CHAPTER TEN

Putting the "I" in Failure

Jennifer Greene

THE SUMMARY LINE ON MY RESUME SAYS THIS: "JENNY GREENE IS AN EXPERIENCED PROJECT AND development manager with a track record of successful projects and teams." And it's true, I've managed a bunch of teams that have been able to create the software they were paid to build on time, within budget, and with quality. I'd bet that a lot of people out there could make the same claim. But when I look back over the work I've done over the past 15 years or so, I know that a number of those successful projects didn't lead to market dominance for the companies I worked for. In some cases, they didn't even solve the problem they were meant to solve. And pretty much everything I've read tells me that I'm not alone. A lot of projects do much worse than the ones I've worked on. Some published estimates put the failure rate as high as 80% of all software projects.

A few years ago, Andrew and I started traveling around to networking events and conferences giving a simple talk called "Why Projects Fail." The point of the talk is to help people make the connection between practices that they might dismiss as heavy or difficult (like code reviews and estimation) and avoiding common pitfalls. We expected it to be a bit irreverent and lively because we tried to use some of the stuff we'd learned in writing Head First books to keep the mood light. And we got what we expected—but more than, that we struck a chord with people who seemed to be dying to talk about the mistakes

they'd made on their projects. I mean, a lot of people want to know: "Really. Why *did* my project fail?" More often than not, the talks turned into an opportunity for people to vent: to tell us their war stories about this insane decision from management, or that terrible programmer who was too egotistical to fix his own code, or that product that was just a bad idea from the start. In front of the audience, we'd always try to turn the conversation around to a particular practice that could help out ("If you've got problems with quality, do code reviews upfront" or "You're always missing your deadlines? Use a consensus-based estimation process like Wideband Delphi"). Those practices have helped both Andrew and me immensely in our careers. But practices won't solve interpersonal problems or make sure that the project you're working on is the right one for your company. They won't guarantee success. And away from the audience, we talked about that a lot.

I've worked with companies where the need to produce software on a regular basis was so encompassing that they'd lost track of the reasons behind that software. Places where people spent tons of money building stuff just to prove that they have a product. I've seen projects get funding without much understanding of what they'll produce, and I've been managed by insecure people who turn the screws on the team to hit crazy deadlines for projects that are really not important in the scheme of things. I've worked nights and weekends and given up vacations for products that have barely been used. I know what it feels like to work on a delivery team when your goals are constantly changing and there's no guarantee that what you're building will ever be useful to anybody. Are you successful if you build that useless product on time? Are you successful if you keep a team together under conditions like that? Frankly, it's tough to be objectively successful in that situation. The best you can do is work to challenge the need for those deadlines. Fight to give people a more reasonable set of goals, and sometimes that's a fight you can't win.

Some of the best teams I've worked with have been in impossible situations and some of the best work I've done hasn't contributed to objective success. It's the people who stick with you, and sometimes those impossible situations, that bring out some really innovative thought. When I was new to New York and still pretty early in my career, I joined the team that'll probably stick with me as one of the most hard-working and genuine groups of people I'll ever work with.

"At Gabfest Software, we think that getting a bunch of really smart people together and putting them in a room with a really tough problem can lead to magical results."

I remember the excited look on the CTO's face as he said those words. He was really enthusiastic about the work that they were doing, and it was disarming. I was interviewing for a job as a QA manager at a small start-up company that was doing amazing things. This company was going to change the way people interacted with computers: it was going to get computers to understand the way people talk. This informal chat came after a full day of meeting half the company and answering more traditional interview questions. He took me to the coffee shop next door to the office and began to sell me on his vision of the company and its future. When he said that bit about smart people and tough problems it seemed not just reasonable, but actually inspiring. Smart people can do anything, right? At the time I thought I was pretty smart. And after spending two years testing internal

software for a really conservative financial company, I was ready to try to tackle more significant problems. So I left that company and prepared to start fighting the good fight.

