We Need the Right Tools for Quality Software

Several years ago, I was supporting a software development effort at a commercial company. My boss decided to add to my responsibilities by having me develop part of the software using a database manager. I did not have any experience with database development, and my management didn't provide me with formal training. However, they did provide me with tutorial textbooks, time to learn, and a customer support contract with the software company so that someone could help me as I needed questions answered. The end result was my software delivered on time, on budget, and with no errors reported from the customer.

Not long after that, I was hired by another organization that wanted me to support their database development. However, they wanted me to use an application of the database manager that I had never used before. I told them that I had never used this application, but I knew about it and wanted to learn it, so they promised to give me that opportunity. I naively assumed that meant the same arrangement as my previous employer: books, time, and customer support. I was wrong. I was given the books and told that I was expected to learn the application on my own time (in addition to the eight hours of uncompensated overtime that they also told me I was expected to work each week). No one else in the organization was familiar with this application, so I also didn't have anyone to answer questions. The results weren't as successful as at my previous employer. I finally left the company while the project was behind schedule, and we didn't know when we would be finished; I don't know what the end result was.

The articles in this month's issue provide some good insights on things that can be done to help an organization develop quality software. I would like to add to these ideas by reminding managers to provide their people with the tools needed for the job. By tools, I'm not recommending more software packages; I'm recommending training, technology, and appropriate time.

Getting to this month's articles, David N. Card starts us out with *Managing Software Quality With Defects*. Card is a leader in the software community on measurement, and he shares his insights for quality measurements in this article. I believe in the concept that if you can't measure it, you can't manage it. If we want quality in our software, then we need to track and manage the quality while the software is being developed.

Next, Dr. Kenneth D. Shere gives an overview of Lean Six Sigma (LSS) in *Lean Six Sigma: How Does It Affect the Government?* He starts his article by discussing what LSS is, then goes on to discuss the benefits realized by several organizations using LSS, and how it can be used for acquisition.

Gary E. Mogyorodi used to work with Richard Bender, the developer of requirements-based testing. Mogyorodi shares the knowledge he gained from this association in *What Is Requirements-Based Testing?* If you would like the opportunity to get more information directly from Mogyorodi, he will be speaking at this year's Software Technology Conference, held April 28-May 1 in Salt Lake City. Don O'Neill also shares his insight in *Determining Return on Investment Using Software Inspections*. If you're not already using software inspections, you're missing the boat; I hope this article will motivate you to start.

The supporting articles in this issue should also provide some useful pointers. Dr. Randall W. Jensen shares his first encounter with pair programming in *A Pair Programming Experience*. Jensen discusses how this experiment was arranged, and how it turned out. We also share a discussion of software architectures in Ingmar Ögren's *Clarify the Mission: A Necessary Addition to the Joint Technical Architecture*. Lastly, Gary E. Mogyorodi presents 20 questions used to determine the maturity of an organization's quality assurance and testing environments in his online article *Let's Play 20 Questions: Tell Me About Your Organization's Quality Assurance and Testing*.

I hope the managers and acquirers who read CrossTalk are using the insights shared by software industry leaders to enable their people and contractors to develop the software they are capable of developing. This requires working with those developers to determine realistic delivery schedules, providing resources for necessary training, and providing other support needed to get the job done right the first time.

Elizabeth Starrett

Elizabeth Starrett
Associate Publisher