Interview with Cliff Reeves

IBM DIRECTIONS

Cliff Reeves is IBM's director of object technology products. He is responsible for object-based products in IBM's Personal Software Products Division, including responsibility for the IBM-Taligent partnership. We talked to Mr. Reeves about the impact he sees from object technology being adopted by software companies and corporate developers.

What makes object technology so important at this time in our industry?

Reeves: The use of personal and desktop computers has only just begun. Even though we've seen a phenomenal growth in the past 10 years, there are many applications still waiting for computers. Increasingly, companies want to distinguish the way they offer products and services by how quickly they react to individual customer requirements and environments. For example, personal bankers need more and more information to say, "Yes, you can have that loan, and here's the interest rate and the credit limit." They must be able to do that without completing a lot of forms.

The person who handles customer complaints at an airline counter needs to recognize the frequent travelers. The last thing airlines want to do is infuriate those who travel the most. They would like to immediately say, "Yes, and by the way, we can put you in first class."

Everyone serving customers should be able to access information about the customer so they can tailor their service to meet the customer's preferences. That's the way many companies plan to distinguish themselves in the future. And this permeates every aspect of industry—from buying cars or airline tickets, to securing health insurance or visiting the doctor. Exceptional customer service is a competitive advantage. That means that everyone will have a computer on their desk for accessing customer information so they can make very quick and timely decisions.

How does object technology fit into that scenario?

Reeves: Since not everyone is computer literate, these new users will need systems that look the way they want them to look. They will need access to information, independent of concepts such as operating systems, databases, files, commands, and prompts. They will want meaningful information presented in context, using highly graphical, tactile interfaces. They will also want to work in a distributed environment, because not all data can be provided centrally or through a homogenous environment.

In addition, this future environment will change very quickly. So we need systems that are unbelievably flexible, decentralized, and good at managing change.

I have just described an incredibly complex development environment. It's very difficult to produce flexible applications. Most applications have their associated operating system and networking environments imprinted in their "DNA."

We need systems that are linked together dynamically. A request for information—satisfied one day by a centralized server in Albuquerque—must still be satisfied if that server is moved to New York. The application should not have to be redone completely to recognize that the server is moved.

Providing these kinds of systems is complicated. Assume that the Application Programming Interface (API) count is some measure of the complexity of writing an application (which is not entirely accurate). When we move to graphical interfaces to provide a friendlier and tactile, high-bandwidth communication between the user and

Cliff Reeves

the machine, the number of APIs increases by a factor of 10.

We have not even begun to discuss the issues of collaborative computing, distributive work, and the effect of the API count there. We need a set of technologies that allows us to deal with that. We need a new approach. Object technology is that new approach.

We will need to be very responsive in the future, and object technology seems to have an answer. What evidence shows that object technology will be adopted as *the* answer?

Reeves: Two or three things are happening that involve some type of standardization. First, we're beginning to see standards emerge for distributed object computing. We can see a mechanism now in the Distributed Computing Environment (DCE) and in the work within the Object Management Group (OMG). Within the OMG, we can define large separate components to handle the issues of security, location, high speed, and all that is needed to run a distributed operating environment.

Object technology allows you to build one application with reusable parts.

Secondly, we're seeing the concept of a reusable software component, which has happened very quickly. We see primitive signs of it in Visual Basic and Object Linking and Embedding (OLE), and more recently in OpenDoc, which specifically defines a reusable software component. It is not necessary for applications to be built from scratch or tied to a particular topology or platform. And more importantly, they don't have to be tied to a particular application, but can be accessible and reusable in combination.

Finally, we're seeing the emergence of a new kind of reuse—design reuse. Object technology allows you to build one application with reusable parts, and then build the same style of application from the same library of reusable components. Once you've done that two or three times, suddenly something else happens: you have a model or template showing how any particular type of application is constructed from reusable components. That produces a new method of software development called *frameworks*.

During the last four or five years, we've been hearing more about frameworks—mostly from the C++ programming world. Setting standards for frameworks takes the concept from niches into general use.

We're seeing the potential for a software value chain that really has three levels. At the lowest level, programmers (using languages like C++ and Smalltalk) will use standard frameworks for constructing core application components. At the next level, system integrators and software vendors will combine the standard components and add value by tailoring a particular program to solve a customer's problem. And lastly, users will receive the benefits from those and, in turn, manipulate the component, record the script, automate standard behaviors, and add more value.

Do you think customers and end users are ready for solutions that object-oriented technology can provide?

Reeves: I just completed a two-week road tour talking to IBM customers, consultants, and the press. IBM, Apple®, and WordPerfect® are collaborating on a software component specification that we plan to deploy across all appropriate operating systems. These companies see several benefits.

Apple has continued to innovate by offering technologies such as Quicktime®. The problem, according to Apple, is the length of time between delivering the technology as part of the operating system and its actual delivery within end user applications. It might take a couple of years to develop the technology. Then they have to evangelize developers to adopt it, and wait for a couple of years until products are available. Apple believes that a better way is to define a standard so that the software is written to a component mechanism, similar to Lego® blocks. Then every time we offer a new set, it would automatically go into applications—similar to a plug-and-play environment. We would like to use that as an operating system extension.

