PART FOUR

Obstacles

SOME OF THE HARDEST PROBLEMS YOU DEAL WITH WHEN BUILDING SOFTWARE ARE THINGS YOU CAN'T control. Sometimes accidents take down your project. Sometimes unforeseen issues do. And unfortunately, sometimes things are done to the team on purpose to undermine a project. Building software is hard enough without dealing with obstacles that are put in the team's way. But we work in the real world, and real-world projects run into problems that are bigger than the team and the software.

One of the biggest obstacles is a bad manager.

Have you ever had a manager who refused to accept a reasonable estimate? Or who consistently undersold your efforts ("The team says it'll take six weeks? I'll promise it in four; that'll light a fire under their asses!") and makes you work weekends to meet the crazy expectations? Bad managers come in all sorts of flavors, and every one of them presents a serious, sometimes even insurmountable, obstacle to the team. Some managers seem bound and determined to make your project fail. They micromanage the team, constantly change the goals of the project, and always seem to be made of Teflon when things go wrong, blaming the team for the problems and never questioning, even for a minute, their own direction or management.

A bad manager can be impossible to work for. But a team that's got a bad manager does have options. Sometimes it starts with recognizing that the manager is working against the team, and working around him. Sometimes you can actually change a bad manager: if you can gain a manager's trust, and get him to have confidence in the team, those things can work themselves out. Either way, the first step is recognizing the obstacle and figuring out how to handle it.

Not all obstacles are managers, or even people. Sometimes an obstacle is as simple as a technical constraint: having to work with terribly old hardware, having to make the software backward-compatible, unexpected and serious performance problems, or impenetrable spaghetti code that's difficult to maintain.

Some problems are an unavoidable result of the environment the team is working in. It could be a problem with the facility or the building. Some teams have to cope with time zone issues, when they're divided into groups across the world from each other. There are weather problems, traffic problems, transportation issues, noise problems … all sorts of obstacles that the team simply can't avoid.

Any of these problems can sink a project. But if you have a team that trusts each other and works well together, they can overcome even the craziest, most seemingly insurmountable obstacles in their way. In fact, getting past a serious problem can bring a team together. After all, you're all in it together. If you've got a serious problem that affects everyone on the team, then you'll all have to work together to get past it. And once you do, you've all got a shared experience that bonds the group tighter—much more than any cheesy corporate team-building retreat exercise ever could. In a lot of ways, it's adversity that makes a team gel.

Obstacles are hard for people to talk about. They're painful to go through, and not much fun to think about. And more often than not, serious obstacles really do cause projects to fail. If you open any software engineering textbook, you can read about all sorts of statistics and studies about project failure. (Some textbooks put this number at a staggering 80%!) But how many people have you talked to who really spend any time talking about their own failed projects?

It's human nature to avoid talking about our own failures. Most people's resumes highlight the successes, and try to find a silver lining in a project that failed. Few (if any) people ever see their own work life or careers in terms of failed projects, and certainly nobody will ever cast his own career as a series of failures, one after another. But if 80% (or even half that number) of projects really do fail, then our resumes and our lives should be littered with them.

And, in fact, when we've gone around to actual teams and talked to them about their own projects, that's exactly what we've found. There really are a huge number of failed projects: projects that went woefully over budget or came in very late, projects that built unusable software that the users rejected, projects that were so behind schedule that the problem they were built to solve substantially changed, and projects that simply failed to deliver anything at all. In fact, we've given many talks over the years about project failure, and at each of those talks we ask the audience, "Is there anyone here who hasn't been on at least one failed project?" We've never seen a single hand go up.

The stories in this section are about teams overcoming all kinds of problems. There are stories that deal with good people facing difficult companies, awful managers, serious technical problems, and true evil. Some of them are preventable, frustrating, and insidious. Some of them are simply heartbreaking and utterly unavoidable. In every case, the obstacles had a profound effect on the team: on how they worked together, and on how (or whether) they got past it. But in every case, there's an important lesson to be learned, and people who were made stronger for the experience, even when they outlasted the team itself.