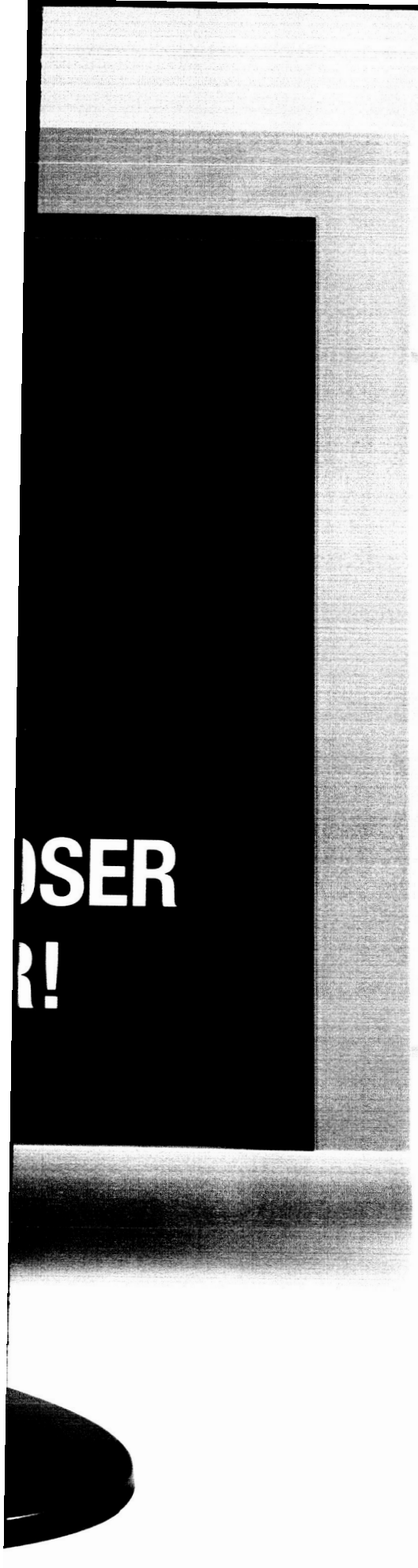

WARNING
PROJECTS MAY BE C
THAN THEY APPE




**I HAD BEEN WORKING FOR TWO YEARS AS THE TECHNICAL PRODUCT MANAGER FOR A LARGE SOFTWARE COMPANY, WHEN THEIR PARTNER COMPANY GAVE ME CALL. THEY NEEDED GOOD SOFTWARE ENGINEERS TO CUSTOMIZE A NEW VERSION OF SOFTWARE, AND THEY THOUGHT I WAS THEIR GUY.**

THEY TOLD ME WHAT THEY WANTED TO DO TO THE software, and they even showed me some prototypes. Their idea was to take the basic software tool that the large company was producing and make it more accessible to the customer. They would do this by building in flexibility based on user skill level and organizational maturity. I thought that was a fascinating approach, and I bought into it in a big way. I decided to leave my job and join up with the smaller company as their director of software engineering.

### YOU'D BETTER KNOW THE SHOT

My former employer was a massive organization with tons of money. They can afford to try, try, and try again—but even they hate to do that. At a startup, you typically have one shot. Screw up, and you're dead.

# MANAGING SOFTWARE HAS A LOT MORE TO D

We were taking a product designed for everyone and trying to make it work for a smaller set of users. Here is something I learned quickly: When you're modifying a commercial-off-the-shelf (COTS) product, it makes a huge difference if you have inside information. Why? Because there are undocumented features, a.k.a. bugs, which only people with an intimate knowledge of the product will know how to work around. It's something to think about when you're going to tailor a COTS product: How extensible is it from the get-go?

We were okay because the person I had recommended to the company, a hotshot at my former company, had gone to work with them. He knew things about the old software that the current team didn't even know. But even with him on our side, we were bootstrapped with limited resources. We needed money, so we had to find a client.

## WHAT THE IDEA LOOKS LIKE

At that point we had our working prototype. (I'm a firm believer that prototypes are the best way to communicate a vision.) We showed it to people and heard things like, "Oh yes, this is interesting." We had a few customers in mind, and we visited all of them. It helps to get feedback from your customers as early as possible (again, the prototype was invaluable). Plus, there's the other side of the equation, which is making certain that your customer has a sense (not a lot of detail necessarily, but a good sense) of the engineering constraints of your development effort.

