Level: All
Works with: Lotus Workplace
Updated: 01-Aug-2003

The Lotus Workplace preview with Jeff Calow

Interview by
Tara Hall

Editor's note: This interview provides a preview of Lotus Workplace. The terminology, concepts, and features discussed reflect the thinking at the time that this interview was conducted; however, they may not appear in or be applicable to the final shipping product.

What is Lotus Workplace? We asked Jeff Calow, lead architect for the Lotus Workplace infrastructure team, that question. We wanted to get the big picture, but we found out a lot more.

What is Lotus Workplace?
Lotus Workplace is a platform for collaboration built on J2EE and relational databases. It has capabilities that we used to sell separately. It's not just duplicating what we did earlier for collaboration, but reinventing collaboration in a new space.

One thing I need to say up front is that because we haven't shipped the next release of Lotus Workplace yet, the names I use in this interview and the offerings I refer to are not final and are likely to change before we ship. We are still in the process of tuning our terms, so this will be a snapshot of our current thinking.

Some of the primary components of Lotus Workplace are the traditional messaging pieces that bring mail, personal calendar, and address book together. The platform also brings together other collaborative capabilities like: team spaces, instant messaging, Web conferencing, and learning. We're building it all on one release schedule, so we have all these pieces together, unified around a single architecture and around a single schedule. Out of that one big collection of technology, we have different offerings, so customers can purchase the combination of functions that they need.

We have a platform that has messaging, team spaces, instant messaging, Web conferencing, and learning. Basically, we are building the platform, and we have offerings that are some subset of the platform. One of the interesting differences with Lotus Workplace is if you buy any of our enterprise offerings, we enable the installation for on-demand. For instance, you can phone your IBM representative to say, "I want to license this extra bit." Then you go to the licensing screen, and with the license to use the capability that you want, you turn it on. You don't have to re-install, you don't have to re-configure the whole system (but may need some configuration tweaking). That said, if you turn on something like learning, there is some extra configuration to be done (setting up the database and some servers, installing courses, defining curricula, and so on), but the other capabilities are configured out-of-the-box.

The analogy that Mike Rhodin—Vice President of Lotus Software—likes to use is this: When you buy hardware, the equipment you receive may have some extra CPUs added on, but they are not active. If you call your IBM representative to say, “I need more MIPS in my machine,” they say, “Type this code into the console.” Suddenly, those CPUs work. That enables customers to buy smaller, but to get an upgrade without having to reconfigure the infrastructure—reinstall the software. It's a more dynamic way of incrementally acquiring capabilities.

Because we’re building everything on one platform and one set of architectures, we’re set up to enable capabilities much more on demand. The other important aspect of that is even with all of these capabilities out-of-the-box, you have a management system based on policies that allows an organization to decide which people get which capabilities. For example, one of the target marketplaces for Lotus Workplace Messaging is the under-served market. Today there are a lot of people in the world who don’t have email. Some of our early business partners have factory floor workers or service workers who don’t have a dedicated desktop or a PC, but they still want to send them email and to let them interact with email. But they don’t necessarily want to give them instant messaging or any of the other common collaboration capabilities that they want to give to other people in their office. So they segment their user population into people who have certain ranges of capabilities.

Administrators can use the administration console to define classes of users and to determine which capabilities they can have. Then when the users come in to the portal for Lotus Workplace, they only see those capabilities that they’ve been allowed to see by policy. Because we are building Lotus Workplace on top of WebSphere Portal, we’re leveraging the dynamic assembly and access control capabilities of Portal to only show people what it is they're allowed to see based on the context that they're in.

We’re also learning from some of the manageability issues we had with QuickPlace and re-casting that into a Portal environment. Web conferencing is a bit of a twist on what we’ve done before with Sametime. We’ve done some studies and found that about 80 percent of all meetings are simple presentations in which someone shows slides. So we've come up with a light-weight way of giving e-meetings via presentations that don’t require any large Java download. We leverage the IM infrastructure and provide a shared "projector" based on an HTTP GET of an image of the slide. We think that we can satisfy 80 percent of the needs of people who conduct e-meetings. We want to optimize that particular experience and make it easier to use and much more reliable.

Finally, there's learning. We’re re-casting the learning experience into Portal over time. In the next release of Lotus Workplace, what we’re attempting to do is bring forth the "learner" experience. The people who come in via Portal are usually employees, so we want to present their experience within the context of the Portal workplace. But the administration of the learning capabilities still takes place in the current learning product. It’s an integration step. Over time, we will be fully integrating learning into the whole architecture.

When you talk about the administrator's experience with learning, are you referring to Lotus Learning Management System (LMS)?