My background was about as far from natural language processing as it gets. The company I'd just come from, Gridline, was a small company that built financial software that tracked stocks. They'd hired me after I moved to New York City (I'd just spent a few years at an online service testing C++ client software). The job at Gridline was great, but a little stuffy. It was the kind of place where casual Friday meant "business casual." Everybody wore suits and ties everyday, the culture was sales-driven, and most of the team had been in finance for a long time.

Gridline was small. The executive team was filled with people who'd been there since the company was founded. They took pride in the company's success and felt a kind of loyalty to its founder that I haven't really experienced anywhere else I've worked. Many of the people who worked there stayed in the same job for years and years. I joined the company as a QA manager testing their internal data scrubbing applications. I was there for about two years and I really enjoyed the work. I hired and trained the test team from scratch, and documented and built the first versions of their development process. That had a big impact on the quality of the work they were doing. When the test team was built, I started taking on more and more responsibility. But I didn't see my paycheck growing along with the challenges. Eventually, it dawned on me that I could probably do better by looking for a new job. The dot-com boom was in full swing, and anybody who knew anything about computers had absolutely no trouble finding a good job.

When I took the elevator up to the Gabfest offices, I was amazed that it seemed to be the absolute opposite of the job I was leaving. There was no security to get into the building, no receptionist, no fancy lobby. Not one person in the office was wearing a suit or a tie. The whole office was set up as one big room full of tables. Everyone sat together with no cubicle walls and no privacy at all. I thought of the fights I'd been through when we were remodeling the offices at Gridline. There were so many levels; small cubes for the entry-level and temporary employees, big cubes for those who'd been around for a few years, small offices with Formica desks for managers and supervisors, and big offices with real wooden desks for directors, VPs, and higher. We had endless conversations about who would sit where and what level should get what kind of office furniture. All of that stuff meant so much to everyone at Gridline. The people who worked in cubes took it very seriously that one day they might make it into an actual office, and the execs were really proud of their wooden furniture. I knew my place: Formica office with about five years of towing the line to get to finished wood furniture. But Gabfest's office made it clear that nobody cared about any of that crap. We were all there to get the work done together. After two years of watching people fight over offices and titles and responsibilities, I was ready for a job that was all about the work.

A lot of people have written about the excesses of the dot-com boom, and now that I've read about the other start-ups in that time, I feel like I got the short end of the stick when it came to amenities. There was no foosball table, no massage therapist on staff, no organic

food chef or free lunch. It was just a big room full of people working really, really hard on some impossible problems.

They were building a product that I still don't fully understand today. A group of computational linguists had created a language processing engine that took phrases that were typed in by users and modeled out responses based on a deep understanding of the domain they were discussing. The first product they built knew everything there was to know about laptops. If you typed in "I want a fast, cheap laptop that weighs under 5 pounds" it would turn that into a SQL query and return the laptops from a database that fit your specifications. These programs would be packaged up and sold to websites that wanted to provide a more personal shopping experience for their users but didn't want to hire people to sit and talk to them.

The work began with one of the senior employees in the company. He led the team of linguists and had built the prototype the software was based on before the company was even formed. He was really dedicated to the software he'd built, and constantly interested in making it better. But he wasn't sold on the idea that it would ever be a commercially viable product. In retrospect, that probably should've been a red flag for everybody working there. But we all really liked our jobs. I mean, we all really liked the work we were doing in the meantime, so it made the direction and overall goal of the product less important.

I instantly liked the people I met. They were smart and funny, most with impressive educations and completely dedicated to their work. The software, for the most part, worked. As long as you confined your conversations to laptops, it was a pretty helpful sales tool. The group had focused on the quality of the software from the very beginning. Along with the product, they'd built an automated test tool to catch defects as early in the development process as possible. There was a sense that the people working there were so dedicated to the problem they were solving that they were really doing their best every day to expand its capabilities.

