71 OFFICE PRINTERS TESTED

BYTE

MAY 1994

THE MAGAZINE OF TECHNOLOGY INTEGRATION

Graphical Front Ends Ease Access to the Internet PAGE 30

21-inch Monitors PAGE 157

IBM's Personal Dictation System Reviewed PAGE 145

ComponentWare

Object-oriented computing has failed. But component software, such as Visual Basic's custom controls, is succeeding. Here's why. PAGE 46

PLUS

- **Wireless Communications Gets Real**
- **Lotus cc:Mail vs. Microsoft Mail**

$3.50 U.S.A./$4.50 IN CANADA
A McGraw-Hill Publication/0360-5280

COMPONENTW

Object technology failed to deliver on the promise of reuse. Visual Basic's custom controls succeeded. What role will object-oriented programming play in the component-software revolution that's now finally under way?

JON UDELL

Tom Button, Microsoft's Visual Basic czar, loves to show how Visual Basic's custom controls have galvanized the component-software business. "Here are the 16 controls we shipped with Visual Basic for Windows 1.0," he says, positioning the Toolbox window in the lower left corner of the screen. "When we shipped version 2.0, third-party custom controls were already becoming common." He sweeps the Toolbox upward to reveal several dozen controls. "And here's the situation today." Now the Toolbox fills the screen with a dense mosaic of all the custom controls the machine's memory and disk can hold.

The fact that VBXes (Visual Basic custom controls) today best exemplify the decades-old notion of reusable software has been a surprise for everyone, including Microsoft. VBXes aren't just for 3-D buttons, gauges, and scrollable grids. National Instruments (Austin, TX) will sell you a VBX that controls GPIB (general-purpose interface bus) instruments. Cimflex Teknowledge (Palo Alto, CA) offers a VBX-based expert system. Distinct (Saratoga, CA) packages its TCP/IP programming kit into a VBX. Diamond Head Software (Honolulu, HI) offers a suite of image-handling VBXes. Stylus Innovation (Cambridge, MA) sells one that you use to build voice-response and fax-on-demand applications.

These are all actual "off-the-shelf" components that you can use to build real applications in a hurry. They are not, however, objects—at least, not the sort of objects that aficionados of C++, Smalltalk, or Objective-C embrace.

Real objects, as OOP (object-oriented programming) experts rightly point out, rest on the tripod of inheritance, polymorphism, and encapsulation, while VBXes stand only on the single leg of encapsulation. But if that's a crippling limitation, why has VBX—rather than OOP—ignited the component revolution? Why have C++ vendors such as Microsoft and Borland had to reverse-engineer Visual Basic so that programmers, lacking reusable C++ objects, can tap the rich VBX component market?

These ironies have spurred all the major players in the software industry to rethink the role of object technology vis-à-vis reusable components. What has emerged is a new, more realistic understanding of how a component-software industry can work.

