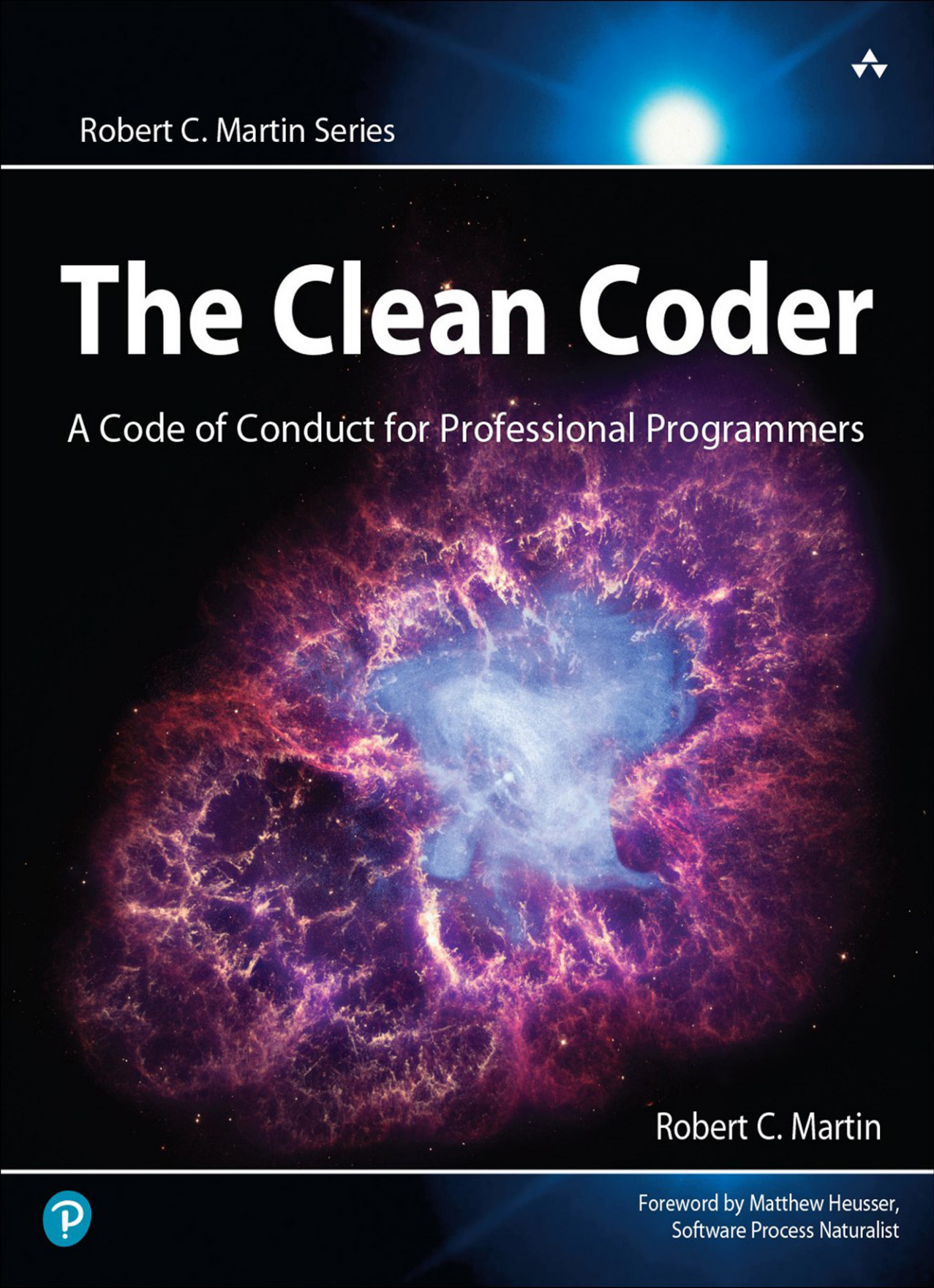
Robert C. Martin Series
The Clean Coder
A Code of Conduct for Professional Programmers
Robert C. Martin
Foreword by Matthew Heusser,
Software Process Naturalist

Praise for *The Clean Coder*

"'Uncle Bob' Martin definitely raises the bar with his latest book. He explains his expectation for a professional programmer on management interactions, time management, pressure, on collaboration, and on the choice of tools to use. Beyond TDD and ATDD, Martin explains what every programmer who considers him- or herself a professional not only needs to know, but also needs to follow in order to make the young profession of software development grow."

