Software Plumbing

Infrastructure. It's easy to ignore, it's often buried, and can tend to be dirty. I learned about infrastructure firsthand while in college. I paid for my dabbling in the dark arts of electronics with a great job on the maintenance crew for a large downtown shopping mall and high-rise. This place was packed with infrastructure. My most memorable lesson came when old Ben had me perched on the top of a really tall ladder armed with an overweight pipe wrench. Ben claimed to have personally known Gen. Patton and served him well in his engineering corps. He shared some great stories on the general's tactical use of engineering, but that's another story. In this instance, Ben was standing below – and some distance away from – my perch, directing his cadet plumber.

It was a bit smelly, but I was doing fine on my task when my outlook changed dramatically – through the open pipe I heard a toilet flush! I was never very athletic, but now I had a vision that made me cover 15 feet before Ben could get a word out. Ben was not amused at my desertion, and sent me back up the ladder. Once I realized my fears of a gushing pipe were unfounded, Ben could see that it was time I learned about infrastructure. "Tony," he said, "you need to know the two rules of plumbing, the first rule being that water flows downhill." I quickly perceived that this first rule was the only reason I was still smiling.

"So what's the second rule?" I asked.

"Don't bite your fingernails!" he said.

Truth be known, it was Ben and his troops that kept the folks upstairs from biting their nails. The whole city block was cooled, heated, lubricated, lighted, and generally functioned because Ben and I were at work. I left Ben's tutelage to begin my career in software and before long found that even software had something akin to infrastructure. The analogy isn't perfect, but just like the undergirding of a building, my initial dabbling in code exposed the underpinnings of the software world, and included in that foundation I found configuration management (CM) and test.

Now, lest the hate mail flow, my reference of CM, test, and sewage in the same article is simply a matter of artistic contrivance to show that these disciplines – like infrastructure in the physical world – hold up the software universe. For example, I heard of one company whose CM was so neglected that the developers literally lost the baseline for their software. Imagine that! They climbed out of this hole by begging the productions guys for a copy of their own software – that's like Julia Child calling Random House for her macaroni and cheese recipe. Talk about missing the essentials!

I've never witnessed such gyrations firsthand; in fact, I became an early proselyte to the ways of CM when it saved my skin. I was merrily cranking code on one of my first assignments when our team lead announced an inquisition; it seemed our latest additions to our product had rendered it, well, useless. As the least experienced on the team, I was the guilty one. But we all marched out to our integration stand where I witnessed the marvel of CM as the build was reconstructed change by change. Everybody wanted to get this over with so, hey, start with the new guy first. My cohorts were warming up a nice selection of ridicule when, horrors – for them anyway – the thing still worked with my code in it. This meant, of course, that someone else was to blame and the cold hand of CM was about to expose him. Which it did, and I don't recall my team lead ever convening another such inquisition.

Illustration by Keith Gregersen.

All right, so exposing the boss was a bit messy, but it would have been a real battle without the structure of CM. Being a test engineer often put me in similarly tight spots, but by then I had learned to enjoy it. We would work endless hours in the lab and if we ever showed up in the office, it meant trouble. You could feel the collective groan as we strode in and toyed with our teammates. Yes, indeed, personal pride is rarely measured in the cost of zero defects. Don't get me wrong, this was gentle professional ribbing – the same sort of professionalism that produced the self-effacing Air Force Acronym Reference Compendium, better known in testing circles as the AFARC.

It might have been a bit rough at times, but the customer was the unknowing beneficiary. Working the infrastructure of software may feel sometimes like working the nameless unseen systems in Ben's building, but that never seemed to bother Ben. He knew.

— Tony Henderson
Software Technology Support Center
tony.henderson@hill.af.mil

Can You BackTalk?

Here is your chance to make your point, even if it is a bit tongue-in-cheek, without your boss censoring your writing. In addition to accepting articles that relate to software engineering for publication in CrossTalk, we also accept articles for the BackTalk column. BackTalk articles should provide a concise, clever, humorous, and insightful perspective on the software engineering profession or industry or a portion of it. Your BackTalk article should be entertaining and clever or original in concept, design, or delivery. The length should not exceed 750 words.

For a complete author's packet detailing how to submit your BackTalk article, visit our Web site at <www.stsc.hill.af.mil>.