

The Muscles, Aches, and Pains of Open Source

In July the University of Wisconsin-Madison had another opportunity to participate in a standards-based, open-source software initiative. To opt in, I had to put up a little of the university's money. To opt out, I had to tell twenty-five of my friends and colleagues around the nation that I did not think this was a wise investment. If enough of us opted out, the project might fizzle. Or it might proceed without input and influence from the higher education community, and those of us in the community might miss the chance to get a product that meets our collective requirements. UW-Madison opted in. So did twenty-four other institutions. But I worry: are we committed enough?

At UW-Madison, we buy rather than build major applications software. These days, this is scary. We are watching our providers fail, merge, and be acquired. Just this past year, one of UW-Madison's major providers merged with another company; a second was acquired. In both instances, the application direction is not what we understood when we purchased the application, and it isn't better. I'm not sure that we would choose the same vendors if we were buying now.

All of us know about Oracle and PeopleSoft (and will know more by the time these words see print). Initially, Oracle said it would not support current PeopleSoft applications; then it backed away from that statement. If Oracle returns to its initial position, all PeopleSoft applications will require expensive enterprise-systems migrations. Many in higher education worked hard in building a relationship with PeopleSoft. The

relationship yielded applications somewhat tailored to meet higher education requirements. A successful Oracle bid for PeopleSoft could cost many of us a lot—but we have no control over the situation.

Microsoft challenges us in other ways. We struggle with its licensing offerings. We struggle with the security of its applications. Because Microsoft owns the desktop marketplace, because it is very large and individual colleges and universities are not, higher education has little opportunity to affect the direction of Microsoft.

That's just how it is, I can hear you say. Fair enough. But these recent vendor events have me pondering bigger collective investments in collaborative software development with my colleagues around the nation. As I ponder, I realize that the investment in development isn't what holds me back: it is the linked dilemmas of implementation, support, and maintenance.

We have a large and talented technical staff at UW-Madison. We are active in national forums and participate with our colleagues in discussions and presentations. We contribute software to open-source pools. We take part in joint software development. We use software

We need to develop creative collaborative solutions to the dilemma of maintenance and support in our shared software-development initiatives.

that our colleagues and friends develop. These are good things. It's pretty easy to pick up a small piece of code, or even a module, from a friend and attach it to an enterprise application. We do this, and we manage the code after we have adopted it. We check in with our friends, put fixes in the greater pool, and benefit from this informal sharing.

But this is all occurring at the margin. In more significant cases, we at UW-Madison contribute our thinking, and we have supplied the occasional piece of code, but we haven't been full partners. Why not? Because we have not been prepared to implement the collaborative products. Even where we have shared a staff member, for example with Internet2 for the development of Shibboleth, we are slow to adopt the outcome.

Why the reluctance to implement shared projects? Because, in my view, the collaborative higher education IT community has not figured out how to provide

ongoing support to complement this great development work. When one college or university leads a development effort, it is usually prepared to implement and maintain the resulting application—but for itself, not necessarily for another institution.

