September/October 2009 $9.95 www.StickyMinds.com

BETTER SOFTWARE

The Print Companion to StickyMinds.com

ROOM TO GROW
Cultivating test diversity

STAKEHOLDER REQUIREMENTS
Think globally and locally

Food for Thought

by Michael Bolton

One can learn lessons about software testing and general systems thinking in surprising places. As a gift, I recently received the book *The Omnivore's Dilemma* [1] by Michael Pollan. This engaging book explores many dimensions of food and our relationships to it—how we produce it, how we obtain it, and how we consume it.

One section of the book tells a story of agriculture and *reductionism*, an approach to science that reduces complex things to much simpler components and interactions. In the nineteenth century, Baron Justus von Liebig determined that soil fertility and plant growth depended on just three chemicals: nitrogen, phosphorus, and potassium—abbreviated as N-P-K, from their initials on the periodic table. Nitrogen encourages chlorophyll production and growth, phosphorus helps in the growth of roots and blooms, and potassium is vital to the electrolyte balance that keeps the cell machinery working. Von Liebig's research led to the development of artificial fertilizers that made agriculture vastly more productive. With the N-P-K approach, plant nutrition and fertilization became a simple matter: Choose the effect you want, dump onto the soil more of whatever the plant is lacking chemically, and watch the crop yields go up.

In the twentieth century, agronomist Sir Albert Howard spent thirty years working with peasant farmers in India. He observed the interaction between the crops, the livestock that helped till and fertilize the soil, and the people who tended the whole system. Howard recognized agriculture as something far more complex than a chemical recipe, noting that "the health of soil, plant, animal and man is one and indivisible." In his 1940 book, *An Agricultural Testament*, [2] Howard points out important weaknesses in von Liebig's model. Artificial fertilizers disrupt the balance of natural processes that produce humus, the complex by-product of the breakdown of organic material. Humus provides minerals for plants and food for microorganisms that are symbiotic to plant growth. Humus also helps the soil retain water, prevent erosion, and keep toxic heavy metals from entering the food chain.

ISTOCKPHOTO

As the statistician George Box said, "All models are wrong; some are useful." [3] Von Liebig's model of agriculture was very useful in that it led to explosive improvements in crop yields, yet it was also wrong in dismissing the role of humus and reducing biology to chemistry and living systems to mechanisms. Reductionist strategies can be very successful for some time, but eventually the system goes out of balance. Plants shoot up when given big doses of artificial fertilizers, but they also become vulnerable to disease and insects, which farmers attack with chemical pesticides that further disrupt the natural balance. Nitrate-based fertilizers run off into lakes and rivers, promoting the growth of algae, depleting oxygen, and killing fish. Increased productivity leads to lower food prices, so that, ironically, the more productive farmers are, the less money they can get for their crops. It's not at all clear that von Liebig anticipated these effects. As Pollan says, "Once science has reduced a complex phenomenon to a couple of variables, however important they may be, the natural tendency is to overlook everything else, to assume that what you can measure is all there is, or at least all that really matters." [4]

This pattern repeats itself in many approaches to testing and test management. No software product stands on its own; every product is a participant in a complex system of interactions between people, computers, and other software. For anything that we test, we face a dilemma. On the one hand, we need to reduce the complexity of the test space so that we can better understand what we're testing. On the other hand, we run the risk of oversimplifying the test space and missing important aspects of the product's relationships with other elements of the system.

Later in the book, Pollan goes on a hunt with a trio of mushroom aficionados. The quarry in this case was the "burn morel," which flourishes in the first spring after a forest fire. These morels are hard to find; they're small, and they closely resemble the burned sapling stumps and pinecones that cover the

"As testers, we have models, techniques, strategies, and biases. All of them are sometimes helpful, but each also is capable of misleading us."

landscape in which they grow. Pollan describes the patterns and techniques that the experts use for the search. One involved approaching the mushrooms from a low angle so that the heads would be visible; another involved looking near the dogwood plants that thrive in soil of the same temperature as morels; another was staying at the same altitude as previous finds that week. "I could see why you would want theories to organize your hunting," Pollan says. "The theories told you when to intensify your attention, scrupulously combing the forest floor with your eyes, and when you could safely rest it. For the hunter-gatherer, high-quality attentiveness is a precious but limited resource, and theories, by encapsulating past experience, help you to deploy it most efficiently."

