Patricia Seybold Group

Editor-in-Chief
Michael A. Goulde

INSIDE

OPEN INFORMATION SYSTEMS

Guide to Unix and Other Open Systems

Vol. 8, No. 11 • ISSN: 0890-4685 • November 1993

Novell's Unix Strategy

NetWare and Unix Converge

By Michael A. Goulde

IN BRIEF: Novell's acquisition of Unix System Laboratories from AT&T affected more than just the two companies' balance sheets. Thousands of computer industry professionals whose careers had been built on the foundation provided by Unix were affected. Having worked with Unix internals in college, they went on to build products and indeed entire companies on Unix. With Novell's acquisition, the relationship between Unix and the industry that has been built on it faces the potential of dramatic change. How drastic that change is, or isn't, depends to a large extent on where Unix fits in to Novell's strategic plans for Unix. Unix, in the form of Novell's UnixWare product, will function primarily as an application server, allowing Novell to address much larger scale client-server applications than it can with NetWare. *Report begins on page 3.*

Migrating to Open Systems

Our Story - Part I: Getting On the Net

BY THE TIME you read this editorial, our internal network at the Patricia Seybold Group will have become another of the thousands of networks that make up that symbol of open systems, the worldwide Internet. Most of our users will be able to connect to systems throughout the world from their desktops and retrieve files, read news, and search databases about thousands of topics. Connecting to the Internet is not difficult. Orienting your IT infrastructure to the Internet when it is not entirely Unix-based is harder. Harder still is figuring out how to transition from where your environment is today to where you want it to be tomorrow with the least amount of risk and disruption.

Our migration began over two years ago. We had built a prototype client/server application using MS Windows and Macintosh clients, a VAX/VMS server, Digital Corporation's Rdb DBMS, Technosis SQLink, Blythe Omnis7, and DECnet and Appletalk network protocols. Lotus Notes was a part of the picture as well, since Notes Mail and access to Notes is an important part of the way we work day-to-day.

The VAX ran Digital's Pathworks, providing common file and print services to Macs and PCs. Digital's X.400 mailbus products provided the mail transport to PCs and Macs and a gateway to MCI Mail.

On paper, everything looked sound. However, the client/server application didn't work. The fatal problem was that response times were measured in minutes rather than seconds. (If you want to know the details, they are contained in our publication *Paradigm Shift*, June 1992; a reprint is $40.) The project was overly ambitious and didn't take into account constraints on network bandwidth, server performance, and client capabilities.

So what next? We considered many options, but they all seemed to be constrained by our VAX and our DECnet. The VAX was difficult and time-consuming to manage, and many applications we were considering did not support DECnet as their transport. We decided that we had to move from DECnet to TCP/IP. While we were at it, we also decided to move from VMS/Pathworks to something a mortal could administer. Microsoft was willing to loan us LAN Manager for OS/2, which was easy enough to manage since Notes was already running on OS/2. LAN Manager came with TCP/IP, so we didn't have to buy it from a third party. It seemed like an easy transition. This approach also gave us the alternative of running Pathworks and LAN Manager side-by-side as a transition, since the former is based on the latter. We even had the option of running LAN Manager on Unix if we desired.

We could have run TCP/IP as well as DECnet on the VAX as a transition step, but we decided that configuring VMS/TCP/IP and configuring Pathworks to run with DECnet and TCP/IP would require too much investment in time and that the steep learning curve wasn't warranted. So we ran DECnet, TCP/IP, and AppleTalk simultaneously on the LAN Manager and Notes servers for a while. This allowed us to test out the configurations while still using the VAX for file and print services. The first service that was switched off the VAX was the MCI Mail gateway. We installed cc:Mail gateways to MCI Mail and to Notes on a LAN Manager/TCP/IP client.

When Notes Version 3 arrived, we tested its ability to run on TCP/IP. Then we switched the PCs to TCP/IP and turned off the VAX. Now we only run TCP/IP and AppleTalk. We selected our Internet Access Provider, installed our router, and are trying a number of TCP utilities for Windows. Now, if the Macintosh version of Notes would run over TCP, then we could switch the Macs to MacTCP, and everyone could have access. If only OS/2 wouldn't crash so often.

OPEN INFORMATION SYSTEMS

Editor-in-Chief
Michael A. Goulde
MCI: MGoulde
Internet:
mgoulde@mcimail.com

Senior Editor
Wayne Eckerson
MCI: WEckerson
Internet:
weckerson@mcimail.com

Publisher
PATRICIA B. SEYBOLD
Analysts and Editors
JUDITH R. DAVIS
STANLEY H. DOLBERG
MITCHELL I. KRAMER
DAVID S. MARSHAK
RONNI T. MARSHAK
JOHN R. RYMER
ANDREW D. WOLFE, JR.
Copy Editors
ALBERT C. D'AMATO
MIRIAM F. D'AMATO
Art Director
LAURINDA P. O'CONNOR
Sales Director
PHYLLIS GUILIANO
Circulation Manager
DEBORAH A. HAY
Customer Relations Manager
DONALD K. BAILLARGEON

Patricia Seybold Group
148 State Street, 7th Floor,
Boston, Massachusetts 02109
Telephone: (617) 742-5200 or
(800) 826-2424
Fax: (617) 742-1028
MCI: PSOCG
Internet: psocg@mcimail.com
TELEX: 6503122583

Open Information Systems (ISSN 0890-4685) is published monthly for $495 (US), $507 (Canada), and $519 (Foreign) per year by Patricia Seybold Group, 148 State Street, 7th Floor, Boston, MA 02109. Second-class postage permit at Boston, MA and additional mailing offices.

POSTMASTER: Send address changes to *Open Information Systems*, 148 State Street, 7th Floor, Boston, MA 02109.

Novell's Unix Strategy

NetWare and Unix Converge

Novell Corporation's dominance in PC networks and the continuing industry trend of migration from mainframe architectures to distributed networks places the Provo, Utah company in an enviable position. However, Novell found that it faced serious challenges in trying to fulfill the requirements of the new computing models, and so it acquired Unix Systems Laboratories (USL) to help bolster its position. (See *Open Information Systems,* Vol. 9, No. 1, January 1993, for more information on the USL acquisition.) Novell's strategy for Unix is broader than just offering it alongside NetWare. Unix represents an important part of Novell's long-range future, particularly as an enterprise supplier.

NetWare Everywhere

Some basic facts have to be considered when we look at Novell's strategy. Novell virtually owns today's market for PC LAN operating systems, with over 10 million users worldwide. Depending on which market research firm's numbers you look at and how you define the market, somewhere between 60 and 70 percent of PC desktop computers that are networked run NetWare client software. Virtually every system vendor in the world sells NetWare in some form. For that matter, NetWare can be purchased from just about anyone selling software. Broad distribution has been one of Novell's strengths. In fact, Novell's greatest asset may be the collection of more than 22,000 NetWare resellers worldwide that have collectively become known simply as "The Channel." Novell has similar plans for broad-scale distribution of Unix, in the form of UnixWare, but it is not certain that Novell's successful distribution strategy can extend to Unix. (See *Open Information Systems,* Vol. 8, No. 5, May 1993, for more information on UnixWare.)

Why Buy Unix?

When Novell acquired Unix System Laboratories from AT&T, it gained ownership of many things, including source code, original equipment manufacturer (OEM) products, licenses, a trademark, and a position as the focal point of the Unix industry, one of the fastest growing industry segments in the world. Although rumors abounded around the time of the acquisition that Novell was acting on behalf of other vendors in buying USL, the fact is that Novell exchanged a significant amount of assets to consummate the acquisition. Valued at approximately $350 million, USL must have been slated for a significant role in Novell's long-range strategy. In fact, USL has been rolled into a Unix System Group (USG), which now has responsibility for Unix, UnixWare, NetWare for Unix, NetWare NFS, and the LAN WorkPlace products. What Novell purchased was not USL, but technology and a position in the open systems market. The mission for USG is to accelerate the growth of the Unix market by enriching and integrating the UnixWare and NetWare environments. The general strategy for accomplishing this includes:

- Establishing NetWare and UnixWare as the leading rightsizing solution
- Driving unification of the Unix industry by working with key system providers, software developers, and industry groups to agree on common Unix system interfaces and technologies
- Driving UnixWare as the volume standard in the Intel marketplace

Novell's Business Model

Essential to understanding Novell's direction for Unix is understanding Novell's business model. The company is in the distributed computing business, selling software that provides network services required by applications and PCs running in a distributed environment, including security, file, print, backup, management, and global directory services. Novell is

not in the hardware business, and it is not in the application software business. It sells its products through multiple channels, and, in fact, broad distribution has been a central component of its strategy. Novell has made NetWare synonymous with PC LANs by delivering shrinkwrapped software that runs on high-volume commodity hardware. (This was not always the case. Until NetWare 386 appeared in 1987, Novell also sold a line of proprietary servers. It quickly got out of that business.) Novell has also leveraged its investments to the hilt.