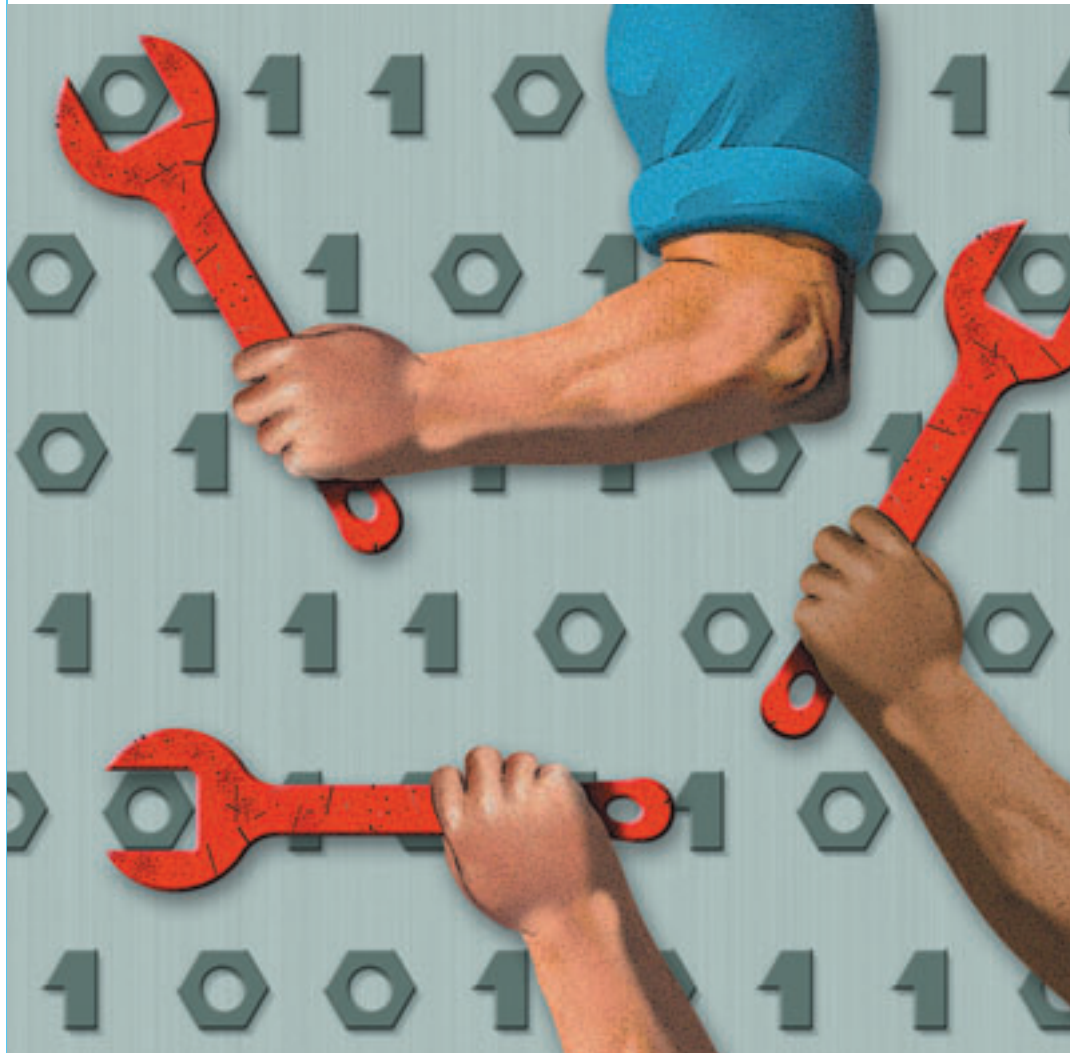


Illustration by Steve McCracken, © 2003

Support is central to enterprise applications. UW-Madison runs Sun's I-Planet e-mail software, serving more than 60,000 people. We had a major systems failure this year. Yet the Sun product developers hung in with us until we were able to isolate the source of the failure. The Sun engineers then offered a fix. If our e-mail software had come from a collaboration among, say, Michigan, Indiana, Penn State, and Wisconsin (a likely group), where would we have gone for that time-consuming, in-depth, post-implementation support?

As I was thinking about this dilemma last night, I remembered CREN, the Corporation for Research and Educational Networking. For years (and for a fee), CREN provided and supported ListProc, an open-source product. Many institutions became CREN members only to get ListProc. Although CREN had limited resources to support ListProc, it pro-

vided an organizational umbrella for a cross-institutional community of technical folks who knew the product and helped keep it running. The Internet2 Middleware Initiative faces the same maintenance and support challenge for Shibboleth. Institutions are beginning to adopt this method of authentication for sharing network resources. The Internet2 middleware people are beginning to think about how to maintain the product and support the user community. If they don't reach at least CREN's standard—and Shibboleth is far more complex and challenging—it will be hard for anyone to adopt the product.

Design and development are fun and exciting. Moreover, at some point in the process, you can declare success and move on. Maintenance and support have neither the glamour nor the defined end points. They're not as much fun, and they last forever.

So why, given the challenges of maintenance and support, did I opt in to a collaborative open-source project last July? And why am I talking to my colleagues about another project that will require an even larger effort? Because, in short, I find the vendor situation so disturbing. But opting in to development is only the first step. As I opt in, I'm challenging all of us to believe in shared, open-source software—but to do so more completely. We need to develop creative collaborative solutions to the dilemma of maintenance and support in our shared software-development initiatives. I want to confidently recommend participation in the development and implementation of great mutual work. For me to do this at UW-Madison—and for all of us in higher education to do this across the nation and the

world—we need to know how we are going to keep things going. And we need to figure this out very soon, or we will remain hostage to an increasingly hostile vendor environment.

Dr. Seuss is my organizational guru and my favorite philosopher. As usual, he's on point: "This is called teamwork. I furnish the brains. You furnish the muscles, the aches and the pains."¹ We are really good at furnishing the brains. Let's figure out how to furnish the muscles, the aches, and the pains.

Note

1. Theodor Seuss Geisel, *I Had Trouble in Getting to Solla Sollew*, by Dr. Seuss (New York: Random House, 1965).

Annie Stunden is Chief Information Officer and Director of the Division of Information Technology at the University of Wisconsin.