Another example comes from a large insurance company. This diversified financial services company just bought an investment banking firm that uses a lot of UNIX workstations and desktop PCs. The insurance company primarily uses MVS™ mainframe applications. An Information Systems (IS) person from the insurance company said, "When I need to provide an IS solution, I must build software from already existing parts, because I can no longer serve the wide variety of my own internal users by starting from scratch—I have to build from reusable software.

He added, "I had an interesting experience the other day. I walked into a department at the investment bank and said we're planning a new corporate time-and-expense accounting system. They almost threw me out. They had been running their time-and-expense reporting locally using a commonly available spreadsheet for almost two years. They print the reports and someone enters them into a transactional system for auditing, and so on. They are happy with what they have and it is familiar.

"They didn't want my new mainframe view of a time-and-expense accounting system. They wanted me to add a button to their spreadsheet to allow them to transfer the data to auditing and into our system." So from that experience, the IS person says that future applications must build from the equity that users have in their personal productivity products, yet they need to be easily extended. For that, we need a software component architecture.

How is IBM addressing the problem?

Reeves: Our answer is a set of services. I'll briefly describe a couple of them, just to give the flavor. It starts with a set of universal system services, including the following:

- Open Blueprint, providing low-level, fundamental linkage and access to common system services
- A set of integration services that totally redefine reusable software
- A set of very powerful and exciting tools that build on this model

We try to avoid building new, completely integrated monolithic applications. We want to help you build from parts—whether it's small parts like you find in class libraries, large parts like a thesaurus, or larger components, such as stock quote access.

We will deliver the functions of our operating systems in totally reusable components. That is not really new to the UNIX community, which has utilities that can be piped together to provide powerful functions. But this component strategy takes us to the next semantic level, in which the components will be more visual and have more rules about their interaction.

Could you explain what you mean by integration services?

Reeves: The integration services provide a specific definition of a reusable software component. Essentially, it's the Lego specification as well as the set of services for a pluggable software component. Integration services address four issues:

- How do you find a software component, make sure it is secure, and locate it on the network? That's based on IBM's System Object Model (SOM) technology that is, in turn, based on OMG's Common Object Request Broker Architecture (CORBA) specification. It's a very thorough, low-level programming interface to access a component.
- What is the visual part of the software component? How does it interact with other visual components? If I drop this component into an object, how do they negotiate space, and which one takes the events and maps them into a different set of actions that can be performed on the object?
- How do you link these components? To build reusable software, these components are often linked together very late in the development cycle. If we're going to have "blister pack" software that links together, then we need a more dynamic means of linking components together—from the local department, to the IS group, to end users who may just want to automate their mail box. Our mechanism for doing this is called the Open Scripting Architecture, a set of templates for linking components.
- How does the component store its data so that it can be interchanged? If applications, such as a bill of materials, are built dynamically from sets of parts, the data that defines the bill is likely to change. We need a technology that allows data to be self-defining so that as it changes, it can still be exchanged and understood. We call this technology Bento™—after the Japanese lunch box that keeps ingredients separated and in order.

We will deliver the functions of our operating systems in totally reusable components.

That's certainly a lot of function. What can we expect to see in the near future?

Reeves: We are quickly moving toward implementation. This year we'll deliver the integration services piece to developers on OS/2®, AIX, Macintosh® System 7, and DOS/Windows™. We'll introduce it early in 1995 on our MVS systems and on OS/400®. It's probably the single, most

fundamental change in the next two years that we'll see in the definition of reusable software components.

Workstation users have dramatically demonstrated that they want open systems with open software. What steps is IBM taking to ensure that other companies adopt these ideas, so that it is not an "IBM-only" solution?

Reeves: The environment I described is heterogeneous, multivendor, and multiplatform. We're deploying it on every important operating system. The technology will be licensed based on standards and partnerships. This technology (from WordPerfect, Apple, and IBM based on a set of industry standards) is being transferred into the ownership of a company called Component Integration Labs (CI Labs). CI Labs was founded on the model of the X consortium. The theory is that because of the major importance of many vendors agreeing on a component standard, the technology must be owned by a forum that is entirely accessible, open, and not controlled by a single vendor.

Although CI Labs will control the branding and certification process, anyone can obtain the source code for this technology and the specifications by anonymous ftp.

CI Labs was just recently incorporated, the bylaws have been signed, and anyone can join it. Just like the X consortium, there are membership levels giving increasing participation in the SIGs and in defining the standards. Although CI Labs will control the branding and certification process, anyone can obtain the source code for this technology and the specifications by anonymous `ftp`. The extreme level of membership is almost a non-membership, but with full access to the technology when it becomes available through anonymous `ftp`.

You can contact CI Labs now, by sending a note containing the word "help" to `majordomo@cil.org`. In fact, on the mail server you can browse all the directories that contain code, specifications, and development plans for the technology (See the sidebar on the opposite page).

