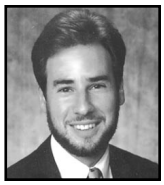

Steve McConnell

Building the Community

In 1984, I began my first full-time programming job as an analyst for a consulting firm with five employees. The job paid well. We got free diet Pepsi. And I got to work on IBM PC projects, which were a lot more interesting than the mainframe projects I'd been working on in school.

The projects were much larger than my school projects, lasting anywhere from a day to a month. I learned a set of skills that I hadn't learned in school. I learned to coordinate my work with others. I learned to contend with a boss who constantly changed his mind about each project's requirements. And I learned to work with customers who depended on the software and actually complained if it didn't work the way they needed it to.

My second job was on a large mainframe aerospace project—a stereotypical document-laden government project. The inefficiency was astounding. I was convinced that three or four of the programmers from my old company could have written in three months what this 30-person project team took three or four years to create. (I still think that's probably true.) One member of the team had a copy of Niklaus Wirth's *Algorithms + Data Structures = Programs*, and he was regarded as the project guru, from all indications largely because he had read this one book. I didn't like this job very much; by the time I left work each day, I happily stopped thinking about computer programming.

Assignment: Fun but Lonely

After the aerospace project, I went back to working in a small-company environment as the only programmer in the office. I began work on an exciting year-long DOS shrinkwrap project in C that pushed the PC to its limits. This project didn't provide free diet Pepsi, but I was happy to be working with microcomputers again. The only hitch was that the new project had renewed my excitement for computer programming, and I couldn't find anyone who was as excited about it as I was.

By this time I had been working full time in the industry about three years and, aside from programming-language and machine manuals, I hadn't yet read any programming books or subscribed to any programming magazines—though I had at least purchased a copy of *Algorithms + Data Structures = Programs*.

The small company I was working for presented itself to the public as the computer programming "A Team." I was told during my job interview that the company had a lot of trade secrets that enabled it to be "The A Team." After I started, I said I wanted to learn the "trade secrets," so my boss handed me a copy of Philip Metzger's *Managing Programming Projects*. I read the book immediately, and was amazed to find that Metzger seemed to have had many of the same experiences I had been having. I had struggled with the planning for my new year-long DOS project. Metzger's book cleared up many of my problems, and I used it as a basis for all the planning I did for the rest of that project.

Clearing the Fog

Shortly after reading *Managing Programming Projects*, I found a copy of Ed Yourdon and Larry Constantine's *Structured Design*. In skimming it I saw an explanation of why I was having such a hard time with the design for my DOS program. I changed my design from transform-centered to transaction-centered, and the whole design fell into place. Then I found Barry Boehm's paper, "Improving Software

Continued on page 126

EDITOR-IN-CHIEF: Steve McConnell • Construx Software • software@construx.com

Copyright © 1998 Steven C. McConnell. All Rights Reserved.

Productivity" (*Computer*, Sept. 1987), which explained in a quantitative way what Metzger's book had explained more subjectively. I started to realize that there might be more information available to help me do my job than I had previously realized.

About this time I foggily recalled some letters my professors had mentioned a few years earlier: A-C-M and I-E-E-E. I didn't have a computer programming degree and didn't feel like a professional programmer, but I decided to apply for membership anyway. I began receiving *Communications of the ACM*, *Computer*, and *IEEE Software*—and *IEEE Software* quickly became my favorite. I discovered articles that addressed the issues I needed to learn about to do my job effectively: how to help customers make up their minds about requirements; how to control complexity on large projects; how to create maintainable code; how to performance-tune code; and so on. Because it was published by the world's largest society of software professionals, the IEEE Computer Society, *IEEE Software* avoided catering to programming fads and hype. The articles weren't as fluffy as the articles in some of the popular magazines I was reading, and I felt they would help me for many years to come.

Prior to joining the *IEEE Software* community, I had viewed programming as just a job. I went to work. I got paid. I went home. I stopped thinking about software.

This was a watershed event in my growth as a software developer. Prior to joining the community of *IEEE Software* readers, I had viewed programming as just a job. I went to work. I got paid. I went home. I stopped thinking about software. After joining the *IEEE Software* community, I began to see that, even if I worked in an office by myself, I wasn't just a lone programmer. I was part of a community of software developers who cared about software development and took the time to share their experiences for the benefit of other software developers.

Professions such as law, medicine, accounting, and engineering have well-defined career paths. You get a specialized degree. You take a qualifying exam, undergo an apprenticeship, or both. If you're good enough, you reach the level of partner, doctor, professional engineer, or other well-defined professional within a preplanned amount of time.

The career path for software developers isn't nearly as well defined. According to a US government report, only about one quarter of the people working as software developers in the US have computer science or related degrees. Compare this to doctors, lawyers, and engineers: virtually every practitioner in those fields has a related degree. It turns out that my working as a programmer without a formal programming background wasn't unusual.

It Takes a Village to Train a Software Practitioner

Considering the extreme variation in education and skill levels in the software world, you might argue that there currently isn't any meaningful software development community, but I think the incredible variation makes it all the more important to build one. With this in mind, last year in Boston the *IEEE Software* editorial board adopted a new mission for the magazine: Building the Community of Leading Software Practitioners.

IEEE Software's goal is to publish articles and columns that put you in touch with other software practitioners. Beyond that, our goal is to provide the information you need to maintain and enhance your skills as a member of the community of competent software practitioners.

Every community is made up of both younger and older members; some are more skilled, some less. One implication of our mission statement is that we must address the needs of leading practitioners both young and old. We must address the needs of practitioners with strong educational backgrounds in computer science, software engineering, and related topics, but also of self-taught programmers—scientists, engineers, accountants, teachers, doctors, attorneys, and others who find that they are now writing programs for a living, even though they never consciously set out to become computer programmers.

Welcome to the A Team

We do not intend to publish run-of-the-mill articles, but articles about cutting-edge, thought-

provoking ideas, practices, and technologies that you need to be a leading software practitioner. Part of *IEEE Software*'s charter is to help transfer leading-edge research results into practice. No other magazine that I know of gives you the chance to see the latest developments presented in a practical way. These really are the trade secrets you need to be on the A Team—except that we don't try to keep them secret.

I have long believed that *IEEE Software* has an important role to play in the software development community, bridging the gap between research and practice, leveraging its association with the world's largest society of computing professionals, and providing a place where the software development community can share its best experiences and ideas. With this issue, I am honored and pleased to accept the role of editor-in-chief of this magazine. Most of the jobs at *IEEE Software* are filled by volunteers (including my job), and we can use all the help we can get. I look forward to hearing from you about how the magazine and I are doing. I hope that you will volunteer to be an article reviewer or submit an article or guest column. You needn't worry about whether you qualify as a professional programmer. If you're reading this magazine, and care about improving software development practices, you'll be a welcome addition to the *IEEE Software* community of leading software practitioners. ❖