Evolving toward the Enterprise

The next stage for Novell's evolution is to extend its influence from the LAN to the enterprise. However, its current products do not support applications at the enterprise level. Tools for developing enterprise-class distributed applications are lacking as well. Because Novell's goal is to become a dominant force in enterprise-class distributed computing, Unix has come to play a key role in Novell's strategy. Unix offers a new technology and a new business model that will help Novell work toward its goal of dominating enterprise distributed computing just as it has dominated LAN-based computing.

Drivers for Novell's Strategy

Although Novell stands in a market-leading position in PC LANs, the industry is going through an important transition that, we believe, drove Novell almost unwillingly into the Unix camp. The significant factors include:

- PC LANs are no longer seen as isolated, resource-sharing conveniences but have become a major part of enterprise internetworks. NetWare requires an array of bridges and gateways to integrate with corporate networks.
- The concept of the Network Operating System (NOS) is fading in importance as distributed capabilities increasingly become part of the operating system itself. Network File System (NFS), as an example, is available from a wide variety of suppliers for virtually every operating system sold, and all of them interoperate across the network.
- File- and print-sharing are receding as important functions of the network, superseded by strategic client/server applications, E-mail, and other distributed application models that are network based.
- Proprietary protocols, like IPX/SPX, which enabled NetWare to deliver extremely fast performance for file-sharing, are less tolerated, particularly by large customers. Standards-based networking is mandated.
- LAN servers must meet increasingly higher standards for robustness, including reliability, availability, and serviceability. All three are notorious weaknesses of NetWare.
- PC LANs must be manageable as a part of the enterprise internetwork. Novell's Distributed Management Services is limited in scope.

In addition to these general trends, competitive forces were increasingly affecting Novell:

- Microsoft's Windows NT, particularly the Advanced Server, is posing a threat from every direction, including distribution, functionality, and, particularly, pricing.
- SunSoft's (Mountain View, California) Solaris for Intel, while not a major threat in the channel, certainly created a diversion by delivering a stable application platform in addition to distributed file, print, and mail services.

Drivers for Novell's Strategy

- IBM and Hewlett-Packard Company were growing increasingly cooperative in licensing one another's technologies and also collaborating with Sun to move the Unix industry forward from its stalemate of standards disputes.

The combination of industry trends and competitive pressures made it clear to Novell that simply pursuing a course of continued NetWare evolution was not going to be sufficient to survive the changes in the industry.

Focal Points for Action

Server as Application Platform

As file and print services recede in strategic importance and distributed application models, including client/server, begin to predominate, the server takes on a different role from that fulfilled by the traditional NetWare server. In distributed applications, the server supplies the platform in which the data management components of distributed applications run. Unix has been important in this context because of its strength as an application platform and its inherent networking capabilities. Novell estimates that nearly two-thirds of applications servers being shipped run Unix. NetWare, on the other hand, has accommodated third-party applications almost as an afterthought. In Version 2.x, third-party applications running on NetWare servers were supported in the form of value-added processes (VAPs), but relatively few were actually ever delivered. When NetWare 3.x came out a few years ago, VAPs became NetWare Loadable Modules (NLMs), and more support was garnered. However, few high-level tools have been produced to support NLM development, making NLM programming an arduous task, particularly for the corporate developer.

In addition, because NetWare is not a protected operating system, NLMs can crash into one another and the operating system, requiring a qualification process that would not be necessary in a protected environment. Although several key applications are available or will soon be available for NetWare—including Oracle7, Sybase SQL Server, and Lotus Notes—when system-level utilities are eliminated, the number is very small.

Novell needs to own a strong application platform, not to bolster just its technical capabilities but its market position as an enterprise platform supplier as well. The company needs a platform with a large number of applications and a large installed base in order to continue to attract development from third parties. Novell needs this platform to be able to play in the new game of distributed applications and to compete effectively against Microsoft Corporation (Redmond, Washington) and others.

Development Tools and Application Frameworks

As Novell has learned with NLMs, simply providing C libraries for developers to use in developing applications is not sufficient to garner the kind of support necessary in today's market. If it is to provide a strong application platform, the company must provide and/or attract many high-level tools for both commercial and professional developers. Those tools, including application frameworks, must address the critical issue of programmer productivity and rapid application development. At the same time, Novell must provide the infrastructure necessary to support those applications. That infrastructure includes so-called middleware, including protocols, transports, messaging services, object request brokers, and the like.

Client Capabilities

As the desktop goes, so goes the distributed computing industry. Or so it would seem. If Microsoft owns the desktop with Windows, many believe that Microsoft will also own the direction of distributed computing. One of the reasons the Unix vendors continue to assault the desktop through their efforts with the Common Desktop Environment (CDE) is to ensure that their model of open distributed computing will be considered a viable alternative to Microsoft's. Without a desktop presence, the Unix vendors would not be taken seriously.

Similarly, Novell must be able to play a role on the desktop. Its acquisition of Digital Research Incorporated gave it a desktop operating system, but DR DOS is essentially a clone

competing in a commodity marketplace. There aren't any significant points of differentiation for DR DOS that Novell can sell to the marketplace. This battle for the desktop has been seen recently in the tug-of-war between Novell and Microsoft over NetWare client support for Windows NT. But this is a battle that Novell probably has to let Microsoft win to prevent NetWare servers from being excluded from Windows NT sites.

OEM Relationships

Novell has been building its OEM business gradually over the years. The licensing of Portable NetWare on Unix platforms was perhaps the zenith of that business. Those relationships were governed by the fact that Novell was licensing a proprietary technology that, unlike Unix, no other party could lay claim to — legitimate or otherwise. Novell, on the basis of its overwhelming market share, held all the cards. There was no other source for NetWare technology. Even with processor independent NetWare (PIN), soon to be released for many RISC architectures as well as for Intel Pentium, Novell controls the critical NetWare code. With Unix, that has changed, and Novell's relationships with its OEMs will change as well.

With the acquisition of USL, Novell acquired an impressive pantry of technologies that USL licensed or had plans to license. Tuxedo, Distributed Manager, and DCE for SVR4 all supplemented SVR4 technology. Novell needed to determine which of those technologies were strategic and which had become black holes into which it should stop throwing money.

Strategies for Success with Unix

Server as Application Platform

Novell sees customers having a continuing need for an optimized network services operating system in addition to an application services platform. NetWare provides the network services, and UnixWare is positioned as the application services platform. Novell will continue with this two-operating-system strategy for some time. It will converge management interfaces and programmatic interfaces long before the two platforms themselves ever converge. The company will concentrate on integration, making management and connectivity seamless and making it easy to use the combined platforms.

Convergence of NetWare and UnixWare. When Novell acquired USL, it acquired the work that was underway in conjunction with Chorus Systems (Saint-Quentin-en-Yvelines Cedex, France) to deliver a microkernel version of SVR4. (See *Open Information Systems,* Vol. 8, No. 8, August 1993, for more information.) Although it would seem logical for Novell to create a multi-personality, "serverized" operating system, with NetWare and Unix personalities sharing a common microkernel, that task is not as simple as it sounds. There are important issues over and above the technical challenge that will take time to address, including understanding what that environment has to look like, what characteristics it has to have, how the development should be funded and the product priced, how distribution will be affected, how OEM relationships might change, etc. Converging NetWare and UnixWare is not a change that can happen overnight.

On the technical side alone, the work that USL and Chorus were doing did not have as one of its goals removing Unix biases from the microkernel, which would have to be done in order to support an optimized NetWare Services server. NetWare customers will not be willing to sacrifice any of the functionality or performance to which they have been accustomed. Therefore, NetWare will continue to evolve on its own base, separate from that of UnixWare. Novell already believes that NetWare has many characteristics of a microkernel operating system, although future versions may share some kernel technology with UnixWare. In the meantime, the USG will proceed with its SVR4 microkernel schedule and probably work on a personality-neutral implementation as a follow-on. Novell's strategic direction is to converge APIs across the two platforms, providing consistency for users and developers. Those APIs will be available to them long before a converged platform server is. Theoretically, Novell

could license IBM's microkernel as its base, but that technology is too new to assess, let alone build the future of Novell on.

Therefore, Novell will continue talking about NetWare and UnixWare as a pair of complementary platforms, with NetWare providing network services and UnixWare serving as the application platform. From the perspective of the desktop client, these two platforms will become increasingly transparent, to the point where a developer will be writing to a single API supported on both platforms. In mid-1994, when UnixWare 2.0 rolls out, its multiprocessor (MP) support will provide Unix-class scalability, along with tighter integration with NetWare services for things like software licensing and distribution, backup and restore, and a common management console.

