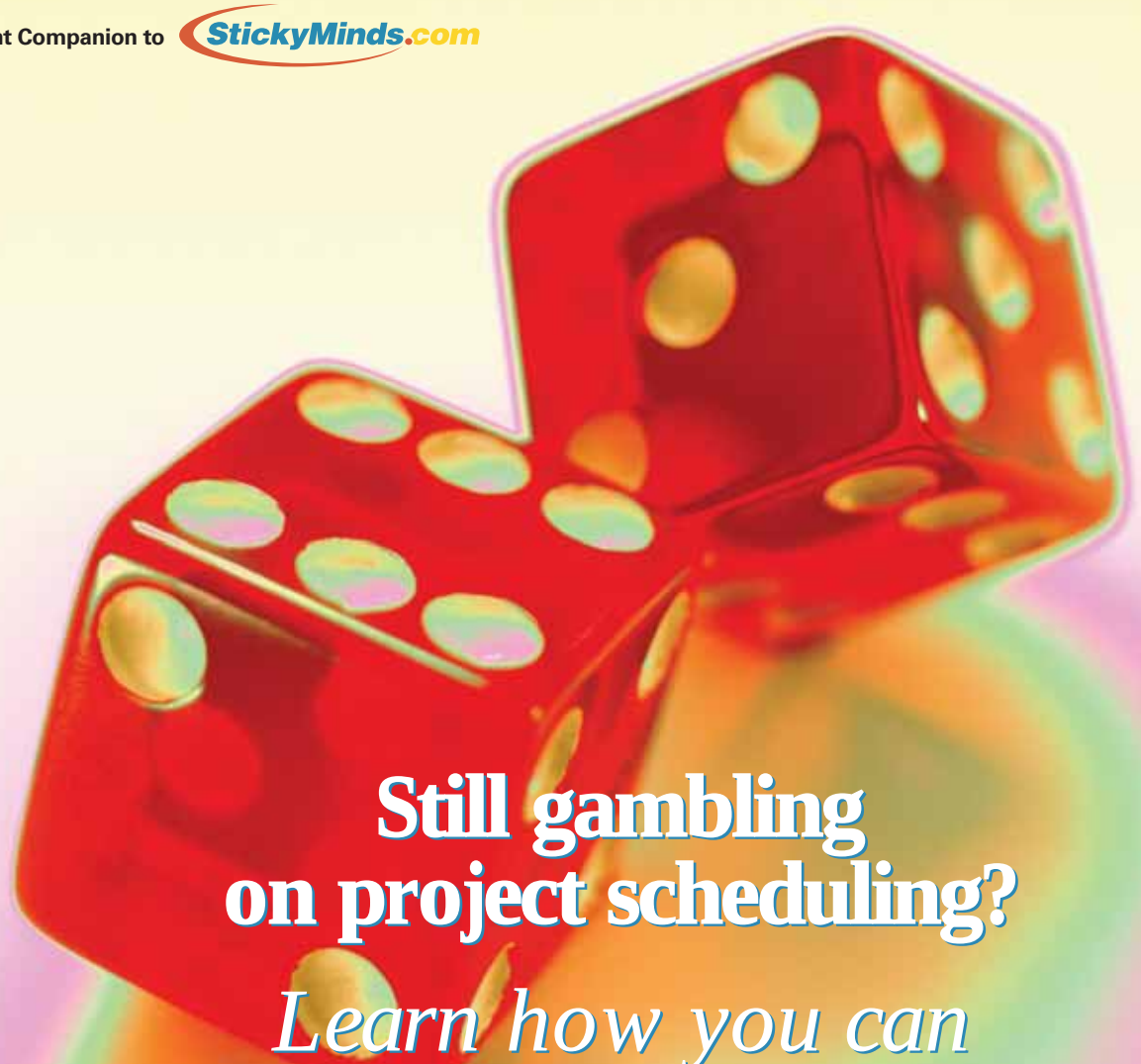
March 2007 $9.95 www.StickyMinds.com

BETTER SOFTWARE

ACCEPTING EXCEPTIONS
. . . and putting them to good use

ESCAPING THE DEATH SPIRAL
Take time now, save time later

The Print Companion to StickyMinds.com



Still gambling on project scheduling?

Learn how you can

Beat the ODDS!

The Proof of the Pudding…

by Michael Bolton

GETTY IMAGES

The first job in which I was called a software tester was at Quarterdeck, but I was a tester there long before I had the official title.

I worked in the technical support department. The company produced two interrelated products, DESQview and the Quarterdeck Expanded Memory Manager (QEMM-386). They were marketed separately, as well as together under the name DESQview 386.

The age of DOS on the Intel 386 processor family was as unruly as the Wild West. Applications could write directly to the display hardware, grab control of the mouse and the keyboard by using BIOS calls instead of going through the operating system, and allocate all of the memory on the system without so much as saying, "Excuse me." There were a dozen software interfaces for accessing the physical memory of the machine, but because of DOS's heritage on earlier processors, the most important kind of memory was in short supply.

