THE OTHER HALF OF CODING

WHAT THEY DIDN'T TEACH YOU

MAX GUERNSEY, III

The Other Half of Coding

What they Didn't Teach You

Max Guernsey, III

This book is available at https://leanpub.com/other-half

This version was published on 2026-01-01

This book, too, is dedicated to my wife and children, who are the driving force behind everything I do.

Contents

Preface

I've been working in software for nearly 30 years. In the course of my career, I've adopted what are now considered the core modern practices of the profession: Test-Driven Development, refactoring, and Design Patterns. I have been a strong supporter of all these things (and more) for most of my career,

It seems like those skills should be enough, yet, I've seen countless teams end up starting out committed to them only to end up struggling anyway. Eventually, they ended up slowing down and having areas of the code they didn't want to change anyway.

Sometimes those teams—even the ones with the strongest developers on them—even decide to give up on good practices and backslide into old habits.

Yet I didn't have this experience, myself.

This book exists because I realized that there was something missing from what those teams were doing. It's not that they were applying the disciplines incorrectly. It's that there was something they weren't doing that they should be.

I'm not interested in replacing existing practices, arguing against them, or proposing a silver bullet. Everything discussed here is meant to sit alongside the techniques you already know. The intent is additive, not corrective.

This is also not a book about tools, frameworks, or process. It's written for experienced developers who already understand the surface area of modern practice but feel that something important remains unarticulated—something they may apply intuitively, inconsistently, or not at all.

The chapters that follow are deliberately structured. They move from diagnosis, to reframing, to concrete techniques. Some sections will feel philosophical; others will feel uncomfortably practical. The book rewards slow, careful

reading more than skimming, and it assumes a willingness to question habits that may feel settled.

If you're looking for quick tips, checklists, or a new methodology to roll out to a team, this is probably not the right book.

If, on the other hand, you've ever felt that changing software is harder—and riskier—than it ought to be, even when you "do everything right," then this book was written for you.

Part I: The Gaping Hole

There's something missing from our industry. Something important.

It's a big empty spot in part of our workflow. It manifests in how we code, but I think it goes much deeper than that.

I think it goes all the way back to how people get their start a software developers.

Chapter 1: Introduction

Think about the top 1% of developers you know. What do they all have in common?

Are they all smart people?

Probably, but do you know any smart developers whom you wouldn't place in the top 1%?

Are they always up to date on the latest techniques?

Maybe, but that could be a correlation. I doubt, very much, that it is a driver.

What about the latest technologies?

Undoubtedly, they are experts in whatever technology they work with most, but that is not what makes them top 1% developers.

As I started to think about this and go through all the developers I can remember in nearly 30 years as a programmer, I became less and less confident in what made someone a really, really good developer.

When it finally struck me, I found the answer surprising.

It was one of those epiphanies that you know to be right because, in retrospect it appears "obvious", even though it eluded you for years leading up to its discovery.

So, later in this chapter, I'll explore my own journey as a software developer.

First, though, I want to give you an example of what I consider to be a top-1% developer.

1.1: Example: The Rewrite that Stopped the Bleeding

One of the best developers I've ever met is someone I worked with over a decade ago, now.

I knew him when he was still "just" a tester. I watched him as he switched over to developer and took off.

In some regards, he's not my best case study, here, because he is, in fact, brilliant. But, at the beginning, his raw intellect was offset by his newness.

Still, very early in his career, he set himself apart by making what people considered a dangerous decision...

There was this one part of the system that made everything slow. I don't mean "3% slower because of this class". I mean - and I write this without a shred of hyperbole - certain customers were waiting 45 minutes for login to complete. Again, that's not a typo or an exaggeration. That's just the reality.

On top of that, was a constant source of waste. Our manager crunched the numbers and decided that the team lost something like 60 hours a month to the new bugs that were constantly being created and found in this single class.

It was a huge class, too. As things got worse, people kept adding to it in an attempt to fix it. That only made it worse, though.

He decided to take it on himself to rewrite it. He produced a system of classes about 350 lines long in total (<10% the original length) that completely solved the problem. He fearlessly changed the coupling over to his new system, knowing that he was taking a risk.

1.2: The Result?

The result was overwhelmingly positive.

The new system he built was easy to understand, easy to change, and fully covered in tests.

The stream of something like 3 high severity defects a month was sealed off– the additional productivity the team got over the next few months more than covered the initial investment of his time.

In addition to sealing up the time-sink, it also solved several critical problems. Not the least of which was bringing login times down to something reasonable.

1.3: Was It Perfect?

Some might ask if this went perfectly. He was dealing with an extremely fragile subsystem that was widely used throughout the system.

He had to identify all the use cases. He had to point all the client code to the new thing. He had to make it all work.

Did he create any bugs? You bet he did.

He created exactly **one** bug of medium severity in that process. A bug he was able to quickly fix.

Since people were accustomed to three severe bugs coming from that system a month when nobody is changing it (aside from previous "bugfixes"), nobody was upset.

He took a risk. There was a cost. It paid off.