Competing against Windows NT. Novell views Windows NT Advanced Server (NT/AS) as a competitor primarily for UnixWare, not NetWare. It believes that the Microsoft product lacks the sophistication, performance, and functionality to seriously compete against NetWare servers. Novell acknowledges that, as an application platform, Windows NT/AS will go head-to-head with UnixWare. The company will rely on UnixWare's Unix heritage, its open systems standards support, and the SVR4-derived portfolio of applications to compete effectively against Windows NT/AS. Attracting broad application support is a key strategy for UnixWare to offset Microsoft's momentum.

Application Frameworks

It is important for Novell to provide mechanisms to help developers build distributed applications. The success of the company's role as a network services provider depends on the delivery of such applications. Today, distributed application development means developing for multiple platforms on multiple networks using many different tools, protocols, and APIs. Novell's strategy is to help developers move to various platforms and have their applications interact using network services provided by NetWare. The company is focusing on where applications are being built today and on providing the tools to build tailored applications that have access to network services, often provided by existing NLMs.

AppWare is Novell's new environment and toolset for building portable, distributed applications. (See *Open Information Systems,* Vol. 8, No. 9, September 1993, for more information on AppWare.) AppWare products are designed to provide portability of client-side application code on all of Novell's supported client platforms, including Windows, Macintosh, and Motif. AppWare is supposed to make it easy to create applications that use the wide variety of shared services available on NetWare LANs. Those services include directory, mail, image management, relational database, and document management. Its ultimate goal is to support completely portable clients across graphical user interfaces (GUIs), networks, transports, and distributed computing services.

The AppWare Foundation Layer supports portability across clients and servers. Novell has pledged to roll out platform-independent APIs for directory services, mail, compound document processing, licensing, and other Novell services. AppWare will be extended over time to increasingly support server applications, with an emphasis on UnixWare as the target for application servers. Novell's goal is to allow NetWare NLMs to be recompiled to run on UnixWare, and UnixWare applications be able to be recompiled to run on NetWare. The work that Novell has taken on to define a portable interface to a full range of services in a distributed environment is ambitious and is being done with several strategic partners, notably WordPerfect Corporation (Orem, Utah) and Borland International (Scotts Valley, California). This process puts Novell in the position of API arbitrator and makes it responsible for delivering a complete set of APIs. Contrary to the Unix heritage it inherited, Novell has not yet implemented a more open process to solicit input from the industry to ensure usability, independence, and complete functionality.

Client Capabilities

In spite of industry skepticism that Unix on the desktop will be an effective counter to Windows NT, UnixWare is Novell's strategic client platform. It is the embodiment of

Novell's work with the Common Open Software Environment (COSE) group and its Common Desktop Environment. The UnixWare client, slated for CDE compliance in its 2.0 release, is aimed at the high-volume, Intel commodity desktop. Novell is positioning UnixWare as the open alternative to Windows NT and OS/2 since it is a 32-bit, multitasking, protected operating system. Its next release will have MP support as well. Its XPG4 compliance, combined with Wabi and other compatibility technologies, makes UnixWare a good candidate for a Microsoft alternative. Of course, Microsoft has its Windows API as a strategic weapon, and Novell is countering with AppWare.

AppWare Bus. A significant client technology for Novell is the AppWare bus. Built around an event-oriented, rather than an object-oriented, model, it is a mechanism for managing events among an arbitrary collection of objects and functions in applications. The model doesn't support either class hierarchies or inheritance. It is different from the object model in Microsoft's Object Linking and Embedding (OLE) 2.0 and in the Object Management Group's Object Management Architecture (OMA).

The current version of the bus operates in a single memory space only. It does not yet handle delivery of messages across networks. Novell plans to add support for distribution to the AppWare bus in a future release. The AppWare bus is not an object request broker, nor does it comply with the OMG Common Object Request Broker Architecture. However, Novell does license HyperDesk Corporation's HD-DOMS and may use it to implement the distributed version of the AppWare Bus. Novell's intention is to evolve AppWare's object model in a direction that is entirely consistent with the OMG standards. That work will appear in a second phase of the AppWare products.

OEM Relationships

UnixWare Is Novell's Intel Shrinkwrap. In addition to continuing its business of licensing SVR4, Novell intends to sign up OEM customers for UnixWare source and binary distributions as well. For vendors of Intel-based systems, Novell will distribute shrinkwrapped UnixWare for uniprocessors, and, with Version 2.0, for many MP architectures. Although many OEMs will have no interest in UnixWare, seeing it as competing with their own distributed computing directions, others will find UnixWare useful in selling to the large Novell installed base. Unisys Corporation has already indicated that it will expand support for UnixWare across its Intel-based product line. Novell will work with OEMs on their Intel-based MP systems, including support for some in shrinkwrap, with OEMs directly providing support for others on their own.

In addition to the shrinkwrap business, Novell will also make a source code version of UnixWare available for customers (OEMs) who wish to port UnixWare to their systems. Novell is eager to have this happen and would not even require those OEMs to call their products UnixWare. Vendors with proprietary MP architectures on Intel base or who have systems using another chip architecture—ICL, for example—will use UnixWare on both their Intel and SPARC product lines. This is a good strategy for OEMs with mixed bases to bridge their platforms. ICL (Bracknell, England) can be expected to work out licensing deals for its Sparc port of UnixWare, either on its own or back through Novell.

As for Novell's branching out beyond Intel with a shrinkwrapped product, that will be driven by the commoditization of a particular platform. Novell's efforts will be strictly volume driven. If a system has adequate volume, either in sales or installed base, Novell will consider creating a product. Without the volume, platform support will have to come from other relationships. Sparc, for example, lacks the volume from Novell's perspective, but, if IBM and Apple Computer Incorporated succeed in creating a clone market for PowerPC-based systems, that architecture could become a candidate.

Overall, Unix spans a very broad range of systems and architectures, much broader than Microsoft can claim with Windows NT. With standardization efforts for the Unix API underway, that breadth will actually come to mean something once the various vendors come

into compliance with the Unix specification. (See *Open Information Systems,* Vol. 8, No. 9, September 1993, for more information on the Unix specification.) One way or another, Novell will seek to proliferate UnixWare on as many Unix platforms as possible.

Other OEM Products and Technologies. Novell intends to continue its OEM relationships for Unix SVR4 and the Tuxedo transaction processing monitor. It sees demand for SVR4 technology continuing, and Tuxedo has a strong potential growth, not just for Unix, but for NetWare as well.

Distributed Manager, on the other hand, will suffer a different fate. Essentially an implementation of Tivoli Systems Incorporated (Austin, Texas) frameworks for SVR4, Distributed Manager will be orphaned, and management of UnixWare will be subsumed under Novell Distributed Management. Most OEMs have their own arrangements with Tivoli, so few should be impacted by this change in direction.

Distributed Services, or DCE, for SVR4 will go on the back burner. Novell ranks DCE support behind Open Network Computing (ONC) support, which is also secondary to NetWare services. Should customer demand be strong enough to make Novell take notice, the company will carefully and deliberately roll out support for DCE in the areas where it is required. But it will not introduce the technology as an OEM product.

Leveraging Source Licensing. When Novell acquired USL, it acquired a source licensing business that will be important in the future. Source licensing has value both in the revenue it produces and in providing feedback for the products that Novell builds. The source licensing business also expands the volume with which the company's products are distributed, and, as we have seen, volume is king at Novell. Licensing will leverage work done for Novell distributed products, so the revenue will be incremental, but licensing is still a key part of Novell's UnixWare strategy.

Marketing and Distribution Issues

The first year of UnixWare distribution was fraught with start-up pains, and the joint USL-Novell venture, Univel, was less than successful. With its absorption of USL and Univel, Novell is paying closer attention to marketing and distribution issues. It is trying to deliver clearer messages about the relationship between NetWare and UnixWare in the form of its "Matched Pair" and "Yes It Runs" campaigns. The company is trying to show customers and its own channel members how UnixWare fits in to the overall strategy. Training of Certified NetWare Engineers (CNEs) has been ramped up, and UnixWare specialists are being trained. A new Unix Masters program is being created to recruit Unix-literate resellers to supplement the established NetWare channel. Novell wants UnixWare to succeed and is putting resources behind its marketing.

The newly established USG will build on the experience of all of Novell's Unix-literate organizations to build the channel for Novell's Unix products and ramp up volume. Apparently the company is beginning to see increased sales and some significant contract wins consisting of large-volume system sales to large customers.

How Good Is the Strategy?