QEMM-386 was a memory manager for the 386 family. Its original purpose was to supply memory to DESQview and to all the applications running under it. DESQview was a multitasking environment that supported regular, off-the-shelf DOS applications, of which there were hundreds. The applications didn't have to be written for DESQview or even be aware of it. The 386 processor introduced a number of hardware features that QEMM and DESQview together could exploit and manage, such as the ability to supply a virtual display that fooled applications into thinking that they were writing to the real display. These features expanded DESQview's power enormously. QEMM also gained a life of its own, supplying memory to DOS programs even when they weren't running under DESQview.

For the longest time, there was no department called "testing" or "quality assurance" at Quarterdeck, yet DESQview 386 was startlingly compatible and robust, especially considering all the applications, hardware, and (non-)standards that it had to support—plus it sold like hotcakes, suggesting that customers clearly valued it. So, how did we test these products effectively and achieve such a high level of quality? The answer is that we had a valuable and powerful stand-in for more formal testing approaches: We *used* the products, every day, all the time, just like our users did.

DESQview and QEMM were highly interdependent. Because DESQview exercised QEMM's memory management features extensively and QEMM provided so many important services to DESQview, each product served as a useful oracle—a principle or mechanism by which we identify problems—for the other. A problem in QEMM had a good chance of exposing itself through a problem with DESQview, and vice versa. *Excellent testing depends on powerful oracles.*

DESQview had a big head start on Windows in terms of multitasking. In the technical support department—as anyone who has worked in one will tell you—we needed a multitasker to do our work effectively. We updated the customer database records, searched the knowledgebase, exchanged email, and took notes. When a customer reported problems with a program, we could install and run that program under DESQview while we were on the phone, even with all of our other programs open. Some tasks were complex, and the order in which we did them wasn't regular or predictable. As a result, we sometimes encountered problems that we wouldn't have seen had we always been doing things in the same order. *A diversity of workflows tends to expose sequence-related bugs.*

Another key advantage, from a testing perspective, was that it was early enough in the age of the PC that employees had come to the company from a wide range of backgrounds. Most of us had been computer users of one kind or another, but we had diversity going for us in other dimensions. There were database designers and programmers, film critics, musicians, retail salespeople, bulletin-board system operators, artists, network administrators, and programmers. We found and fixed many of problems because we were set up to see them. *Diversity of the test team and its ways of using the product makes testing more powerful.*

All of us at Quarterdeck used DESQview 386 as it was meant to be used—in a variety of user models, for a variety of tasks, and in challenging environments where plenty of things were going on. Our salespeople, marketers,

network administrators, and graphic designers had different tasks to perform and applications to run but ran them under DESQview, too. *Using the product under real (or realistic) circumstances tends to reveal problems—and tends to help identify the kinds of problems that are important to real users.*

Quarterdeck grew slowly in its first ten years, and in that time, the company didn't have money to buy test platforms. Most of the computers that we used in the support department were provided to us on an exchange basis; we gave software to hardware vendors for compatibility testing, and they gave us hardware in return. *Consider alternate strategies for obtaining test resources.*

Few machines had covers on them, since we were forever swapping video cards, network cards, and other peripherals. Quarterdeck's IT managers recognized the information value of the testing that we were doing and allowed each of us to choose our own tools. Some of us preferred the very latest versions of Microsoft DOS, some the oldest, and some liked third-party offerings like Digital Research's DR-DOS. Some of us used DOS command-line enhancers, like 4DOS, or other utilities, like Sidekick or XTree. We used different personal information managers, word processors, databases, spreadsheets, utility packages, and so on. Rapid testers define the platform for a product as "anything on which the product depends that is outside the scope of our current development project." DESQview depended on the hardware platform and the operating system, but its success also depended on its compatibility with applications that ran under it. *Diversity and responsiveness to change are keys to good platform and compatibility testing.*

QEMM came with a support tool called Manifest. With Manifest, we could see how memory was allocated, how DESQview and QEMM were set up, and a lot of things about the state of the operating system and the hardware. If we had trouble or questions about the system, Manifest could help us get answers. *Visibility into the workings makes a program more testable.*

In addition to all of this, there was an application programming interface (API) to DESQview, upon which some of our support applications, like our fax client, depended. Programmers (and support people who programmed) used the API to write DESQview-specific programs and thereby tested DESQview. *A scriptable or programmable API to a product makes it more testable. One of the more important users of your software might be another piece of software.*

Quarterdeck wasn't a big company in those days. Whenever anyone in the company had an issue with QEMM or DESQview, it was easy to get access to the developers—we'd simply walk over and demonstrate the problem. Our developers were very responsive, since we could quickly explain the importance of the task and, to their credit, they were eager to help solve any problems. *Shortening the feedback loop between developer and end-user is a powerful force for beneficial change.*

In my experience, companies tend to do better work when they're engaged in using the products that they're producing. A product that's useful to us is at least useful to someone. We might choose to create elaborate plans, scripts, test automation, or even testing departments, and those things can be valuable. But we get real information—important information—when we use the thing ourselves. **{end}**

Michael Bolton lives in Toronto and teaches heuristics and exploratory testing in Canada, the United States, and other countries worldwide as a consultant and co-author of James Bach's Rapid Software Testing Course. Michael is also program chair for the Toronto Association of System and Software Quality. He is a regular contributor to Better Software *magazine. Contact Michael at mb@developsense.com.*

How would things be different if the people who produce products or provide services were obliged to be their own customers? Got any good stories?

Follow the link on the StickyMinds.com homepage to join the conversation.