The Abolition of Ignorance

by Alan Page

I've been a software tester for more than fifteen years, but I still remember when I realized I didn't really know anything about testing. It's not that I was a bad tester. I found a ton of bugs, I used automation and other test tools appropriately (most of the time), and I received great feedback from management. Despite these signs of success, three or so years into my testing career, I decided I would like to *study* testing. Since I was getting to a point where I thought testing may actually become a career for me, I decided I wanted to learn more about the hows and whys of testing and build a formal base of knowledge. I started by reading a book on testing that a colleague recommended. In many ways it opened my eyes. I felt that the knowledge I gained from this book, combined with my experience, would make me some sort of "super-tester" (it didn't). A few weeks later, flushed with success, I read another book on testing. I enjoyed reading the second one, too, but it stressed different practices and even contradicted the first book in places.

Now, I was confused.

By the time I completed a third book on testing, two things happened. The first was that I realized not every book on testing was very good at teaching testing. The second was that I began to form my own opinions on what works and what doesn't when testing software. I've read dozens of books and thousands of articles since then on testing and on software engineering, and all I've figured out from all this reading is that I've barely scratched the surface of what there is to know about testing.

I question much of what I read now—not necessarily as a skeptic who thinks that none of this stuff really works in practice but as a learner who wonders how the ideas might apply to situations I have experienced or could imagine. Theory isn't enough when you're trying to make a career out of testing software. So I try things. I experiment. The more I learn through reading and practice the more I realize how much there is still to learn and how many undiscovered paths remain for me to explore.

"Theory isn't enough when you're trying to make a career out of testing software."

Phillip Armour first wrote about the five orders of ignorance (5OI) in 2000 [1]. The 0th order of ignorance (the orders are 0 based, of course, because Armour is a programmer) is *lack of ignorance*. You have 0OI when you know something ("I know how to speak English"). The 1st order of ignorance is *lack of knowledge*. You have 1OI when you know you don't know something ("I know that I can't speak Chinese"). The 2nd order of ignorance is *lack of awareness*. You have 2OI when you are unaware of what you don't know. The 3rd order of ignorance is *lack of process*. You have 3OI when you don't even have a way to figure out what you don't know. The final level of ignorance—*meta ignorance* (4OI)—is when you don't know about the levels of ignorance (a level of ignorance that readers of this article can cross off now).

When I first heard of the five levels of ignorance, I realized that testing lives at 3OI and 2OI. Our job in examining software is to figure out what questions to ask about the software and then to determine what the answers are. Chances are that the programmers with whom you work don't write your test cases for you or suggest, for example, that you "try a really big number in this field." Instead, we testers examine the software, hoping to learn enough about it to ask that question, and subsequently, find the answer. Our job as testers isn't to find bugs; it's to find knowledge!

The more I think about this, the more concerned I get about testers who live entirely at 0OI and think they know all the answers—not in a snobby, know-it-all kind of way but in a way that relies only on the knowledge they have today in order to test each new piece of software that comes their way. The state of the art in test has not kept up with advances in product design and implementation because too many testers have stayed inside their 0OI comfort zone when their true place in the food chain is 3OI. Too many testers think that they "get" testing or that they know "enough" to do the right thing. If you think you have testing all figured out, you are part of the problem!

The 3OI processes, of course, contribute new knowledge to software engineering, but the consummate 3OI process is the critical thinking of intelligent and experienced testers. Testers who can navigate the unknowns of 3OI will elevate the science and craft of testing to a level it deserves but too often does not enjoy. The greatest testers I know realize there is much more to learn about testing than they know today. They know that there are questions they haven't thought of yet and know that there will always be new means to discover these questions. They never assume that they know "enough" about testing to always make the best choices. Obviously, if you are reading this article, you have *some* interest in learning more about testing, but how often do you challenge yourself to discover something new about testing? Do you have a vision of what software testing can become and strive toward that vision? Or do you think that merely repeating and refining what you're already doing is enough?

A passion for learning drives the work of every great tester I know. What drives you? **{end}**

REFERENCES:

Armour, Phillip G. "The Five Orders of Ignorance." *Communications of the ACM*, October 2000/Vol. 43, No. 10.