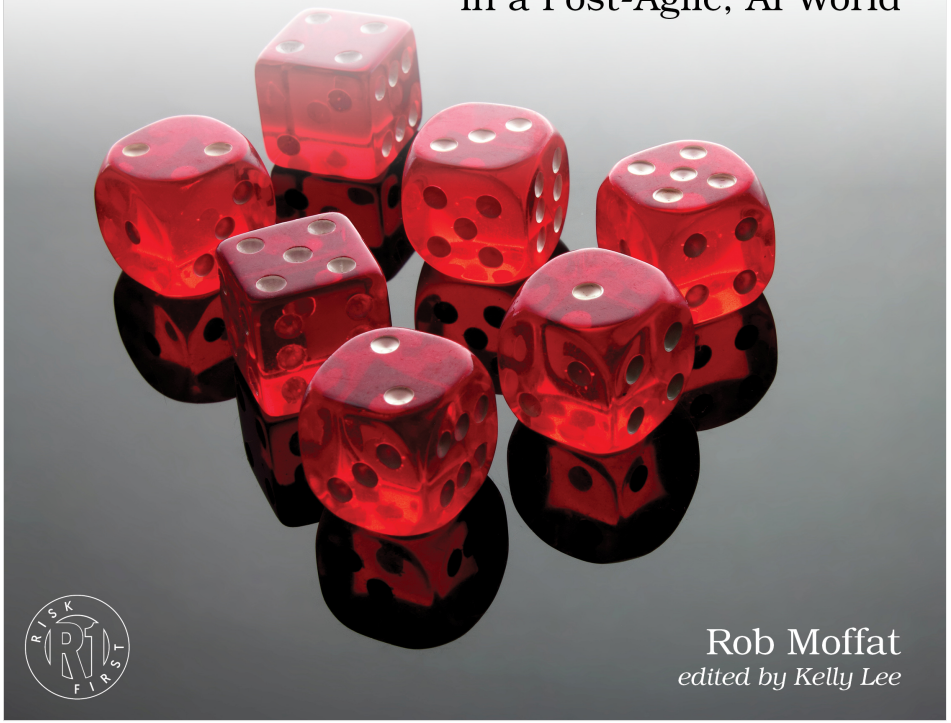
The Pragmatic Programmers

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.

The Cost Of Meeting Reality

Meeting reality *in full* is costly. There are lots of tasks that are *expensive*:

- The Release Process
- Training Users
- Getting users to use your system
- Gathering feedback

These steps take a lot of effort and time, but you don't have to meet the "whole of reality" in one go. You can meet it in a limited way which is less expensive.

Tip: Making Meeting Reality Manageable

You should try to meet reality in the four following ways:

- Sooner: so you have time to mitigate the hidden risks it uncovers.
- More Frequently: so the hidden risks don't hit you all at once.
- In Smaller Chunks: so you're not over-burdened by hidden risks all in one go.
- With Feedback: if you don't collect feedback from the experience of meeting reality, hidden risks stay hidden.

Example 3: User Acceptance Testing (UAT)

Putting your software in front of users in production will definitely put you into contact with reality. You'll learn a lot. But sometimes, this can be dangerous: if the release contains bugs that affect the user population, your reputation may suffer and you might end up fire-fighting support issues.

UATs, beta-releases, incremental (small) releases and phased releasing to small populations of users also allow us to meet reality too, but dial down the stakes (see the following figure). They give us a more limited exposure to reality, more cheaply and sooner.

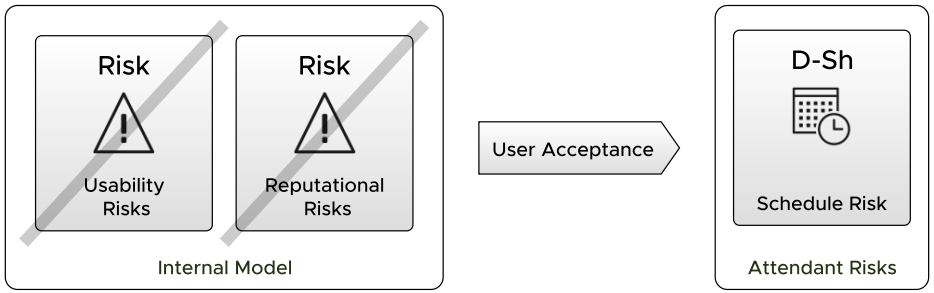


Figure 3.4: Testing flushes out hidden risk, but increases schedule risk

Evaluating and Making Trade-Offs

Making a move on the Risk Landscape is about accepting a trade-off. The examples above are all classic software development trade-offs. If you're an experienced software developer, you'll understand that any technology decision (whether it's unit testing, database choices or release processes — the examples we've seen here) means accepting a trade-off.