1.4: His Career

I won't say that this launched my friend's career. I'm sure it had a positive impact, **but if not this, then some thing**.

What I mean by that is that this is a pattern for him.

It wasn't a one-time risk he took that paid off and launched him: It was an expression of the way he works.

This approach to software has served him well. He quickly climbed the ranks of individual contributors and then started moving up in leadership.

The last time I checked, he was a VP of Engineering at a well-funded and growing startup with a strong team underneath him. A team that trusts his skills and his abilities.

A consistent pattern of making the right decisions with good long-term payoffs—and a solid plan for how to implement those decisions—is what put him where he is, today.

That got me thinking about my own trajectory.

1.5: My Story

Now I want to take you back to my beginnings as a software developer.

I've been coding for a long, long time. I've been doing it seriously since I was a child.

I used it to weasel my way out of a number of classes in high school that I thought were beneath me.

Why should I take an elective, when I was the one teaching programming? I'll just convince the administration that that was worth a math credit.

Why should I take a foreign language? Isn't defining and building my own programming language foreign enough?

I started coding professionally when I abandoned the "education" system once and for all in my freshman year of college.

I've always thought of myself as pretty good at this. Humility only really entered into the picture once I'd been doing it long enough—and sticking with one project for sufficient time—to look back on how my younger self worked and be able to shake my head in judgment of that younger man.

But, professionally, others recognized my abilities as well (which did little to help me develop humility).

Over the decades, my perspective on what software development—especially *good* software development—was evolved.

1.6: How I Saw Coding as a Child

This is barely worth mentioning, because I'm pretty sure this is just the description of how any child sees anything that interests him.

When I was a child and I first started to get involved with programming, I saw it as a plaything.

It was a source of mysteries to be unraveled with the potential to serve as a creative outlet.

I had been playing games for a little while and I wanted to make my own. I'm not the first developer with that as the initiating force in his backstory, nor am I the last.

Nevertheless, I had too little understanding of what programming (let alone software *development*) was to have an opinion on what was good and what was bad.

1.7: How I Saw Coding as an Adolescent

As a budding youth, I saw coding as one big exercise in mathematics. The goal was to get the *right* answer.

There may have been a sense of craftsmanship but only in so far as it served in the pursuit of perfection.

Mathematics was appealing to people like me because there were correct answers. It's one of the only spaces where that is true in its absolute sense.

My idea of what good coding was descended directly from this fact.

There's just one problem. In mathematics, each proof, each theorem is correct because we **make** it correct.

Software is too connected to the real world for us to have that kind of stipulative power.

1.8: How I Saw Coding as a Young Man

Upon leaving college and joining a firm called Rogue Wave Software, I was presented with a new force: Someone else was paying me to code.

These people weren't really interested in perfection. Instead, they were mostly interested in these annoying things called "problems".

Customer problems served as a crucible in which my actions were held for judgment.

The real world had come calling and the idea of frittering away who-knows-how-long on who-knows-what-but-it's-beautiful withered in the fires of that judgment.

So my new perspective quickly became something like "good coding is solving problems efficiently".

Yet "efficiently" did not really have a universally-accepted definition for software development back then. Honestly, it might not even today.

My previous experience and my instincts told me that solving problems with quality was essential to efficiency. Intuitively, I understood that quality code was easier to change later.

Mentors and collaborators along the way also knew this and helped me learn to externalize this idea.

In the years that followed, something happened: A lot of developers kept doing things the hard way. People would cut corners, knowing it was going to cost more in the long run.

1.9: How I Saw Coding as an *Angry* Young Man

As someone to whom interpersonal interaction never came easily, I did what was obvious. I got angry.

People were being obstinate. I told them what to do. Luminaries far above me and widely respected told them what to do.

But they weren't doing it.

They weren't stupid people. So malice (which I called "laziness" at the time) was the only explanation I could imagine.

So I blustered. I castigated. I whipped my prowess out, slapped it on the desk, and dared anyone I could to go get a measuring tape.

For some reason, this didn't work any better than telling people nicely.

Go figure.

1.10: How I Saw Coding as a New Coach

Despite my many, many flaws, a relatively elite consulting firm ended up hiring me as a technical coach.

They took me under their wing. They helped me learn how to interact with people. They taught me that it's not my responsibility if someone can't or won't adopt a skill.

They taught me to **accept** how people grow and change.

They helped me grow as a coach and as a team member.

Through all that, though, they didn't really change my concept of what good software development was.

I still saw the fundamental problem of being a software developer to be solving problems with high quality, so you can keep solving them rapidly later.

1.11: How I Saw Coding as an Established Coach

After a while, I started to see things a little differently. Once I had enough samples, the pattern became clear to me.

People in the top 1% often...

- ...are very smart, but they're not all geniuses.
- ...have some kind of authority within their organizations, but not always.
- ...adhere to best practices, but some of them are basically just cowboys.
- ...study deeply, but not always.
- ...have a mastery of multiple technologies, but some of them hyperfocus on a single stack.

They can work differently, talk differently, act differently. They even wake and sleep differently.

1.12: So What is It?

