April 2005 $9.95 www.StickyMinds.com

BETTER SOFTWARE

The Print Companion to StickyMinds.com

PLUNGE IN
Automating tests with Watir
PAGE 40

DEPEND ON IT
Injecting testability into your designs
PAGE 26

IT TAKES TWO TO TANGO

What every software manager should know about pair programming and how to implement it without missing a step. PAGE 34

A publication of
SOFTWARE QUALITY ENGINEERING
www.sqe.com

The Pleasure of Finding Things Out

by Michael Bolton

My interest in science is to simply find out more about the world, and the more I find out the better it is. I like to find out. —RICHARD FEYNMAN

Every now and then someone asks me what I do, and I reply that I'm a software tester, and that I teach testing, and that I write about testing, and that I *love* testing. At that point, some people look at me as though I have tulips growing out of my ears. And at *that* point, I usually feel as though a little explanation is in order, so I tell them about Richard Feynman.

Many years ago, I saw a *Nova* episode called "The Best Mind Since Einstein?" that featured a profile of Dr. Feynman, perhaps the greatest physicist of the latter half of the twentieth century. His career more or less started at Los Alamos, on the project that resulted in the development of the atom bomb, and was capped with his participation in the investigation into the crash of the Challenger, an account that appears in his book *What Do You Care What Other People Think?* Through most of the period in between, Feynman was a professor at Caltech and worked on some of the hardest physics problems of his time, including quantum electrodynamics. He was famous for having developed "Feynman diagrams"—little drawings that simply and clearly illustrate the interactions between subatomic particles, time, and energy. He had a particular genius for reframing ideas and numbers in ways that allowed previously intractable problems to be solved.

Feynman's first collection of memoirs is called *Surely You're Joking, Dr. Feynman: Adventures of a Curious Character*. There are at least three possible interpretations of that title. The adventures themselves had a curious character to them; Feynman himself was eccentric or a curiosity; and for sure, Feynman was curious—relentlessly, eagerly, wonderfully interested in the world. He wanted to know how things worked, and he took an inordinate joy in discovery. He told delightful stories about learning and discoveries. These stories often involved discussions between adults and children. Feynman was lucky enough to have a very good amateur scientist for a father. In particular, the elder Feynman was wise about the extent of his own knowledge. He pointed out that people often bandied about names for things that they didn't understand, so he encouraged his son to learn about the nature of things, and not just to learn the words for them. The younger Feynman grew up in a household that encouraged and valued asking questions and attempting to answer them with experiment and observation.

ANNE SMITH/ILLUSTRATION WORKS/GETTY IMAGES

I've often described Richard Feynman as the patron saint of software testers. Like a tester, he took great pleasure in pulling off tricks that others couldn't. In *What Do You Care What Other People Think?*, he describes his career as a safecracker. He became something of a legend at Los Alamos, a highly "secure" installation. The staff kept important papers in identical safes with rotary combination locks. Every now and then someone needed something that someone else had locked away. Feynman found that he could get into the safes, but not through any magical manipulation of the tumblers. He had managed to find a little documentation on the safes, which included the default combination. Feynman soon figured out that a significant number of people never changed the default combination. He also came to realize that they tended to choose obvious combinations based on things that were memorable to them, such as birthdays, and that they often left the combinations written down somewhere in or near their desks. When no useful information was obviously available, he tried lots of other techniques: listening, feeling, and, above all, experimenting. Besides using the brute-force approach—trying all possible combinations—he also found that one didn't have

to be precise when manipulating the dial; getting within three numbers (out of sixty) would do, which reduced the size of the problem space. Come what may, he was always able to get the safe open eventually, even when the owner wasn't around.

In this story alone, it should be easy to see that Feynman had all kinds of testing skills. He was very good at developing and using heuristics—fallible guidelines for solving a problem—picking them up, using them, and then dropping them when they were no longer useful. He sought clues and information and tried lots of different approaches. He didn't needlessly restrict himself to "fair" ways of solving a problem; he was happy to cheat and snoop and hack his way around it. He thought nimbly and scientifically, experimented readily, and was always ready to work out a harder problem next time.

He also liked to tell the story of a conversation that he had with an artist friend. The artist derided scientists for looking too closely at things like flowers and not appreciating their intrinsic beauty. Feynman argued that the beauty was available to anyone, artist and scientist alike, but that the scientist had a more comprehensive understanding of the nature of the flower: its cells, its processes, its reproductive system, its different colors under black light, colors that insects can see, leading to more questions—might insects have an aesthetic sense too? As Feynman said, ". . . all kinds of interesting questions which a science knowledge only adds to the excitement and mystery and the awe of a flower. It only adds. I don't understand how it subtracts."

Amen. I like to think that, when we're at our best, we're testers because we want to know how things work; we want to appreciate things on a deeper level of awareness; we want to interact and experiment with them; we think like scientists. We want to learn and to appreciate and expand and show off our knowledge. We want to observe things from plenty of angles and to connect things with other things. Most importantly, we want to ask questions that help us understand the nature of the things that we're testing. When we're testing well, our brains are turned on and we're connected with the world.

That will be the motivation for this series of columns. I'm going to focus on something that Cem Kaner calls the investigative side of testing. Testing has a confirmatory aspect too, in which we make sure that things work as they're supposed to. That's important work, for sure, but there's already a lot of literature on that subject. So in these columns, I'll focus on investigation—anticipating and identifying risks, searching for problems, and expanding notions of coverage. My goal is to help you think critically about software, not disparagingly, but comprehensively, as a good literary or film critic would. That will require us to look at many different aspects of the program and to examine the program from a variety of perspectives. Because software appears in so many different contexts, we'll need to develop skills that help us choose appropriate techniques and use them wisely—and invent new ones when we need them. Most importantly, I'll examine the fastest, least expensive way to fulfill the testing mission, something that my colleague James Bach and I call Rapid Testing.

We testers are lucky: Our organizations and our managers hire us specifically to give them information about the software we test and to shine light in all of its dark corners. Like Feynman, we have the pleasure of finding things out. **{end}**

My goal is to help you think critically about software, not disparagingly, but comprehensively, as a good literary or film critic would. That will require us to look at many different aspects of the program and to examine the program from a variety of perspectives.

Times haven't changed much. Some people still don't change their defaults; they use obvious passwords and leave information lying around. Problem solving hasn't changed either. A rough approximation can be sufficient to solve a problem well enough, and the more approaches you take to it, the more likely it is you'll solve it. The test of a theory is an experiment, and if the experiment doesn't fit the theory, then the theory is wrong and it's time for a new one. As testers, we need to continue to create and test theories about our product, all the time.

Feynman was never afraid to look silly, and perhaps in part because of that, he never stopped learning new things. Over the years, he kept adding skills to his repertoire of science and safecracking, including drumming, dancing, and drawing.

Michael Bolton lives in Toronto and teaches James Bach's Rapid Software Testing course all over the world. Contact Michael at mb@developsense.com.