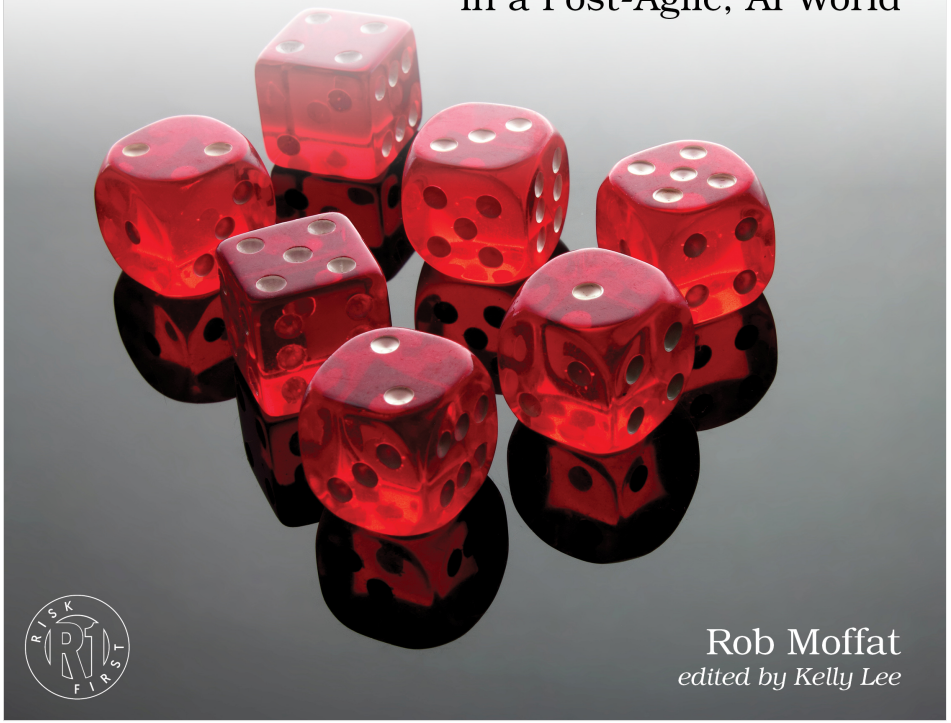
The Pragmatic Programmers

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.

Communication Risk

Risk associated with getting messages heard and understood and the transfer of information.

If we all had identical, perfect knowledge, there would be no need to do any communicating at all, and therefore no Communication Risk. But people are not all-knowing oracles: we rely on our *senses* to improve our Internal Models of the world. There is Communication Risk here — we might overlook something vital (like an on-coming truck) or mistake something someone says (like "Don't cut the green wire").

For *people*, Communication Risk includes:

- Not having the information you need.
- Being presented with the wrong or erroneous information.
- Being "out of the loop" of important updates.
- Not being able to speak the language, or misunderstanding what's being said.

For *our software*, it includes:

- Issues with connectivity, accuracy and relevance of information.
- The problem of contradictory versions of information.
- Protocol incompatibilities.

Communication Risk applies equally well as a concept in human scenarios between *people*, *teams*, *organisations* as well as software scenarios such as *libraries*, *services*, and *data formats*.

A Model Of Communication

In 1948, Claude Shannon proposed in his seminal paper, "Mathematical Theory Of Communication", that "the fundamental problem of communication is that of reproducing at one point, either exactly or approximately, a message selected at another point."

And from this same paper we get the following figure: we move from top-left ("I want to send a message to someone"), clockwise to bottom left where we

hope the message has been understood and believed. (For completeness, the last box, *reconciliation* has been added to Shannon's original diagram.)

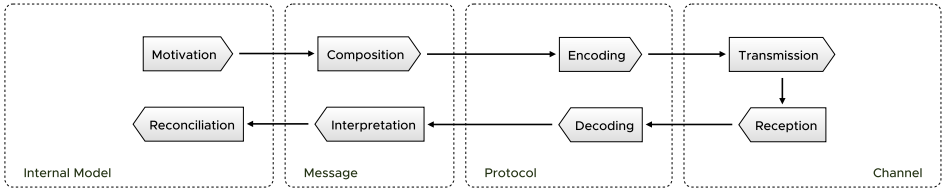


Figure 10.2: Communication Risk, broken into four areas (Shannon's Model)

One of the chief concerns in Shannon's paper is the risk of error between *Transmission* and *Reception*. He creates a theory of *information* (measured in *bits*), sets the upper-bounds of information that can be communicated over a channel, and describes ways in which Communication Risk between these processes can be mitigated by clever *Encoding* and *Decoding* steps.

But it's not just transmission. Communication Risk exists at each of these steps. Let's imagine a human example, where someone, *Alice* is trying to send a simple message to *Bob*.

Step	Potential Issue / Threat
Motivation	*Alice* might be *motivated* to send a message to tell *Bob* something, only to find out that *he already knew it*.
Composition	*Alice* might mess up the *intent* of the message: instead of "Please buy chips" she might say, "Please buy chops".
Encoding	*Alice* might not speak clearly enough to be understood.
Transmission	*Alice* might not say it *loudly* enough for *Bob* to hear.
Reception	*Bob* doesn't hear the message clearly (maybe there is background noise).
Decoding	*Bob* might not decode what was said into a meaningful sentence.
Interpretation	Assuming *Bob has* heard, will he correctly *interpret* which type of chips (or chops) *Alice* was talking about?
Reconciliation	Does *Bob* believe the message? Will he *reconcile* the information into his Internal Model and act on it? Perhaps not, if *Bob* forgets, or thinks that there are chips at home already.