—Markus Gärtner
Senior Software Developer
it-agile GmbH
www.it-agile.de
www.shino.de

"Some technical books inspire and teach; some delight and amuse. Rarely does a technical book do all four of these things. Robert Martin's always have for me and *The Clean Coder* is no exception. Read, learn, and live the lessons in this book and you can accurately call yourself a software professional."

—George Bullock
Senior Program Manager
Microsoft Corp.

"If a computer science degree had 'required reading for after you graduate,' this would be it. In the real world, your bad code doesn't vanish when the semester's over, you don't get an A for marathon coding the night before an assignment's due, and, worst of all, you have to deal with people. So, coding gurus are not necessarily professionals. *The Clean Coder* describes the journey to professionalism . . . and it does a remarkably entertaining job of it."

—Jeff Overbey
University of Illinois at Urbana-Champaign

"*The Clean Coder* is much more than a set of rules or guidelines. It contains hard-earned wisdom and knowledge that is normally obtained through many years of trial and error or by working as an apprentice to a master craftsman. If you call yourself a software professional, you need this book."

—R. L. Bogetti
Lead System Designer
Baxter Healthcare
www.RLBogetti.com

- **Add features by faulting the team for not recognizing their necessity.** In this case, the hardcoded content was going to require app updates to change. How could I not realize that? I did, but I'd been handed a false promise earlier, that's why. Or a client will hire "a new guy" who's recognized there is some obvious omission. One day a client will say they just hired Steve Jobs and can we add alchemy to the app? Then they'll …
- **Push the deadline. Over and over.** Developers work their fastest and hardest (and BTW are at their most error prone, but who cares about that, right?) with a couple days to go on a deadline. Why tell them you can push the date out further while they're being so productive? Take advantage of it! And so it goes, a few days are added, a week is added, just when you had worked a 20-hour shift to get everything just right. It's like a donkey and carrot, except you're not treated as well as the donkey.

It's a brilliant playbook. Can you blame them for thinking it works? But they don't see the God-awful code. And so it happens, time and again, despite the results.

In a globalized economy, where corporations are held to the almighty dollar and raising the stock price involves layoffs, overworked staffs, and offshoring, this strategy I've shown you of cutting developer costs is making good code obsolete. As developers, we're going to be asked/told/conned into writing twice the code in half the time if we're not careful.

Code Impossible

In the story when John asks "Is good code impossible?", he is really asking "Is professionalism impossible?" After all, it wasn't just the code that suffered in his tale of dysfunction. It was his family, his employer, his customer, and the users. *Everybody* lost[3] in this adventure. And they lost due to unprofessionalism.

So who was acting unprofessionally? John makes it clear that he thinks it was the executives at Gorilla Mart. After all, his playbook was a pretty clear indictment of their bad behavior. But *was* their behavior bad? I don't think so.

3. With the possible exception of John's direct employer, though I'd bet they lost too.

The folks at Gorilla Mart wanted the option to have an iPhone app on Black Friday. They were willing to pay to have that option. They found someone willing to provide that option. So how can you fault them?

Yes, it's true, there were some communications failures. Apparently the executives didn't know what a web service really was, and there were all the normal issues of one part of a big corporation not knowing what another part is doing. But all that should have been expected. John even admits as much when he says: "Despite years of constant reminders that every feature a client asks for will always be more complex to write than it is to explain . . ."

So if the culprit was not Gorilla Mart, then who?

Perhaps it was John's direct employer. John didn't say this explicitly, but there was a hint when he said, parenthetically, "In business, client importance matters." So did John's employer make unreasonable promises to Gorilla Mart? Did they put pressure on John, directly or indirectly, to make those promises come true? John doesn't say this, so we can only wonder.

