PART ONE

People

WHEN WE STARTED WORKING ON *BEAUTIFUL TEAMS* BACK IN LATE 2007, WE THOUGHT WE HAD A pretty good idea of where the book would go. We're techie people: we've implemented lots of practices, done lots of software projects, and seen things get better over time as we did it. We've seen our own teams improve, we've learned from our mistakes over time, and we thought we had a handle on what makes a team work.

So we thought we'd sit down with a bunch of people who felt the way we did, write down some insightful tricks of the trade and pearls of wisdom, and everyone would be happy.

As it turns out, we were in for a surprise.

Now, we weren't exactly wrong or even misguided when we started working on this project. Results do speak for themselves, and we'd gotten good results. One of the things that made us feel quite confident about our reasonably complete understanding of what makes a good team work is that we'd read plenty other people who agreed with us.

One thing that we both agreed on when we started this project was that a good team has to have a focus on the technical skills of the people on the team. So when we started recruiting authors and looking through their writing, we expected a lot of stories about how to make sure your team is technically capable: recruiting and hiring the right people, finding the right practices (like writing specs or doing code reviews) and training people on them, and generally helping people who already show talent and promise to develop their skills to become better programmers.

It didn't exactly work out like that. In fact, there's very little talk throughout the whole book about training people, improving their skills, or being very selective about your team members.

Now, let's be clear here. Nobody said that wasn't important. In fact, just the opposite seems to be true: all of our contributors seemed to assume that a good team starts out with at least a kernel of skilled people with the right training, background, and attitude for the job. It seemed obvious to everyone that if you have people who are incapable of doing the job, the team and the project will clearly fail. But what surprised us was that rather than being the ultimate goal, getting the right skills for the team was really more of a starting point to build on.

The truth, as it turns out, is a lot more complicated than we realized at first. It has more to do with things that are a lot harder to quantify. Sometimes the best people on the team aren't always the most technically skilled, and sometimes the most technically skilled people on the team can actually be detrimental to getting the work done. People need to be able to work together, and if they can't, it doesn't matter how good their skills are. In fact, some of the best programmers (from a technical perspective) are in danger of getting too cocky, and can actually make it more difficult to get the project done. But those are often the most important people on your team.

We were starting to get a little nervous. We were used to seeing the world in black and white: either you have a team of the best people (the smartest, most technically skilled, capable developers, testers, architects, etc.) who can do the job best, or you have a less-than-perfect team and you have to deal with it. But what we found is that *all* teams are less-than-perfect. Even the ones that are full of highly skilled, top-notch people can have serious problems that can cause the code to suffer. And *all* teams have some drama. Even if they produce something solid and amazing, they all have to overcome the same challenges that come with people working with other people.

Once we realized that, we started looking back through our own careers, and our own teams. What we found was that our most successful teams really weren't always the ones with the best programmers. They were the ones where everyone worked really well together.

In the end, teams are made up of people, and people are the most important part of the equation. Sometimes we even fell into the trap of thinking that people with similar skills were somewhat interchangeable. If you treat building software as a bean-counting exercise, it's way too easy to think of people as resources to be assigned to tasks, and their work as effort hours to be expended; you could easily come to the conclusion that good people are just cogs in a machine, and that their technical skills are just attributes that can be measured or turned on with certifications, training classes, and experience hours.

As we worked with our authors and interviewees, we were reminded just how untrue that is. And, in fact, we realized that on a few occasions, both of us have fallen into the trap of thinking of our people as interchangeable cogs, and it inevitably led to our projects running into serious problems.

The stories and interviews in this section are the ones that reminded us of those important lessons. They're about what happens when people clash on teams, and what happens when they learn to work together well. Reading these stories and talking to these people, we learned that no team is a storybook team of brilliant engineers who work together perfectly and professionally 100% of the time.

More than that, the most influential people on your team aren't always the best programmers. In fact, they're not the managers or leaders, either. They're the people who are *genuine*: they're there to get the work done, and are able to be themselves while doing it. They listen to the people around them, and understand what motivates them and makes them tick. And it's not because they read a book or took a seminar on how to be a motivational leader. It's because they're considerate to the people around them. They take the time to make sure that everyone's on the same frequency—not necessarily nice or even helpful to each other, just compatible in a way that works for them. And they recognize when they simply can't work together (for whatever reason), and try to do something about it.

Teams are messy. They're full of emotional connections, often between people who are at their wits' end trying to solve problems that may not necessarily be solvable. When you have people who, in the midst of a situation like that, are willing to be themselves, put themselves on the line, listen to the people around them, and help everyone get through the late nights and the frustration, that's what makes a team great. Not necessarily *beautiful*, but effective and good to work on. Without those kinds of connections between people on the team, the job is just a job, and the team isn't one that everyone on it will remember for the rest of their lives.