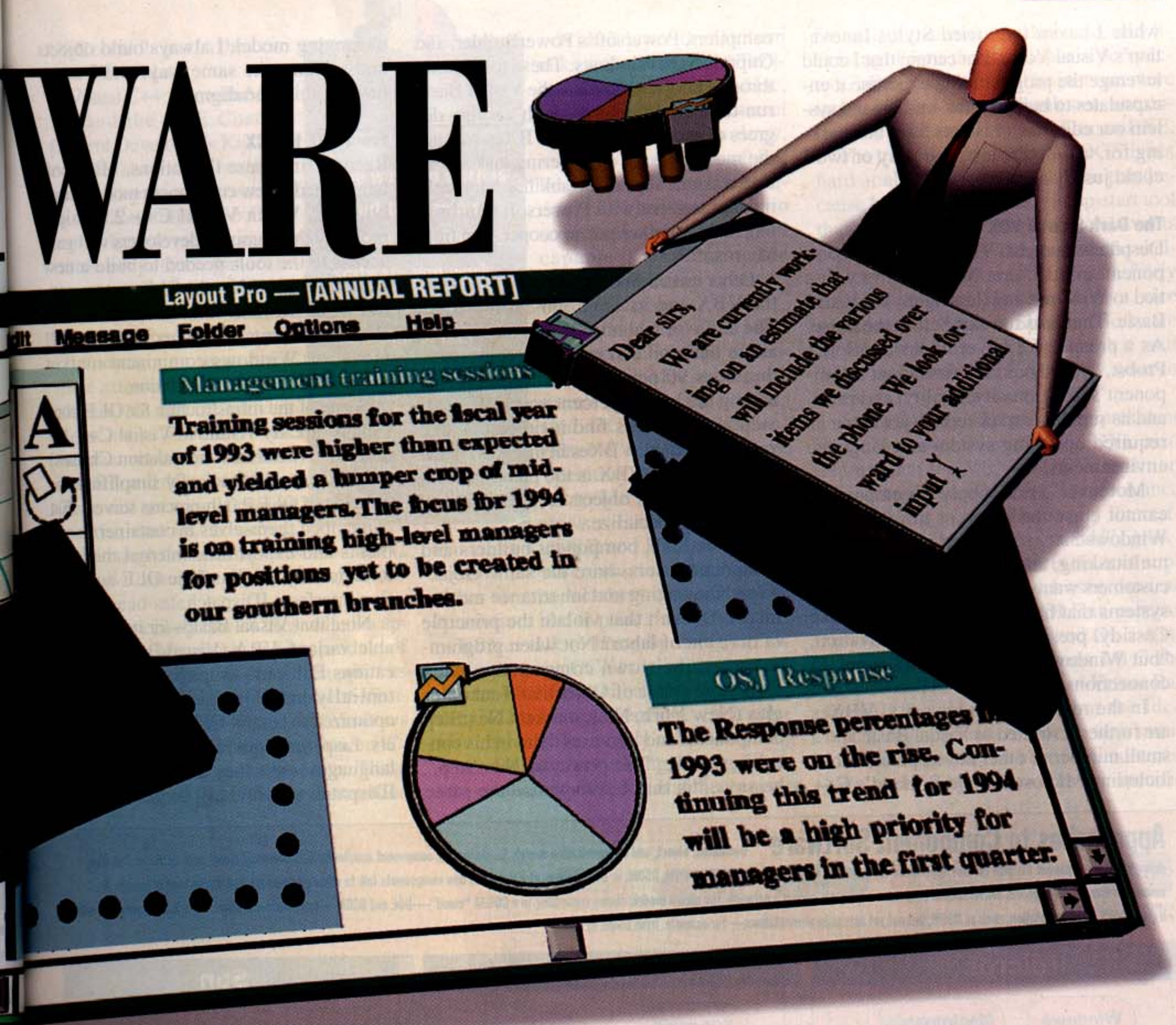
Rethinking Reuse

The traditional OOP vision was, at best, vague on the subject of reuse: Objects would appear as by-products of software development, a market would emerge, and programmers would become producers and consumers of objects. Why didn't this happen? There were two major roadblocks. Most OOP language systems, including C++, lack the means to package and distribute objects effectively in binary form. More subtly, the skills and disciplines needed to build components are often quite different from those needed to use them.

Apple, DEC, IBM, Microsoft, Novell, Sun, and others are busily revamping their system software and tools in an effort to break through these roadblocks. Despite incessant bickering, they're all headed down the same path.

An alphabet soup of standards, including Microsoft's COM (Common Object Model), IBM's DSOM (Distributed Sys-

WARE


Layout Pro — [ANNUAL REPORT]
dit Message Folder Options Help
Management training sessions
Training sessions for the fiscal year of 1993 were higher than expected and yielded a bumper crop of mid-level managers. The focus for 1994 is on training high-level managers for positions yet to be created in our southern branches.
Dear sirs, We are currently working on an estimate that will include the various items we discussed over the phone. We look forward to your additional input
OSJ Response
The Response percentages in 1993 were on the rise. Continuing this trend for 1994 will be a high priority for managers in the first quarter.

tem Object Model), Sun's DOE (Distributed Objects Everywhere), Hewlett-Packard's DOMF (Distributed Object Management Facility), Next's PDO (Portable Distributed Objects), Novell's AppWare Bus, and the Object Management Group's all-embracing CORBA (Common Object Request Broker Architecture), will provide the mechanisms for component exchange that pure OOP failed to deliver. Meanwhile, a restructuring of the software industry will define niches and appropriate technologies for the component builders who create reusable packages, as well as for the solution builders who assemble components into end-user applications.

At the end of the day, applications are all that really matters. Thirty years ago, we began hearing about a software crisis. The crisis was, simply, an applications backlog—too few programmers, too little time, and too much demand. Since then, every new paradigm for the construction of software—structured programming, CASE, OOP—has been billed as the way out of the software crisis. Yet we've hardly made a dent in the backlog. True, we're far better served by commercial software than in the past, but it remains horribly expensive to build custom software that automates processes unique to particular industries or individual companies.

To drive down the cost of custom software development, you have to apply a principle that software theorists have known for years. The best programmers aren't just a little better than average programmers; they're shockingly better—10 times, maybe 100 times more productive. And yet, says Richard Probst, SunSoft's manager of business development for project DOE, today we see virtually no division of labor in the software industry. "The way you work is about the same no matter what kind of software you work on," he says, "and that's a sure sign of an immature industry."

The VBX phenomenon is an important first step toward maturity. VBX-enabled programming differs markedly from conventional programming. You create applications by arranging controls on forms, editing the controls' properties, and writing a few—often surprisingly few—lines of event-handling code in Visual Basic, or C++, or whatever language is native to the environment that hosts the VBX.

This simple discipline, which is standard across all domains served by VBX controls, enables average programmers (like me) to build custom applications in hours or days. It took me just two days to put together a useful client/server database application using Coromandel's Integra VDB. And

ILLUSTRATION: JOHN CORBITT © 1994

while I haven't yet tried Stylus Innovation's Visual Voice, I'm certain that I could leverage the programming expertise it encapsulates to build the fax-on-demand system our editorial assistants have been asking for, and do the job in the day or two I could justify spending on it.

The Dark Side of VBX

Despite its success, VBX is a flawed component architecture. Most glaringly, it's tied to Windows and (less tightly) to Visual Basic. That puts the cart before the horse. As a prospective buyer, notes SunSoft's Probst, "you should ask first about a component's functionality, quality, and price, and its supplier's track record, not about its required operating system and language environments."

Moreover, a rich supply of components cannot erase the inherent limitations of Windows 3.x—segmentation, cooperative multitasking, and fragility. "Some of our customers want to build T1 voice-response systems that handle 24 lines," says Mike Cassidy, president of Stylus Innovation, "but Windows can handle only about 15 connections."

In the realm of Windows 3.x, VBXes are further restricted to Visual Basic and a small number of other development tools, including Microsoft's and Borland's C++ compilers, Powersoft's PowerBuilder, and Gupta's SQLWindows. These tools jump through hoops to emulate the Visual Basic run-time environment—with varying degrees of success. "Hosting VBXes was not the most pleasant engineering task we've undertaken," says Bill Rabkin, senior technical evangelist with Powersoft (Burlington, MA), "and we got no cooperation from Microsoft."

