# An Ounce of Prevention …

by Matthew Heusser

*"Trying to improve software quality by increasing the amount of testing is like trying to lose weight by weighing yourself more often…. If you want to lose weight, don't buy a new scale; change your diet. If you want to improve your software, don't test more; develop better."*

— Steve McConnell in *Code Complete*

**Matthew Heusser believes prevention is the best cure for bugs.**

THE BEST WAY TO ENSURE THAT THE CUSTOMER never sees a bug is to never create one in the first place. Yet, improving the quality of development, or "developing better" as McConnell calls it, is rarely even discussed. I remember one annual review in which I gave myself a poor score for creating too many bugs. My manager responded, "Yeah, well, everybody has bugs." That's the crux of the problem. Bugs are expected. Everybody has them. We've created "testing" to help find these bugs so that we can "cure" software applications before they ship.

We all know that prevention would be better. Studies show that the cost of fixing a defect grows exponentially over time, and the earlier in the lifecycle the defect is detected, the cheaper it is to fix. Shifting the focus of the industry away from cure and toward prevention requires an entirely different mode of thinking, but it's the next obvious step. The software industry has survived other big mindset changes. We've moved from structured thinking to object-oriented thinking, from mainframe to desktop to client/server to Internet, from text-based to GUI. In fact, in many cases, developers were the ones driving the changes.

Many people in the industry are already "developing better." Agile methods admit that humans make mistakes and then focus on compensating for those mistakes when they happen. Techniques such as pair programming and two-person design make review a creative teaming endeavor instead of an IRS-style audit or a teacher with a red pen. Source code control systems stop developers from "stepping" on each other's code. Hallway usability tests step through the user interface of the software before it is even written. Test-driven development forces developers to focus on inputs, outputs, and boundary conditions of the software before they write a single line of code. Project velocity helps with improving estimates over time and picking realistic ship dates. Iterative software models get a simple working system into the hands of testers fast, then build onto that system, instead of discovering all the integration, logic, and other errors in one big chunk of time at the end of the project.

All too often, "quality problems" are blamed on the mysterious "quality people" instead of the person who actually introduced the defect. If we are to move toward prevention, management must create an environment and culture of quality. Developers must choose to do the right thing instead of the easy thing because they believe doing the right thing now will save time and effort later. To ensure quality requirements, design, and code, the tester role must become one of a quality champion who is involved in the project from the outset. It's surprising that it has taken us so long to adopt a prevention mindset—the idea is not new. The automotive industry and most of American manufacturing has been dealing with it for decades. Maybe in companies that build *things*, the pile of scrap that is thrown away after rework makes its own point. In software, with such intangible products, it may feel far too easy to do sloppy work and "put a patch on the website if we have to." Moving toward prevention is a challenge. Yet, if we are to win in this increasingly competitive world, it is a challenge we must rise to meet. **{end}**

---

*Matthew Heusser is a programmer/analyst at Priority Health, where he is also chair of the QA committee for software engineering. He can be reached by email at heussers@datawise.net.*

SANTA FABIO/REDUX PICTURES