medium.com /@jgreze/thinking-probabilistically-about-engineering-a26cc4c0a7e6

Thinking probabilistically about engineering

Jean-Denis Greze

4-5 minutes

One of the most empowering aspects of being an engineer is the control we exert over the final product — how it is built, how it looks and works, and the myriad small decisions that add up to a beautiful, engaging experience. Such is the joy of being a craftsman with (generally) deterministic tools. Of course, this control cuts both ways. Both engineering successes and failures can be attributed to us: as an individual contributor ("IC"), bugs caused by my code are my fault. Hence, engineers quickly learn how to use testing, QA, fast iterations and deployments, and other methods to minimize both the bugs and their effects on the products we build. We strive for **ex-ante prevention** of bad outcomes.

At scale — be it machine or people scale — this ex-ante prevention proves increasingly unattainable.

Consider an engineer who is managing a system that runs on a very large fleet of servers. Hard drives fail, and although the overall failure rate is known, any one failure is random. You cannot stop hard drives from failing and must instead focus on how to quickly recover from hard drive failures. Bad outcomes are no longer something that can be avoided — they are simply a **probabilistic inevitability** and near certainty as fleets get larger.

When some ICs decide to become managers, they will often take with them the deterministic view described above, attributing the "full control" they once exercised over work to their teams. Unfortunately, there too the "expected unexpected" rears its head. Consider attrition. As a manager, it's inevitable that someone will leave your team, you simply don't know when (or at least, generally not with much advance notice). You can lower the probability of someone leaving, but however hard you try full deterministic ex-ante prevention is unattainable.

As teams scale, the unexpected goes from rare to commonplace. Like ICs, the sooner managers adapt their thinking to incorporate probabilistic thinking and solutions, the better.

What does it mean to think probabilistically?

Let's work through an example: how to manage quality. Every engineer, at some point, introduces a bug. This is unavoidable. At a small scale, this isn't a big deal — if you have 12 engineers and on average each introduces a big bug that makes it past tests into deployment twice a year, then you'll introduce a bug intro production every two weeks. Ideal? No. Manageable? Sure. However consider what ensues if you have 120 engineers: you're going to have to revert/rollback/fix a buggy live system every work day. That's a very real and very big problem.

Thus the rare events that didn't really affect your velocity at a small scale can become totally debilitating as your team grows.

Thinking through solutions, if it's unavoidable for bugs to make their way into production, then we should not just focus on processes and training that minimize them before-the-fact, but also on ways to quickly push fixes or revert to older versions of the code base after a bug is detected. **Ex-ante prevention, which makes sense in a world of full control, needs to be coupled with after-the-fact recovery.**

This coupled approach works broadly across engineering at scale — be it on the technical or the management side — for both are rife with unpredictable events. And the sure sign you are working with experience engineers and managers is that they'll think not just in terms of prevention, but also in terms of the monitoring, tools and process required to fix things after the worst inevitably happens.