We found a few small clients willing to pay for some of the research and development we needed to keep going. After a while, we got enough traction to go after the real capital—we needed VC money. That was a scary proposition. We couldn't just go in and say, "Give me some money because I have a great idea." These people were very technical and very sharp. But we apparently said the right things, because in the end we got the money we needed. That took some of the pressure off. We now had enough money to survive on, but a very short period of time (about a year and a half) to develop a product.

## CLOSING IN ON THE DEADLINE

This was early 1999. At that point, I was trying to hire the development team that would take our product to the next level. Because we had to get things done quickly, we couldn't afford to sit around thinking about the design for eight months before we wrote any code. We decided to take a milestone approach, where you basically pick a number of features or a quality bar—something—put a date on it, and run towards that date.

We chose a rapid development programming language as the first technology to implement this software. We did this because the software was more readily modified as customer feedback came in. When we would show the software to the customers, they would say, "I think you should have this," or "Based on my experience it should do this..." Then we could turn around, go to engineering for a couple of weeks and send a version of the software back to the customer that did what they were describing.

## NO TIME TO PLAY

That was helpful early on because we could get the customer's viewpoint almost immediately. It's also a wonderful way to force your developers to buddy up with the Quality Assurance guys and get things done. That's important because the natural instinct for software developers is to treat the project like a box of Legos. They want to take it home to see what they can create. You have to ask, "Okay now, is the customer really going to have a better experience if you put one more layer of lacquer on that sucker?"

Their answer is usually, "What's a customer?"

As a project manager, you have to be able to take these people who think they are just playing with Legos and get them to do things on schedule and on budget. You have to make certain they understand the business purpose for what they're developing. The role of the project manager has to be to articulate business requirements to the developers in a way that gets them jazzed—like reminding them that they're doing something good for the company, or whoever the target

# ITH PEOPLE THAN IT DOES WITH PROCESSES.

customer is in the end. When it comes to managing software, I've found that it has a lot more to do with the people than it does with the processes.

## GETTING TOO CLOSE

Our ultimate constraint was a window of opportunity, which was rapidly closing. Our one competitor was getting ready to release. We had some money, but we still had to do what's referred to in the industry as a "death march" towards our deadline.

I was the development manager, or what is effectively the project manager—and I made the absolute worst mistake a manager can make. As development manager, your job is to delegate, but I couldn't keep my hands out of the coding.

Originally, I didn't think that we had the manpower to separate the functions of management and development. In hindsight, we should have. I was working on a number of low-level subsystems, and other developers couldn't get started on their work until mine was flushed out. I was supposed to be managing these people, and here I had put myself on the critical path.

It was just a crazy time. I lost 20 pounds. My team kept asking, "What happened to our development manager?" I would go into my office before it was light, and I would leave after it was dark. Instead of meeting with them to talk, I was "instant messaging" my leads to get status reports, because I was busy cranking out all of this code.

## A HIGH PRICE FOR SUCCESS

We launched in October 2001, and it was a big success. But I had to take a hard look at myself and ask, "At what personal cost?"

I had previously thought of the developers as the guys who got wrapped up in creating rather than efficiently tailoring a product to the needs of the customer. Now I was faced with the truth: I had become one of the Lego people, obliviously adding layer after layer of blocks. Don't get me wrong—we software people do it because we love it, we have a passion for it. But I had let myself become overtaken by just one aspect of the project rather than the whole. I was losing weight, never seeing daylight, having virtually no person-to-person contact with my team...

It was a great experience, but I wouldn't do it again. I had to figure out the hard way that software itself is not the answer; project management is the answer. It's all the people, processes, and technologies working together. Rather than being one of the Lego people, I knew I needed to figure out exactly how to motivate them so that together we could create products that make good business sense. •

LESSONS

- When tailoring a COTS software product, you should recruit developers who have an in-depth understanding of the intricacies of the product.
- Establishing open communication with your customer is not only intended to understand customer requirements but also to convey challenges you face on the project.

QUESTION

*How does an active project manager effectively delegate and manage without "sticking his hands in" the actual project?*

"I wrote my first piece of software when I was 11," says **COLBY AFRICA.** "When I was 15, I wrote my first production software, which was promptly rewritten by a 13-year-old, who went on to work at the media lab at MIT. The lesson there is that someone is always smarter and younger than you are."

In spite of his prodigious gifts, Africa never intended to become a software developer. At Microsoft from 1994 through 1999, he rose through the ranks to become a product manager—making time to write a little software on the side. Today, he works for Robbins-Gioia, a project management consulting firm in Washington, D.C., and is too busy assisting project teams around the country to write any software code even if he wanted to.

"I have a bunch of developers who work for me now," he says. "Whenever I want to get my fix of software, I have them show me something cool they're working on."