In retrospect, there were a lot of problems with the software we were building—and not necessarily technical ones. For instance, we've learned a lot about how to sell people laptops over the Internet, and it's pretty obvious that people don't shop the way Gabfest wanted them to. If you want to buy a laptop online, most companies that sell them have configuration tools that let you select the features you want from a list of what they offer. When you think about it, an interface like that is probably a lot more direct than the tool we were building. Let's face it: most people don't want to have conversations about laptops with NLP engines just to narrow down the six or seven optional features they can choose from. But it was still a really interesting problem for linguists to solve, and one that it was possible to get funding for back then.

I worked really hard while I was there. Whenever there was a demo or a business development deal in the works, I would put in ridiculous hours making sure that everything worked correctly. I remember putting in overnights and weekends on an almost routine basis. Everybody did. You could rely on everyone there to do his or her part in a way that I haven't really seen since. I was responsible for starting to gather metrics on the way the

product was behaving, tracking down some nasty bugs and planning out the work of the rest of the QA team to hit some aggressive delivery deadlines. Because the testing itself was automated from the start, I needed to figure out the best way to evaluate all of the test results and track them over time. I started to find habitual problems with the development process and worked with the team to correct them. That work was interesting for me and it's where I started to piece together some of the practices that would make such a difference in the way I managed development teams going forward.

I had the freedom to make any suggestion I thought might help our products directly to the CTO, the development leads, and the developers themselves. I was able to suggest changes and see their impacts right away, although every change was discussed and argued by the team before we did it, and if an idea had merit, more often than not we did it. I trusted the people I worked with, and they trusted me, so all in all, it really worked. After working in some pretty political places, having that kind of access to the decision makers was a welcome change.

But as rewarding as the work was, there were some major problems, too. Alongside the natural language projects were some AI projects that were being developed without buy-in from the linguistics teams. We also had a data mining team that was spinning up to track the conversations live users would have with the software when it went live. There were often heated battles about whether it made sense for us to focus on all of those goals at once.

As the company went on, the three groups of people really emerged as almost warring factions. Everyone believed really strongly in their own cause, and honestly believed that they were going to be the difference between success and failure for Gabfest when it finally hit the marketplace. There were the computational linguists. They fought to make the NLP engine as good as possible, expanding its vocabulary and its knowledge domain. There was no real end point. There are an unlimited number of ways you can express an idea, even an idea as simple as "I want a fast laptop." They knew their engine would never be as smart as a human being, but they also knew that people wouldn't trust it to work properly if it didn't understand "I want my laptop to be fast" to mean the same thing as "I don't want a slow laptop." It seems like an easy problem to describe (for anyone who hasn't really worked with it), but it turns out to be a deviously difficult linguistic problem to solve. And that's what the computational linguists were doing. They felt that the richer the engine was, the better the product would be in the end.

Then there were the artificial intelligence people. They wanted the software to be able to learn new vocabulary and even new behaviors and ideas from the people who were talking to it. Their work wasn't nearly as well envisioned as the linguists'—they were trying to tack their functionality onto an already working language processor. But what it lacked in vision, it made up for in cool factor. Everyone loved this idea. They wanted the software to get smarter and smarter, not because people were working on it, but just because they were talking to it.

If the company had been pulled in only two directions, by the linguists and the AI folks, it might have turned out differently. But there was a third faction who had their own goal for the company: data mining. They wanted the software to go through all of the conversations it had had with all of its users, and use that information to target advertising to specific users based on their profiles and current conversations. Everyone had an intuitive feeling that there was a lot of potential profit there. But we were all worried—even the data mining proponents—about the possible privacy issues that this raised. But it was one of the big selling points of the product. Gabfest's clients were pitched the idea that they would learn a lot more about their own customers based on their conversations with the software.

That was the three-way battle in a nutshell. The linguists basically thought, "None of this works if our stuff doesn't work." The AI people had a goal that wasn't well thought out, but everyone felt it was worth pursuing just because it was so cool. The data mining people considered themselves the profit center for the company. The problem was that we weren't big enough to do all three things well. And as smart as we were, these were very, very tough goals. We all knew that, as a company, we really needed to choose one direction. But we didn't, and the leadership encouraged all three factions to go in their own separate directions.

