Managing Technical People (When You're No Techie)

by Elisabeth Hendrickson

DURING THE "MENTORS, MODELS, AND the Making of Managers" panel at a recent software management conference, a participant asked, "Do you need to be technical to manage technical people?" The panelists were united in their answer: "No." I agree, and I would like to explore the question in a little more depth. To what extent do technical skills help managers? And what other skills help managers even more?

Susan manages a team of developers who are working in C++ on Windows. The last time she programmed, it was in COBOL on a mini computer. No one in her office even knows the term "mini computer." She originally felt out of touch technically with her people. But despite her concerns, she has been successful in leading her department because she has earned the respect of her developers.

Frank, a highly successful sales manager, was promoted to be the vice president of a software engineering organization. The executive staff wanted him to bring his gung ho attitude and customer focus to the development group. Frank eagerly accepted the challenge—then began wondering what he'd gotten himself into. But soon he found that the non-traditional skill set he brought to this position turned around a floundering department.

Cheryl manages a team of testers working on billing software. She doesn't have much test experience, but she has more than a decade of experience working with billing systems. She was concerned that her lack of test experience would make it difficult for her to manage and guide her testers; however, her domain expertise gave her a different perspective on the problem and

made her a very effective manager.

Susan, Frank, and Cheryl had a common problem: they didn't understand the day-to-day details of what their people were doing. They worried that their lack of understanding would undermine their ability to manage.

So what can you do if you're in a similar situation to Susan, Frank, or Cheryl? How do you manage technical people when you're either not technical or not up to date on current technologies?

Acknowledge What You Don't Know

The first thing that Susan, Frank, and Cheryl did was admit what they didn't know. They understood that misrepresenting their skills to their staff would backfire.

I once worked with a non-technical manager who kept programming books on his shelf so he'd look more technical. I was fooled for a little while. Then I discovered that he didn't understand basic programming constructs like loops and case statements. He was trying to bluff, and I lost all respect for him. He would have been better off admitting that he didn't know how to program. Instead, I began wondering what else he had hidden.

Use What You Do Know

Frank knew how to motivate people. Managing sales people is a motivation-intensive position. He didn't know a macro from a make file, but he knew what the customers needed and how to motivate the developers to give it to them. The developers much preferred working for Frank rather than for his predecessor, a process-oriented former developer who spent most of his time developing waterfall-like process models instead of talking with his people. Frank helped them understand what the customers needed and improved the relationship between sales and development.

Susan knew a lot about how to build software. She knew how to manage risk

and had good instincts about when to delay a release and when to ship. She also used her past experience as a developer in negotiating with the executives to get the engineering organization better salaries, more space, and better equipment. She didn't have to know much about the current technology to be a good advocate for her group.

Cheryl knew just about everything there was to know about billing systems. She used her knowledge in prioritizing testing tasks and assessing risk. Her domain expertise was more valuable to her than a broad background in testing would have been.

Any job has numerous facets. If you are a non-technical manager in charge of technical people, don't despair. You achieved your position because of your strengths, not in spite of your weaknesses. Your task is to figure out how you can use your strengths in managing your group.

As an employee and a consultant, I've worked with more than a hundred managers. All of them achieved their positions by demonstrating their capabilities in a variety of areas. The ones who succeeded in their positions did so by being honest about what they knew, using their skills to the full extent of their abilities, and actively seeking to grow their skills in new directions. Not understanding the details of the technology doesn't get in the way if you use your strengths well.

Ask Questions

If you're too busy hiding your lack of knowledge, you won't feel comfortable asking questions. You'll think you should already know the answers. Let me tell you a secret: no one knows all the answers. No one. Not knowing the answers isn't a sign of weakness—not asking questions is.

Frank was particularly adept at asking questions on unfamiliar topics. His favorite phrases seemed to be "Help me understand…" and "Explain this to me." His technique worked. He didn't

QUICK LOOK

- Ways to work around your technical shortcomings
- Building on your strengths

This article is provided courtesy of *STQE,* the software testing and quality engineering magazine.

Career Development

always understand the answers to his questions, but he kept asking questions until he got the answers in terms he could understand.

When asking questions, it's important to ensure that you are genuinely asking a question and not simply phrasing a statement in an inquisitive way. Just because a phrase sounds like a question doesn't mean it is. The phrase "What do you mean, we're going to be late?" isn't a question. That's a statement that means, "You idiot. We can't be late." Frank never asked rhetorical questions. When asking questions he accepted the information he got whether or not he liked it. That's why Frank's staff never balked at answering his questions; Frank queried them in a way that indicated he really wanted to know the answers.

Asking questions helps you understand what your staff does on a day-to-day basis. It also enables you to gauge the effort your staff put into solving a given problem. Finally, asking questions causes your staff to question their own assumptions—and that helps them improve their own work.

Focus on the People

No matter what the technology, people are at the heart of software development. That means that as a manager, your ability to work with people is far more important than your ability to sling code. As Jerry Weinberg says, "Every problem is a people problem." If you're a non-technical manager, you may have strong people skills that your technical counterparts do not. Play to your strengths; focus on the people.

Both Susan and Frank focused on their people. Susan worked hard to get her best developers rewards they hadn't previously received. Frank encouraged communication and strengthened the re-

lationship between development and other departments. Your people skills can help you lead and manage, and can also point out potential problems.

Without understanding the content of a technical discussion, you may be able to spot technical risks by listening to the tone in meetings and watching postures. You can tell a lot by watching people interact. Perhaps your lead programmer is an intellectual bully or your best tester has stopped sharing information. If you were focusing on the technical details of the discussion, you might miss the subtle body language or masked hostility that could warn you of communication problems.

Learn Something about the Work

Cheryl made a point of taking testing classes and reading books on software testing. If you're truly non-technical and find yourself in charge of a technical organization, it's a good idea to learn something about the work your people do. You don't need to become so proficient at it that you could take over for your people. You just need to understand the ins and outs of the job well enough that you can anticipate what your people need and understand the importance of what they're saying.

So where do you get technical experience?

First, decide what technology would be most beneficial for you to learn. You might start with programming, testing, system administration, databases, networking, or a variety of other technical skills. If you're unsure what to learn first, consider asking your people, "I'd like to become more technical. Where do you suggest I start?"

Consider finding a class in the technical skill of your choice. Community colleges and university extension programs

offer a wide variety of such courses. You can also find online courses through companies like Digital Think (www. digitalthink.com). Although you could try to learn to program from a book, I've found that formal classes are often better for beginners. You can ask questions in a class. Further, the interaction with other students can help speed your learning.

Conclusion

There's a lot more to managing than understanding the technology. Do you know how to elicit requirements from users? Do you work well with management? Do you have a knack for asking the right questions at the right time? Not knowing where to put the semicolons in a line of code isn't a big deal. Knowing how to lead people, now *that's* a big deal.

You don't have to become a super geek to succeed in an engineering organization. You do need to bring your own unique talents and skills to the table and continue to add to the value you bring. Whatever your personal strengths are, they will serve you well in your role. It's up to you to figure out how. STQE

Elisabeth Hendrickson (esh@qualitytree. com) *is an independent consultant specializing in software quality assurance and management, with twelve years of experience working with leading software companies. An award-winning author, Elisabeth is a frequent speaker at major conferences on software quality and management. You can read more about her ideas on quality and testing at* www.qualitytree.com.