Other critics find the boundary between the VBX and its environment too rigid. The allure of real object technology, after all, is that you can modify a component that does 90 percent of what you need, adding the last 10 percent yourself. NextStep programmers find it ridiculous that you can't extend VBXes in this way. Their equivalent to a VBX is the palettized object, which other objects can freely inherit from and specialize.

In NextStep, component builders and component users share the same Objective-C messaging and inheritance mechanisms. Doesn't that violate the principle of division of labor? Not when programmers use their own components. Alex Cone, president of Objective Technologies (New York, NY), markets NextStep components and also uses them in his consulting work. "The power of NextStep," he says, "is that I always use the same messaging model, I always build objects and systems the same way, and I never have to shift paradigms."

From VBX to OCX

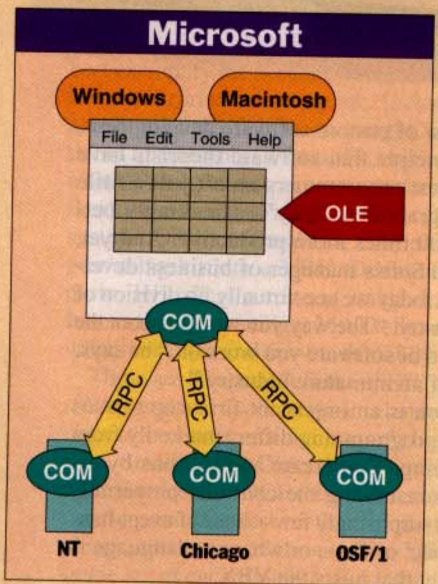
Recognizing these limitations, Microsoft has created a new component model based on OLE. When Visual C++ 2.0 ships, probably this summer, developers will gain access to the tools needed to build a new generation of VBX—the OLE custom control, or OCX. OLE controls won't silence all the criticisms of VBXes, but they will move the Windows component market onto a much firmer foundation.

Some of the infrastructure for OLE controls is already visible in Visual C++ 1.5 and MFC (Microsoft Foundation Classes) 2.5. That tool set radically simplified the creation of OLE 2.0 in-process servers that can embed themselves in container documents and export their internal methods to callers by means of the OLE automation interface, IDispatch.

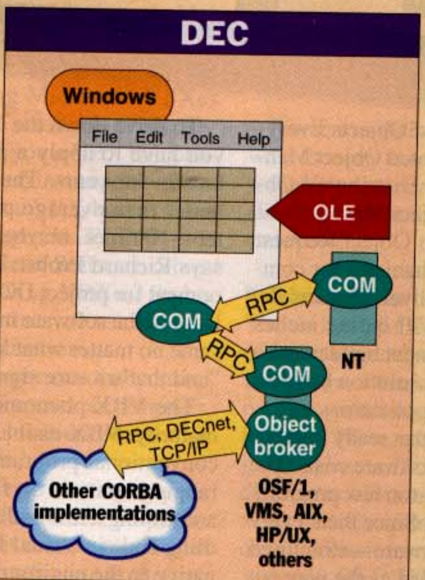
Note that Visual Basic—or its embeddable variant, VBA (Visual Basic, Applications Edition)—is only the first of potentially many languages that will be optimized to control OLE automation servers. Lisp, Smalltalk, and other interpretive languages, once they are retrofitted with IDispatch support, will be able to wield

Approaches to Component Software

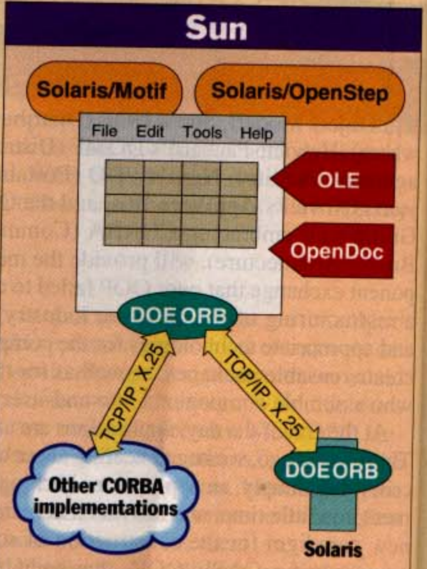
Document, object, and communication models for six leading component architectures. A document model, such as OLE or OpenDoc, defines how a component fits into the GUI application environment. An object model, such as COM, DSOM, or DOE, defines at a high level how components talk to other components that may be local or remote. A communication model supports conversations between components across a network. The object models shown connecting to a CORBA "cloud"—DOE and DSOM—are CORBA-compliant. There is interoperability within a given CORBA implementation, such as DSOM, but not yet across implementations—for example, from DSOM to DOE.



Today, OLE 2.0 and COM support Windows (and soon, Macintosh) components. A distributed version of OLE 2.0 that will support networked components is also in the works, and it has been demonstrated using a version of COM licensed to DEC.



DEC plans to bridge the worlds of OLE/COM and CORBA. ObjectBroker, acting as a gateway, will enable OLE components to communicate with CORBA components running on a variety of platforms, including OSF/1, VMS, AIX, and HP/UX.



OpenStep brings to the Solaris platform the rich application environment of NextStep and its wealth of object-oriented components. Sun plans to support OLE and OpenDoc. DOE components will communicate with each other and with other CORBA implementations.

OLE controls just as they can now call DLLs.

Visual C++ 2.0, with MFC 3.0 and the OLE Custom Control Developer's Kit, will enable such automation-aware in-process servers to mutate into full-blown OLE controls that maintain editable properties, generate events, and can bind to data the way VBXes do today. The redistributable run-time DLL containing support for these extensions will be available in 16- and 32-bit versions; unlike VBXes, OCXes will run natively on Windows 3.x, NT, and their successors. They will not initially exploit multithreading, however, even though the enabling substrate—MFC 3.0—will itself finally be thread-aware and thread-safe.