The one thing I noticed in every developer I've met who consistently made highly-visible impact was this:

They all were willing to tackle the hard stuff.

They weren't afraid of any code, or, if they were, they didn't let it stop them.

They have a pattern of seeing that something needs to change, **and then making that change**. Often, this carries significant risk so they come up with strategies to mitigate that risk.

Some of them work in ways so careful, that they prevent those risks from manifesting at all.

Yet all of them make the hard changes and make them work.

1.13: The Top 1%

Throughout the remainder of this Part, I will clearly define what it means to be in the top 1%, what doesn't work, and what holds people back.

I'll also establish fear of short-term consequences as a major player in what keeps people out of the 1%.

For now, let me establish this definition:

Top 1% Developer

> Someone who consistently solves the problems the rest of a team is unwilling to tackle.

In other words, they aren't magicians or savants—at least they don't have to be. They're just the people who keep rushing in where others fear to go.

By the end of the book, not only will you know what it takes to be one of them, you'll know what you need to do to get there.

Chapter 2: Change

> We're all adrift on the choppy seas of the market.

Mankind craves stasis. We want a world that stays put.

There's just one problem: It doesn't exist. Even in fiction, creating stasis usually involves stopping time itself.

Change is so intrinsic to our existence that suspension of disbelief is **easier for stopping time** than it is for eliminating change.

2.1: Early Hints

Around the turn of the century, coaches would often use this question:

> If you had to choose between writing new code and integrating what you wrote with existing code, what would be your preference?

Usually the goal was to help the person being asked realize that they needed to do something *now* in order to make integrating new functionality easier *later*.

Nowadays, I don't ask that question. I'm not sure if others do, but I definitely don't.

Ask yourself this question instead: How much time do you actually spend "writing new software" or "writing a new function"?

For me, it's virtually nil. Almost all my time is spent changing something that I've already created. The vast majority of that is spent changing something I've already changed a few times before, too.

I bet, for you, it's probably about the same. Maybe you spend some of your time making shiny new things and then some of it integrating those new things into existing code, but I bet you still spend more of your time on the integration than on the creation.

If not, this might not be the right book for you. Also, in that case, please reach out to me and tell me how I can get into the mystical Garden where you probably reside.

2.2: The Mythical Software Project

As I said, despite the fact that it's an impossibility, people crave stasis.

The very idea of a "software project" is based on the notion of a sort of "plastic stasis": Things are the way they are until we change them and then they are—and will stay—the new way.

The goal of "software project management" was to treat software like something that was built... the same as a bridge or a building.

Yet pursuit of this goal led to repeated and massive failure.

2.3: Blaming Complexity

So people started to wonder why making software wasn't working.

One factor that was identified was "requirements uncertainty".

"It's impossible to know everything you need," they'd say, "because software is so abstract." Or, sometimes, "...because software is too complex."

This is tantamount to saying "People can't come up with big plans and make them happen."

Yet as I write this, I am sitting in my home in a master planned community. I push data to my git repo on a route that sometimes includes a satellite, which was lifted into space by a rocket.

People devise and execute complex plans all the time. The result is, quite literally, history.

2.4: Consider This

Try this thought experiment.

Imagine the best product manager you've ever known. Heck... imagine the best product manager of whom you can even conceive.

To me, this product manager has a superlative ability to decide what is "in" and what is "out". He does a good job of making decisions about what really deserves to be in a product and what doesn't.

Now let's imagine a product. Maybe a drone courier service to connect businesses within local regions.

Can he make good, solid "in vs. out" decisions on scope?

My business partner, Luniel de Beer certainly would be able to do this. He could scope it out and have all the behaviors defined correctly to within 99.5% accuracy before development began.

A team could then consume a list of .feature files and just start making the software until they had implemented the solution.

There are other good reasons to **not** do things this way, but I know he **can**.

So I don't think "requirements uncertainty" or "complexity" is really the issue.

2.5: Consider a Little More

Now let's extend our thought experiment:

A competitor beat us to the punch and captured the Los Angeles market. It looks like they're going to be able to roll out to other major cities before us, too.

We were expecting to be able to sell into a large, dense environment with a relatively temperate climate. Now, we're faced with an uphill battle wherever we turn because they're going to have brand recognition even if we start serving a city they don't serve when our platform is ready.

To keep our venture afloat, we need to do something radical and the C Suite has decided to pivot from dense populations with mild climates to sparse populations with periods of hostile weather conditions. We'll make robust drones that travel over land and keep people connected even when they have high winds or are snowed in.

A "perfect plan" for the original goal is still a "perfect plan" for that goal, but it's no longer suited to the **new** goal.

This brings me back to the idea of "requirements uncertainty".

2.6: Requirements Uncertainty Revisited

It's not that people couldn't predict the requirements for what they wanted to build. Really, it's that people can't actually predict what they want to build.

If we could know—truly **know**—what would be needed in the future, we could easily plan out building it. We might still choose an Agile work-management strategy, but it wouldn't be a result of supposedly "not being able to plan ahead".