The Risk-First diagram helps us in two ways. First, it clarifies the details of this trade-off. What's lost? What's gained? Second, by describing our trade-offs in terms of *risk*, we are also making clear the fact that up front, we're never certain whether the trade-off will be worth it.

It's All About Risk

Taken to its logical conclusion, you should be able to see that every action you take on a project is a trade-off, ideally moving you on the Risk Landscape to somewhere preferable. From there, it's not really much of a stretch to ask the question: is *everything* productive you do on a software project Risk Management?

In the previous chapter on page ? we focused on a "toy" development process, maybe not a huge leap from the one you use at work, which might involve weekly releases, a continuous-integration system, unit-testing and beta-testing. As we saw in the previous chapter on page ?, these activities all have a part to play in managing different risks. They work to expose hidden risks early, before they lead to bigger risks in production.

Can we go further? Is it just those parts of our development process that manage risk, or is it actually *all* of them?

Pulling Apart The Project Manager's "RAID Log"

In order for me to make this case, lets look at one of the tools in the Project Manager's tool-box, the RAID Log,[2] and observe how risk-centric it is. Then, we'll generalise from there.

Many project managers will be familiar with RAID Logs. It's simply four types of item on a spreadsheet: *Risks*, *Assumptions*, *Issues* and *Decisions* allowing you to track what (if anything) you're going to do about each of them.

Category	Description	Impact	Owner	Priority
Issue	Complete logo approval		Debbie	HIGH
Risk	Supply Delivery Time	Project Schedule	Bill	MEDIUM
Assumption	Master data consistency		James	MEDIUM
Decision	Go/No Go with ABC Ltd.		Internal Hiring	LOW

Table 1—A RAID Log

Let's keep things simple for now and zoom in on just the first item in the RAID Log:

> "Debbie needs to visit the client to get them to choose the logo to use on the product, otherwise we can't size the screen areas exactly."

- Is it an *issue*? Yes, because it's holding up the screen-areas sizing thing.
- Is it a *decision*? Well, clearly, it's a decision for someone.
- So, is this an *assumption*? Possibly. There's the assumption that it's Debbie's job, and that the client will make a decision.
- Is it a *risk*? Probably. Debbie might go to the client and they *still* don't make a decision. What then?

This example is deliberately chosen to be hard to categorise. Normally, items are more one thing than another. But often, you'll have to make a choice between two categories, if not all four.

This *hints* at the fact that at some level it's all about risk:

Every Action Attempts to Manage Risk

The reason you are taking an action is to manage a risk. For example:

2. http://pmtips.net/blog-new/raid-logs-introduction

- If you're coding up new features in the software, this is managing Feature Risk (which we'll explore in more detail in Chapter 8, Feature Risks, on page ?).
- If Debbie is getting a business sign-off for something, this is managing the risk of everyone not agreeing on a course of action (a Coordination Risk, discussed in Chapter 10, Model Risks, on page ?).
- If you're writing a test, then that's managing a type of Implementation Risk (also covered in Chapter 8, Feature Risks, on page ?).

Every Action Has Attendant Risk

At the same time as managing risk, anytime you take action you're also taking on new risks. For example:

- How do you know if the action will get completed successfully?
- Will it overrun, or be on time?
- Will it lead to yet more actions?
- What Hidden Risk will it uncover?

Consider *coding a feature*. The whole process of coding is an exercise in learning what we didn't know about the world, uncovering problems and improving our Internal Model. That is, flushing out the Attendant Risk of the Goal.

And, as we saw in the Introduction, even something mundane like organising a dinner party led to uncovering new risks. When Debbie goes to see the client, will she uncover some new unforeseen risks? Is the client thinking of pulling out of the project, going bankrupt or undergoing a change in management?

Actions with attendant risks are also called "threats" in the risk management world. We'll see the term threat come up a lot later. For example, a hacker is trying to break into your database (an action) or the possibility they might (a threat) increases *security risk* — a topic we'll return to in Chapter 11, Environmental Risks, on page ?.