"The way you work is about the same no matter what kind of software you work on, and that's a sure sign of an immature industry."

Richard Probst, manager of business development, SunSoft

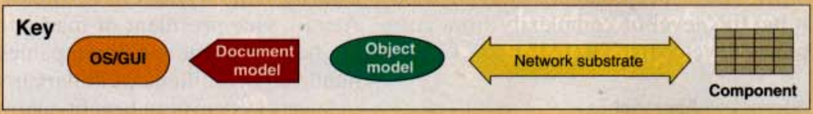
It's very likely that OCXes will also appear on the Macintosh, as a by-product of work that Microsoft is doing to support its own Mac applications. Versions of Visual C++ 2.0 that are hosted on Windows NT—but target 680x0- and PowerPC-based Macs—are in the pipeline. These tool sets support an MFC layer that rests on a Win32 layer that in turn talks to the native Mac Toolbox. FoxPro 2.5 for the Mac, which is built with an internal version of this technology, validates what Microsoft has long claimed: that the Windows API has the ability to serve as a cross-platform API capable of expressing the core of substantial commercial applications.

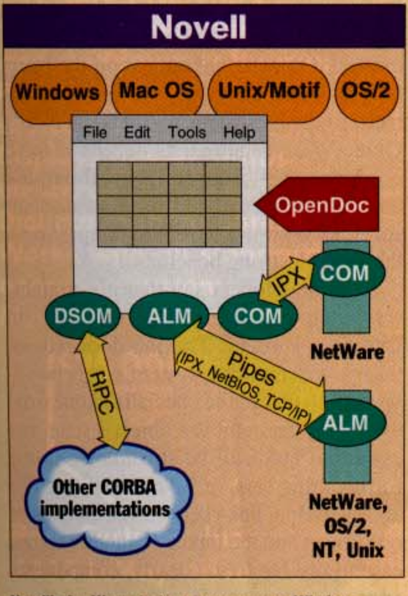
If Visual C++ 2.0, MFC 3.0, and OLE 2.0 all materialize on the Mac as planned, there's every reason to expect that OCXes will become portable, at least across those operating systems that matter to Microsoft—the Windows variants and System 7. Prospects for OS/2 and Unix, where Microsoft has no commercial interest, are rather dim, as members of the OpenDoc consortium like to point out.

VBXes talk to hosts by firing events. To implement OCXes in a similar way, the run-time DLL will add new interfaces to OLE 2.0 to implement an event mechanism. It will also supply common dialog boxes for property editing; stock properties, events, and methods; and mechanisms for self-registration, persistence, and licensing. From the developer's perspective, an OLE control will be just one more target the tool set can crank out, not noticeably different from a DLL or an EXE.

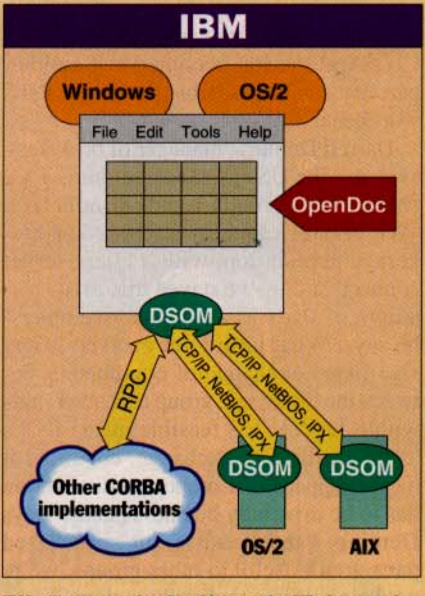
The transition from VBX to OCX is not hard at all, VBX vendors say, in part because Microsoft provides a jump-start tool that can look at the properties and events supported by a VBX and generate the skeleton of a compatible OCX. "It's a fairly mechanical port," says Joe Modica, vice president for R&D at Sheridan Software Systems (Melville, NY), "although if your VBX was written in C, you may want to think about converting to C++."

Dorai Swamy, executive vice president of Coromandel (Forest Hills, NY), also reports that the VBX-to-OCX transition is a no-brainer and that OCX performance seems fairly snappy. Especially interesting to him, in view of Coromandel's growing consultancy business, is the tool support for building OCX hosts. Just as controls are specialized OLE in-process servers, hosts are specialized OLE containers—thanks, again, to new MFC abstractions. The next version of Visual Basic will be one such host, but the idea is that any application should, with minimal effort, be able to host OLE controls. "We're defining frameworks for specific

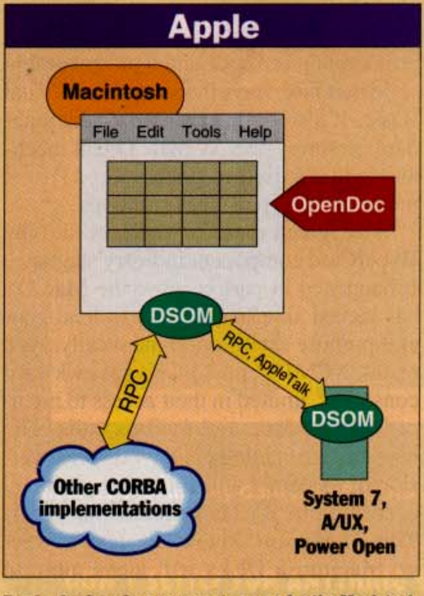




Novell's AppWare strategy encompasses Windows-, Mac-, and Motif OpenDoc-based components. ALMs running on multiple operating systems will communicate across multiple network substrates. Novell is also considering DSOM and COM support.