Novell is faced with serious challenges to its preeminent position as supplier of network software. However, the bulk of those challenges is coming not from competitors but from changes in the way technology is being used. When laser printers, large disk drives, and tape backup subsystems were expensive and valuable resources, NetWare solved a critical customer problem. Today, however, the critical customer problem lies in the development and deployment of distributed applications across a variety of server and client platforms.

How Good Is the Strategy?

Is UnixWare a Unix Strategy?

Novell's UnixWare strategy clearly has strengths and weaknesses. Unix on Intel places Novell in direct competition with Santa Cruz Operations (SCO) and, to a lesser degree, with SunSoft. But if the market for Unix application servers is growing as fast as Novell claims, there should be ample opportunity for all. Unless, of course, Windows NT/AS grabs a significant share of that growing market. If UnixWare is going to succeed in gaining its fair share in competition with the other Unix vendors, then it must do so on the basis of distribution and integration with NetWare. If it is going to succeed against Windows NT/AS, it must be a better technology with more applications. Novell is working in both directions, so it may be able to achieve success.

Novell must have an application platform. Clearly, among all that were available to Novell, it probably chose the best in UnixWare.

AppWare— Ambitious but Necessary

Novell's acquisition of USL and its promotion of UnixWare is designed to offset what would have been a very weak position in application services. The strategy makes sense, and Novell's work with AppWare makes sense—conceptually. It is in the implementation that the rough edges begin to appear. First, Novell is being very ambitious in what it is trying to achieve with AppWare. The company is approaching portability by defining a superset of all platform APIs and then delivering those APIs as a part of its base platform. On the other hand, Microsoft defines portability as bringing its APIs to other platforms. Novell faces a more complex task because it has to mediate different APIs and toolkits. Microsoft is more in control of its own future because of its parochial focus. And Microsoft has an easier task.

AppWare is a developer-oriented strategy. The key developers to drive acceptance of technology such as AppWare are independent software vendors (ISVs), not corporate developers. ISVs can get the ball rolling for Novell by writing full applications and AppWare Loadable Modules (ALMs) based on AppWare, making the technology meaningful to customers. The only way for the average ISV to survive is to write for the most widely installed platforms. That means AppWare is threatened with being put near the bottom of the list of Windows, Macintosh, Solaris, VMS, AIX, HP UX, etc. However, AppWare gives ISVs a free portability layer. ISVs want volume; hence the thrust behind UnixWare and volume platforms.

AppWare is built to the NetWare 4.x interfaces. It will be positioned as a benefit of migrating to 4.x. But, by itself, it doesn't provide a stimulus to move. Customers have other options available to them for developing portable applications.

Sticking with Open Systems

Novell's traditional approach has been oriented toward de facto standards. It has crafted an approach that retains a view that was true in the past: What customers use and buy is what establishes a standard. NetWare is an important de facto standard. Yet, now that Novell owns Unix, it will have to find a middle ground in the way standards are set. The work that has taken place around COSE and Spec 1170 is indicative of the middle ground. Leading vendors serve as catalysts for new specifications which, we hope, are based as much on customer requirements as on vendor need. The specification agreed to by the inner circle is then placed in a public forum, like X/Open, for comment and revision. In this way, no single vendor controls direction, yet a strict consensus process, which often fails to meet user needs in a timely fashion, isn't necessary. We believe that Novell, by supporting this new open process, is in a good position to continue to advance open systems standards in ways that will have significant benefits to users. ◐

Next month's *Open Information Systems* will address
Open Online Transaction Processing.

For reprint information on articles appearing in this issue,
please contact Donald Baillargeon at (617) 742-5200, extension 117.

Unify Vision

New Product from a Renewed Company

Unify Corporation (Sacramento, California) was among the pioneers delivering relational database management systems (RDMBSs) and tools for building portable, event-driven, client/server applications. Its Unify RDBMS was introduced in 1982, and the Accell development tool came to market in 1986. Both arrived and established themselves long before downsizing and client/server development became the leading industry trends of the 1990s, but neither has kept pace with those trends. While database companies, such as Oracle Corporation (Redwood Shores, California) and Sybase Incorporated (Emeryville, California), and tools companies, such as Powersoft Corporation (Burlington, Massachusetts) and Gupta Corporation (Menlo Park, California), have been at the forefront of these trends, Unify had all but disappeared. Technology apparently passed it by. The company did upgrade its RDBMS in 1989 to include recovery and integrity features and expand platform support, but the product never got back into the mainstream. Its development tool retained the look and feel of an 80s product. But in September 1993, Unify rejoined the mainstream with the announcement of Unify Vision, a thoroughly modern, graphical, portable, event-driven, client/server development tool. Unify Vision offers capabilities that can improve productivity in the development of client/server applications. We believe that Unify Vision can be the means to bring Unify into the top tier of client/server suppliers.

Unify Vision is a development tool for client/server applications. Its graphical development environment may be run on Windows, Motif, or OpenLook workstations. The product's toolset can be used to build Windows and Unix-based clients as well as RDBMS servers. Unify Vision applications can be characterized as graphical, event driven, and object based. They are built on a forms-based paradigm. Applications comprise an object-oriented user interface, a procedural 4GL, and SQL access to a variety of RDBMSs. Unify Vision is targeted at professional developers for IS-style and IS-size applications. Typically, applications built with Unify Vision will implement commercial business functions.

Automatic generation features are strong differentiators for Unify Vision. They can improve the productivity of the development process significantly by reducing the time for development and the programming skills required. The user interface and most aspects of database access can be automatically generated. This approach frees a developer to concentrate on implementing the business function performed by the application. We've seen many tools that simplify the specification of SQL statements, facilitate access of data in related tables, and perform transaction management and locking, but none that does these things to the extent that Unify Vision does.

Unify Vision Structure

Unify Vision has four components: Unify Vision Designer, used to create the user interface; Unify Vision Script, the product's scripting language; Unify Vision Debugger, a tool for debugging all aspects of the application, not just the code; and Unify Vision Manager, which creates the environment for running Unify Vision applications. Illustration 1 shows Unify Vision's components and structure and their relationships to the underlying platform, network, and database.

Unify Vision Structure

Unify Vision Structure

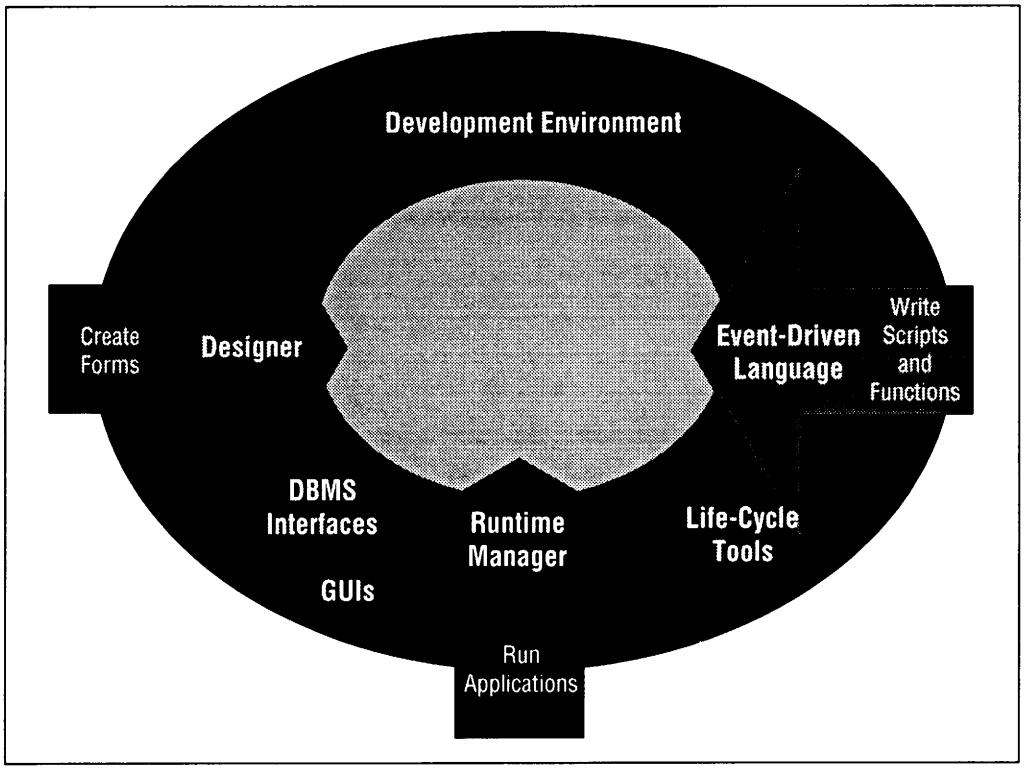

Illustration 1. Unify Vision components are Unify Vision Designer, Unify Vision Script, Unify Vision Debugger, and Unify Vision Manager.

