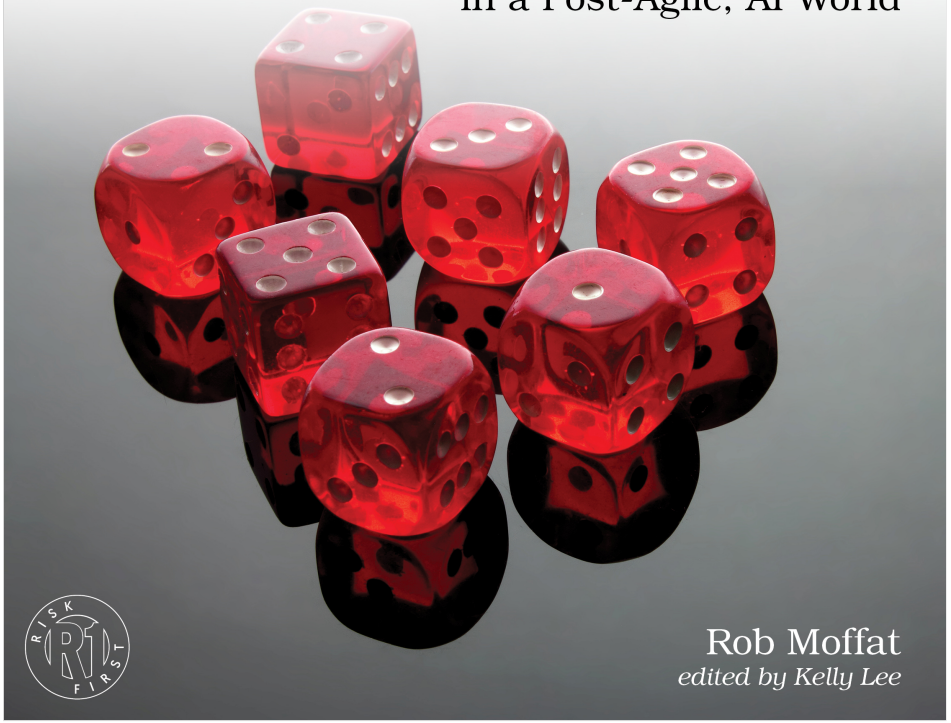
The Pragmatic Programmers

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.

Introduction

If you've ever felt that your project is going off the rails despite following all the "best practices," this book might be for you. It challenges the idea that success in software is down to choosing the right methodology. Instead, it offers a practical and powerful mindset: that software development is a continuous process of identifying, managing, and responding to risk.

As a reader, picking up this book for the first time, you'll likely find yourself in one of two camps. For some, this will be a new perspective on software development — perhaps unfamiliar, even a little suspect. Others who already operate with this mindset may be wondering if there's anything new here worth committing precious time to — hearing ideas you've embraced explained in a more formal, structured, or rigorous way is, after all, often both affirming and clarifying.

The job of this book is therefore two-fold: first, to justify this perspective for the uninitiated—to show how adopting the lens of risk explains why decisions are made, where projects tend to stumble, and how to deliver software more purposefully. And for the second camp, it aims to buttress the believers with new language, new techniques, and new clarity to make the effort of reading worthwhile.

This perspective is especially timely. The introduction of generative AI tools is reshaping the developer experience. While these tools can dramatically boost productivity, they also introduce novel risks: hallucinated code, overreliance on automation, gaps in explainability, and a shifting role for human judgment in design and review.

In a world of accelerating change, the ability to think clearly about risk might be the most important skill a software developer can possess.

Positioning Risk-First Software Development

Risk-First Software Development reframes software development as an exercise in managing uncertainty rather than merely executing process. Traditional

methodologies often treat risk as a peripheral concern—something to be noted in passing or tracked in spreadsheets. This book argues that risk is not peripheral; it is the core driver of every decision made during software creation.

By placing risk front and center, Risk-First provides a framework that transcends methodology and technology. It doesn't compete with Agile, Waterfall, or DevOps—instead, it explains them. Each is revealed to be a response to certain categories of risk. By understanding this, you can move beyond dogma and apply the right tools to the right problems, with full awareness of the trade-offs involved.

If you are a seasoned software professional, you will have seen this before and probably already see your job as a risk manager. Nevertheless, sometimes its important to step back and explain why we work in a certain way. This book is an attempt to do that.

Background to the First Edition - A Quick History

The first edition of Risk-First Software Development was written in 2019 and occurred due to a specific set of circumstances. I'd spent my career working in investment banking technology, specifically building systems for estimating risk — how much money a bank might lose on a given day. But, burnt out after many years of this, I switched gear and began working on an exciting new project with some talented co-workers in building chatbots and applications based on chat platforms (specifically, Microsoft Teams and Symphony). We were trying out Scrum as a way to build the software. I'd spent most of my career trying to adopt agile practices in my industry and over the 25 years since Kent Beck wrote *XP: Extreme Programming Explained [Bec00]*, momentum had been building.

