Playing the Expert Game

Jonathan Bach, Microsoft

My brother Jonathan used to wash dishes at a country club in the Seattle area. I remember him explaining the rhythms and efficiencies of his sinks, optimal cleaning techniques, and how social status among kitchen workers affects quality of service. A little later, he worked at a Borders bookstore, ran the self-help section, which stocked a book he himself wrote, and came up with an idea to use barcoding to streamline their inventory system. Borders ignored his proposal. Jon went to Microsoft. Whatever you think of the folks in Redmond, they need this guy.

—I James Bach

I showed the developer a bitmap of an error.

“This isn’t possible,” he said. “Could be a mismatched DLL. What version are you using?”

“How do I check?”

“The standard way is fine.”

“Okay. What’s the standard way?”

“Never mind,” he said as if he hadn’t heard me, “I’d actually like to check something, so just get it into a debugger.”

“What kind of debugger?”

“Just the basic settings. Nothing fancy. All I need is a stack trace.”

I returned to my office in a cloud of worry. I didn’t know what he was talking about. I guess I should have asked him what he meant by “basic settings,” but the way he said “nothing fancy” left me with the impression that it was something I was supposed to know. I assumed it was my responsibility to figure it out.

This situation was typical of my early experiences as a tester at Microsoft, and I suspect thousands of people face the same thing every day. Today I recognize that the developer was asking for a simple thing, and the particular words and ideas he used no longer sound like an obscure Klingon dialect.

But for every technical idea I understand, there are 50 more I don’t. What’s changed for me in the past four years is not so much that I’ve come to master all the technical skills I need on the job, but rather that I’ve learned how to play the game of expertise well enough that my company pays me to keep playing.

Self-doubt and self-education

Thankfully, they don’t call it a campus for nothing. There were a million resources at my disposal. I’d fight my self-doubt with self-education. A Microsoft badge is like a student ID card, qualifying its bearer to take classes and order books for free. On my first day, I rush-ordered eight different Microsoft Press books and signed up for four classes.

Software was also available. I installed Microsoft Developer Studio, and while I was at it I installed Visual Basic and C. I saw my manager using something called the Knowledge Base to research bug trends. I installed it too. I queried the intranet for “test tools” and installed whatever looked impressive.

Then I went to the Internet to find free test applications. I ran Usenet queries for “test methodology.” I printed all the relevant white papers I could find about my product and borrowed professional magazines like InfoWorld and PC Magazine that were on the tables in the lobby.
Learning the things that matter

Once I showed a co-worker that I couldn’t see any files in a network file share. The folder was empty. He said, “Well duh, don’t you know? You don’t have permissions.” Permissions? Why do I need permissions if it’s supposed to be a shared folder? He then explained NT security to me and told me to read a chapter in NT 4 Server Unleashed. I read it, learned about account security, and came up with a whole new battery of tests to run. What if a user is locked out from a share after five failed attempts? What if the share is read-only? What if it is read/write? Can the setup files be overwritten with DLLs from another app? Unlike the other times I’d tried to read technical books, this time the information stuck. Somehow the direct relevancy of the information made it much easier to absorb and retain.

Another problem assigned to me was to retest any fixed bug with our product under NT. I discovered that only one person in our group knew how to install NT, so I watched him do it. He wanted to share the information anyway so he wouldn’t be bothered all the time. It was surprisingly easy. The funny thing is that I became sought after to do NT installations after that. A couple of people even heard that I was an “expert” on NT. Expert?

Playing the game

My learning progresses through these kinds of events, as well as through direct interaction with and exploration of the technology I test. This is the learning I do every day. As the months passed, I began to experience a phenomenon that my brother calls the Expert Game. The game goes like this: Under certain circumstances, we will treat other people as knowledgeable even if we know they probably aren’t. See “The Expert Game Equation” sidebar.

The Expert Game stems from the nature of problem-solving organizations: When there is a need, anyone seen as useful will be put to use, regardless of his or her level of knowledge. At Microsoft, the people who can are encouraged to do more. I develop resources, relationships, and choice bits of technical knowl-

---

The Expert Game Equation

If...

- you are able to get important things done
- you are seen learning things on your own
- you are seen trying to do things even if you aren’t sure how
- you share freely the things that you know
- you don’t hide your ignorance, but also don’t rest on it
- you honor what other people know
- you know more often than not how to find out what you don’t know
- you know how to ask for help
- you offer to help people on their own terms

Then...