Unify Vision Designer

Through the tools provided in Unify Vision Designer, the developer can create forms, place graphical objects on the forms, assign values to forms and graphical objects, and enhance the appearance of forms and graphical objects with lines, boxes, and labels. Unify Vision's Designer Palette provides a point-and-click interface for "painting" a window. Graphical user interface (GUI) support is provided through a partnership with Visix Incorporated (Reston, Virginia). Unify has licensed Visix's Galaxy GUI builder as the basis of the Unify Vision GUI. (See *Open Information Systems,* Vol. 7, No. 10, October 1992, for more infomormation on Galaxy.) It also provides the mechanisms for Unify Vision's help functions. We applaud this approach. Little differentiation is possible through the GUI of a development tool, yet good graphical support is required. Because Unify is late to market with GUI support, the decision to buy instead of build is a good one, allowing Unify Vision to get to market faster with the features of one of the best GUIs available.

Unify Vision Script

Unify Vision Script is an event-driven language used to "code" the methods for Unify Vision objects. Unify Vision forms, menus, toolbars, radio groups, check boxes, sliders, list boxes, and text fields are all objects for which events can be defined and methods coded. The language comprises statements for array definition and manipulation, assignment, command execution, debugging, decision, deletion, function declaration, input and output, loop control, screen control, selection, transaction control, update, and form sequence control. SQL statements may also be coded and executed from Unify Vision Script. Unify Vision provides many high-level functions to simplify scripting, including systems-oriented and user-defined functions. Unify Vision Script also supports linkage to 3GL routines. Scripts may be associated locally with individual forms or accessed globally from all Unify Vision applications.

Unify Vision Script is similar in structure, size, and complexity to the scripting languages of most other client/server development tools. There's a core of a few language statements to handle assignment, looping, and branching. Then there are dozens, sometimes hundreds, of product-specific functions and commands oriented to the manipulation of graphical objects. Because these languages are awkward and hard to learn, it's extremely important that tools reduce the need to use them. Unify Vision Script is no better and no worse than other scripting languages. However, with Unify Vision's level of automatic generation, developers won't use it too often.

Unify Vision Debugger

Unify Vision Debugger supports breakpoints and watchpoints, where values or attributes can be set, variables tested to determine whether they've been initialized, the names of activated forms listed, the name of the current field displayed, variables displayed, and script files browsed. Breakpoints may be set for application object events as well as for Vision Script statements. The debugger runs with Vision Manager, so applications must be running for the debugger to be active.

Unify Vision Manager

The Unify Vision Manager is the "engine" that provides the environment for Unify Vision applications. It controls events, system functions, and global functions for applications. It anchors the application to the underlying system, manages the interface to databases, and enforces the transaction model. Unify Vision Manager is based on technology originally used for Unify's Accell/SQL product.

Unify Vision Application Components

Unify Vision applications comprise forms, menus, and toolbars. These components are object oriented, and they have attributes and methods. In fact, "menu" and "toolbar" are attributes of forms. As a whole, however, Unify Vision is object based. The methods of forms, menus, and toolbars are implemented within a procedural 4GL. Note that Unify calls object attributes "properties." We'll use the Unify term in this article.

Forms, menus, and toolbars are defined in classes. Classes are managed in object libraries. The terminology is somewhat misleading because the initial release of Unify Vision supports only classes of a single type. Any number of classes may be defined, but there is no support for subtyping (inheritance) within a class. Future versions of the product will offer greater levels of object orientation, including inheritance. For now, use of this object-oriented terminology paves the way for the more robust object functionality yet to come.

Classes may be reused. Reuse involves opening an existing class, modifying it to address application requirements, and saving it as a new class.

Unify Vision Application Structure: It's All in the Forms

Forms are the fundamental entity in Unify Vision applications. Developing a Unify Vision application involves defining forms, setting form properties, establishing relationships between forms, and adding 4GL code to script events for forms and other Unify Vision objects. Key to the value and appeal of Unify Vision is the level of application capability that is built into forms based on the selection of form properties. The power and productivity benefits of Unify Vision are delivered through these properties. Property selection is accomplished through ten graphical dialogs. The forms property selection process is what Unify calls SmartView. The properties are accessible through the Forms Property Sheet, shown in Illustration 2, a dialog activated by double clicking on a form. Let's take a look at what the developer can accomplish with them.

Unify Vision Application Structure: It's All in the Forms

Unify Vision Form Properties

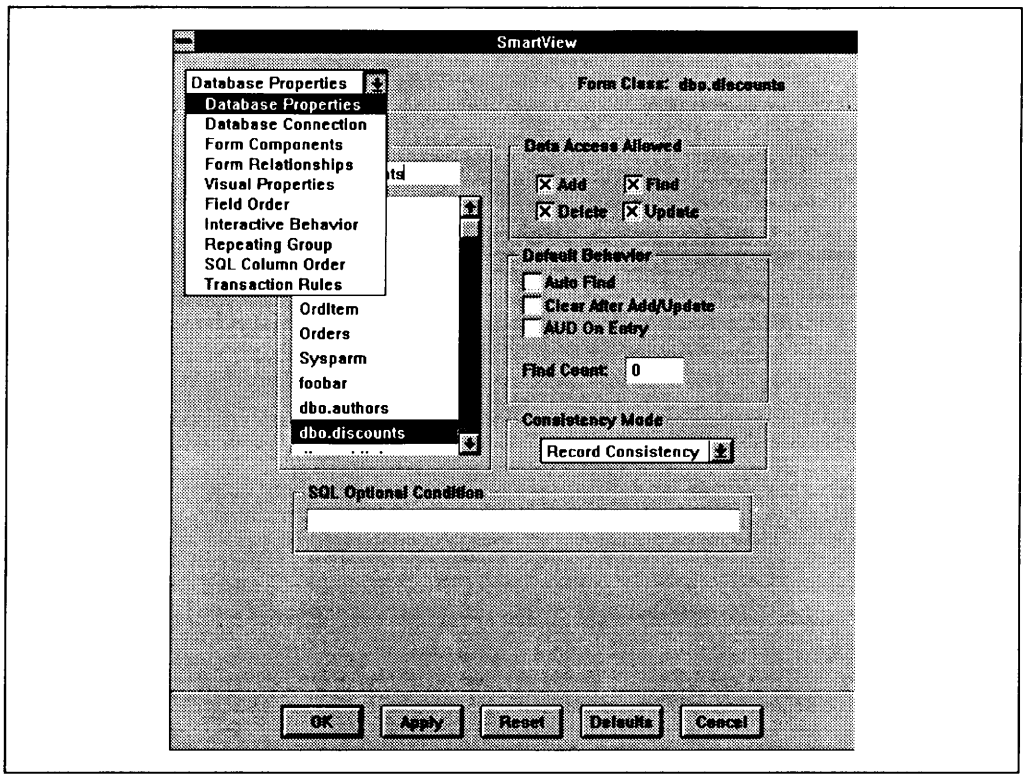

Illustration 2. Form properties are used to accomplish many development tasks and specify many application functions. Properties are set via point-and-click selection from the Forms Property Sheet, a dialog activated using the mouse or Development Environment menu.

Form Properties Dialogs

Five form properties dialogs are used to specify the visual appearance and user interface to the form. They provide the functionality that has come to be expected with graphical development tools. Their capabilities make Unify Vision competitive, but there is no differentiation here.

- **Form components.** The form components dialog is used to select the menu and toolbar that appear on the form at runtime.
- **Visual properties.** The visual properties dialog is used to specify the size of the form and its foreground and background colors.
- **Field order.** The field order dialog specifies the tab ordering of fields on the form. Form fields are automatically linked to columns in the database table connected to the form.
- **Interactive behavior.** The interactive behavior dialog specifies where and when the user can interact with the form through the mouse and keyboard. Properties include the form's mode and whether context-sensitive help will be available.
- **Repeating group.** The repeating group dialog specifies the number of occurrences of a graphical object and the spacing between those occurrences.

The other five dialogs address database support and relationships between forms. This is the area where Unify provides significant productivity advantages.

- **Database properties.** For every form linked to a database table, Unify Vision automatically generates the SQL statements required for finding, adding, updating, inserting, and deleting data. The functions that execute these SQL statements are "built in" to the form, as are the capabilities to generate queries through query-by-example (QBE). The database properties dialog is used to specify the database table that is linked to the form; to restrict user access to the database find, add, delete, update, and insert functions; to specify the number of rows in the selected set; to specify transaction consistency mode; and to specify an SQL "Where" clause to further qualify each query.

- **Database connection.** Unify Vision supports concurrent access to multiple databases from within a single application. The database connection property specifies whether the built-in database access should inherit the prior form's database connection, use a different database connection, or use the application's default connection.

