Interview: Bob Balaban Inventing the Future

Interview by Betsy Kosheff

[Editor's note: This interview resides in "Notes Today", the technical Webzine located on the http://www.notes.net Web site produced by Iris Associates, the developers of Domino/Notes.]

Bob Balaban doesn't waste time. First, he built the Version Manager feature in 1-2-3 for Windows; then, he wrote the Notes back-end class libraries and was a key contributor in getting LotusScript and agents into Notes 4.5. Besides earning a U.S. patent and recently being awarded the Lotus Commitment Award, he's now working on the next generation of programmability in Domino -- the Domino releases that support Java. What will he dream up next?

So, you earned a U.S. Patent. What was that like?
Weird. We'd submitted the patent application in the spring of 1992, and then our lawyer spent lots of time explaining it to the examiners. Years went by. Then, in December of 1994, I found in my mailbox what looked like junk mail. It was from a company that makes plaques, congratulating me on my patent award. That's how I found out. So, my wife bought me a plaque for my birthday.

What made you invent the Version Manager?
I was only one of the people who brought it into being. Irene Greif (Director of Research at Lotus) was really the parent of the idea. Paul Haverstock hired me to be the architect, and my job was to figure out how to implement it. I was sold on the idea immediately: alternative versions of spreadsheet ranges, seamlessly integrated. It even caught on with upper management after a while.

What brought you to Iris?
Nearly four years ago, I was just finishing 1-2-3/W and there was a project starting up at Lotus to do "agents" for Notes. They wanted it because, conceptually, it seemed like a good idea , plus, agents were all the rage. The first thing we had to do was figure out what it means -- what are agents, and what would be the architecture for agents in Notes?

Bob has x-ray vision when it comes to the future of Notes and Java.

What was the answer?
Well, what we call agents in Notes is different from intelligent agents or agents in the UI that monitor what you're doing and then try to figure out a better way to do it. I have a philosophical problem with the whole

AI concept that this is something people really want. The biggest problem people have is finding a good way to get a computer to do what they want it to do in a reliable, non-virus way. I don't want little things popping up in my face saying "you could do this same thing in three keystrokes." Get outta my face... but, that's just my opinion. The design center for Notes agents is relatively simple. It's a program that you write which you use to tell Notes what you want to do, whether it's time- or event-triggered. These programs can run on either the client or the server, they can be called from other agents or LotusScript.

A lot of the work was just getting LotusScript into Notes, because we needed a language and a programming environment. Then we needed Notes objects, so people could access the Notes UI and data objects, without being a C programmer. And that meant developing the Notes Object Interface, and writing the front and back-end classes.

So, you wrote the original LotusScript back-end classes?
Yes, I developed some prototype classes and a simple agent execution environment to use to sell Iris on the idea that this kind of thing should be in Notes. They liked it well enough to go ahead, and then (Iris founder) Len Kawell helped me re-design the entire object model that we shipped in Release 4.0. Jen Kidder wrote the front-end classes. After 4.0 shipped, I transferred from being a Lotus employee to being an Iris employee.

How has the whole LotusScript thing been received?
Pretty well, on the whole. It sure is being used a lot now, at Iris, to do lots of work in our standard templates, and by customers and business partners. Most of the feedback I get is requests for more functionality. I take that as a good thing; people are really using it and stretching the capabilities to do real work.

What's the Lotus Commitment Award?
I think it's when they take a vote that you be sent the hospital for a "vacation".

Jeff Papows and Mike Zisman have been giving out two Commitment Awards each year for a couple of years now. Mike wrote a very nice letter about what a great guy I am and how much they appreciate my contributions to the company. Then he read it out loud at a meeting of all the Lotus and Iris managers, and shook my hand. It was nice, though slightly embarrassing.

