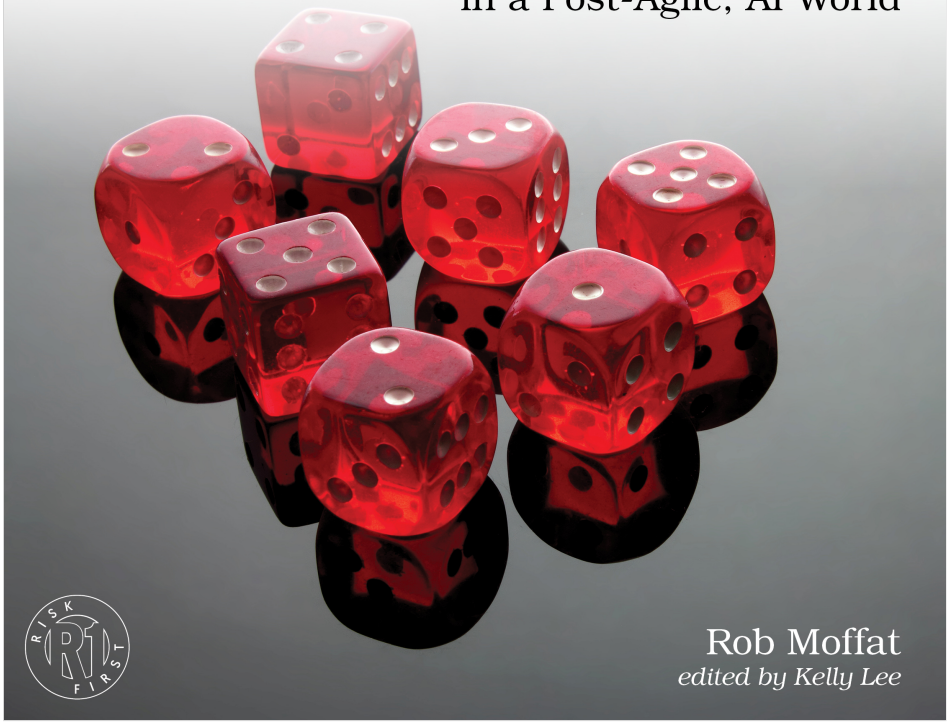
The Pragmatic Programmers

Risk-First Software Development

Second Edition

Deliver Better Systems in a Post-Agile, AI World



Rob Moffat

edited by Kelly Lee

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

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

Preface

If you've come looking to learn how compilers work, how to build three-tier apps, or how to optimise database design, you're in the wrong place. In today's open-source world, there are *millions* of technology choices and *tens of thousands* of books, guides, and reference manuals to help you get the most out of them. This book isn't one of those. But it *does* speak to the phenomenon behind all that choice - the growing complexity of the software development landscape - and to the deeper issues that make software development hard.

As software systems and the societies we are constructing with them grow more intricate, interconnected, and fast-moving, many software practitioners are coming to recognise that risk management isn't a side concern but *central* to what we do. Whether we're choosing a new framework, prioritising features, architecting a system, or responding to production issues, we're constantly navigating trade-offs under uncertainty. Yet, we rarely talk about our work this way and we don't explicitly talk about what it is we're trading off. This book aims to make that implicit reality explicit.

In fact, if you're already an experienced software developer, you may finish this book feeling like you haven't learnt anything new at all. If that happens, then both of us have done our jobs well. Chapters often begin with a potentially controversial idea, but the aim is that by the end, you're nodding along, thinking: *well yes, that's obviously what I'm doing anyway.* (That's the mark of my favourite tech talks - they reveal something we already know intuitively, but give us the language and structure to understand why it matters.)

This book is about making visible the behaviours and instincts we nurture and rely on as software developers and giving them names. Once we can name things, we can talk about them clearly. And clarity matters: if we can move from a vague discomfort - "I've got a bad feeling about this..." - to clearly articulating the underlying risks, trade-offs, and expectations, then we can make better, more collaborative decisions.

By the end of the book, it should be clear that risk management isn't just the future of software development - it's the fundamental human activity that has scaffolded the civilisation we live in today. And in a world of accelerating complexity and speed it is the skill most urgently in need of attention and development - a world increasingly shaped by the hands of software developers.