- **SQL column order.** The developer may specify the sort order for a database column that is displayed on the form through the SQL column order dialog. This becomes the SQL "Order By" clause.

- **Form relationships.** The sequence of Unify Vision application processing is determined by the "next form" function. From the current form, the next form may be accessed programmatically from events on the current form or may be selected manually by the user from a menu on the current form. The form relationship dialog is used to specify the next form to be displayed, the mechanism to be used to cause its display, the transaction state and consistency level to be set for it, and the keys to be used for establishing master-detail relationships between the current form and the next form (a description of master-detail is given below). A form may have several next forms specified for it. Application logic will determine which next form is to be displayed. Optionally, a menu of next forms may be automatically displayed for user selection at runtime. Illustration 3 shows the form relationships dialog.

- **Transaction rules.** Transaction control in Unify Vision applications may be accomplished through setting form properties. Transaction control commands are executed on the activation of the next form. The transaction rules dialog allows the developer to set the state of the current transaction when the next form begins execution. Options are continue, commit and release locks, and commit and hold locks. Alternatively, default values can be used to insulate the developer from the complexities of transaction control.

Master-Detail and Zoom

Master-detail relationships between forms and zoom forms provide the Unify Vision developer with powerful and useful capabilities in accessing and manipulating related data.

Zoom Forms. Zoom forms are sub-forms that are properties of check boxes, list boxes, radio groups, sliders, and text field objects in top level forms. Events on those objects can activate and display zoom forms. Zoom forms may be linked to a database table other than that of their top level forms. So, zoom forms can be used to access related data based on processing logic, data values, or computations.

Master-Detail. Master-detail relationships are common among database tables in commercial applications. For example, a customer table may have a master-detail relationship with an orders table. An order entry application may want to know all the current orders for a particular customer. Master and detail form properties in the Unify Vision forms relationship dialog can be used to specify this foreign key type of relationship between forms and the tables to which they are connected. The master property and the detail property specify the database table column to be used as the foreign key. That specification is all that's needed; Unify Vision does the rest automatically. Master-detail performs logical joins between tables. Master-detail/detail/.../detail relationships may be established among groups of forms.

Unify Vision Application Structure: It's All in the Forms

Unify Vision Form Relationships Dialog

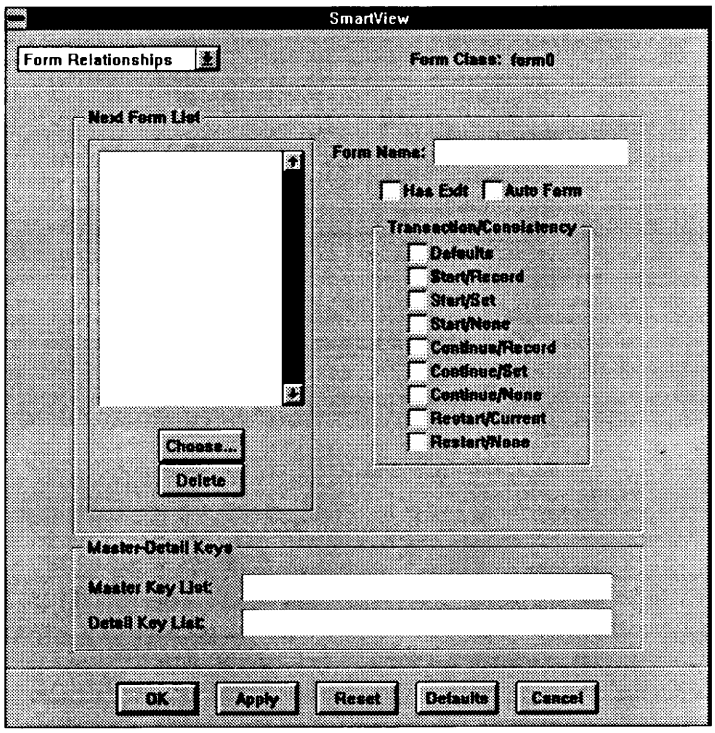

Illustration 3. The form relationships dialog is used to set application processing sequence, transaction control, and relationships between forms.

Database Support

When Unify Vision first ships to customers, the product will offer native database support for Oracle, Sybase, and Unify 2000 RDBMSs. By "native" support, we mean that Unify Vision will access the databases through their own APIs and all functions of those APIs will be supported. Native support requires a sizable development effort for Unify, but the payback to the developer, both in function and performance, easily justifies it. Unify Vision also supports the Microsoft Open Database Connectivity (ODBC) interface. ODBC can increase the portability of applications across many databases and can expand the number and types of data supported by a development tool. For example, through Unify Vision's ODBC support, any of several development platform databases can be accessed. Local access improves support for prototyping. However, ODBC can't offer the function or the performance of the native API. Unify's approach of combining native interfaces for the key RDBMS for function and performance and ODBC to expand database support, prototyping, and portability is practical. Native support for additional databases will be increased as the market demands. In fact, native support for both the Ingres and Informix RDBMS will likely be delivered within the first half of 1994.

Portability and Transparency

Based on form properties, Unify Vision automatically generates SQL Data Manipulation Language (DML), locking requests, and transaction control commands to the database to which the form is attached, as we mentioned earlier. Should a form originally developed for Oracle be switched to connect to Sybase, Unify Vision will support that switch as transparently to the application as possible. Differences between the two RDBMSs in command syntax and semantics and differences in locking and transaction control are accommodated by the Unify Vision Manager. This is a big portability boost. However, as

portable and transparent as Unify Vision is across RDBMSs, there are some fundamental differences between RDBMS that can't be addressed from the client. For example, in the switch from Oracle to Sybase, the Oracle Triggers used by the application must be ported to Sybase. Database programming portability may not be as transparent as client application portability.

Prototyping

Unify Vision offers good prototyping capabilities. An application prototype can be generated and enhanced without coding and without requiring connection to the server database. Based on an existing database schema, Unify Vision will automatically generate a user interface and the function to access and manipulate the database. No coding is required, and that user interface may be executed directly from the development environment. This very basic prototype can be easily and iteratively enhanced. Unify Vision Designer can be used to customize the user interface to meet user requirements. ODBC support for several small, development platform-style databases can be employed to accomplish most of the prototyping locally to the development environment.

Client/Server Architecture

Unify Vision supports the distributed SQL model of client/server architecture. Unify Vision clients perform all application processing between the user interface and database access. Unify Vision servers are RDBMSs. The client/server interface is SQL. The interface is supported across all networks supported by the RDBMS used within the application.

With its older Accell/SQL product, Unify had offered Accell/TP. With Accell/TP, Accell/SQL applications could be built for more generalized distributed processing capability within an OLTP environment. Accell/TP provides the tools for the development of Accell/TP servers and the adaptation of Accell/SQL clients in an OLTP environment managed by a third-party transaction manager such as Tuxedo. Accell/TP servers may offer processing or database functions. Accell/TP clients may access these "global" transaction managed servers or the "local" servers of standard Accell/SQL clients.

We believe that, as larger, more complex client/server applications are being deployed, the capabilities of products like Accell/TP will become increasingly important. Already, users are beginning to have performance problems on client platforms, especially Windows clients. The popular way of offloading clients and improving client performance is to code part of the application as stored procedures in databases. Stored procedures can be effective only for database-related logic, and the tools to build them don't offer the productivity of client development tools like Unify Vision. In addition, it is hard to anticipate how long it will be before the increase in application complexity and the number of users result in performance problems in the RDBMS on the server. The requirement boils down to support for distributed processing in the client/server environment, and Unify is investigating alternative approaches. Transaction processing managers offer one way to address the requirement. Alternative solutions include RPCs (Remote Procedure Calls) and distributed object computing frameworks. We think it would be a good idea to make Accell/TP functionality available for Unify Vision applications as a short-term fix interim to the longer-term strategic solution being expored by Unify.

Application Management

Unify Vision applications are managed within files on the underlying development platforms. Components are managed in a hierarchy of projects, folders, applications, and libraries. Libraries contain the objects that comprise applications: forms, global functions, menu bars, and tool bars. Applications contain libraries and application profiles. Application profiles define the search path, the entry point, and the database connections. Folders contain one or more applications, and projects contain one or more folders. Unify Vision provides a check-in/check-out capability at the level of application objects. Third-party application management products may be used to manage Unify Vision application files. Unify is seeking a third-party management product to tightly integrate with Unify Vision. The company hopes to have established the relationship and integrated the technologies in time for the next release of Unify Vision.