What people call "requirements uncertainty" is really just the uncertainty innate in our universe… Changing goals, shifting markets, emergent technologies, fluctuating conditions, and more all conspire to create choppy conditions for software development (or maybe even for product development in general).

Change is an external influence we can't control.

2.7: Impact on Software Developers

So, as a result, we spend most of our time as coders changing what we already made. It's okay. It turns out it works better in a lot of ways than the other would, if it were even possible.

...but make no mistake: The reason we take an Agile approach is that the other approach simply **does not work**. An iterative and incremental way of managing work does.

All other benefits are ancillary.

2.8: On Building

There are things that we make and let stand the test of time. Books are one example. For centuries, a book was written and released as is. Small changes were handled with errata sheets. Major revisions became new editions, which were effectively treated as books.

Bridges are another example. Sometimes new access points are added and maintenance is definitely needed to keep them from collapsing, but basically you build a bridge and there it is.

These are things that are "built" - we make them and then they are the way they are.

2.9: We Build Things that Don't Need to Change

Books are not built with change in mind because, generally, no *need* to change them. Bridges are the same.

A book is at once timely and timeless. When you read *The Space Trilogy* by C.S. Lewis, you don't think "Well that's not how it is!" at his description of the surface of other planets. At most, you think "Oh. Right. They didn't have the data on that, yet."

A bridge is there to help people cross some kind of obstacle and such obstacles seldom change. In the case of a river that might meander, efforts are made to control or absorb that change by either forcefully channeling the river along a certain path or putting the entry and exit both out of the range of the river's wandering.

When we build something, we build it to stand the test of time because it is solving a problem or addressing a need that basically does not shift or evolve.

Now I'd like to look at coding by comparison.

The Challenge of Learning to Code

At the time of this writing, I am currently going through the process of teaching my children to code. The contrast between basically linear natural language and a highly-structured programming language is actually big stumbling block.

This serves as a reminder for me that natural human languages and the mathematical constructs we call "programming languages" are extremely different.

Natural language is extremely linear. It's an excellent way of conveying a thought.

The thought doesn't need to be reshaped later. It just is exported from one mind, digested by another, and (hopefully) understood.

If there was no need to change software, source code would probably read a lot more like prose.

We would have found easier and easier ways to describe what someone wanted until it was no longer difficult for a child to learn.

As a result, computer languages would be primarily linear. They would be ways of people to export thoughts from their heads and translate them into instructions.

In fact, if you look at very primitive ways of programming (and the programs that were written that way), you'll see that this is how things used to be.

But nobody really uses those tools anymore.

2.10: Modern Programming Languages

Programming languages quickly evolved to have rigid *structure* - something that you can easily, predictably, and repeatably attach behavior and descriptions.

An emphasis is often placed on "understanding" code. Many people claim one of the primary aspects of good structure is that it allows you to understand code.

Indeed it does...but why is it that you would need to understand code?

It's always to change it. Either to change the existing code into something else or to write a replacement for the existing code. Both are changes to the system in which a piece of software lives.

2.11: Change as the Primary Factor

Change is the primary force driving all things in software development.

So I believe it should be the main factor to which we attend.

I'm far from alone. There have been many attempts to address change as a force.

In the early days, as we just saw, people devised programming languages themselves as a way of making software easier to change.

After that came a number of other innovations. To name a few:

- Object-Oriented Programming
- Design Patterns
- DRY and SOLID principles
- Test-Driven Development
- Domain Driven Design

Software developers have been building tooling and methodologies to support change since practically the beginning of software. We've been doing it since long before the business and process people caught up and realized they needed to actually *manage* change.

...but there's been something missing the whole time.

Chapter 3: What's Missing

When the only tool you have is a hammer...

So what is it that is missing from software development that makes changing software still so difficult after all these years and developments?

Let's analyze some of the most effective things that have been tried.

Of course, I'll recognize the impact of each, but I also want to help you see a pattern. It's the negative of that pattern that tells us what is missing.

3.1: Programming Languages

Early and modern programming languages have, as I said in Chapter 2, trended toward ever-increasing powers of change.

The code we wrote in C was harder to change than what was expressed in (well-written) C++. The code in C# or Java is easier to change than the C++ that was available at the turn of the century[1].

Good languages let you write code that is easier to change later.

The better the language, the more support it gives in this way.

[1]Modern C++ is extremely powerful and expressive. I can't rightly say modern C# and Java are more or less powerful than it. But that really only strengthens the point of this paragraph.

3.2: Parametric Polymorphism

Parametric polymorphism was another advancement in software development.

The fancy-schmancy academic term for things like C++ templates or what C# generics is "parametric polymorphism".

This advancement also allows one to write code that isn't coupled to concrete dependencies. It only really cares what it can ask that other thing to do.

Consequently, a separation is forced between the client and the provider entities. The client can't make assumptions about what happens inside and the provider can't make assumptions about how it will be used.

Good use of templates/generics lets you set up your code so that it's easy to make a single change in the right place when you need to later.

The technology has only advanced from there, with things like type constraints and specializations allowing for an extremely rich solution to some very hard problems.

3.3: Object-Oriented Programming (OOP)

The advent of object-oriented programming gave us mental models and code structures that made it easier to understand and organize our code.

