KV the Loudmouth

To buy or to build, that is the question. Of course, it's rarely that cut and dried, so this month Kode Vicious takes time to explore this question and some of its many considerations. He also weighs in on the validity of the ongoing operating system wars. Have an equally controversial query? Put your thoughts in writing and shoot an e-mail to kv@acmqueue.com.

Dear KV,

I was somewhat disappointed in your response to Unclear Peer in the December/January 2006/2007 issue of *ACM Queue*. You answered the question, but I feel you missed an opportunity to look at the problem and perhaps expand Unclear's professional horizons.

What requirement is being satisfied by having Unclear *build* a P2P file-sharing system? Based upon the answer, it may be more effective, and perhaps even more secure, to use an existing open source project or purchase commercial software to address the business need. Indeed, if the definition of P2P is loose enough, encrypted e-mail would meet your security criteria and might solve the business problem.

If Unclear is just a koding gnome, content to write kode as specified and not ask why, then I withdraw my concerns. Otherwise, it seems to me that an opportunity to teach Unclear, and your readers, was missed.

Sincerely,
Buyer not always a Builder

Dear BB,

Perhaps I've missed the marketing hype around this, or it has wound up in my spam box like all those ads for enlargement technology, but last I checked there wasn't an off-the-shelf P2P system one could buy. That being said, you bring up a good, if tangential, point—and one

Got a question for Kode Vicious? E-mail him at kv@acmqueue.com—if you dare! And if your letter appears in print, he may even send you a Queue coffee mug, if he's in the mood. And oh yeah, we edit letters for content, style, and for your own good!

A koder with attitude, KV ANSWERS

YOUR QUESTIONS.

MISS MANNERS HE AIN'T.

interesting enough to prevent your letter from winding up with all those aforementioned enlargement ads.

The buy-vs.-build, or as I like to think of it, the integrate-vs.-build question touches just about every part of a product. I like to say *integrate* because that can take into account using open source software, as well as buying software from a commercial vendor. Although many people might like to build everything from scratch—the Not Invented Here school of software construction—that is rarely an option in most projects because there is just too much to be done and never enough time. The problems that need to be addressed are the cost of integration and the risks.

Cost in this case is not just that incurred in buying a piece of software. Free or open source software often has high costs. The number of people on a local team required to maintain and integrate new releases of a component is definitely a cost that must be accounted for. Producing documentation is also a cost. For commercial products the costs include those just listed, as well as any money required to license the software in question.

In reality, the cost could be seen as just one of the risks involved when making the decision on whether to integrate or build a component of a system. The risks of integrating a component include the likelihood that the company or project that provides that component will continue to exist, and whether the component owner will change the system in a way that doesn't agree with your product over time. Plenty of people have been bitten by software that was changed underneath them.

It all comes down to control. If you can architect your system in such a way that the risks of integrating a component can be mitigated successfully, then integration, barring exorbitant costs, is probably a reasonable way to go. If you need absolute control over how a component works now and in the future, then you'll have to build it yourself. There is a spectrum of choices, but those are the two poles that you must navigate between.

KV

Dear KV,

I suspect that you don't get many letters from CFOs, but one of my people left a copy of *Queue* in my office the other day. I read your column and thought you might be interested in this question. Getting directly to the point, does the operating system still matter? I ask this because every time we initiate a project in my org, a small but loud group of people push me to pick an open source operating system for the project. It seems that they care more about that than about the application we're rolling out to our staff.

Reading over the trade press, I see claims and counter-claims about various operating systems, based on security and total cost of ownership, but all these claims seem

system is the ultimate arbiter between your application and the underlying computer. The operating system controls access to the CPU, memory, and all the devices. A good operating system is like good service in a restaurant: there when you need it and invisible when you don't. A poorly designed or implemented operating system is like the waiter who constantly asks, "Is everything all right?" when your mouth is full.

Two of the most important measures of operating system quality are security and efficiency. Does the operating system you want to use have a good security track record? No operating system, or piece of software, is perfect, but there are clearly classes of problems that may affect your application and these are the ones you, or likely your staff, need to study to make an informed decision on which operating system to put under your application.

Efficiency is also important. Although there are plenty of micro-benchmarks that show that one operating system is better than another, the speed of a context switch is unlikely to impress you—though I would be impressed if you knew what it meant. For an application, the question is one of a macro-benchmark. Simply put, "How much work can people do in the application in a given unit of time?"

Another question would be around how integral the operating system is to your product. If your company builds products where the operating system is an integral component, such as a consumer device or piece of networking equipment, then the quality of the code, your ability to modify it and distribute your changes, documentation, and how long you think the company or project that supports it will last all come into play. These concerns were addressed in the previous response to the letter from Buyer not always a Builder.

So, the short answer is, "Yes, the operating system matters." And, please, don't just fire the loudmouths. I might be one of them.

KV

A good operating system
is like good service in a restaurant: there when
you need it, invisible when you don't.

to be written by proponents of one of the systems in question. At this point, it seems like the operating system doesn't really matter anymore, just so long as my application runs on it. What do you think? Should I just fire the loudmouths?

Tired of Zealots

Dear TZ,

You're right, I don't receive many letters from CFOs unless they're printed on pink paper and include words like "...please empty your desk by..." I also rarely condone firing the loud ones, for what must, by now, be obvious reasons.

Many pundits (i.e., people paid to have opinions) now claim that the operating system is a commodity that, in itself, has little intrinsic value. I don't get paid to have my opinion, but I claim that pundits have little intrinsic value.

Let me try to answer this question without going too deep into Operating Systems 101. The reason that the operating system matters, and will continue to matter as long as there are operating systems, is that the operating

KODE VICIOUS, known to mere mortals as George V. Neville-Neil, works on networking and operating system code for fun and profit. He also teaches courses on various subjects related to programming. His areas of interest are code spelunking, operating systems, and rewriting your bad code (OK, maybe not that last one). He earned his bachelor's degree in computer science at Northeastern University in Boston, Massachusetts, and is a member of ACM, the Usenix Association, and IEEE. He is an avid bicyclist and traveler who has made San Francisco his home since 1990.

© 2007 ACM 1542-7730/07/0500 \$5.00