Java and Other Productivity Jolts

Users, vendors and analysts alike are only now beginning to understand the versatility of Java and its relationship to other languages for general- or special-purpose applications development in an environment that includes Hypertext Markup Language (HTML) clients.

The way Java was originally presented, it appeared to be aimed at a futuristic, electronic commerce scenario. Some corporate users took one look and declared they wouldn't touch it with a 10-foot pole. Security is the big issue. No matter what Sun might say about "no holes in theory," in practice any new software requires a shakedown period. Java is no exception. Skeptics are going to stay in the wait-and-see mode for quite a while.

Other enterprise computing buyers were attracted by the cross-platform capabilities of Java. But they wondered whether Java could solve more than a small part of the desktop problem. The really big problem, as *Fortune* 1000 MIS sees it today, is not the cross-platform desktop but the obese desktop.

When user organizations begin to move "fat-client" applications from pilot to production, product and management costs balloon. And when it comes to supporting remote users or reaching out to customers or suppliers, client/server is a real test of fortitude. Administrivia is the heaviest burden, even for an organization with mostly Windows on the desktop.

Accordingly, when the Web came along, many organizations were already seriously questioning their investment in two-tier client/server applications and development tools. They were already weighing options for slimming down existing applications and developing new apps with lighter-weight clients.

As Netscape first hit the radar screen, MIS grew misty-eyed with nostalgia. Sure, the original Navigator was a little on the dumb side. But it was lightweight, versatile and low-maintenance. It had the advantages of the terminal, but it was hip! "Could an HTML browser possibly serve as a generic client interface?" MIS wondered.

By Nina Lytton

In 1995, some organizations with TCP/IP infrastructures already in place decided to try to make the browser the front end of their multitier client/server environments. These early experimenters got straightforward applications up and running by slapping an HTML server between a new, dumbed-down front end and an existing back-end service or database. Essentially, it was a '90s twist to good old-fashioned mainframe screen-scraping.

Ups, Downs and an Example

In taking stock of the 1995 experience, two facts are clear. First, the upside is enormous. An incredible amount of business value can be derived from retrofitting existing apps with a "lipstick on a bulldog" approach. A 100 percent fat-free client is particularly appealing for infrequently accessed applications, remote access situations and any application that reaches out to customers or the supply chain. Simply providing ready access to existing information, without using any of the fancy features as they came out from Netscape, made a day-and-night difference to the business. In a large division of a leading telco, for example, the option of a fat-free client was the decisive enabling factor for a customer-care application used by every employee.

Second, the downside is equally clear. Application performance, versatility and maintenance quickly became issues for early adopters of Web-based client/server apps. The HTML/CGI programming environment is crude—far below today's standard of development productivity. And rolling your own tools and utilities is not the answer for most *Fortune* 1000 MIS shops.

In 1996, public awareness of the Java language has grown considerably. In-house development shops are now asking the same types of questions about Java that they had been asking about HTML browsers last year.

- How broadly applicable is Java?
- Who says a Java applet has to run on the desk? Why can't Java applets run just as effectively on a server?
- Why can't Java applets (or "servlets") be compiled as well as interpreted?
- Is Java's ability to call other languages really a bug, or is it a feature?
- Can Java become the basis for a general-purpose development environment?

Because Java development environments are just now becoming available, the answers to these questions remain to be seen, let alone proven. The ratio of hype to reality in the Java world remains high.

Meanwhile, however, one can make useful observations based on the success of another interpreted language, Smalltalk. Last year, ParcPlace-Digitalk of Sunnyvale, CA, introduced a VisualWorks Smalltalk extension called VisualWave, which brings the complete Smalltalk applications development environment to rapid development and maintenance of serious business applications on the Web. In addition, it provides database connectivity and adds state (continuity of user sessions) to the Web server. VisualWave users can maintain a single code base for all client/server development on and off the Web.

Two conclusions emerge from the Smalltalk example. First, the time-to-deployment benefits of a robust, full-featured development environment are substantial. Expect to see more new languages and environments for the Web, and expect these to interoperate with existing systems.

Second, it is not necessary to move to a totally new language to get benefits from Web-based clients. Today's popular development environments will be extended to the Web soon. And there is quite a business in opening existing legacy applications, even mainframe apps, to Web users.

So what to make of all this? Java can be served in many different ways. Folks who won't drink Java brewed espresso style will be able to chose a decaf skinny cappuccino. Java is going to be a lot more useful a lot sooner than we skeptics thought.

Even so, some people are going to think that coffee smells better than it tastes. Java will have plenty of competition. IT

***Nina Lytton** is president of Open Systems Advisors in Boston and producer of the Crossroads Conference.*