What can we expect from other vendors?

Reeves: Endorsements have come from Lotus®, Novell®, Xsoft™, Borland™, and Taligent. Although each vendor has a slightly different deployment plan, you will see alpha code in the late spring and early summer of 1994, beta test in the late summer and early fall, and general availability from the different vendors in the late fall through early 1995.

You mentioned there would also be new tools.

Reeves: We are producing a new breed of compilers of "power tools" that assume the developer is building or using software components or frameworks. Most of these tools are highly visual.

We're getting an exciting set of tools from Taligent. These tools are for professional programmers who want to build the reusable software components and who are very focused on high-level function and productivity. These developers want to develop quickly and efficiently in their choice of programming languages, and to leverage their investment across several operating systems. Taligent will support the OpenDoc standard and extend it into a complete new environment for both developers and end users. An example is IBM's VisualAge™, a component-based tool that will move to the SOM and OpenDoc standards during 1994 and 1995.

That requires a powerful and not highly abstract technology. Taligent is producing all of its technology in C++ frameworks for professional programmers. They've taken the concept of what you would do with a class library of reusable software and used it to consider problems such as addressing text with very rich text functions.

How does IBM's Open Blueprint, announced in March (see *AIXpert*, May 1994), fit into the overall plan?

Reeves: The Open Blueprint is a set of low-level system services. When there is no consensus of what model to use for a particular low-level system service, Open BluePrint provides a recommended standards-based interface and hides all the complexities of the implementation. If you are working on a new emerging area for a system service, it recommends using a particular API since there is no dominant standard. Because it is based on an X/Open, OMG, or ANSI® standard, Open Blueprint is very explicit in almost every area except some emerging areas that do not have standards.

Since they are based on standards, SOM and Distributed Computing Environment (DCE) are the interfaces and system services that we recommend. We will support them in all of our systems. DCE is based on the standards delivered by

the Open Software Foundation® (OSF®) and is increasingly being adopted by system vendors. We are freely licensing IBM SOM to other system vendors. It will be part of the OpenDoc standard. It is based on, and fully compliant with, the OMG's CORBA specification for distributed objects.

At a lower level, Open Blueprint includes a technology called *AnyNet*. AnyNet is based on Berkeley Software Distribution (BSD) sockets. It's a *de facto* standard because of its simplicity and its adaptability to various underlying protocols. Open Blueprint supports all standard protocols underneath that level of interface, including TCP/IP, NetBIOS, IPX, System Network Architecture (SNA), and so on.

IBM is delivering a broad spectrum of object-oriented technology along with a migration strategy during the next year. What are the major benefits for developers and end users when they follow this strategy?

Reeves: Software and system vendors need cross-platform, multivendor integration so they can be confident that their software conforms to standards. They also want to leverage their development efforts by using software across platforms. When they write to the OpenDoc standard and use the Taligent frameworks, the result is productivity and portability across OS/2, AIX/6000®, HP/UX, and OS/400. It also allows them to focus on their added value.

System integrators and solution providers can depend on software that comes from multiple vendors and multiple platforms to have some real semantic and business value in terms of how it might be linked together.

Users can get new function quickly. When they need new function for an application, they will not be limited to the vendor that provided the original package. Other vendors' software components can extend that package very gracefully.

That summarizes what we're delivering this year. Everything I've mentioned will ship to developers in 1994. A couple new items on midrange and mainframe systems will come later.

Building Tomorrow's Solutions Today

Component Integration Laboratories (CI Labs) is the result of several leading companies banding together to openly develop and promote component software. This non-profit organization was founded by Apple Computer, IBM, Novell, SunSoft, Taligent, WordPerfect, and the XSoft Division of Xerox™. CI Labs will provide the technological specifications and foundation technology for developing and integrating component software through the OpenDoc architecture (see article on page 16).

Supporting Technology Development

CI Labs will license and promote the technologies contributed by the sponsor companies, and distribute them as reference source code. Members will continue to compete on other technologies that differentiate their products, as they contribute to CI Labs their "non-differentiating" infrastructure technologies. CI Labs will manage the specifications for the technologies under its control, and will perform the following functions:

- Adopt and promote the key technologies essential for software components to integrate information and media from different applications within a networked environment and on multiple platforms
- License reference source code to developers and system vendors
- Provide open access to decision making and priority assessment in a vendor-neutral forum
- Manage and facilitate software contributions, design discussions, technology definition and evolution
- Support developers in adopting these new technologies by providing test suites, documentation, and training
- Validate the interoperability of components and platforms through a comprehensive testing program
- Assist in marketing efforts
- Collaborate with standards organizations (such as the OMG)

CI Labs Internet Lists

CI Labs provides related services and forums regarding CI Labs technologies and component software in general on the Internet. To request information, address a message to `majordomo@cil.org`. In the first line of the body (not in the subject line) enter `help` to receive E-mail describing majordomo or `lists` for a list of CI Labs mailing lists. For more information, send mail to `cil@cil.org`.