IBM will support visual and interactive Windows- and Presentation Manager–based components using OpenDoc. The real payoff, though, will come from DSOM's ability to integrate these components into large-scale distributed systems.



For Apple, OpenDoc represents a way for the Macintosh to duplicate—and, it is hoped, improve upon—the kinds of document- and component-based applications that have flowered on Windows, thanks to OLE.

industries, such as retail and finance," Swamy says, "and the ability to plug OLE controls into our own applications will be very important to us."

Doesn't all this dependence on MFC and OLE extensions tie developers to the Microsoft tool set? In principle, no. "Everything works in terms of OLE 2.0," says Eric Lang, Microsoft program manager for OLE custom controls, "so anyone who understands those interfaces can reproduce what we've done." In practice, however, that will be very difficult.

In principle, OLE controls, unlike VBXes, can be extended—not by inheritance, which COM doesn't support, but by aggregation of interface pointers. In practice, that, too, will be difficult. Most developers say that, until Microsoft evolves tools that simplify the mechanics of aggregation, they aren't willing to wrestle with it. Note, however, that the audience for whom OCXes are intended won't necessarily find multiple inheritance any more congenial than inheritance. The VBX model succeeded precisely because it hid this level of complexity from corporate developers.

"The power of NextStep is that I always use the same messaging model, I always build objects and systems the same way, and I never have to shift paradigms."

Alex Cone, president, Objective Technologies

Alternative Approaches

Years ago, Visual Basic's spiritual ancestor, HyperCard, popularized the visual-prototyping, script-oriented programming style that now fuels the success of Visual Basic. It also defined a component standard of sorts—the XCMD/XFCN mechanism that's used to surface C or Pascal modules as HyperCard primitives.

Why, given these ingredients, did the HyperCard component industry stagnate? It happened in part because the Mac OS has lacked an effective way to load general-purpose extensions dynamically. As a result, XCMDs and XFCNs are awkward constructs, limited in their access to memory, global data, and toolbox calls. The new DLL technology in the PowerPC version of System 7 will solve that problem, according to BYTE's Macintosh expert Tom Thompson. At this late date, however, Macintosh DLLs will mean more to the emerging OpenDoc standard than the aging XCMD standard.

More recently, Parts from Digitalk (Santa Ana, CA) has stirred up many of the same ingredients—visual programming, scripting, and components—to create an attractive Smalltalk-based development kit for Windows, NT, and OS/2. Like Prograph CPX from Prograph International (Halifax, Nova Scotia, Canada) and Novell's Visual AppBuilder, Parts pushes visual programming to the limit. In these systems you program, at the highest level, by creating diagrams. You drag icons representing components, functions, and syntactic constructs onto the surface of a form and connect them with links. To encapsulate a complex part of a diagram—thereby creating a component—you draw a boundary that hides what's inside, exposing a few inputs and outputs.

In Digitalk's system, the simplest parts to create are those that encapsulate code expressed entirely in the diagrammatic language. More advanced parts wrap objects written in the system's native language, Smalltalk, or wrap DLLs that employ C, COBOL, CICS, SQL, or other languages. "Many of the most critical components come from technology specialists who cannot necessarily write VBXes in C," notes Mike Arrigo, vice president of marketing for Digitalk. Fortune 1000 companies with hundreds of millions of dollars invested in legacy systems can benefit enormously from the ability to componentize those systems so that, for example, a host-based CICS transaction becomes just another part that can be assembled using the Parts workbench.

Darrell Deming, manager of brokerage systems for USAA (San Antonio, TX), finds this approach highly productive. "We've built a sophisticated discount brokerage application, with a client/server connect, and we've stayed true to the paradigm of Parts as an object assembler," he says. What hasn't yet evolved is any significant exchange of components between the brokerage group and other units within USAA. It's feasible, using DLLs as the medium of exchange, but so far it hasn't happened. "Component distribution has to be driven by business needs," says Deming, "and because our specialized parts aren't useful to other groups, we're not being driven to distribute them."

While Parts today is a proprietary component model, Digitalk plans to support OCX controls and has already demonstrated that Parts can work with DSOM, the exchange standard that underlies OpenDoc. Another proprietary component model that's portable across Windows, NT, the Mac, Unix, and OS/2 is available today from XVT Software (Boulder, CO). XVT's thin, operating-system-neutral GUI layer is the linchpin of one of the premier cross-platform development toolkits.

Last year, XVT Software extended the kit with XVT-PowerObjects, which are modeled on VBXes but are portable across all XVT-supported platforms. The first set of components included toolbars, toggle buttons, status windows, and table and spreadsheet widgets. XVT Software, its partners, and its customers are busily extending the PowerObjects catalog. Because the company has always preferred to use native services where available, it's eyeing the emerging component-exchange standards with great interest.

Despite the incessant "Windows everywhere" mantra, Roger Oberg, XVT Software's marketing vice president, reports no slackening of Mac sales, and particularly strong demand on the Motif side. "If Microsoft's portable object technology doesn't address Motif," he says, "then it won't meet the XVT need." OpenDoc's multivendor heritage and CORBA-compliant DSOM foundation appeal to the company, however.

Rediscovering NextStep

One of the great strengths of the VBX approach is that it boils down a lot of traditional programming tasks to simple design-time editing. NextStep's Interface Builder was doing that—and in a more sophisticated way—long before VBXes ever existed. Objects that appear on the Interface Builder palette are true first-class citizens in the NextStep environment. Next's ObjectWare catalog lists dozens of these components—some for general-purpose use, others specialized for the financial-services realm, where NextStep has established a strong beachhead.

