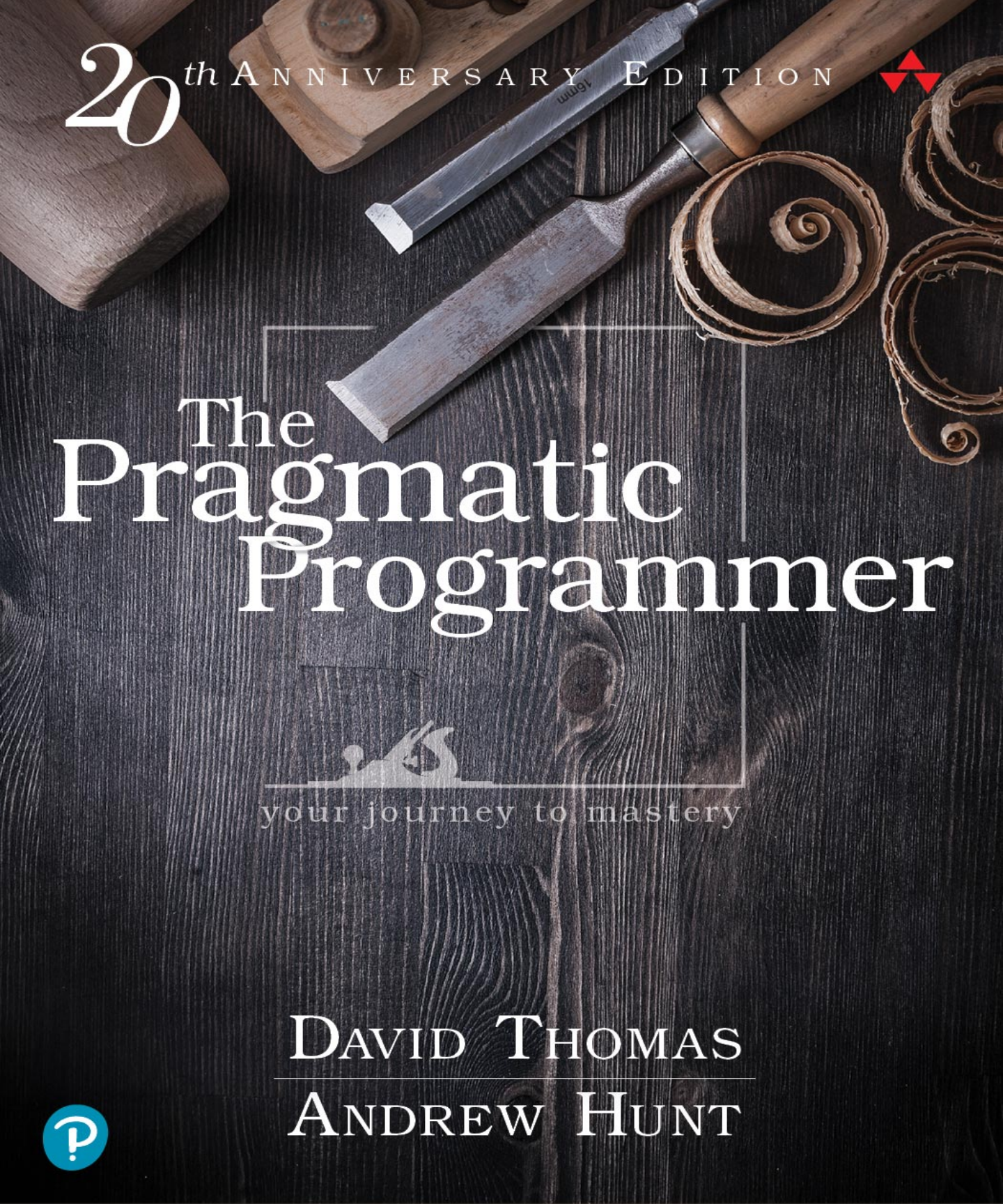
The Pragmatic Programmer

your journey to mastery

20th Anniversary Edition

Addison-Wesley

Boston • Columbus • New York • San Francisco • Amsterdam • Cape Town
Dubai • London • Madrid • Milan • Munich • Paris • Montreal • Toronto • Delhi • Mexico City
São Paulo • Sydney • Hong Kong • Seoul • Singapore • Taipei • Tokyo


20th Anniversary Edition
The Pragmatic Programmer
your journey to mastery
David Thomas
Andrew Hunt

The Pragmatic Programmer

your journey to mastery

20th Anniversary Edition

Dave Thomas
Andy Hunt

Addison-Wesley

Boston • Columbus • New York • San Francisco • Amsterdam • Cape Town
Dubai • London • Madrid • Milan • Munich • Paris • Montreal • Toronto • Delhi • Mexico City
São Paulo • Sydney • Hong Kong • Seoul • Singapore • Taipei • Tokyo

Library of Congress Control Number: [to come from ITP]

ISBN-13: 978-0-13-595705-9
ISBN-10: 0-13-595705-2

1 19

Preface to the Second Edition

Back in the 1990s, we worked with companies whose projects were having problems. We found ourselves saying the same things to each: maybe you should test that before you ship it; why does the code only build on Mary's machine? Why didn't anyone ask the users?"