Query and Reporting

Query

As mentioned earlier, Unify Vision forms automatically support query by forms (QBF) capabilities. End users may key in data on forms to specify query conditions. Unify Vision will retrieve all the records in the tables connected to the forms and the records in the tables connected to related forms that satisfy those conditions. The end user may interactively and iteratively qualify the query. This interactive query capability approaches the level of function and the ease of use for QBF capabilities of the popular database access tools such as Q+E Data Editor from Q+E Software Incorporated and PowerViewer from Powersoft Corporation. Query conditions may be specified in one form field or in more than one field; conditions within one field are implicitly ORed; conditions among fields are implicitly ANDed. These conditions include full-field value specification, partial-field value specification, specification of value ranges, expressions comprising arithmetic and logical operators, and any function in the Unify Vision Script language. More complex queries can be handled through scripting or through the "where" clause of the query's SQL "select" statement. The "where" clause is specified as a property in the database properties dialog. Unify Vision's query capabilities are powerful and easy to use from within Unify Vision applications.

Reporting

The Unify Vision Report Writer is designed for the generation of tabular reports. Report input is a stream of records comprising delimited columns and ending with a new line character. The input is processed by a procedural report script. The script language bears a striking resemblance to RPG. Reports are formatted with headers and footers. Formatting commands are print, skip, sort, and need; "need" ensures a minimum number of lines per report page. The report writer also provides functions for column calculations: average, count, min, max, and total. In addition, the report writer may call external C language routines for additional functions.

The key advantage of Unify Vision Report Writer is its ability to be used in batch mode on Unix servers to generate production reports. The product is not a particularly attractive tool for graphical, client/server-style reporting, especially in contrast to the highly graphical, powerful, yet easy-to-use query tool/report writer products like Impromptu from Cognos Corporation (Burlington, Massachusetts), PowerViewer from Powersoft, or Quest from Gupta Corporation.

Third-Party Query and Reporting Support

Unify has established marketing and development relationships with vendors of reporting/query tools in order to address general and end-user query requirements and to provide simpler, more graphical reporting. Through these partnerships, Unify hopes to offer a complete tool environment for its customers. IQ Software Corporation (Norcross, Georgia) and ReportSmith, Inc. (San Mateo, California) are two of the first of these partners. IQ

Software's Intelligent Query (IQ) and ReportSmith's ReportSmith product can be integrated with Unify Vision to address customers' query and reporting requirements.

Competition

Competition for Unify Vision exists in three areas: tools for portable client/server applications, tools of the RDBMS vendors, and Windows-based development tools.

Portable Tools

The distinguishing feature of Unify's products has been portability. Accell/SQL, Unify Vision's predecessor product, was known for its support of a broad range of Unix and proprietary systems. Accell/SQL also offers good database portability. Unify's traditional competitors have been JYACC (New York, New York) with its JAM development tool and Uniface Corporation (Alameda, California) with its Uniface development tool, and portability has been the basis of their competition. All three products share platform and database portability features and have been on the market for some time. However, their similarity ends with platform and database portability. The products differ significantly in their strengths and weaknesses and in the sorts of applications best suited to them. JAM is strongest in platform and user interface support and weakest in database support and the power of its 4GL. JAM applications usually contain a considerable amount of C code. Although JAM supports many databases, database access frequently requires explicit SQL coding. In contrast, Uniface is strong in database support, and its 4GL can be used to address most application requirements without the addition of 3GL code. However, Uniface is a complex product. Its applications are based on the ISO, three-schema architecture. Their development is repository-based and requires the definition of an application model. Where Accell and JAM can effectively be used for prototyping and rapid application development (RAD), Uniface application development takes a far more structured approach. Significant training and up-front design are required.

With the introduction of Unify Vision, Unify has jumped ahead of its traditional competitors. Unify Vision's graphical development environment and the development productivity made possible through its database support features have created some distance from JAM and Uniface, at least for the time being. However, in October 1993, JYACC announced the next version of JAM, JAM 6, which will become commercially available in early 1994. The product was about to enter beta testing when it was announced. JAM 6 promises a fully graphical development environment and repository-driven development that should vastly improve its database support. JYACC has also added a debugger to the JAM 4GL. JAM 6 features look very similar to Unify Vision features, and a JAM 6 demonstration looks an awful lot like a Unify Vision demonstration. Choosing between these products will be hard. For now, Unify has a several-month window of opportunity to establish itself in the market while JYACC presses on with the development of JAM 6. We suspect that JYACC was pressured into this early announcement of JAM 6 in order to compete with the announcement and early success of Unify Vision and with the continuing success of the Windows-based client/server development tools. JAM 5 had become uncompetitive. Uniface has not yet responded. We understand that a major new version of the Uniface development product will be introduced sometime in 1994. Not surprisingly, among the new features will be a graphical development environment.

RDBMS-Linked Tools

For a given RDBMS, Unify Vision competes with the development tools offered by the RDBMS vendor, a situation where portability is no longer an advantage. For customers willing to consider a "third party" development tool, Unify Vision, ironically, offers productivity advantages in database support. Unify Vision performs more automatic generation for database access and transaction control than the Ingres Windows4GL, the Informix 4GL, and the newly-announced Build Momentum from Sybase. Oracle Forms 4.0 provides database support very similar to that of Unify Vision. In fact, we believe that, for accessing Oracle databases, Forms and Unify Vision are comparable alternatives.

Competition

Windows-Based Tools

Windows-based development tools for client/server applications like PowerBuilder from Powersoft Corporation and SQLWindows from Gupta Corporation are the rage of the industry. PowerBuilder owns client/server mind-share and is quickly building market share. It has become the industry client/server benchmark. Every new tool gets compared to PowerBuilder, and the vendors of new tools must differentiate their products from PowerBuilder. Here's a brief comparison of the two:

- Unify Vision supports Windows and Unix development and runtime platforms. PowerBuilder is only Windows-based.
- Unify Vision provides "native" database support for Oracle and Sybase (Ingres and Informix next year). Native support takes best advantage of RDBMS-specific features and provides better performance. PowerBuilder supports more databases, but many are supported through ODBC.
- To access data and control database transactions, Unify Vision provides significantly more automatic generation than PowerBuilder. Unify Vision developers write much less SQL code than PowerBuilder developers.
- Unify Vision objects will not support inheritance until the next version of the product. Inheritance facilitates reuse; reuse improves productivity. PowerBuilder objects support inheritance.
- Unify Vision and PowerBuilder offer similar protoyping facilities. Both can run applications from the development environment. Both can execute applications before adding code. Both support a variety of RDBMSs, which can be accessed on the development platform to aid in protoyping.
- Unify Vision reports can be run on the server without client involvement. Server-only execution is well suited to large, long-running, regularly scheduled production reports. However, through its PowerViewer add-on product, Powersoft offers easier to use and more powerful query and reporting capability than Unify Vision.
- Unify has few third-party partnerships but is working on establishing them. Powersoft has built third-party relationships for enhancing PowerBuilder's capabilities in application management, CASE integration, and communications.
- Unify Vision's runtime engine and transaction model are based on Accell. Accell has been used to develop and implement applications supporting hundreds of users. Unify Vision applications will therefore probably scale to this size. PowerBuilder has had problems in scaling.

Unify Marketing

Founded in 1980, Unify has always been a technology-driven company, not a marketing-driven company. Its approach to the market has been more like Digital Equipment Corporation's than IBM's, more like Apollo's than Sun's. The technology-driven approach leads with product features and functions. Features direct the company toward the right customers who would intuitively recognize the value of the technology, buy it, and apply it correctly.

The technology-driven approach resulted in steady growth for Unify at the beginning of the 1980s. Then, in the late '80s, industry-wide growth in the RDBMS and development tools market left Unify behind. As mentioned at the beginning of this article, neither the tools nor the RDBMS were enhanced to keep up with technology trends. By 1991, growth had slowed

considerably. The company hired a new CEO, Jim Hammock, who set about rebuilding the company. Now, with a completely new management team, Unify has become a new company. Unify Vision is a new product, the first product of the new Unify Corporation, which is moving toward a balance between technology and marketing. Some of the key marketing aspects of Unify are described below. Considerable marketing work is needed. The good news is that it has begun.

Product Positioning

Unify has positioned Unify Vision as a development tool for building open, client/server, database applications. Unify Vision is designed to assist MIS application development managers in their efforts to downsize mainframe applications and to upsize personal applications onto open systems. Development productivity and platform and database portability are the key Unify Vision features.

Target Applications

Unify would like Unify Vision to be considered as the preferred tool for building large applications. Unify Vision is based on the Accell runtime engine, and Accell has supported applications with large development teams and hundreds of end-users. Therefore, Unify Vision should also support these large applications. This strategy moves Unify Vision away from Windows-based development tools which appear to be only departmental solutions. This strategy positions Unify Vision against products like Uniface and JAM, over which Unify Vision has significant productivity advantages.