So, then came the Internet. What was your first move?
My very first move was to buy a book on Java and sign up for a course. Then I developed a prototype of the back-end classes in Java to see what that would involve, and posted it to our Web site to let people see that we'd done it and what it could do. Which was essentially a Java interface to Notes/Domino through these classes, provided you were running a Notes machine. The next thing I wanted to try was to see how I could use those classes from a non-Notes machine talking to a server -- a way of remoting the interfaces. JavaSoft has this capability called RMI -- Remote Method Invocation. I tried using that and I got to the demoware stage, so I had a working demo of a non-Notes machine talking over the 'Net to a Notes server in Java, manipulating the Notes objects. But RMI wasn't -- how should I say it? -- robust enough. Plus, it's Java to Java, just like Microsoft's answer, DCOM, is C++ to C++.

Is that how you arrived at CORBA?
Yes. A more general architecture for doing the same type of things on an "anything to anything" basis is CORBA (Common Object Request Broker Architecture). It describes how to implement objects so that they can be shared across machines, even when the machines involved are running different operating systems.

What's so great about CORBA?
CORBA is simply another way of remoting interfaces to objects, so that from one machine, you can call into objects that are really running on another machine. The nice thing about it is that it's language

independent, so that on the server side, the objects can be implemented in any language, and on the client side, I can be talking to them using any programming language.

What will this buy me as a developer?
With a remote interface to a collection of Notes/Domino objects, you can write Java applets stored on a Domino server. These can do all the things the Notes back-end classes can do in a client-server environment today -- create, store, retrieve and manipulate data in a Notes/Domino database. Plus, they can be linked to HTML pages, downloaded to and launched from any Web browser. Any applet written in Java on any Web server will run in any browser, on any other kind of computer!

You seem pretty excited about the combination of Java and Notes!
It opens the whole Internet to groupware. The stuff you can do now with JavaScript and Java on a Netscape or Microsoft server is new and interesting, but it's nothing compared to what you can do in Notes, because they don't have all the infrastructure Notes has -- scheduled agents, security, replication, collapsible sections, the object store. You're forced to use JDBC if you want a database, which is okay for certain things, but what if you don't want a relational database? What if you want secure agents that are not viruses? You have to make CGI scripting and Perl available to more people, and that's just where the market is going.

How does CORBA work with Domino?
With CORBA, you define your objects in CORBA's language neutral definition, called IDL -- Interface Definition Language, and then use compiler tools to generate the language targets you're interested in. For the server, you use a compiler to generate a bunch of C++ code, which you use to implement the objects. For the client, you generate a bunch of Java stubs that someone can download as part of an applet. The stubs represent the client side of your objects and they know how to talk to the server, where the real work happens. You can get stubs in any supported language -- Java, C++, Object COBOL, whatever. Obviously, we're most interested in Java.

So, what does this Java-ized set of back-end classes bring to Iris, Lotus and IBM?
They're going to reveal some amazing power for the developer community. Clearly, Web site developers, especially people developing Java applets for HTTP-based servers sites, want to run these applets in the context of a browser. And they'll get that capability from us in a nice way because we'll have the Java implementation that's remoteable using the CORBA IIOP protocol. The site developer can create sever side applets that are integrated into Domino and the Notes object store. The applet, at runtime, makes remote calls to Notes objects that really live on the server, and can access data stored in a Notes database, or in several Notes databases, from the browser. All the full text searching, fast lookup of records and so on are processed on the server, with the results transmitted back to the applet using IIOP.

That's one way we're servicing Web site developers. Another group is people wanting to do server-based applications -- especially things like agents. We're going to let people write agents in Java, or LotusScript. Or they may want to write Java-based server applications. So, we're going to have this Java interface on the Notes machine, as well as remoted.

Why should developers care about LotusScript, now that there's Java?
Well, first, Java didn't exist when we needed an agent language, so we had to use LotusScript, and now there's a large installed base of LotusScript applications that we're obviously not going to obsolete. But one nice thing about the back-end classes is that they're implemented in LSX -- the LotusScript Extension Architecture. I wish I could say it was by design, but it turns out that because of the way we've implemented it, it's easy to support interfaces to those classes from other languages. So the code that does all of the work on these Notes objects is identical, whether you call it from Java or LotusScript. So it was a very small amount of work to export a Java interface. The result is going to be that enhancements to the class hierarchy in the future are almost automatically going to be available in both LotusScript and Java.

