EDITOR DAVID ALAN GRIER
George Washington University; grier@gwu.edu

Do We Need a Software Czar?

David Alan Grier, George Washington University

We need to temper authoritarian leadership with the insights of individual programmers.

In my travels through the world of technology, I've met many people who believe that software has gone astray and that we desperately need to take action to put things right. My colleague Curtiss is one such person. He believes that our efforts to produce open, flexible, and reusable software have utterly failed. Instead of developing a body of software that's getting better and better, he says we've been creating programs that are brittle, incomprehensible, and nearly impossible to change. Our efforts to solve this problem with open standards and the free flow of information have been misguided. According to him, we need to appoint a software czar, someone vested with absolute authority to bring some order to the field.

I suspect that Curtiss would like to be that software czar. I also suspect that he'd like me to help him get this position. I acknowledge some of the flaws he sees in software, but I'd argue that we need to be more subtle in our leadership of software projects. We need to temper authoritarian leadership with the insights of individual programmers.

We start designing software by drawing boxes on a whiteboard that represent different system modules, and then we divide our programming staff into groups that correspond to those boxes. Each of these groups is responsible for their own code. They can't concentrate on their assignment if they have to know every detail about every other module. Yet, the quality of the system will be determined by how well those groups and the modules they create work together.

> The connections between software modules are more than technical specifications.

Pioneering computer scientist David Parnas famously observed that the connections between software modules are more than technical specifications. These connections are assumptions that the modules make about each other—or, more accurately—assumptions that programming groups make about the work that other programming groups are doing.

Although the free exchange of information often helps clarify assumptions, it can also have side effects. It can allow individual programmers to impose their ideas upon others. Programmers can find unintended uses for libraries, exploit coincidences in structures, and conclude that data follows a pattern where none exists. Through any of these actions, programmers can introduce an unanticipated assumption into the system.

Over the past few decades, we've increasingly used open, market-based techniques to manage software projects. However, markets don't anticipate change well, and open information can hide important facts. Software development projects can't be successful without some form of leadership, which helps a programming team decide which ideas will produce a strong, flexible system and which won't. Many developers find it difficult to provide that judgment, as it's easy to let a project drift from one extreme to another by either dictating every detail of the system or by accepting every contribution from every programmer.

One successful CTO claims that his software teams produce the best work when his leadership style is closer to the role of a judge. "If you want to make any change to the plan," he told his team, "you have to bring it to me and convince me that it's a good change. If you believe the idea isn't worth bringing to me, then it isn't worth putting in the system." However, this role isn't as easy or as passive as he makes it sound. He gets his team together to make sure that information about the design has been thoroughly reviewed and is understood by all. He also reviews the code to make sure the developers are following his direction. He's a strong leader but not the type who would dictate the future of software—he's not a software czar.

See **www.computer.org/computer-multimedia** for multimedia content related to this article.

DAVID ALAN GRIER is the author of *When Computers Were Human* (Princeton Univ. Press, 2007) and the producer of the new podcast series "How We Manage Stuff" (http://itunes.managestuff.net). Contact him at grier@computer.org.

0018-9162/15/$31.00 © 2015 IEEE