To save time with new clients, we started jotting down notes. And those notes became *The Pragmatic Programmer*. To our surprise the book seemed to strike a chord, and it has continued to be popular these last 20 years.

But 20 years is many lifetimes in terms of software. Take a developer from 1999 and drop them into a team today, and they'd struggle in this strange new world. But the world of the 1990s is equally foreign to today's developer. The book's references to things such as CORBA, CASE tools, and indexed loops were at best quaint and more likely confusing.

At the same time, 20 years has had no impact whatsoever on common sense. Technology may have changed, but people haven't. Practices and approaches that were a good idea then remain a good idea now. Those aspects of the book aged well.

So when it came time to create this *20th Anniversary Edition,* we had to make a decision. We could go through and update the technologies we reference and call it a day. Or we could reexamine the assumptions behind the practices we recommended in the light of an additional two decades worth of experience.

In the end, we did both.

As a result, this book is something of a *Ship of Theseus*.[1] Roughly one-third of the topics in the book are brand new. Of the rest, the majority have been rewritten, either partially or totally. Our intent was to make things clearer, more relevant, and hopefully somewhat timeless.

1. If, over the years, every component of a ship is replaced as it fails, is the resulting vessel the same ship?

We made some difficult decisions. We dropped the *Resources* appendix, both because it would be impossible to keep up-to-date and because it's easier to search for what you want. We reorganized and rewrote topics to do with concurrency, given the current abundance of parallel hardware and the dearth of good ways of dealing with it. We added content to reflect changing attitudes and environments, from the agile movement which we helped launch, to the rising acceptance of functional programming idioms and the growing need to consider privacy and security.

Interestingly, though, there was considerably less debate between us on the content of this edition than there was when we wrote the first. We both felt that the stuff that was important was easier to identify.

Anyway, this book is the result. Please enjoy it. Maybe adopt some new practices. Maybe decide that some of the stuff we suggest is wrong. Get involved in your craft. And give us feedback.[2]

But, most important, remember to make it fun.

How the Book Is Organized

This book is written as a collection of short topics. Each topic is self-contained, and addresses a particular theme. You'll find numerous cross references, which help put each topic in context. Feel free to read the topics in any order—this isn't a book you need to read front-to-back.

Occasionally you'll come across a box labeled *Tip nn* (such as Tip 1, *Care About Your Craft*, on page ?). As well as emphasizing points in the text, we feel the tips have a life of their own—we live by them daily. You'll find a summary of all the tips on a pull-out card inside the back cover.

We've included exercises and challenges where appropriate. Exercises normally have relatively straightforward answers, while the challenges are more open-ended. To give you an idea of our thinking, we've included our answers to the exercises in an appendix, but very few have a single *correct* solution. The challenges might form the basis of group discussions or essay work in advanced programming courses.

There's also a short bibliography listing the books and articles we explicitly reference.

2. https://pragprog.com/titles/tpp20

What's in a Name?

"When I use a word," Humpty Dumpty said, in rather a scornful tone, "it means just what I choose it to mean—neither more nor less."

➤ *Lewis Carroll, Through the Looking-Glass*

Scattered throughout the book you'll find various bits of jargon—either perfectly good English words that have been corrupted to mean something technical, or horrendous made-up words that have been assigned meanings by computer scientists with a grudge against the language. The first time we use each of these jargon words, we try to define it, or at least give a hint to its meaning. However, we're sure that some have fallen through the cracks, and others, such as *object* and *relational database*, are in common enough usage that adding a definition would be boring. If you *do* come across a term you haven't seen before, please don't just skip over it. Take time to look it up, perhaps on the web, or maybe in a computer science textbook. And, if you get a chance, drop us an email and complain, so we can add a definition to the next edition.

Having said all this, we decided to get revenge against the computer scientists. Sometimes, there are perfectly good jargon words for concepts, words that we've decided to ignore. Why? Because the existing jargon is normally restricted to a particular problem domain, or to a particular phase of development. However, one of the basic philosophies of this book is that most of the techniques we're recommending are universal: modularity applies to code, designs, documentation, and team organization, for instance. When we wanted to use the conventional jargon word in a broader context, it got confusing—we couldn't seem to overcome the baggage the original term brought with it. When this happened, we contributed to the decline of the language by inventing our own terms.

Source Code and Other Resources

Most of the code shown in this book is extracted from compilable source files, available for download from our website[3].

There you'll also find links to resources we find useful, along with updates to the book and news of other Pragmatic Programmer developments.

3. https://www.pragprog.com/titles/tpp20

Send Us Feedback

We'd appreciate hearing from you. Email us at ppbook@pragprog.com.

Second Edition Acknowledgments, Year 2020