When the time came to change it, we[2] often knew where to look.

It also provided some limited degree of reuse. This meant that we didn't have as many cases of needing to repeatedly make the same change in multiple places.

As a result, if you structured your code according to good OOP principles at the beginning, you could change it more easily later.

The better you were with object-oriented programming, the easier it was to change a system down the line.

[2] I say "we", but I'm of a generation that was born after OOP was commonplace. I mean the royal "we", I suppose.

3.4: Design Patterns

While the Gang of Four claims to have merely measured (rather than invented) design patterns, they certainly did their part to make them a part of how we write software: Design Patterns aren't fully adopted to this day, but they keep gaining traction.

Design Patterns allow developers to invert dependencies and encapsulate variation **regardless of the language they are using**.

Design Patterns give you a completely different relationship with change. A properly recognized and implemented pattern makes it so that you can change behavior by addition instead of modification.

If you find all the Design Patterns in your problem-space and implement them, it is much easier to swap new functionality in later.

3.5: Principles and Slogans

As we learned more about software design, people began to distill the principles into slogans.

I don't mean to disparage the slogans. Even a cursory glance at history reveals slogans (and propaganda in general) to be a necessary component of any kind of transition within a population.

Three big ones float around our industry, doing good wherever they touch a developer's mind:

- DRY - **D**on't **R**epeat **Y**ourself
- SOLID - 5 principles: **S**ingle-Responsibility, **O**pen-Closed, **L**iskov Substitution, **I**nterface Segregation, **D**ependency Inversion
- KISS - **K**eep **I**t **S**imple, **S**tupid

If you understand these principles and adhere to them, you will write code that is **much** easier to change later.

3.6: Test-Driven Development (TDD)

Test-First Programming and, later, Test-Driven Development have revolutionized how software is made.

You start by defining a specification in the form of a test. Often this test doesn't even compile and those compiler errors tell you what about your design is required by the spec.

At some point, you get the test to the point of failing—that tells you the software doesn't already do what was specified—and you make the changes required to get it to pass.

As a result, you end up with every important behavior in the system specified by tests. Hopefully, there are very few behaviors that aren't specified by tests, if any at all. In addition, you end up with a much better design than you often would without it.

When you write code with TDD, that code is so much easier to change later that it's a wonder anyone builds anything without this discipline.

3.7: Did You See The Pattern?

These are the shifts with the largest impact I can see and they all have something in common.

Did you see it? In case you didn't, I'll capture the most important line from each section and change the emphasis to highlight the commonality.

> Good languages let you write code that is easier to change **later**.
>
> Good use of templates/generics lets you set up your code so that it's easy to make a single change in the right place when you need to **later**.
>
> ...if you structured your code according to good OOP principles at the beginning, you could change it more easily **later**.

> If you find all the Design Patterns in your problem-space and implement them, it is much easier to swap new functionality in **later**.
>
> If you understand these principles and adhere to them, you will write code that is much easier to change **later**.
>
> When you write code with TDD, that code is so much easier to change **later** that it's a wonder anyone builds anything without this discipline.

All the major, widely-adopted approaches to change involves preparing yourself to withstand change in the future. They treat change as something to survive, rather than master.

Perhaps this echoes our instinct to do things like build shelters to protect us from weather.

Change is recognized by all the aforementioned things, but it's something you have to (ironically) plan for. That means that, when change happens, it was **your approach in the past** that mattered.

That's not to say that **nobody** made an effort to impact how people make changes, though.

3.8: Other Innovations

There are a few things that I left out of the previous sections. I think these discoveries are major for our industry and I don't believe they've received the attention they deserve.

One such innovation is refactoring.

While Martin Fowler did not invent it in his book, *Refactoring*, his position as an authority on the subject and one of its primary advocates cannot be denied.

Yet how many people have apocryphalized the word "refactoring"? How often does someone "refactor" in a manner consistent with what he wrote?

I'm not saying it never happens, but it's a marginal fraction of the time in my experience. Most people who "refactor" are really just changing design willy-nilly and hoping for the best.

Another work of great import is *Working Effectively with Legacy Code* by Michael Feathers. This book gives deep insights into how to restore legacy code to modern status. Or, at least, how to maintain and extend it without having to succumb to the temptation of a rewrite.

3.9: Test-Obsession

Both *Refactoring* and *Working Effectively with Legacy Code* are focused on change, but they seem preoccupied with test coverage, to me.

In *Refactoring*, Fowler says that the first thing he needs to refactor is tests.

Feathers takes the less-strenuous position of defining legacy code as code not covered in automated tests.

Now, I am a huge proponent of TDD and test automation in general. So I don't want this section to be taken as resistance to testing.

But I do think that the fact that the two greatest works we have on **changing** code seem test-focused tells us a few things:

1. Automated tests are really, really important. Seriously: If you're not practicing TDD, start.
2. Even the few attempts to focus on change in the now are somewhat focused on the past… like an attempt to time-travel back to when the tests weren't written and retrofit them to support current changes.