Next developers say that it's straightforward to palettize an object for use with Interface Builder. The job does require that you extend the system's generic inspector to create the specialized one used to display and edit the object's state, but even that task will be streamlined in the forthcoming version 3.3 of NextStep. The ability to drag links between the outputs of one object and the inputs of another comes essentially for free. Clearly, components arise more naturally from the normal NextStep development process than VBXes do from routine Windows development.

Moreover, says Dirk Fromhein, president of Watershed Technologies (Marlborough, MA), objects that use the NXConnection class get networking for free. That means that Watershed's GraphRight, a charting component, is able to serve both local and remote clients. "The NXConnection object traverses the whole netinfo domain automatically," says Fromhein, "so the client literally does not know whether the service is being provided locally or remotely."

Despite NextStep's undisputed virtues and its much-heralded port to Intel hardware, it continues to struggle for mind and market share. It's too early to know whether SunSoft's licensing of Next's application framework and tool set will turn the tide, but the once-unthinkable alliance between former rivals is a resounding affirmation of the value of Next's technology. "I used to have trouble getting IS people on Wall Street or in health care to consider a Next-based solution," says Objective Technologies' Cone, "but now the Sun deal has validated the whole concept."

SunSoft has intentions of grafting the NextStep (or rather, the *OpenStep*) application framework and tools onto its own

Object Wars

C++ created a peculiar sort of doublethink in the software industry. The object-oriented features of C++ presumably explain why it is slowly but surely eclipsing C. Yet it clearly has not delivered on one of the major promises of OOP (object-oriented programming): software reuse.

In terms of packaging and distributing binary modules, C++ arguably represents a huge step backward. Statically linked C libraries have always been an effective way to exchange reusable code. Dynamically linked C libraries worked even better. Freed from the burden of static linking, DLL-based operating systems such as Windows and OS/2 became collections of field-upgradable parts.

When C++ class libraries change, however, their clients typically have to recompile to accommodate them. Tom Pennello, vice president of MetaWare (Santa Cruz, CA), says that operating-system developers working in C++ are hard-pressed to explain to their managers why they can't release object-oriented libraries. The reaction, Pennello says, is invariably something like, "What do you mean? We've been doing this for years with procedure libraries!"

There is also the vexing problem of link compatibility across compilers. "This is where the type safety of C++ comes back to bite you," says Tom Keffer, president of Rogue Wave Software (Corvallis, OR), a leading vendor of C++ class libraries.

Although we reflexively equate OOP with reuse, Jim Bonine, former vice president of engineering and now a consultant to StepStone (Sandy Hook, CT), says that C++ was never even meant to solve the problem of large-scale component exchange. "When we pressed Bjarne on that point," says Bonine, recalling a 1987 debate between AT&T's Bjarne Stroustrup (inventor of C++) and StepStone's Brad Cox (inventor of Objective-C), "he admitted that the appropriate scope for reuse of C++ modules was probably [at] the project or department level."

StepStone's Objective-C was one response to this problem; IBM's SOM (System Object Model) is another. Each of these technologies employs a runtime engine to enable objects to bind dynamically while preserving the flow of inheritance across object boundaries. "With SOM, you can add virtual functions, or even refactor the class hierarchy," says MetaWare's Pennello.

The DirectToSOM feature Pennello is adding to MetaWare's High C/C++ parallels the compiler's internal object model with IBM's SOM, solving the problems of component exchange and link compatibility. The benefits of this tactic, Pennello says, more than repay its cost in performance. When IBM's C++ compiler adds the same feature, says Cliff Reeves, IBM's director of object-technology products, "you'll see the first [direct] binary exchanges between different C++ compilers."

StepStone's Bonine thinks that SOM makes sense for harnessing components written in multiple languages, but he questions the notion of retrofitting C++ in this way. "It's unclear how much more baggage C++ can take," he says. Mike Potel, vice president for technology development at Taligent (Cupertino, CA), says the DirectToSOM compiler won't initially handle the full complexity of C++. Taligent has built extensions to its C++ compilers to get the dynamic binding capability that is needed for its forthcoming object-oriented operating system.

Not surprisingly, the harshest criticisms of IBM's SOM and DSOM (Distributed System Object Model) come from the Microsoft camp. "We take seriously the idea that interfaces are signatures, separate from implementations," says Microsoft's Mark Ryland. "None of the CORBA [Common Object Request Broker Architecture] schemes, including SOM, face up to what it really means to have millions of binary objects out there."

SOM will break, he says, in cases where vendors supply competing implementations of the same interface—implementations that are at first equivalent but diverge over time. Microsoft's COM (Common Object Model), he argues, avoids such problems by spawning interfaces: A single object can simultaneously express multiple versions and varying sets of capabilities.

CILabs' (Component Integration Laboratories) executive manager Jed Harris responds heatedly. "It's a broken example," he says. "The two implementations would no longer be valid subtypes, and that's just a bug that you can detect mechanically." Neither SOM nor CORBA requires a singly rooted inheritance tree, he adds. Clients can use mixed-in multiple inheritance to select from a smorgasbord of components.

While that's possible with SOM, says Mark Bramhall, DEC's technical director for distributed computing services, it's less efficient than with COM. (DEC has licensed COM so that its own component toolkit, ObjectBroker, will be able to act as a gateway between OLE and the CORBA technologies.) "In the distributed case, with DSOM, thousands of remote objects means thousands of proxies," he says. "With COM, on the other hand, you can quantize these into a smaller set of interfaces so that things scale nonlinearly. You can get away with just tens or hundreds of proxies and avoid exploding the type environment."

Although the debate rages on, with no end in sight, there is a subtext of tacit consensus: Components are crucial; C++ alone can't deliver them; and new mechanisms need to evolve. The ferment is a sign of healthy growth.