That reminded me of focusing strategies that we use in testing: trigger error conditions, go backward and forward, force state transitions, try 'em all if you can, consider the opposites, overwhelm the inputs. But then the experts spoke again about an overriding heuristic: When hunting mushrooms (or hunting bugs) you should "be prepared to jettison all previous theories and go with whatever seems to be working in this particular place at this particular time. Mushrooms behave unpredictably, and theories can only go so far in pushing back their mystery." [5]

Bug hunting, for me, works the same way. As testers, we have models, techniques, strategies, and biases. All of them are sometimes helpful, but each also is capable of misleading us. We never know which one is going to prove most fruitful in a given testing situation, because bugs by their nature are unpredicted and unpredictable.

Still on the mushroom hunt, Pollan describes learning how to spot the morels—a phenomenon known both to mushroom hunters and psychologists as the "pop-out" effect. "…when we fix in our mind some visual quality of the object we're hoping to spot—whether it's color or pattern or shape—it will pop out of the visual field, almost as if on command." Humans acquired this pattern-recognition capability through evolution; it's essential to survival to be able to spot food in a complex and chaotic field. Can we take advantage of that evolutionary adaption to find bugs? I think to a great degree we can, and we can strengthen it by practice and priming. Emotional triggers like surprise, confusion, frustration, impatience, or fear provide important clues that point to the existence, meaning, or significance of problems. Reviewing, discussing, and celebrating bugs (as James Whittaker suggests in *How To Break Software* [6]) helps prime us to spot patterns of familiar problems. When we teach Rapid Software Testing, we recommend that people memorize, review, and practice using the Heuristic Test Strategy Model [7] to link test techniques, product elements, and quality criteria. Specific guideword heuristics in the model help us focus on aspects of the product that may contain bugs, while the entire list reminds us to de-focus from time to time. Moreover, we encourage people to develop and master their own heuristics and to share them with the community.

Pollan notes that learning about mushrooms is a life-and-death matter, and that not even the best field guides or photographs can convey the learning and confidence provided by direct experience and close collaboration with mentors. [8] The same can be said for testing, programming, and all sorts of other human endeavors.

For me, the most important testing lesson from *The Omnivore's Dilemma* is a reminder: We can learn important things about testing from outside the field of testing. Ross Ashby was an influential member of the general systems movement in the early days of cybernetics. He coined the law of requisite variety, [9] which says for any pair of systems in which one is controlling the other, the controlling system must have more states available to it than the system being controlled. Karl Weick applied this idea to people, saying, "Complicate yourself if you want to understand complicated environments." [10]

It's crucial for us as testers to be complicated—to cultivate diverse interests, to learn about the world, and to bring that knowledge back to our products and our processes. Learning is our job. Isn't that wonderful? **{end}**

References

[1] Pollan, Michael. *The Omnivore's Dilemma: A Natural History of Four Meals.* Penguin Books, 2006.

[2] Howard, Albert. *An Agricultural Testament.* Rodale Press, 1972.

[3] Box, George E. P. and Norman R. Draper. *Empirical Model-Building and Response Surfaces.* Wiley, 1987. p. 424.

[4] Pollan. pp.147-148.

[5] Pollan. p. 384.

[6] Whittaker, James. *How to Break Software: A Practical Guide to Testing.* Addison-Wesley, 2002.

[7] Bach, James and Michael Bolton. Rapid Software Testing Course Notes. 2009. www.satisfice.com/rst.pdf

[8] Pollan. p. 372.

[9] Ashby, W. Ross. *An Introduction to Cybernetics.* Chapman & Hall, 1956. pespmc1.vub.ac.be/books/IntroCyb.pdf

[10] Weick, Karl. *Sensemaking in Organizations (Foundations for Organizational Science).* Sage Publications, 1995. p. 56.

From what books or disciplines have you taken lessons about testing?

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