This focus on testing not only tells us something about what's missing in the software industry, it presents a material obstacle to refurbishment of old code.

On Testability

The sad reality is that, since many software developers didn't and still do not practice TDD, a lot of code isn't tested.

Worse, much of the code out there is untestable **as it is written today**.

That qualification is very important. It's not that the problem being solved is untestable. It's not that the code can't be changed to make it testable. It's that the current system as written is either impossible or prohibitively expensive to cover in automated tests.

This is a key impediment, but not directly on the path of the point I'm trying to make in this chapter.

3.10: It's Really about Focus

That focus on "make it like it should have been" really speaks to the mindset of the industry as it has been for my entire career.

The ambient message of the elite appears to have been:

> Do it right, today, and you can change it tomorrow.

That focus on supporting future change is not *wrong*. It's an important aspect of software development.

But it's only half of the story.

3.11: The Other Half of Coding

The other half is a focus on how you change software. That is, the how of the change itself. I believe that Feathers and Fowler both made a serious effort to

move people toward more discipline in this regard. The other luminaries no doubt had their impact as well.

Yet, it wasn't explicitly called out. This is what we do *most of the time*, but it is still treated as a natural consequence of decisions made in the past, rather than given the kind of explicit treatment it deserves.

There are a few books, but it doesn't seem like colleges or bootcamps really spend any energy on this. You have to figure it out for yourself and you may not even think about it directly. For most, it appears to be an accumulation of intuitions that are never formalized or made explicit.

`the other half of coding` is to shift your mindset from *what* you want to code to *how* you want to get there.

We've spent half a century building systems to make change possible later, now it's time to learn how to make change safe *now*.

To really understand it, though, you'll need a clear picture of what is in your way.

Part II: What Stands in Our Way

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Chapter 4: The Root of Decay

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

4.1: Can't Understand It

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

4.2: Don't Know How it Should Work

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

4.3: Messiness and Unreadability

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

4.4: Too Much Coupling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

4.5: Structures are Too Large

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

4.6: No Tests

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

4.7: Reasons for Lack of Tests

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

4.8: No Time to Write Tests

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

4.9: Not Feasible to Write Tests

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

4.10: No Time to Write Tests Revisited

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

4.11: What about the Other Problems?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Chapter 5: Why Designs Rot

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

5.1: Code Can't Change Itself

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

5.2: All Software Work is Design Work

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

5.3: So, Back to Why People Don't Change It?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

5.4: What Aspects People Tend to Freeze

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

5.5: What's Really Happening

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

5.6: The Engine of Decay

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

5.7: We Know It in our Hearts

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

5.8: For the Most Basic of Reasons

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

5.9: Are We Stuck?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

5.10: Change Never Changes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

5.11: Risk Is Different

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

5.12: The Key

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Chapter 6: Decline

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

6.1: The Rotting Agent of Rot

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

6.2: Rejecting Inherited Wisdom

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

6.3: Terminological Decline

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

6.4: What is Design

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

6.5: What is Refactoring?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

6.6: Vibe Coding

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

6.7: Legacy Systems

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

6.8: Connecting the Dots

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Chapter 7: Powerful Tools Can Become Crutches

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

7.1: What Tests Do

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

7.2: What Tests Don't Do

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

7.3: What Tests *Can't* Do

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

7.4: What Good Design Does

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

7.5: What Good Design Can't Do

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

7.6: What Design Oughtn't Do

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

7.7: Pair Programming or Mob Programming

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

7.8: Requirements Analysis, Especially Behavior-Driven Development

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

7.9: Useful, All; Sufficient, None

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Part III: The Solution

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Chapter 8: A New Ontology

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

8.1: Change as a First-Class Citizen

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

8.2: Design-Focused

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

8.3: Creating Safety

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

8.4: Preserving Continuity

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

8.5: Reversibility

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

8.6: Goal-Seeking

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

8.7: The New Philosophy

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Chapter 9: Related Works

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

9.1: *Refactoring*

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

9.2: *Working Effectively with Legacy Code*

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

9.3: *Design Patterns*

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

9.4: *Test-Driven Development*

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

9.5: Transformation Priority Premise

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

9.6: Addition, Not Replacement

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

9.7: On the Shoulders of Giants

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Chapter 10: Principle: Design-First

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

10.1: Structure Before Function

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

10.2: Levels of Design

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

10.3: Structure of Algorithms

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

10.4: An Example

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

10.5: Call Graphs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

10.6: Higher Levels

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

10.7: As Above, So Below

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

10.8: Physics is Governed by the Low

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

10.9: The Principle

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Chapter 11: Principle: Pathfinding

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

11.1: The Destination

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

11.2: An Example of Bad Design

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

11.3: An Example of Good Design

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

11.4: The ~~Destination~~ Destination(s)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

11.5: Shifting Focus

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

11.6: Navigating Difficult Terrain

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

11.7: Navigating Difficult Code

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

11.8: Pursuit, not Approach

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Chapter 12: Principle: Transformation of Code