The justification of Unify's strategy is flawed, however. First, Unify Vision does not provide the mechanisms to support large development teams. It offers only a simple check-in/check-out facility against objects managed within files on the development platforms. In fact, Unify is seeking third-party relationships with vendors that have more robust team development technology. Second, Unify has historically marketed its products at the departmental level. Departmental applications, even the largest ones, don't support hundreds of users. The goal for this strategy is, however, excellent. IS development managers want tools for building large applications. Tools like PowerBuilder and Visual Basic are proving unscalable beyond twenty or so users. Tools like Uniface, although seemingly scalable, do not offer the productivity advantages of graphical development and RAD. There is a tool vacuum, and Unify Vision is expanding into it. We'll soon discover exactly how suitable it is for the biggest applications. If it works, Unify will be an industry leader. If not, the company still has a nice tool for high-productivity development of department-size applications.

Channel Strategy

Unify is developing a mixed-channel strategy. The company is beginning an effort to build a reseller channel by recruiting value-added resellers (VARs) and systems integrators that will base their products and services around Unify Vision. Unify's direct sales force will focus on large strategic opportunities and on "pulling sales through" the VAR channel. Unify already markets Unify Vision through distributors in Europe and in Japan.

Like Unify, many of the client/server-tool vendors are building reselling channels. The industry is in the midst of a major shift away from direct sales and toward reselling. Competition is the driving force from two perspectives. Among vendors, increased competition puts extreme pressure on product prices. Pricing pressures can be relieved by reducing sales costs. Among users, time to market is critical to establishing a competitive edge. Buying applications and customizing packages instead of building them can shorten the time to market. VARs and systems integrators can help vendors and users. They absorb much of the cost of selling development tools, and they deliver solutions in less time than users can build them. Building a reseller channel takes some care and planning. It's easy to recruit and sign up VARs. It's not so easy to produce actual sales and to maintain VAR loyalty. Strong sales and marketing programs, quality technical support, and attractive pricing are required.

Pricing

The Unify Vision pricing model includes a development license charged per developer independent of platform and a platform-independent, runtime charge based on the number of

end-users. Unify's typical runtime charges for an installation are 10 percent of the development license charges. Unify is considering changes to this pricing model.

Windows-based development tools vendors have all but eliminated runtime charges for their products but have begun to beef up their maintenance charges. Unix-based vendors like Unify, JAM, and JYACC have maintained runtime charges. The two approaches walk the increasingly fine line between competitiveness and profitability. We would recommend that users consider total costs in comparing products. We consider maintenance agreements to be very important. Client/server development tools change rapidly, and product cycles are less than one year. With the tools being used for critical applications, maintenance is an essential cost item.

Product Plans

Unify plans a new release of Unify Vision for the second half of 1994. Key new features will be development and runtime support for the Macintosh and Windows NT platforms, and additional object-oriented features including support for subtypes and associations within the product's class libraries. By the next release, Unify hopes to have established partnerships in application management and CASE support, and to have developed and implemented a distributed processing strategy.

Conclusion

Unify Vision is a very attractive client/server development tool. Its productivity features for database access are as good as any currently available on the market. For the next several months, Unify Vision will position Unify uniquely in a client/server market niche. Unify Vision appears to be more scalable than Windows-based tools and less complex than the current versions of portable tools. Unify must act quickly to take advantage of this position. It must find customers to build big applications with Unify Vision and do everything possible to ensure that those customers succeed. Customer success will bring success to the company. At this point, Unify Vision is just coming into general availability, and Unify needs to demonstrate proof of its scalability. A lot of work must be done, and the competition is not resting.

Longer term, Unify must work on its marketing because Unify Vision will face stiff competition. In the shift away from a technology orientation, the company must develop and implement a marketing plan. Product positioning, marketing strategy, target market, application, buyer, channel strategy, and pricing must all be more crisply defined and developed.

Open Systems: Analysis, Issues, & Opinions

APPLIX

Applix*ware*: Modularizing Aster*x

Aster*x Becomes Applix*ware*

Applix's (Westboro, Massachusetts) Aster*x, a three-year old product, is a Unix-workstation-based, client-based, compound document editor that supports text, graphics, and spreadsheet objects. Basically, Aster*x was a single environment which included multiple editors; the user, however, never needed to be aware that he or she was dealing with different editors—the feel was that you were working in a single editor which happened to offer the desired functionality based on what you were editing (text, graphics, spreadsheet, etc.). We had positioned Aster*x as an integrating environment because you can embed documents from almost any outside application, including them in the compound document, but maintaining links to the source file.

The entire product sits on top of the Extended Language Facility (ELF), a powerful scripting-based programming language as well as a macro recorder and a point-and-click macro definer that is the underlying architecture for all Aster*x components. The major attraction of Aster*x over products such as the Island Graphics suite of Write, Draw, and Paint is the extensibility of the functionality with ELF. ELF is a full programming environment—thus, complete applications can be written in ELF using the Applix applications as the user environments.

However, the other side of the coin is that users, today, are reluctant to buy single products that include multiple functions, preferring the Island Graphics model where you can buy only those pieces you want—with, of course, the assurance that they all work together seamlessly. Applix has recognized this buying pattern, and has decided to take apart the pieces of Aster*x, selling them independently as part of the Applix*ware* line.

The Components of Applix*ware*

The components that made up Aster*x and which become separate products in Applix*ware* include:

- Applix Words—full-featured word processor
- Applix Spreadsheet—full-featured spreadsheet
- Applix Graphics—full-featured graphics (draw) package
- Applix Mail—a Unix sendmail front end

All of these components have been significantly enhanced from the latest version, 2.1, of Aster*x. (Originally, before the company decided to separate the components, the functionality that is being delivered was to be Version 3.0 of Aster*x.)

Applix has added new components to Applix*ware*:

- Applix Data, a data access product that supports access across multiple databases in a single query. The product completely hides SQL, using a point-and-click front end. The first version of Applix Data supports Oracle, Sybase, Ingres, and Informix databases.
- Applix Open Mail, an X.400 mail client based on HP's OpenMail.
- Applix Builder (coming by year end), an object-oriented version of ELF. Builder is an applications developer's tool which is closely linked to the databases supported by Applix Data. Rather than being the next generation of ELF, Builder is a complementary tool.

Applix plans to build and add more components that fall into the target vertical business lines, such as real-time spreadsheets. The company is also encouraging third party developers and value-added resellers (VARs) to write line-of-business components for Applix*ware*. These components are written in ELF.

In addition, the Applix*ware* products can establish live links to popular desktop applications via ELF. The resulting integration is seamless to the user.

Since Applix*ware* is no longer a single product, Applix has developed a Personal Desktop from which all Applix*ware* components can be accessed.

Aster*x Isn't on the Critical List

Although Applix and Aster*x haven't been very visible over the past year, the product and the company have been doing just fine, thank you. That's the message Applix wants to send to the industry. Aster*x currently has over 200,000 users in the federal government and an additional 150,000 nongovernment users worldwide. The product has become very verticalized in three areas where there are heavy concentrations of Unix workstations:

- Financial/banking
- Telecommunications
- Engineering/manufacturing

For those of you wondering about Alis, Applix's first integrated office suite—which was the first Unix-based GUI office environment to come to market—the product is still alive. Although it is only being maintained in the United States, it is still being actively sold in Europe. However, most customers have migrated to Aster*x.

(For background information on Alis and Aster*x, see *Unix in the Office*, September 1991, Vol.6, No. 9.)

Pricing Structure

One of the problems with Alis was that you had to buy the entire office environment even if you really only needed some small piece of the product. Applix took a large step toward solving that problem with Aster*x, which was a slimmed-down version of the integrated office system. Now, with new modular product packaging, customers can really buy only those pieces they need.

ELF is given away with every component (except Mail and Open Mail). Applix recommends that you buy both words and graphics as the base product, though this isn't required.

- Words, graphics, ELF, and macro library: $695
- Spreadsheet, ELF, and macros: $495
- Data, ELF, and macros: $995
- Builder, ELF, and macros (not yet available): $2,495
- Mail and Open Mail: $195 and $295, respectively

Conclusion

The message that Applix is trying to get across is that it is not entering new markets; the company has always been in the integrated office, document publishing, E-mail system, workflow automation, application development, and relational data access areas. But, by and large, the industry was unaware that all these capabilites were available in Aster*x (using ELF as the application development and workflow builder). The new modularity of the product emphasizes the wide range of functionality available in Applix*ware,* allowing you to buy only the pieces you need.

But Applix's main message is the emphasis on the strategic positioning of ELF as the underlying shared architecture. Using ELF and, soon, Builder, Applix*ware* becomes a completely extensible environment for developing Unix-workstation-based customized applications. — *R. Marshak*