Even so, where is John's responsibility in all of this? I put the fault squarely on John. John is the one who accepted the initial two-week deadline, knowing full well that projects are usually more complex than they sound. John is the one who accepted the need to write the PHP server. John is the one who accepted the email registration, and the coupon expiration. John is the one who worked 20-hour days and 90-hour weeks. John is the one who subtracted himself from his family and his life to make this deadline.

And why did John do this? He tells us in no uncertain terms: "I hit Send, lay back in my chair, and with a smug grin began to fantasize how the company would run me up onto their shoulders and lead a procession down 42nd Street while I was crowned "Greatest Developer Ev-ar." In short, John was trying to be a hero. He saw his chance for accolades, and he went for it. He leaned over and grabbed for the brass ring.

Professionals are often heroes, but not because they try to be. Professionals become heroes when they get a job done well, on time, and on budget. By trying to become the man of the hour, the savior of the day, John was not acting like a professional.

John should have said no to the original two-week deadline. Or if not, then he should have said no when he found there was no web service. He should have said no to the request for email registration and coupon expiration. He should have said no to anything that would require horrific overtime and sacrifice.

But most of all, John should have said no to his own internal decision that the only way to get this job done on time was to make a big mess. Notice what John said about good code and unit tests:

"To make up for the extra work, the coding will have to go a little faster. Forget that abstract factory. Use a big fat for loop instead of the composite, there's no time!"

And again:

"I spent those eight days writing code in a fury. I used all the tools available to me to get it done: copy-and-paste (AKA reusable code), magic numbers (avoiding the duplication of defining constants and then, gasp!, retyping them), and absolutely NO unit tests! (Who needs red bars at a time like this, it'd just demotivate me!)"

Saying yes to those decisions was the real crux of the failure. John accepted that the only way to succeed was to behave unprofessionally, so he reaped the appropriate reward.

That may sound harsh. It's not intended that way. In previous chapters I described how I've made the same mistake in my career, more than once. The temptation to be a hero and "solve the problem" is huge. What we all have to realize is that saying yes to dropping our professional disciplines is *not* the way to solve problems. Dropping those disciplines is the way you create problems.

With that, I can finally answer John's initial question:

"Is good code impossible? Is professionalism impossible?"

Answer: I say *no*.

This page intentionally left blank

3 Saying Yes

Did you know that I invented voice mail? It's true. Actually there were three of us who held the patent for voice mail. Ken Finder, Jerry Fitzpatrick, and I. It was in the very early 80s, and we worked for a company named Teradyne. Our CEO had commissioned us to come up with a new kind of product, and we invented "The Electronic Receptionist," or ER for short.

You all know what ER is. ER is one of those horrible machines that answers the phone at companies and asks you all kinds of brain-dead questions that you need to answer by pressing buttons. ("For English, press 1.")

Our ER would answer the phone for a company and ask you to dial the name of the person you wanted. It would ask you to pronounce your name, and then it would call the person in question. It would announce the call and ask whether it should be accepted. If so, it would connect the call and drop off.

You could tell ER where you were. You could give it several phone numbers to try. So if you were in someone else's office, ER could find you. If you were at home, ER could find you. If you were in a different city, ER could find you. And, in the end, if ER could not find you, it would take a message. That's where the voice mail came in.

Oddly enough, Teradyne could not figure out how to sell ER. The project ran out of budget and was morphed into something we knew how to sell—CDS, The Craft Dispatch System, for dispatching telephone repairmen to their next job. And Teradyne also dropped the patent without telling us. (!) The current patent holder filed three months after we did. (!!)[1]

Long after the morphing of ER into CDS, but long before I found out that the patent had been dropped. I waited in a tree for the CEO of the company. We had a big oak tree outside the front of the building. I climbed it and waited for his Jaguar to pull in. I met him at the door and asked for a few minutes. He obliged.

I told him we really needed to start up the ER project again. I told him I was sure it could make money. He surprised me by saying, "OK Bob, work up a plan. Show me how I can make money. If you do, and I believe it, I'll start up ER again."

I hadn't expected that. I had expected him to say, "You're right Bob. I'm going to start that project up again, and I'm going to figure out how to make money at

1. Not that the patent was worth any money to me. I had sold it to Teradyne for $1, as per my employment contract (and I didn't get the dollar).