Worked Example

You're about to roll out new software in an organisation and you're worried that staff within the organisation won't bother to read the documentation on how to use it. You decide to organise a demo. However, there is the risk that by doing this you divert the staff's time and attention away from hitting some critical release milestones, not just through the wider activity of attending the demo but through your colleagues' time spent preparing for it.

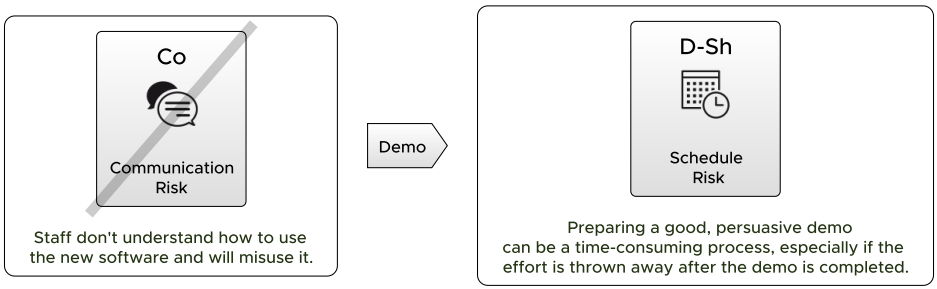


Figure 10.3: A demo helps overcome Communication Risk

Example Threats

Each of the above stages in Shannon's model represents a potential threat vector, increasing Communication Risk.

1. Channel Threats

Threat: Channel is noisy, containing much that isn't the signal you're after.

Threat: Channel is low-bandwidth and important messages that should be sent and received aren't.

2. Protocol Threats

Threat: The protocol isn't what is expected (e.g. you're understand English but they're speaking French, or you're expecting HTML but get JSON)

Threat: The protocol changes to a new version (e.g. people start using words you don't understand, or the network changes to use IPv6)

3. Message Threats

Threat: Someone is sending a message with malicious purpose

Threat: The message contains only part of the information you need, leading to the wrong decision.

4. Internal Model Threats

We'll cover these in more detail in the Internal Model Risk section later in this chapter, but examples might include the internal model being wrong, incomplete or out of date.

Practices Addressing Communication Risk

Documentation (as the output of say *requirements capture* or *analysis*) is an obvious aid, though it must be kept up-to-date to remain useful. *Demos* facilitate clear communication of features and concepts, while *stakeholder management* or ensures consistent communication channels between different parties. Techniques like *reviews* or *meetings* can help get people on the same page while *protocol standards* are very important for both communicating systems and people.

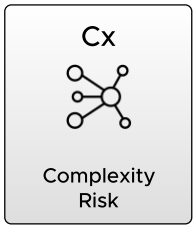
Anecdote Corner

The 1998 Mars Climate Orbiter software disaster was caused by two teams (Lockheed Martin and NASA) collaborating on building different parts of the orbital insertion control program. The two components needed to communicate with each other but there was a discrepancy: one piece of software was expecting *pound-force seconds* of thrust, while the other was producing *newton seconds*: one imperial, one metric measurement. This is a great example of Communication Risk occurring at the protocol level.

During flight, two navigators had spotted discrepancies in the orbital insertion but failed to correctly fill out forms relaying their concerns to the project managers in charge. This is a further example of Communication Risk, this time occurring at the channel level.

The orbiter and lander were both lost at the cost of hundreds of millions of dollars.

Complexity Risk



Risks caused by the weight of complexity in the systems we create, and their resistance to change and comprehension.

Complexity Risk is the risk to your project that arises from its inherent complexity. Factors like code, documentation, issues, features, algorithms, diverse user types and use cases, and, most importantly, the relationships *between these elements* all contribute to it.

Looking at the living world, society or software in general, we can see that over time complexity increases. There is a trade-off wherein we can capture more resources, more value or more user requirements (respectively) through increasing complexity.

Complexity Risk makes our software more brittle, harder to change and more expensive to maintain. A further downside of this (as shown in the following figure) is that complex systems are hard to reason about and therefore often contain unanticipated weaknesses (Hidden Risks). In production systems this leads to Operational Risk (see Chapter 11, Environmental Risks, on page ?).

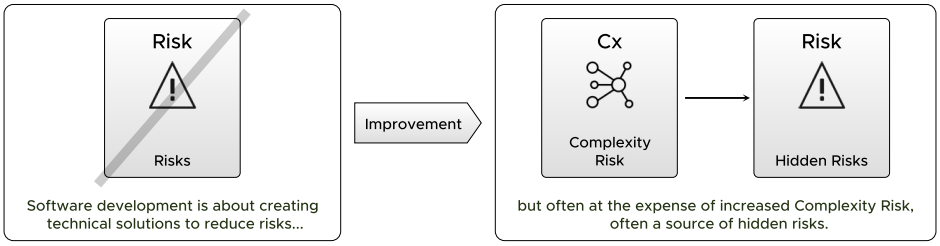


Figure 10.4: Complexity Risk is a source of Hidden Risk