Yes. There are two sides to LMS. There's the side of LMS that includes the course administrator's job to set up courses, to create course catalogs, to schedule the courses, and so on. Then there's the experience of someone who wants to take courses. The consumption of courses, or the learner experience, is done via Portal. We're expressing that part of the LMS experience through the Portal. The learning delivery service remains part of the existing technology.

So the LMS experience is split in two between small numbers of users who are the administrators of the learning system and the large number of users who are the people in your organization. Lotus Workplace provides a Portal location where all of your collaboration needs can be realized in one place. That is the Workplace. You don't have to separate applications to fill all of your collaboration needs. Lotus Workplace brings you one central organizing principle. Your email is there and your Buddy List is there. We're also doing analogous things right now with our current product offerings.

With Notes and Domino 6.5, we're trying to unify the experience to bring in the Sametime Buddy List and a new Welcome page. We're trying to bring together this concept of a workplace—the place where you spend your time as a knowledge worker—across all of the Lotus offerings. Or even as a non-knowledge worker, it's a place where you can go to get all of your needs met within the context of a corporation. Companies may also have other information they want to present to the employee inside of their workplace. We enable them to do this using Portal technology; we let you add other information, like company specific applications, HR portlets, and so on.

Let's talk a little bit about the technologies behind Workplace. You mentioned WebSphere Portal and J2EE.

What we're primarily building on is Portal, and there are a couple of tiers to our architecture, so maybe we need to talk about the generic architecture first and then drill down into the technologies that we're using to realize that.

We've broken up the problem into, essentially, three different tiers: the User Tier, the Service Tier, and the Resource Tier. The Resource Tier is the back-end database where you store information; the Service Tier is where you can make programmatic service calls; and the User Tier is where you present information to the user. Our User Tier is based on WebSphere Portal, so we're building our user interface capabilities using portlets. Also lumped into the User Tier architecturally are the standard email protocol handlers like IMAP and POP3.

The User Tier talks to the Service Tier that implements the EJBs (Enterprise Java Beans). The Service Tier talks to databases that are implemented primarily on DB2. Portal provides a light-weight content management system on top of DB2, so we're interacting with that instead of talking directly with DB2 for those areas related to document management (like team spaces). We also have some other specialized stores for storing full-text indexes and are leveraging indexing technology from across IBM to do our full-text indexing.

Our primary store is DB2. Over time our intention is to support more of the databases that Portal and WebSphere support, such as Oracle. Our intention is to be agnostic when it comes to data stores. One of the things that we're not going to be agnostic about is the application server. Most of our technology is built using standard J2EE, Java Beans, and so on. However, a bunch of our capabilities, like our protocol servers, plug into WebSphere at a level below the J2EE container. We leverage WebSphere as a server framework, as opposed to leveraging it as a J2EE application server. We're taking full advantage of all of the WebSphere management infrastructure and of their administrative console. Our primary administrative console plugs into the WebSphere administration console to perform system administration tasks. Our user administration tasks and other administration tasks are done via Portal administration or, in the case of learning, the LMS Web application.

For our User Tier, we use the open source struts and tiles framework that's been modified to run inside of WebSphere Portal and that's shipping with WebSphere Portal 5. We're basing Lotus Workplace on WebSphere Portal 5, which is based on WebSphere Application Server 5 and DB2 8.1. Obviously, we're also building some of our own underlying frameworks, which is what my whole team is working on for unifying security, administration, people, and some other common pieces.

You mentioned the security architecture. What can you tell me about that?

Our main thrust of security for the next release of Lotus Workplace is something called seamless authentication. We want to be able to log in to the Portal once and to get to all of the resources that you have rights to access without getting prompted again for another login. As a Web browser-based system, we want to leverage the underlying capabilities of the platform to provide a seamless authentication experience. We don't want you to have login prompts popping up. A lot of that we get for free because we leverage the infrastructure that supports it; we need to make sure that all of our components play properly within the infrastructure. That's one part of what our security team is dealing with.

Another really important area includes email and instant messaging. Suppose you're getting messages that could be in HTML format from unknown sources. You don't know whether or not it may contain malicious code embedded within the HTML. So we have two technologies that we leverage to help combat that. One is called an Active Content Filter. It takes any potentially dangerous content and nullifies it before it puts it on the screen. This is differentiated from a virus scanner—it doesn't check for viruses, it just disables all active content. If anyone sends you an email with embedded JavaScript, it'll be disabled before being displayed.

The second thing that we're doing is something called dangerous URL protection. A dangerous URL can delete all the email in a folder, delete a folder, delete an account, send a message to somebody else under your name, and so on. What a particularly crafty person can do is send you an email that has a link in it that drives that particular URL, so it's still in the context of your session. Because of our seamless authentication, we think it's still you coming in, so we execute that command. For every URL that can execute a potentially dangerous command, we've added something called a security nonce to it. This is a cryptographically secure random number associated with your session that an attacker cannot guess. Therefore, if a command comes in that does not have that nonce attached to it, then the command is rejected. Therefore, no one can send you an email with this malicious content.