The only problem was that I was building one application whilst everyone else was working on something completely different. Nevertheless, reporting lines being what they were, I attended the retrospectives and the stand-up meetings and tried to help out with story points.

It was very clear that this wasn't how things were supposed to work: the meaningless rituals and behaviours we were engaging in so that we could variously say "we're doing Scrum properly" or "I'm a Scrum-Master" or "Two years of experience managing Agile projects" (on a CV) were consuming time that could be better spent elsewhere. Worse than that, these practices amplified the divisions and tensions in the team and became a stick to beat people with: politics had taken over and instead of focusing on what was best

for the project, adherence to process became a tool for tracking obedience and delivering functionality to customers took a backseat.

This was a trying project and not because of Scrum: political problems required political solutions. Some of us were eventually able to "work the org chart" in our favour and escape into less divisive teams and deliver amazing products. Nevertheless, I am thankful for the experience since it allowed me to understand first-hand some of the rumblings I had observed on internet forums about the misapplication of Agile practices and how they were not the panacea that many consultancies and Agile thought-leaders claimed them to be.

Back To Risk

The irony is that in looking at these problems I realise that I had come back to risk. When I compared the tasks I was assigned to do with the ones I felt it was most critical to do, the underlying factor was the amount of risks those tasks addressed. Having spent a decade before writing risk systems for banks, I now realised that this was exactly what was missing from our approach to software development.

Obviously, I was not the only one. Other teams around me were beginning to talk in abstract terms about risk and de-risking. Risk Management has always been part of the project manager's lexicon and remit. The very start of *XP: Extreme Programming Explained [Bec00]* makes the point that "it's all about risk." Bubbling under the surface, this has always been there, but perhaps it doesn't sell training courses or workshops? Perhaps it isn't "cool" enough? It isn't really affected much by technology trends or the new shiny. Maybe it isn't glamorous enough? The idea of careful risk management runs counter to every Hollywood action movie plot (although see chapter 4 on page ? for an exception). It's the antithesis of the "rockstar programmer" ideal, but my thinking is that if you really want to be a rockstar programmer, building your risk management instinct is the way to get there. (As far as I know where "there" is, anyway.)

Introduction to the Second Edition

I'm writing this second edition in 2025, working for the Linux Foundation in an organisation called "FINOS". The goal of FINOS is to act as an open source software foundation for the financial services industry. The industry finds itself in an interesting place: on the one hand, the business of money is being re-thought by crypto-currencies and FinTech startups. The technology landscape is increasingly fast-moving and turbulent. But equally, the burden of regulation is greater than ever and consumes more and more of the budget.

There is a palpable excitement/fear around AI tools and how they might reshape not just the industry but society as a whole. Open Source offers an opportunity to mutualise risk across the industry. Unfortunately, there are lots of well-meaning regulations (e.g. insider dealing, customer confidentiality and data leakage prevention) which inadvertently make it very hard to engage firm-to-firm on public projects. But increasingly, the types of software project we see at FINOS are focused on risk management for one form of risk or other: cyber-security risks, AI risks or open source risks, for example. From this unique vantage point, the case for software development being all about risk management has never been stronger. So this second edition is able to look at how the finance industry, the software development community and society as a whole are reacting to these technological forces and comment on the increasing importance of risk management in dealing with them.

How This Book Is Structured

This book is structured in three parts. Part One guides you through a comprehensive exploration of the idea that risk is the central concern in software development. It begins by laying the conceptual groundwork, introducing risk as the lens through which all software development activity can be understood. The early chapters build a foundational language and toolkit: terms, diagrams, and ways of thinking that allow risk to be discussed explicitly and productively. You are gradually shown how risk already underpins development processes and how acknowledging that directly can lead to healthier projects and more deliberate decision-making. As it progresses, it shifts from theory, to tools, to application within the project team and wider organisation.

Part Two starts with examining how different methodologies (e.g. Agile, Waterfall or DevOps) are essentially collections of practices for managing various kinds of risk and looking at the importance of *patterns* as a way of classifying the different risks we see. From there, it delves into specific risk patterns you find in software projects: the risks inherent in building features, relying on dependencies, constructing models, and operating in broader legal and social environments. Each risk is framed as a pattern and comes with a spotter's guide to common causes of the risk and practical tools on how they can be mitigated or leveraged. The concept of a "risk pattern language" emerges here as a unifying structure.

In its final part, the book turns to the broader implications of a Risk-First mindset. It looks at how risk management is becoming the discipline for dealing with the increasing criticality and complexity of software-based systems in the modern world. It culminates in a forward-looking discussion of AI-

related risk specifically and technology risk generally and analyses how society will need to deal with them.

Online Resources

The Risk-First website[1] contains a lot of additional resources and is frequently mentioned throughout this book. If you want to join in discussions or help with content, then you can get an invite to the Github Risk-First team by adding your star to the Github repository.[2]

You can also find this book's online forum and errata on The Pragmatic Bookshelf website.[3]

1. https://riskfirst.org
2. https://github.com/riskfirst/website
3. https://pragprog.com/titles/rmrfsd