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

12.1: The Sketch Artist

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

12.2: An Example

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

12.3: A Proposed Change

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

12.4: Ctrl+F

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

12.5: Compiler-Driven Development

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

12.6: Just Leave The Old Design In Place?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

12.7: Little by Little

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

12.8: Transformative Change

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

12.9: The Quantum Barrier

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

12.10: Spanning the Gap

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

12.11: Proof

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Chapter 13: Principle: Abort & Retry

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

13.1: Why Tests Help

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

13.2: Commitment

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

13.3: Freedom

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

13.4: Extracting a Principle

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

13.5: Necessity

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

13.6: Loops in Loops

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

13.7: Effect on Development

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Chapter 14: Flow

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

14.1: Creative Flow

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

14.2: The Flow Requires a Plan

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

14.3: Decomposition

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

14.4: Recursion

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

14.5: Interruption

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

14.6: Switching Modes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

14.7: One Process with Two Uses

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

14.8: Setting a Goal

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

14.9: Planning the Transformation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

14.10: Recursive Breakdown

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

14.11: Bottom-Up Changes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

14.12: Accumulation of the Small Into the Large

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

14.13: Disruptions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

14.14: Triggers and Responses

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

14.15: Going Back Isn't Failure

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

14.16: Reaching the Goal

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

14.17: A Coherent Mindset

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Part IV: Skills

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Chapter 15: My Little Puzzle

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

15.1: Refactoring

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

15.2: Levels of Difficulty

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

15.3: Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

15.4: Gameplay

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

15.5: Transformative Flow in the Game

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

15.6: Complexity

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

15.7: Why It's in This Book

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Chapter 16: Recursive Disaggregation of Changes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

16.1: In *Refactronica*

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

16.2: Breakdown

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

16.3: The Basic Structure

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

16.4: How Far Is Far Enough?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

16.5: How to Disaggregate

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

16.6: Why It's Important

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

16.7: How It Fits In

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

16.8: Intrinsic to Refactoring

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

16.9: Applying to Refactoring

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

16.10: Example of Applying Disaggregation to Refactoring

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

16.11: The Goal

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

16.12: The First Step

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

16.13: Continuing the Continuity

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

16.14: Continuing the Continuation of the Continuity

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

16.15: Popping Frames

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

16.16: A Red Flag

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

16.17: As in the Large, So in the Small

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Chapter 17: The Power of Duplication

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

17.1: "...But Isn't Duplication Bad?"

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

17.2: Redundancy

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

17.3: It's Just a Tool

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

17.4: Bypasses

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

17.5: Support for Movement

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

17.6: An Example

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

17.7: The Transformation Connection

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

17.8: Adding It to Your Flow

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Chapter 18: Compounding Changes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

18.1: In *Refactronica*

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

18.2: Microcosms

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

18.3: Macrocosms

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

18.4: Continuous Flow

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

18.5: Revisiting Recursion

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

18.6: Recursive Composition

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

18.7: Planning as a Part of Pathfinding

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

18.8: Example

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

18.9: The First Step

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

18.10: Building Up

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

18.11: Compounding Further

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

18.12: Emerging Modules

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

18.13: Expanding Offerings

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

18.14: Scaling All the Way Up and All the Way Down

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Chapter 19: Puzzle-Solving

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

19.1: The Accidental Analogy

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

19.2: Constraints

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

19.3: Reasoning & Understanding

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

19.4: In Coding

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

19.5: Inspection

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

19.6: Reduction

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

19.7: Reduction in Coding

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

19.8: Hypothetical Reasoning

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

19.9: Refutation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

19.10: Confirmation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

19.11: Confirmation in Code

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

19.12: Is Confirmation Just The Refutation of Alternatives?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

19.13: Ordering

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

19.14: Lemmas

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

19.15: Lemmas in Software Development

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

19.16: The Logical Lemma Lexicon

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

19.17: Experimentation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

19.18: Integration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Part V: Mechanisms

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Chapter 20: Foundation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

20.1: A New Part and a New Path

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

20.2: The Treasures

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

20.3: How Each Chapter Works

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

20.4: Pouring the Foundation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Chapter 21: Refactoring

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

21.1: What Is Refactoring

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

21.2: Why Is Refactoring Important

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

21.3: The Book

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

21.4: The Formal Operations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

21.5: What the Operations Have in Common

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

21.6: Exploiting the Transitive Property

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

21.7: An Example Refactoring Operation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

21.8: Preserve Whole Object

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

21.9: Adding the New Parameter

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

21.10: Transferring Coupling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

21.11: Repeating for Each Parameter

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

21.12: How It Fits In

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

21.13: Refactoring to Drive Behavior

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

21.14: Further Study

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

21.15: Building on the Discipline

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Chapter 22: Clever Use of High-Bandwidth Refactoring Tools

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

22.1: Richer Use of Tooling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

22.2: Example

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

22.3: Testability

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

22.4: Extraction

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

22.5: Creating a New Object

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

22.6: Making the Method an Instance Method

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

22.7: Extracting an Interface

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

22.8: Externalizing Object Creation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

22.9: Changing Coupling to the Interface

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

22.10: Conditional on Tooling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