Of course, it's important to keep in mind that you'll use LotusScript when building applications that need to work with the Notes client. LotusScript will be the way to build to applications that use all the power of the Domino server and Notes client.

Where should third-party developers be headed?
People are going to be writing Java agents and applets that use calls to these CORBA-based Notes objects. We're working on an authoring environment for Web sites where you'll have events in the browser trigger Java and call back into Notes objects that same as you can do with the Notes client today.

Then you'll start to see more server-side application programs being done in Java. One thing we're looking at is how the Domino server could allow Java 'servlets' to be added in, so that, for example we could write a program that gets a last crack at some HTML that's about to go out to a client, like final filtering of data. These are the kinds of tools people are going to need. We'll also let people write their own server objects, and use our CORBA setup to export the remote interfaces for them from a Domino server. A year or two from now you might very well see Domino servers being used to act as universal database connectors, where applets are written in Java that talk not only to Notes databases, but to Oracle and DB/2 systems as well. Even CICS, via IBM's MQSeries objects.

What's the most needed Java agent?
Definitely the Bozo Filter -- if people send you junk mail you can just add them to the bozo list so future mail from them gets deleted, and won't clutter up your inbox.

What are you doing with JavaBeans?
We're thinking that our Notes objects should be Beans, though what is and what isn't a Bean is a slightly fuzzy concept. Still, if our objects support introspection in a nice way, they'll be more usable in the various Java builder tools. The back-end classes, of course, will be invisible Beans, because they exhibit no UI. Jack Ozzie is also working on implementing some UI-oriented Beans that emulate parts of today's Notes client interface. He'll have nice looking renderings of the familiar Notes view UI that you'll be able to embed in an HTML page and interact with on a browser.

What do you think about JavaSoft's approach to app development "without interacting with class libraries?"
Sounds like they're just replacing "class libraries" with Beans. I think they should work with Iris to develop a standardized set of groupware Beans that are downloadable, but which use CORBA and IIOP to implement real Client-Server applications. Sure, some people want client-only apps, which we call brochureware, and others want server-only apps, such as agents, but lots of people really want and need distributed, client-server apps, just like you get with Notes and Domino today, using browsers as the client.

What's the timeline for the Java deliverables?
In 4.5, we delivered support for Java applet execution in the Notes browser, and a way for developers to deliver Java-based applications to Notes and Web browser users through the Domino server.

Next, we'll expose the Domino back-end classes via Java this summer. (You can look at an early non-Java Beans example at **http://www.notes.net/javacl.nsf**.). Using these classes, you'll be able to write applications and agents in Java. You can use whatever development tool you like; we won't be providing one in that release.

We'll expose the back-end classes via IIOP around the end of the year, or early next year. Domino will include an ORB at that time. We're also looking at providing some integrated developer tools in that release, particularly a Java debugger.

When will you make the Notes client a Java Beans container and the Domino servers a Java Beans publisher?
That's going to happen in Release 5.0.

What's keeping you busy right now?
Writing the real Java implementation of the back-end classes for 4.7, fixing LotusScript and Agent Manager bugs for 4.51 and 4.52, doing design and prototyping work for the CORBA stuff in 5.0. Trying to keep up with all the new stuff coming out.

You mention Agent Manager -- that's a Notes 4.0 feature we don't hear much about. What's new?
The biggest new thing is that the Agent Manager will (as of Domino 4.7) be able to run Java agents, as well as LotusScript agents. Also, we're focusing on fixing problems for the next maintenance release. Beyond that, well, there's lots of things we could be doing that I want to start investigating, like mobile agents.

What are mobile agents?
The idea is that you write an agent, probably in a portable language like Java, and it has the ability to transmit itself over a network to another machine and perform actions there.

What about security?
Yes, that's the big issue. We don't want mobile agents to be viruses. Another issue that we want to look into is whether these things are really going to be useful to people. We won't spend cycles on it just because it's a cool idea.

What's your dream for the future of Notes?
That someday everyone, not just the 15 million customers we have now, will realize how great it is (and buy it). That Iris will stay loose enough and sharp enough that we can just keep re-inventing the product as the world changes around us.