Those are the primary areas of security. We're also doing a bunch of low-level integration with things like access control, so we're standardizing on the instance-based access control using Portal. We're doing a bunch of internal work for that level of integration. And our security team is responsible for doing end-to-end evaluations of security and is validating the security architectures of all the underlying components as well; that's another big task that our security team does.

Workplace will eventually support a rich client, and like the tools for WebSphere, this client is based on the Eclipse framework.
That's correct, but let's back up and ask why a rich client at all? Why can't the Web do everything you want it to do? The Web is good for all sorts of really important things. I think it has revolutionized certain aspects of people's experience for getting and publishing information and for accessing applications that are only used occasionally, like doing my expense report or my online banking. For me as an online banker or for me as an individual going to my brokerage account—buying and selling stocks and such—the Web is perfect. I don't need the fanciest interaction. If it's a little bit slow because of the download, that's fine because I'm not interacting with it on a day-to-day basis.

On the other hand, I live in my email. I think I'm getting close to 150 messages a day, so to process all these emails, the responsiveness of my application is critical to my productivity. If I have to wait while the Web page reloads, I'm not being productive. In my former life, I worked on PowerBuilder, so I have a bias towards very rich desktop clients. You can't get the kind of information-dense applications with complex interactions that you could build with PowerBuilder on the Web today. You can get relatively simple applications on the Web (and some folks are pushing the limits of the Web, like iNotes Web Access), but the most complex applications that I've seen in PowerBuilder you can't even come close to on the Web.

One of the problems with the client-server infrastructure was that the distribution model was really bad. The real issue was how do you get the applications that you've written on to every desktop (and also get the database access configured on each machine). That's really what was co-opted by the rise of the Web. So our goal with the rich client platform is to provide the richness of something like PowerBuilder or VB or a desktop application that gives you all the functionality that you require. The Web is really designed around keeping the content away from the rest of your desktop. All the things you come to expect from a real rich client (drag and drop, rich copy/paste, and so on), you can't do on the Web because browsers purposely sandbox you. On the other hand, Notes has a very secure model of how you run externally-sourced content securely inside of a particular

container without sacrificing interaction richness.

Then there was the choice of platform. We looked around. IBM is very oriented towards Java. But Java on the desktop has actually not had a very stellar reception from the industry. Also, one of the things that IBM was doing on the tools side was revamping all of our tools in Java. And the tools team had the same set of requirements; they had to compete against Visual Studio. It wasn't good enough to say, "I'm pure Java. I can run anywhere." They had to compete with the native applications. Not only did we need to compete with the native applications; we needed to be cross-platform to support both Windows and Linux clients.

As we were investigating, we realized that the WebSphere Studio team faced the same constraints—they built a framework on top of Java—a very powerful framework that I think can compete easily with what Microsoft has to offer. It provides all the desktop integration and desktop look and feel; it looks like a native application, but you write your stuff in Java. It was really a good starting point for what we need to do.

But we did have to make a few changes to Eclipse so that we could dynamically download applications and start them up. You want to have an on-demand application so that you click a link and open the Portal—because we're providing this in the context of Portal—and it says, "OK, you need these plug-ins to run the rich client experience." You don't have the plug-ins on your client, so it goes off to some central manifest and downloads the plug-ins that are required. Over time that will include full security and signing and all of the things that you want to run third party plug-ins. In the early releases, we're mostly distributing trusted applications, but in the fullness of time, it's our intention to move to a model very similar to Notes, where you have trusted roots, and the corporation can indicate trust for particular signers using code signing technologies to make sure that the mobile code is safer.

That was the other thing with client-server: You didn't have to worry about the safeness of code because it doesn't get on your desktop unless IT gives you something to install. Because we're dynamically installing code, the tag-line we like to use is "All the richness of a rich client with the administration experience of the Web." You get the same benefits you get out of having Web pages and Web applications, but you get all the richness that you want to have.

Will customers be given the choice between the rich client or a Portal experience?
Absolutely. But it's not an either-or situation. In some sense, it is either-or for certain classes of users. The factory floor workers will not get a rich client email experience; they'll walk up to a kiosk and log in over the Web. They fit the pattern of the occasional user of an application. There are other users for whom email is their life, and those sorts of people will get a rich client experience. And there's obviously going to be a corresponding price differential for that. It's still not going to be disparate from Portal though because it's really just a different surfacing of the Portal experience.

