I Am an Abstraction Layer

from the editors

Edward Grossman, Editor, *ACM Queue*

I'm not going to "sully" *Queue*'s special report on databases by attempting to talk about them myself. Doing so would be like a sportscaster giving color-commentary to a Dalai Lama sermon: embarrassing. And while I won't go so far as to call this issue of *Queue* spiritual, read Jim Gray's "A Call to Arms" and you'll see what I mean. And put your hip-waders on.

As we worked on this month's issue, a deeper signal began to resonate around the edges: the database as an *abstraction layer*. I don't need to know anything about how a database stores my data; I just issue queries and get responses (and I'll be the first to admit after working on this issue of *Queue* that my simplistic description of a DB is quaint at best).

And the signal traveled farther. High-level programming language: I don't care about the bits and bytes, you compiler-folk worry about it, I'll just issue instructions. Operating system: I don't care about how you open/close/read/store files, just open what I want when I want. And the list goes on, and on.

In fact, developing abstraction layers is really just a very natural process, an *instinct* if you will. And we don't only invent abstraction layers in "technology," we surround ourselves with them: I don't really know the details about how the HR department produces my paycheck, and I don't want to know. I'm lazy, solve it for me—that's the abstraction layer mantra.

But turning the mantra on my self produces an interesting result: For whom am I "just handling all the dirty details"? For whom are we acting as abstraction layers?

I recall a previous job I held as a software development project manager. My daily routine was altogether predictable: meet with business decision makers and get their needs, translate that into geek-speak, and explain it to the software devs. The opposite was also true: get the response from the devs about why it couldn't be done that way (just kidding!), translate it into suit-speak, and explain it to the VPs. Never was I more an abstraction layer—and a bi-directional one at that! Neither side wanted to dig into details of the other; they just wanted to express their needs and have me handle the details.

Truth be told, I always felt more akin to the devs than

Queue's DATABASES

SPECIAL REPORT

SPARKS A THOUGHT.

the suits—more amongst my own. That's why it struck me with alarm that the ultimate Holy Grail of all this abstraction is

to do away with us software developers altogether. What happens when we finally give the business managers the tools they *really* want—ones they can use themselves, and define business processes without the need of "all those messy details" on which we devs thrive? Are we working toward our own obsolescence?

My answer to that question is a resounding NO!

Why? Yes, it's true that the straight *implementation* of lines of code is abstractable—note trends in outsourcing/offshoring and even the rise of reusable .NET/EJB components. But while we might abstract out some of the implementation details of this or that portion of the spec, the architecture or design of the system is a *long* way from something we can just brush off and say, "Oh, you worry about the system architecture, network design, platform choices, and all those messy details." Application design, including the methods by which business processes are handled, is still hard.

Second, the idea that we'll *eventually* reach that goal of building the perfectly efficient system for nontechnical decision makers to implement business processes themselves is simply erroneous, for the simple fact that the end zone is receding. As soon as we build our pals systems they can point and click independently to implement their business process desires, they'll want a layer further removed. Remember the mantra: I'm lazy, solve it for me. So now build me a system that I can just speak to so I don't have to point and click. Next, build me a system that can predict options I'm likely to choose. An engineer's work is never done.

Queue is pleased to be an abstraction layer in its own right: bringing you insight into important technologies from important technologists—and don't worry, we'll take care of all the details. All you have to do is read. Enjoy! ☺

EDWARD GROSSMAN is an abstraction layer.