CORBA-compliant distributed-object-plumbing layer, DOE. SunSoft's Probst likens this to IBM's plan to layer the Taligent application framework on top of the CORBA-compliant DSOM. A wrapper of CORBA IDL (Interface Definition Language) around Next components, says Probst, will enable them to plug into the same sockets that accept C++, Smalltalk, or Ada components.

Won't that destroy the seamlessness of pure Objective-C development that Next programmers so highly prize? Not necessarily. It's true that Objective-C objects can't converse intimately with foreign objects. But a library that internally exploits all the power of Objective-C can export multiple interfaces and so appear, to clients, as a collection of independent components.

The OpenDoc Alternative

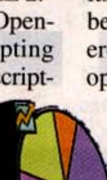
OpenDoc is the cross-platform compound-document standard that will be licensed by CILabs (Component Integration Laboratories), with the backing of Apple, IBM, Novell, WordPerfect, and others. OpenDoc's charter, like that of OLE 2.0, goes beyond compound documents; it defines a full-blown component architecture (see "A Close-Up of OpenDoc," March BYTE). OpenDoc parts, like OCXes and OLE servers, can load dynamically, embed themselves in containers, and respond to commands issued from a variety of languages.

Four foundation technologies underly OpenDoc—a compound-document framework for OLE-like embeddings, a compound file format (Apple's Bento), a language-neutral automation architecture (modeled on Apple Events), and a language-neutral run-time mechanism for dynamic object linking and binary component exchange (IBM's DSOM). (See "IBM's Assault on Distributed Objects," November 1993 BYTE.)

CILabs will license the source code for all four technologies to interested parties. "There aren't any secrets," says David Austin, Apple's manager of OpenDoc development. The first developer's releases of OpenDoc for at least three platforms should start appearing around the time you read this, from Apple (for the Mac), WordPerfect (Windows), and IBM (OS/2).

CILabs claims that OpenDoc will have a number of advantages over OLE. In the realm of compound documents, these include support for nonrectangular content and multiple active objects. "OpenDoc's screen-brokering technology is much better than OLE's," says Doug Donzelli, vice president for AppWare foundation technology in Novell's AppWare Systems Group. "ClarisWorks, which internally uses a highly sophisticated component integration scheme, was one of the benchmarks for the OpenDoc designers; you could not build ClarisWorks with OLE 2."

As a general component model, OpenDoc's strengths flow from its scripting technology and DSOM. To support scripting, OpenDoc will support the registration of standard protocols, or *event suites*, for major classes of applications. The event "advance to next word," for example, will mean the same thing in any word processing application. OpenDoc proponents argue that this discipline, like the Apple Events model, ensures at least some level of script reusability across applications and components, whereas OLE's approach guarantees none.



"If Microsoft continues to ignore the non-PC Motif platforms, then OpenDoc will have the edge."

Kimbo Mundy, Interleaf's chief architect

Microsoft's response? "We wanted to standardize on suites of OLE automation verbs," says Mark Ryland, senior program manager on the Cairo project, "but the major independent software vendors couldn't come to a consensus. Do you leave the cursor at the beginning of the next word? The end? What about punctuation? We would have had to mandate these things like Apple does, and we chose not to."

IBM's DSOM, say the CILabs backers, will endow all OpenDoc platforms with a network-capable, language-neutral mechanism for packaging and distributing components. The CORBA-compliant interface-definition language used to describe their interfaces means that users of components can extend them—without access to source code—using multiple inheritance. Microsoft disputes these claims, and the COM-versus-DSOM debate has lately turned into a pitched battle (see the text box "Object Wars" and "Extensible Software Systems" on page 57). Microsoft's Ryland argues that COM's aggregation, unlike DSOM's inheritance, cleanly separates interfaces from implementations.

A related argument is that while inheritance is useful—perhaps even essential as a private discipline for builders of components—it's inappropriate as a public discipline for users of components. OpenDoc proponents vehemently disagree. "Obviously, a well-encapsulated object has value," says Cliff Reeves, IBM's director of object-technology products. "But at what point does it stop being something for which inheritance is useful?"

Jed Harris, executive manager of CILabs, argues that Microsoft's approach forces the programmer to predetermine the boundary between a component and its environment. But that boundary can't be known in advance; it must be discovered during iterative, exploratory development. "You can never get it right the first time," says Harris, "and that's why you don't want two different programming models."

The uses of OpenDoc are as varied as the companies backing it. Apple, focused on the desktop, needs to enable the Mac to duplicate—and hopefully improve upon—the kinds of document- and component-based applications that have flowered on Windows, thanks to OLE. IBM, focused on the enterprise, wants to build complex, heterogeneous, distributed systems using standard interchangeable parts. WordPerfect sees OpenDoc as a platform-neutral way to decompose a monolithic application into pieces that can be specialized for particular markets and to enable that application to accept pluggable third-party extensions.

So far, developers have mixed reactions to OpenDoc. "I'm focusing on OLE 2," says Ray Côté, president of Appropriate Solutions (Antrim, NH). "It's the holy grail of reusable code, and it will be mature on the Mac and Windows by the time OpenDoc arrives." Interleaf's (Waltham, MA) chief architect Kimbo Mundy says, "If OpenDoc does a few more things than OLE 2, then, frankly, I don't care; I just want one interface that will keep me competitive on multiple platforms." But Mundy cautions that "if Microsoft continues to ignore the non-PC Motif platforms, then OpenDoc will have the edge."

Acceptance of OpenDoc will certainly depend, in part, on how effectively it can interoperate with OLE. Can two such complex standards really play together? "After months of analysis, we're convinced it will work," says Novell's Donzelli. "Otherwise we wouldn't have backed OpenDoc."