We like to phrase it as the rich client is just another kind of device for Portal that happens to have a lot richer experience. It really is tied in with the Portal experience. You can't just buy the rich client and run it with anything; you need our whole Lotus Workplace infrastructure to run it because it has affinities. Many of the reasons for that is the policy system I was telling you about. The access control system of Portal will control what's going to be expressed within the rich client. There's still a unified management model—it's all about making it as manageable as the Web.

Will there be capabilities in later releases to build your own applications?
Yes, absolutely. We heard loud and clear from our customers that if all we did was provide a bunch of PIM applications, that's nice, but they want to build their own applications. Our architecture is designed around multiple levels of integration. The easiest level of integration lets customers build their own portlets or get portlets from third parties and aggregate them together within the context of Portal. Secondly, we will have a bunch of services that we use to build our applications, and people are going to want to leverage those services to build workplaces of their own—team spaces of their own. They want to interact at a much higher level and they want to create applications by assembly, which is not programming—it's still sitting down and saying, "I want to use this portlet and this portlet, and I want to wire them together." But the interaction capabilities are much greater.

Portal, again, has some starting points for this, and we're going to take it to the next level in the subsequent

releases. The capability won't be in the next release of Lotus Workplace, but in subsequent releases we'll provide a tool that allows you to assemble applications. If you think of the query by example case, you put together something that looks like what you want. Turn it into a template so that other people can stamp out their own versions of it and all the data underlying it. It's very similar to taking the team room template that we ship with Notes, modifying it to meet the needs that you want, and in some ways, creating your own template out of it. It's not going to be the same kind of technology, but it's the same sort of idea as what we're trying to do.

Finally, down the spectrum there are those who want to build their own portlets. We'll provide some capabilities for system programmers who want to build more components, so they can build components that plug into our assembly model. Third parties and business partners can take our assemblies, take our components that you can assemble within them, and add their own components in to build vertical offerings.

There'll be mobile abilities in Lotus Workplace as well.
Having a mobile projection is very important. Some of the ways that we're going to have the projection is via mobile browsers. That's coming in later releases as well. The next release of Lotus Workplace won't have mobile capability, but in subsequent releases we plan to add in mobile. We will enable capabilities to synch down to your particular devices, obviously, because it's all part of the full PIM experience. On the further edge, which is still very exploratory, is the ability to write signature applications that run on the particular device. We're investigating that in concert with our pervasive division to explore what different options we have.

Of course, Java is write once, run everywhere, so we have some ability to leverage it on those smaller devices. Obviously, the same application cannot run in both places. For mobile users who are laptop oriented, the full suite of the rich client, including working offline, is definitely supported in the early releases, and then improved as time goes on. That's one of the key strengths of Lotus—our offline story.

Is there any duplication of effort with the QuickPlace portlet and the Sametime portlet that already exist in the Portlet Catalog and Collaboration Center, which has a Web Conferencing portlet, a Team Workspaces portlet, and People Finder portlet?
What's happening is the existing portlets are not going to go away because they're providing access to particular back-ends. We're writing new portlets that have access to our new back-ends. Some of that is just the nature of the way the portlets are done. The intention of our portlets and workplaces is to fit into a particular architecture, to fit together in a seamless whole, and to allow them to be aggregated into other customers' applications, more so than just dropping a portlet down. We're using the property broker capabilities of Portal to enable us to provide the experience that people want.

Our intention is to evolve over time. As we evolve into this new infrastructure, we don't want to remove the ability for you to take advantage of your existing infrastructure. Suppose you buy Lotus Workplace and you happen to have Domino and want to use the Domino portlets—you can interject the Domino portlets into Lotus Workplace and have them sit side-by-side with the other Lotus Workplace portlets. You don't have to pick an either-or. That's why there's an explosion of the number of portlets that are out there—to provide access to individual back-ends.

ABOUT JEFF CALOW
Jeff is an IBM senior technical staff member with a focus on Web technologies. Jeff has been at IBM since 2000. He worked on J2EE integration and the JSP tag library for Notes/Domino 6 and has been helping define and deliver J2EE-based collaboration since 2002. Prior to IBM, Jeff spent his career working on application development tools. He has worked on PowerHouse 4GL from Cognos, a stillborn development tool from Sybase, and PowerBuilder from Sybase/Powersoft. Just prior to IBM, Jeff helped architect and build a personalization server using J2EE server technologies at an Internet startup. Jeff's professional interests include programming language design, distributed computing, and software engineering practices. When not working, he helps to raise a family and enjoys meditating, watching movies, and reading science books. His main topics of interest are philosophy (both eastern and western), cognitive science, evolution, and psychology. He holds a BS in Computer Science and Physics from Carleton University in Ottawa.