- no one will care whether you succeed by learning or succeed by already knowing
- no one will care if you mess up occasionally because they assume you learn from it
- no one will mind if you forget (or don’t know) any given fact or method at any given time
- you will be treated as if you’re smart and useful, even though everyone knows you have a lot to learn

---

A mountain of knowledge

I’m sorry to report my strategy failed. The most important thing I learned from the mountain of knowledge I gathered was that it doesn’t help much to gather a mountain of knowledge. I don’t have time to study it from beginning to end, and even when I find some time, I don’t retain enough of what I do study.

Being a tester is a lot of work. There are bugs to log, strings to defog, matrices to design, sign-offs to sign, regressions to make, meetings to take, schedules to keep, modules to sweep, machines to clean down, debuggers to hook up, and tons of metrics, heuristics, and statistics. These things take a lot of time. They take more than all my time, meaning that any time to read books or take classes is time that I have to steal from something else I feel an urgent need to do.

I did manage to take a few classes, and the books I ordered arrived. In between meetings and test passes, I flipped through them, hoping to see something pertinent and obvious that I could learn quickly. But I didn’t retain much from those efforts. Today I can barely fill a Post-it note with what I learned from them: something about how doing Ctrl-D after a query will display the results in grid format, which is easier to read. And be careful with some queries because “IS NULL” is not the same as “= NULL.” That’s about it, friends.

DISCOVERING THE EXPERT GAME

Now for the strange part: I’ve learned a lot in the past four years. I feel comfortable in my job, and I’ve been rewarded for doing good work. How is that possible? The answer is either so obvious or so subtle that only when I sat down to write this column did I become conscious of it. Despite all the problems I’ve outlined, I do learn a lot on the job: I learn from people and problems that are directly part of my work, and most of all from things that are directly necessary to do my work.
and opportunities to learn. The game is helpful, because it gives me time lentlessly better at my work. Still, the continue my education and become r e-
tions can substitute for broad technical knowledge.

I like to think of this process as a game because it helps me not take it too seri-
ously. What is serious to me is that I con-

• I am continuously learning without my manager needing to push, prod, or pay much attention to my learning process. I manage my own learning process.
• My learning strategy helps other people learn because it unfolds every day on the job, as I work with the team. I’m contributing to the over-
all capability of the team.
• My learning strategy means that people can trust me with bigger problems than they otherwise could.

They know I will figure it out some-
how, get someone else to figure it out, or give up gracefully.

The dialogues I have with developers today are more confident:

“Found this,” I said recently, showing the developer a bitmap of an error. “It happens only when exiting the program using Alt-FX. I filed the bug report, assigned it to you, and sent you an e-mail with the stack trace.”

The developer looked at the error. “I know what this is. Win9x machines only?”

“O h. G ood point. H aven’t tested NT, but I will.”

“O K. Does it have a reliable repro?”

“I’m working on that now. I just didn’t want to wait for that before giving you the heads up.”

“What version of the runtime DLL”

“M ay 18, ’99.”

“W ere you shutting down?”

“Yep, but it also happens when just logging off.”

“O K.,” he said, turning back to his
screen, “just e-mail me about NT.”

That developer got good service. I knew the basic jargon. I anticipated what he needed to know. And, most important, I was not paralyzed by the fear of what I don’t know.

I have to confess, I never did read the Excel 3.0 macros book. In fact, I just recently put it back in the recycle bin. Maybe another newbie will pick it up and learn the same thing I learned from it.

Solo represents a novel approach to compilation that combines system-level control and dataflow analysis, aggressive optimization techniques, schedule-based algorithms, and flow management.

At Improv, we’re developing several applications for system customers to use on top of the PSA, such as the Improv HDTV video decoder. Developed over the past few months, the decoder targets a single high-speed PSA and offers several features, including full support for the Grand Alliance HDTV specification, modular software configurations that convert output to native display formats, a bit-stream interface, and a raster scan digital video output section.

Highly reconfigurable hardware designs that are tightly coupled to flexible software tools will be key to success in the competitive world of consumer electronics. To make our PSA even more competitive, we will be integrating it with mixed signal, bus interface, and microcontroller image processors. We will also offer more audio, video, and image-processing applications, and will work with our licensees to develop new implementations that support higher clock speeds.

Oz Levia is chief technical officer at Improv Systems. Contact him at ozl@imпровsys.com.

Jonathan Bach is a test lead and network administrator in Microsoft’s SMS Dogfood team, which is responsible for testing and administering Microsoft’s Systems Management Server as it’s being developed. His first testing job was in 1995 as a contract setup tester on Microsoft Bookshelf. In 1997, he joined Microsoft’s Systems Management Server team, where he became a test lead and a full-time employee. Contact him at jbach@microsoft.com.