An Issue is Just A Type of Risk

The RAID log tries to make you choose between putting something in the "issues" column or the "risks" column. That turns out to be hard, because:

- Issues need to be fixed…
- And fixing an issue is an action…
- Which, as we just saw also carries risk!

One retort to this might be to say: "an issue is a problem I have now, whereas a risk is a problem that *might* occur. " But ask yourself:

- Do you know *exactly* how much damage this issue will do?
- Can you be sure that the issue might not somehow go away?

Issues then, just seem more "definite" and "now" than *risks*, right? But his classification is arbitrary: they're all just part of the same spectrum, they all have inherent uncertainty, so there should be no need to agonize over which column to put them in.

In a way, this is a blessing because it means if you are maintaining a backlog[3] (the Scrum term) or tracking work in an issue-tracking tool such as GitHub Issues[4] then you're tracking risks.

Let's look at a real-life example. The following figure shows a selection of issues logged in GitHub for an open source project called FDC3.[5] The first one, "Create more FDC3 developer tools" is written as if it's a piece of work to be done (an action to take). However, there's an implicit risk being addressed by this piece of work: the risk that developers using the project are under-served by the functionality of the tools they've been provided, and will be dissatisfied by the project. This issue was likely raised by developers facing this problem: frustrated by insufficient tooling and therefore complaining about it.

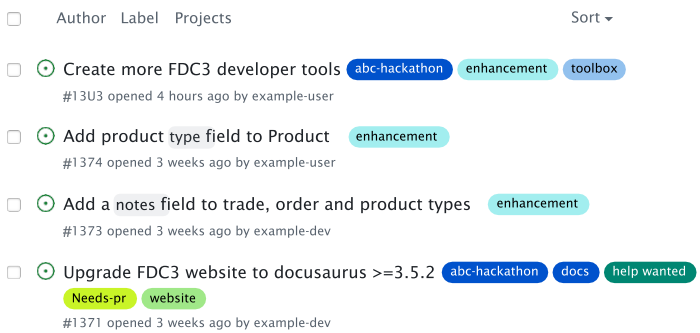


Figure 3.5: A selection of issues from a GitHub project

3. https://www.scrum.org/resources/what-is-a-product-backlog
4. https://github.com/features/issues
5. https://github.com/finos/FDC3

And Assumptions Are Risks, Too

Finally, when we log assumptions we're also logging the risk of that assumption being wrong. Essentially, this is making note of a known risk that we're taking but have also decided not to do anything about (at least, for now).

Your Goals Are Risks, Too

Hopefully, the above has convinced you on at least the notion that issues are just types of risks and recording issues in an issue tracker is really just a way of logging risks. Later on in this chapter we'll investigate in a bit more depth how we prioritise these risks / issues and decide what to work on. However, before we get there, let's look at the flip side of risks and talk about goals.

In software development, tracking issues is a day-to-day activity. Tracking goals happens much more infrequently, but it should happen. There are various tools for this which you've probably come across, such as:

- Specific, Measurable, Achievable, Relevant and Time-Bound Goals (or "SMART" Goals)
- Key Performance Indicators (or "KPIs")
- Objectives and Key Results (or "OKRs")

Goals live inside our Internal Model, just like Risks. And functionally, Goals and Risks are equivalent. For example, the Goal of "Implementing Feature X" is equivalent to mitigating "Risk of Feature X not being present".

Let's try and back up that assertion with a few more examples:

Risk	Goal	Action
Risk of someone breaking in	Feeling of safety	Install a home security system
Risk of looking technically inferior during the cold war	Feeling of technical superiority	Land a man on the moon
Risk of the market not requiring your skills	Job security	Retrain

Table 2—Risks, Goals and Action Equivalence

There is a certain "interplay" between the concepts of risks, actions and goals. On the Risk Landscape (and in a Risk-First diagram), goals and risks correspond to starting points and destinations, whilst the action is the movement through the risk landscape.

Starting Point	Movement	End Point
Goal, risk	Action	Attendant risk, achieved goal

Table 3—Risks Goals and Actions

From a redundancy perspective, if you know any two of the start, end or movement you can figure out what the third would be. At different times, people like to focus on different parts of this. Sometimes, humans are very goal-driven: they like to know where they're going, and are good at organising around a goal (like "landing a man on the moon"). However, by focusing on goals ("solutionizing") it's easy to ignore alternatives. Sometimes, we focus on the risks. As politicians know, fear is a great motivator for voting!