Novell's AppWare Bus

Another toolkit for the construction of portable components, due out by the time you read this, is Novell's AppWare. Version 1.0 will include about 70 bundled components, or ALMs (AppWare loadable modules), and will support development of ALMs—for Windows and the Macintosh—in C and BASIC.

Novell's AppWare Systems Group will

unite technology from two acquisitions, Serius and Software Transformation. Serius provided the pictorial programming environment, which is now called Visual AppBuilder, and the AppWare Bus, which defines how ALMs plug into and communicate with hosts. Software Transformation supplied cross-platform foundation classes that will make existing Windows and Mac components more robust, and it will also extend AppWare's reach to OS/2 and Unix platforms.

Central to the AppWare foundation is the notion of scalable families of components. AppWare's text widget, for example, comes in several API-compatible versions, ranging from a lightweight multiline edit control to a near-full-function word processor. This foundation won't be part of the initial AppWare products; Novell plans a developer's release of foundation-based versions of the AppWare Bus and Visual App Builder by the end of 1994, with final versions due in 1995.

Concurrently, Borland is working to graft its OWL (Object Windows Library) framework onto the AppWare foundation, transforming OWL into a cross-platform API and making ALMs an easy target for Borland C++ developers. Eventually, claims Novell's Donzelli, "you'll be able to write an OLE part or an OpenDoc part or an ALM from a single source. In fact, OWL programmers are doing this today, although they don't realize it."

While Visual AppBuilder's pictorial approach to programming will likely receive the lion's share of attention at first, Novell says the product's main purpose is to expose the AppWare Bus and jump-start the ALM binary standard. "We're giving away the bus—the tool interface, the runtime event engine, the messaging system," says Joe Firmage, vice president for AppWare Bus technology in Novell's AppWare Systems Group. Early adopters include Gupta, which has announced that a future version of its SQLWindows will be able to accept plug-in ALMs.

Why might developers prefer AppWare over OLE or OpenDoc? These technologies are tuned for the desktop—for visual, interactive tasks, Firmage argues—whereas AppWare's inherently asynchronous approach favors distributed, communications-intensive applications. But version 1.0 does not let you distribute an application based on ALM components. The next version, due around September, will provide two mechanisms—PeerLogic's Pipes, and one of the CORBA-compliant technologies, possibly DSOM.

"How will component vendors compete when a few hundred dollars buys you a whole application—or even a suite—that's also an integrated development environment with most of the objects you want and very few missing pieces?"

Jeffrey Tarter, publisher of Softletter

Specialized Applications

Mainstream applications can profit from component technology, notes Mark Ericson, object architect for WordPerfect (Provo, UT), who thinks that internal use of OpenDoc parts will enable his company's word processor to handle new kinds of content and thus appeal to specialty markets. "WordPerfect has a general equation editor," he says, "but scientists or engineers may require specialized equation editors."

In an era of shrinking margins, the ability to create and manage premium products could become critical to vendors of what has truly become commodity software. The same OpenDoc technology used internally to specialize WordPerfect for particular markets, Ericson adds, will give WordPerfect users access to third-party components. That means the company won't have to invent, maintain, and evangelize a proprietary extension mechanism.

But Microsoft desktop marketing product manager Mike Risse doesn't yet see a need to differentiate Excel by varying its core components. "We give you the object set and the tools to customize Excel for a medical office or an electrical-engineering firm," he says. The next version will be even more customizable, he adds, because you'll be able to export user-written Visual Basic for Applications functions to OLE automation controllers.

However, in Microsoft's development labs, experiments are validating an OLE building-block approach to applications. "We've been playing with a word processor built out of components," says OLE architect Tony Williams, "and it's blindingly fast." What's more, the user interface of Cairo, the Windows NT successor due in 1995, is literally a collection of user-customizable OLE components.

Applications vs. Components

Today's applications not only are in competition with the new pluggable components but also are growing increasingly component-like themselves. That's especially true on the Macintosh, where Apple Events are now widely exploited.

Symantec C++, for example, is actually a collection of independent components that talk to each other by means of Apple Events. When the debugger needs to evaluate an expression, it pipes it to the compiler. As a result, the debugger is able to handle very complex expressions, and it automatically benefits from compiler upgrades. Now that the Think Class Library encapsulates the Apple Events APIs, it's easier than ever before for users of Symantec C++ to achieve the same effect in their own applications.

Excel 5.0, Windows' bellwether application, exposes dozens of objects—and hundreds of methods and properties—to programs written in its own internal scripting language or in Visual Basic 3.0. This OLE automation capability, coupled with Excel's OLE embeddability, lets Visual Basic programmers use the application as though it were a high-powered custom control for charting or data analysis.

The increasing programmability of mainstream applications raises some interesting questions. "How will component vendors compete," asks Jeffrey Tarter, publisher of *Softletter* (Watertown, MA), "when a few hundred dollars buys you a whole application—or even a suite—that's also an integrated development environment with most of the objects you want and very few missing pieces?"

Peter Mullen, development manager for Shapeware (Seattle, WA), shares that concern. Shapeware's Visio, an intelligent business graphics application, was one of the first implementers of OLE automation. Visio Express, the first pure OLE 2.0 server, exists only to embed the graphics within OLE 2.0 containers.

Will Shapeware also cast its technology in the OCX mold? Mullen's not sure. "Resellers love Express because it's a secondary sell along with an application like Word," he says. "But where's the mass market for an OCX?"

While the VBX example proves that component vendors can find comfortable niches, these questions are extremely pertinent. Nearly everyone agrees that the issues of cost, distribution, and support will have to be worked out before a software-components market can really thrive. The technical foundations are being laid, but the business model is still up in the air. ■

Jon Udell is a BYTE senior technical editor at large. You can reach him on the Internet or BIX at judell@bix.com.