

Fight Trendy Decisions and Use the Best Technology for Your Business

The decision was made. We were heading down the road of rewriting our financial package into the latest Java, JSP, Servlet, J2EE, web services-enabled, STRUTS framework application. Just like many other companies, our customers were demanding that we deliver our application to the web. The decision was a simple one, jump on the Java and company bandwagon and start rewriting quickly. Or, was it that simple?

The existing application was an object-oriented application written in Smalltalk with about 3,000 model classes and 1,000 views. It was back-ended by Oracle, SQL Server and Informix database platforms and originally developed by nine developers.

The prototype effort had begun; we embarked on the journey to build our cutting-edge J2EE application. A year and five man-years of effort later, there was merely a "Hello World" proof of concept to show for these efforts. The project plan projected that there would be 20 developers required to deliver this product in a year.

What was going on? Why was this taking so long? Why did we not yet have a web prototype? What would current customers and prospects think of the stagnation of features in this time frame? Market analysis projected our small company would lose over a million dollars if we did not deliver in a year, not to mention the competition's advancements we may never recover.

The excitement and energy of providing crucial business functions to customers deteriorated to simply the effort of re-coding one application into another language. The decision to go to J2EE was made without the careful evaluation that was done when Smalltalk was chosen. That evaluation was the beginning of the best solution for delivering applications to the web.

A decision to make a fundamental shift to a new technology platform would obviously

have a great benefit to the company, right? Without even performing a detailed and careful analysis, you could assume such a shift to have a cost. The prognosticators of Java did an excellent job of masking this cost to management, with much of the cost not being considered. The wholesale retraining effort was not adequately scoped and the loss of opportunity for new business was overlooked. Building a web application was the focus. The impact to the whole organization was ignored during this massive change.

Beneath these higher-level costs, are subtler, but nevertheless crucial, technology costs. Crucial because technology allows the flexibility to integrate with the business. The quicker and easier this integration, obviously the faster we keep up with the market demands. Questions about the following areas were key to us:

- Servlet ability (Web Server/Servlet interfaces)
- Web Services capabilities
- IDE (Integrated Development Environment) ease of use
- Configuration Management reliability
- Developer productivity
- Interoperability in a web world

And most importantly, **reuse**.

When you have an application with over 50 man-years of code base, serious consideration needs to be made before throwing that out and starting over. Can you manage a single code base? Can you reuse what you already have? Smalltalk's flexibility and the enormous support from the vendors allow this. Does it make sense to abandon your code for the latest thing? No!

Even if you are concerned about being left behind, Smalltalk will not allow it. There is full interoperability with RMI, CORBA, Web Services, HTTP, COM and more. These

interfaces will allow existing and new Smalltalk interfaces to talk to the external components. Not being tied to a specific language and staggering the implementation is a possibility.

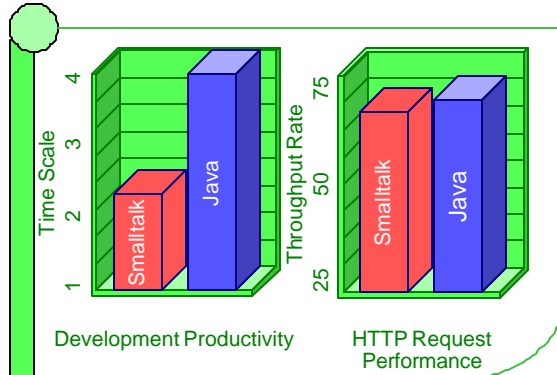


Figure 1: Smalltalk Benchmarks

Look at the productivity of Smalltalk vs. Java. Figure 1 illustrates the comparison of Java productivity to the Smalltalk productivity that we experienced. Even though we had a couple of seasoned Java developers on this prototype team, we did not assume our Java development experience was average. Our productivity seemed in line with what our peers experienced. Our Smalltalk productivity rates were three times those of Java.

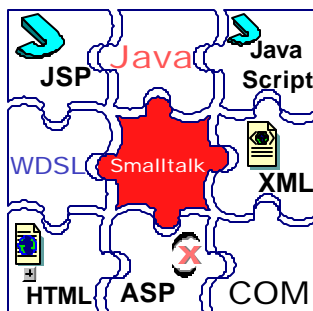


Figure 2: Smalltalk Interoperability

So Smalltalk does not lend itself to being a web application? Wrong, Smalltalk is not a medieval language. Smalltalk does Servlet and JSP, Web Services, XML and RMI as well as many other interoperability protocols. And as any Smalltalk developer will tell you, the Envy configuration environment and the Smalltalk IDE are unmatched. And how did Smalltalk's reputation for being slow measure up? Smalltalk and Java both ran our benchmarked processes at 75 requests per second.

Smalltalk for a web application development tool was not only a viable possibility, it was also an excellent choice. Years of existing business functions and new modules were made accessible as a web application to users. Getting away from the latest trend resulted in one key accomplishment, success in the market!