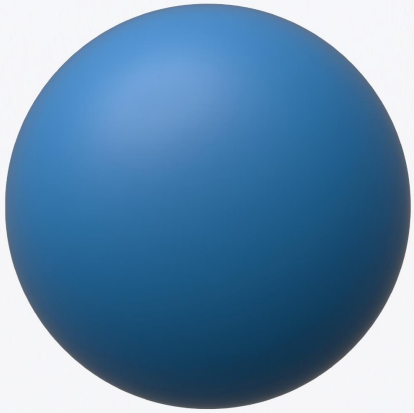
PragmaticBookshelf

simplicity

sustainable, humane, and effective software development



dave thomas

edited by
Susannah Davidson

This extract shows the online version of this title, and may contain features (such as hyperlinks and colors) that are not available in the print version.

For more information, or to purchase a paperback or ebook copy, please visit https://www.pragprog.com.

Let's Change Our World—Again

Over the last decade I've been looking at the state of software development with a combination of awe and terror.

We have achieved amazing things, things that change the world. While doing it, we've also defined new ways of working and interacting, ways that often make the world smaller and that build bridges across cultures. Never doubt that software development is a potent and powerful force in today's world.

But I'm also scared, because I know that we are teetering on the brink of being out of control. We are all running as fast as we can just to stay in the same place. We repeat the mistakes of the past because we have no time to learn our history. We fight our own personal obsolescence as technologies reshape themselves around us.

The fact that we're surprised (or at least relieved) when something works is a good indication that we don't have the control that we need.

Why haven't we tamed the development process? The answer is obvious:

Idea 1 **The world of software development is inherently complex**

Unfortunately, we seem to delight in making it not just complex, but complicated, too. We feel the pressure to adopt bright, shiny, new technology for fear of missing out. We feel empowered by adding bells and whistles to our projects: look what I can do! We don't invest the time to make things simple, because we have no time.

Ironically, we have no time because things are complicated.

We have to make the time; if we don't, we won't learn and improve. We have to discover how to embrace simplicity and create simple things, otherwise we'll spend our time like the plate juggler, running between the poles to stop things crashing. We need to learn that it's OK not to follow the influencer-lemmings off the cliff of new fads. They earn eyeballs by hyping the new and

untested; you should take note of what they say, but also feel good using less shiny but more reliable (and better known) tools and frameworks you already know.

Ward Cunningham coined the phrase "do the simplest thing that could possibly work." That's not just a way of writing code; it applies to everything we do.

Simplicity doesn't mean simplistic. It doesn't mean naïve. It means producing things that are easy to understand and change, and that have that vague quality of "feeling right." Christopher Alexander, the architect who wrote the original *A Pattern Language: Towns, Buildings, Construction [AIS77]*, called this the *Quality Without A Name.* Achieving simplicity rewards you with this feeling.

If you ever come across someone trying to sell The Simple Methodology, laugh in their face and slam the door. Simplicity isn't the way you do things; it's the spirit with which you do them.

How To Approach This Book

There's a problem writing about simplicity; there are no rules or procedures that make things simple, no coding standards or design methodologies that guarantee the things you produce will not end up complex.

Why? Because what's seems simple to me might not seem so to you, and something you find simple might go over my head.

In the end, I took an indirect approach. I don't tell you "do this, because it is simpler than that." Instead, I describe situations that I've been in that felt more complex than they should be, and then outline the steps I took to simplify things. I call each of these scenarios a *practice*.

A practice is not a part of some overall methodology—perish the thought. Nor is a practice something you should blindly adopt.

Instead, a practice is an example of the way I personally deal with some aspect of software development. I don't expect you to adopt any practice unthinkingly. Instead, I hope that you'll read each as a kind of story.

Each practice starts with some kind of problem. Then there might be an elaboration of that problem, perhaps explaining what issues it causes, or what other approaches I've taken.

Finally, there's the description of a solution. To be honest, this is possibly the least important part of the story, because these solutions work for me,

doing what I do. They are there not as an answer, but as an illustration of one of many possible approaches to finding an answer.

How To Get The Most Out Of The Practices

If you just read the practices as a sequence of "what Dave did next" blog posts, then I will have failed.

Approach these practices as if they were parables: stories which are a kind of metaphor. As you read, pause to consider the parallels between my problems and your problems, my solutions and yours.

When you're reading about a complexity I encountered, take my problem and project it into your context—your code, your team, or your project. Think about the symptoms of my problem; are there parallels in your world? My intent here is to help you to learn to spot complexity by giving you examples of things that messed me up.

If I elaborate on a problem, I'm illustrating the way I think about issues after I've identified them. Again, I may be right or wrong; what matters is that you get into the habit of treating each complexity with respect, and not just go for the quick fix.

And when you come to my suggested solution, well, that's your opportunity to find all the holes and contradictions. Be critical (but nicely). The better you get at thinking critically about possible solutions to a problem, the easier it will be to filter the solutions to your own complexities.

The Order Is Up To You

These practices can be read in any order; most are free-standing, and they're also fairly short reads.

To impose a little structure, however, the practices are divided between four parts: projects, environment, interactions, and code.

1. *Simplify Your Projects* looks at the higher levels of development; how we can reclaim the original spirit of agility without relying on full corporate buy-in? Here you'll also find approaches to making software simpler by reducing the amount we write, and also the amount we depend on.

2. *Simplify Your Environment* explores the areas over which you have direct control: your tools and your personal development.

3. *Simplify Your Interactions* is a short part, just one chapter, but it's an important one. You don't work in a vacuum, so let's look at making it easier, less time-consuming, and more pleasant to work with other people.

4. *Simplify Your Code* sits at the lowest level. Truthfully, it could be a book by itself, but instead it contains just two chapters. The first is a series of practices involving swapping the roles of code and data when developing; I've been using data-driven development as a tool for years now, and I won't go back. The second chapter is a fairly short list of things that work for me when I program.

There's a fifth topic in this book, but it has no explicit part or chapter. Instead, it appears on every page.

Courage

Kent Beck kicked off modern software development with *Extreme Programming Explained: Embrace Change [Bec00]*. He lists the five values of XP as Communication, Feedback, Respect, Simplicity, and Courage. I believe these are core values not just for eXtreme Programming, not just for programming in general, but for life.

Sometimes I wish I could call this book simply "Courage," but I'm afraid to, because it might put potential readers off.

Ultimately the ideas in this book all make great demands on you as a person. They ask you to create and follow a set of values, often in the face of peer pressure. They ask you think about what you're doing, rather than going with the flow, and they inform you when you have to make a stand. And they ask you to fight against all the factors that make things complex as you try to bring simplicity into your life.

All this takes courage. All great changes do.

Enjoy the process.

See You Online

https://pragprog.com/titles/dtcode
: The book's home page, with links to its discussion forum and the errata list

articles.pragdave.me
: Where I write articles and notes about simplicity and whatever else catches my fancy.

pragdave.me

My personal homepage.

pragprog.com

My business and the last 20 years of my life.

Part I

Simplify Your Projects