22.11: Further Study

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

22.12: Scaling All the Way Up and All the Way Down

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Chapter 23: Fastidious Application of Automated Transformation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

23.1: An Unforeseen Challenge

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

23.2: A Positive Example

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

23.3: An Illustration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

23.4: The Non-Transformative Option

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

23.5: The Transformative Option

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

23.6: Pinning Invariants

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

23.7: Continuing Onward

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

23.8: How it Fits & Why It Matters

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

23.9: Further Study

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

23.10: Powerful Tooling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

23.11: Tooling Is Everywhere

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

23.12: Reminder: Not Just One Tool Set

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

23.13: Use It Or Lose It

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Chapter 24: Careful Misuse of Low-Bandwidth Tools

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

24.1: Bailing Wire and Chewing Gum

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

24.2: Change is Ephemeral

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

24.3: The General Pattern

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

24.4: “Strong” vs. “Weak”

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

24.5: An Example

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

24.6: Some Extractions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

24.7: Verified Duplication

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

24.8: A Safe Replacement

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

24.9: Other IDEs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

24.10: The Importance of Flexibility

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

24.11: On Breadth & Depth

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Chapter 25: Exploitation of Language Features

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

25.1: Nobody Likes Transients

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

25.2: Balanced Imperfection

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

25.3: Special Strengths

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

25.4: How It Fits & Why It's Important

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

25.5: Example

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

25.6: Using Language Strength to Balance Tool Weakness

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

25.7: Enlisting the Arguments

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

25.8: Removing the Parameter

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

25.9: The Basic Structure

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

25.10: Universally Applicable

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

25.11: Further Study

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

25.12: From the Specific to the General

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Chapter 26: Using TDD to Prove Interchangeability

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

26.1: Using Tests to Support Transformation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

26.2: Pinning Tests

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

26.3: Abstraction Motivation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

26.4: A Problem

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

26.5: A Partial Solution

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

26.6: Interchangeability Tests

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

26.7: Example

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

26.8: Caveat: Complexity

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

26.9: Have You Heard the Good News About Complexity?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

26.10: How It Fits & Why It's Important

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

26.11: Up Next

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Chapter 27: Favoring Transformation Priorities

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

27.1: Different Possibilities

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

27.2: A Deep Look

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

27.3: Transformations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

27.4: Prioritization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

27.5: Modifications

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

27.6: Example

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

27.7: One Option

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

27.8: The TPP Angle

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

27.9: The Invariants Angle

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

27.10: Why It's Important

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

27.11: How It Fits Into Transformative Flow

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

27.12: Further Study

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

27.13: Constraining Motion

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Chapter 28: Using TDD to Constrain Behavioral Changes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

28.1: The Fountainhead

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

28.2: How Much TDD This Book Covers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

28.3: "Red-Green-Refactor"

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

28.4: Tests as Specifications

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

28.5: Tests Specify Behavior

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

28.6: Reframing “Red-Green-Refactor”

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

28.7: Test-Driven Development Example

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

28.8: The First Test

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

28.9: Completing the Boundary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

28.10: Constrained Motion

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

28.11: Caveat: Legacy Code

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

28.12: Mitigation: “Islands” of TDD

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

28.13: Wrapping up *Mechanisms*

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Part VI: Context

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Chapter 29: Patience

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

29.1: An Analogy

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

29.2: Pace

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

29.3: Corner-Cutting

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

29.4: An Example of Corner-Cutting

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

29.5: Rushing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

29.6: When You Rush, That's When Accidents Happen

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

29.7: Speed

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

29.8: Patience of the Right Kind

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Chapter 30: Requirements Maturation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

30.1: “It’s Just a Signal, People!”

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

30.2: Permission to Succeed

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

30.3: Jumping the Gun

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

30.4: Principles of RMF

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

30.5: Further Reading

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

30.6: Shelter for a Budding Discipline

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Chapter 31: The Other Half of Patience

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

31.1: Weightlifting

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

31.2: Frustration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

31.3: From Goals to Growth

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

31.4: Do the Work

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Chapter 32: Rewiring your Habits with Deliberate Undos

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

32.1: “Make it Again!”

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

32.2: Retraining Your Habits

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

32.3: Pick a Thing to Learn

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

32.4: The Retraining Trigger

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

32.5: Surprisingly Effective

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

32.6: Example

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

32.7: The First Incident

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

32.8: The Trigger

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

32.9: The Retraining

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

32.10: The Tradeoff

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Chapter 33: Investment in Practice

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

33.1: “It Costs too Much”

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

33.2: Misattribution

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

33.3: Decoupling Costs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

33.4: Martial Arts

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

33.5: In Software

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

33.6: On the Job

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

33.7: On the Sly

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

33.8: Occasional Pilot Items

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

33.9: On Your Own Time

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

33.10: Dedicated Sabbatical

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

33.11: How to Choose

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

33.12: Make the Investment

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Chapter 34: It's Up to You

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Backmatter

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Appendix A: Search Page Code

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Appendix B: Adoption Reference

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Appendix C: Getting Others Onboard

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Glossary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.

Bibliography

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/other-half.