What we ended up with was a bunch of almost "covert" projects happening simultaneously. The AI people built lots of prototype neural networks, but didn't really add anything to the main product. The computational linguists kept adding functionality to the existing product, even though its sales were still struggling to take off. And this, to me, was the real story of the dot-com boom: it wasn't about foosball tables, flexible hours, wacky job titles, open office plans, free food, or even the rise and fall of venture capitalists. It was about people being given more free reign to take bigger risks and do more interesting work than they could have imagined. It was a time when we were really encouraged to be creative. And for us, that meant doing real, cutting-edge linguistics and artificial intelligence research and trying to come up with a real business model out of it.

Everyone kept working on their own thing. But as things progressed, we had a lot of trouble coming up with a single cohesive product. I think most of us knew this, but wanted to ride it out because each and every one of us was learning something, and it was the most fun job any of us had ever had. In a way, we all got to choose our own adventure, and we were all rewarded by it—not just with paychecks, but in our careers. Every single person I worked with at Gabfest was better off for being part of it. But everyone did feel very strongly about their opinions, and the value of the work they were doing. Each person tried to exert as much influence as possible over where the company was going. There were a lot of knock-down, drag-out fights, sitting in conference rooms screaming at each other about where the company was headed.

Then one Tuesday afternoon the CTO told us we'd run out of money and the company was going to dissolve in two weeks.

I remember the night the announcement was made that we all sort of hung around talking to each other about what we'd do, for hours afterward. We opened beers, sat around and hung out, and for the first time in a long time we had a relaxing night. The fights didn't matter anymore. There was no more goal for the company. We shifted to being just people who'd gone through something traumatic together.

Then something weird happened. The last two weeks that the company was in existence, everyone who worked there showed up every day as if nothing had changed. We just kept working to try to preserve as much of the product as we could. Nobody even questioned it. We were there working 12-hour days, even though we knew we'd be out of work in two weeks. Every single person cared so much about the work they were doing that it didn't matter what would happen to the company. We had a new goal: get everything archived so that we could pick up the work again some day. And in a way, I believe that every one of us thought that would really happen.

Even afterward, we helped each other get jobs. We were all references for each other. One of my friends was eight months pregnant at the time of the company closing, and the whole company came together even after we'd disbanded to throw her a baby shower, chipping in money to get gifts for her. In the months and years after, we've all stayed in touch. Even more than a year later we were getting together. As ill-fated as the original idea was, we were all so engaged in our pieces of it that I think we all think back on the experience and the relationships fondly. We were bonded by that company. I'd happily pick up the phone and give a glowing reference for every single person I worked with there, without exception. There isn't another company I've worked with that I could say the same about.

I know I've never worked with a team before or since that was so singularly dedicated to their work. They're still an inspiration to me. But the truth is that it takes a lot more than smarts, gumption, and money to solve big problems. First it takes a unified, viable goal. We didn't commercialize natural language processing, or build experts that made the web shopping experience much more personal for everybody. We didn't revolutionize the way people communicate with computers. But the work we did at Gabfest did result in something sort of magical.

Despite the industry's disturbing track record for failed projects, nearly everybody's resumes are full of sentences like the one I quoted earlier. In my case, this particular failure was worth a lot more than many of the successes I've had since. Would I consider my part in this successful? On a personal level, definitely. I grew immensely as a result. But objectively, the company was a huge failure. As far as I know, our archived software never got dusted off. I know that we were smart enough and skilled enough to build any software we wanted to—the team at Gabfest was easily the smartest group of people I've ever worked with. But I've seen much dumber, less skilled people make an awful lot of money in the software industry, because they chose a product that would sell and they got it to customers who would buy it.

In general, none of us seem to think that the failures we've lived and worked through are ever our fault. And I don't necessarily think that's a bad thing. I guess I don't think that the failure I've lived and worked through is my fault, either.