Release 1.0

88-11 9 November 1988

VAPORWORDS: IF YOU DON'T HAVE THE PRODUCT, SELL A CONCEPT

You could say that the computer industry is operating on a higher plane than ever this year, since fashionable talk centers not around unavailable products but unavailable concepts such as standards, (binary) compatibility, portability, scalability, interoperability, cooperative processing and vaguest of all -- openness. The problem is that these concepts, unlike at least some vaporware of prior years, are fated never to be realized in pure form as people imagine them, since they carry in them inherent contradictions. That is, while concepts exist unfettered, implementatations of concepts are subject to trade-offs, installed-base pressure and technical constraints. Moreover, such goals shift -- now that we're approaching portability, we're beset by proliferating incompatible user interface "standards."

At this Comdex there are few new products worthy of note (but we'll try to find them). Instead, we have interest groups rallying around would-be standards that incorporate or provide the vaporvirtues listed above. There's the Open Software Foundation and its counterpart the Archer Group; there's the Open Bus Foundation or the EISA Gang of Nine, promoting a counterstandard to IBM's MicroChannel. (Savvy customers such as New York's Microcomputer Managers Association have noted that now we'll simply have incompatible memory cards at a time when most peripheral devices won't come on cards but will instead be built into the machine or available from a network server.) And there are smaller vendors or groups proposing other commendable standards: Intel and DCA with CAS; SoftView with the SoftView Tax Data Standard for financial information; Hewlett-Packard with NewWave; and everyone with a unique yet standard user interface for UNIX applications.

We can't possibly sort through all these things, but we will attempt to provide a user's guide to the vaporwords for consultation by those susceptible to claims of vaporvirtue. Can it really be as good as that once it's in a real-live product? Or is it best left as a concept, unsullied by the trade-offs and commercial considerations of reality?

To illustrate, let's start with a tried-and-true vaporword: compatibility. Compatibility is now taken for granted in the pc world, so it has lost its power there -- but the workstation vendors want it and the magic market power it im- ⟹

INSIDE

VAPORWORDS 1
Portability, scalability.
Interoperability.
Openness.
Grand unified vaporvirtue.
SOFTWARE ACROSS SYSTEMS 8
Hunter Systems.
UNIX interface wars.
FREE ADVICE 16
Be specific.
A rationale for groupware.
NITTY GRITTY EXPERTS 19
Report conversion.
Drawing by the rules.
FORUM/RESOURCES/CALENDAR 22-27
Coming to Palm Springs...

plies. Compatibility means building your hardware to match the big guy's so that you can share the third-party support he has garnered. (For users and software vendors, it means that their software will be portable as long as they limit themselves to that range of compatible machines.) The problem is, this approach doesn't work in a market where there is no big guy, such as the market for UNICes, all the versions of UNIX. And no one really wants to create a big guy to be compatible with -- unless it can be oneself. So when it seemed that AT&T and Sun were about to get together to become a big guy, some other little guys (including Apollo, DEC and IBM) banded together to create the Open Software Foundation. And so forth. Now we have several groups trying to tell the UNIX world what standard to be compatible with.

This compatibility, moreover, can't be total. For one, UNIXes run on a variety of different processors (not just Intel-architecture variants, as DOS and OS/2 do) with different instruction sets, so shrink-wrapped software is not going to run across all UNIX machines. The source code for an application must be recompiled for each class of processor -- Motorola 68000, Motorola 88000, Intel 386, MIPS, SPARC, etc. There's just no way around this that makes any sense, although Hunter Systems (page 8) makes the process fairly painless. Also, there will be shrink-wrapped software for processor-defined classes -- the binary compatibility Sun and AT&T are working on. OSF has promised to produce a non-machine/processor-specific source code version of its operating system, so as to favor no one, which everyone can compile for a specific machine from the source code. (Although it will have to be implemented on something first if only for testing.)

Portability

Even without binary compatibility, a common operating system does at least foster application portability (across). As with hardware, "portability" is not an absolute. It implies that something is possible, not necessarily easy, to carry (hardware) or to move from machine to machine (software).

Common wisdom posits that software portability is inherently good. Vendors like it because they can address a larger target market; users like it because they can run the same software on a variety of different machines, and because it encourages competition among the vendors of those machines. Scalability is portability across similar machines of different performance levels. In essence, it means that a hardware vendor has built a range of compatible machines at different performance levels: Thereby, even software not portable across different hardware lines can run on different-sized machines across the same line, offering varying levels of performance -- the Digital Equipment approach par excellence, so why is it now doing deals with Tandy, Apple and even Compaq? (Applied to hardware, scalability means that an architecture is easily implemented at different performance levels, implying indifference to semiconductor manufacturing techniques and a simplicity of design that will allow the use of ever-finer wafer integration.)

How can applications be made portable? One, they can be purely logical applications (text-based, no I/O) written in a standard language. Then they can easily be recompiled for a variety of target machines. Alternatively, they can be database applications written for, say, Oracle, which like the so-called "portable" operating systems has been implemented on many different machines. Or they can be written with some high-level tools that will automatically generate environment-specific code, down to providing graph-

VAPORWORDS DEFINED

At several recent conferences devoted to vaporvirtues (Sphinx's Open Software Forum in London and Adapso's Emerging Software Technologies conference, among others), there was widespread confusion among *portability*, *scalability* and *interoperability* (tightly defined). In fact, these are not three separate things, but two different approaches to *interoperability* (loosely defined).

Interoperability, a choice vaporword, has two meanings in use. It means both several applications/environments talking to one another (the more appropriate, "true" definition), and one application running in a variety of environments (portability and scalability). *Portability* and *scalability* allow you to run the same software on machines of, respectively, different architectures and different sizes, while true *interoperability* allows different software programs, possibly on different machines, to work together and to remain different and optimized for their environments. True interoperability is what Ashton-Tate's Ed Esber means when he says, "We're committed to having an Ashton-Tate/dBASE solution across the board from pcs to VAXes. But it won't necessarily be dBASE itself, the same code -- just the same tools and language and data structures." In fact, it could be InterBase, with which A-T has a deal on undisclosed terms.

Compatibility is the hardware approach to the issue, making machines alike enough that software doesn't notice the differences. (If you're willing to forgo *binary compatibility*, then making software operating environments enough alike offers portability through automatic recompilation of source code.)

Cooperative processing is independent of but related to all of these, the highest form of interoperability. Cooperative processing allows tasks to work together with some "understanding" of each other, regardless of their platforms or environments, while *interoperability* merely assumes that different entities can effectively trade data or otherwise communicate at a low level. It is the notion of separate tasks or programs working together through the exchange of messages to accomplish a single unit of work.

Finally, *interoperability* is (1) a matter of degree and (2) implies compromise. It's not binary. In fact, all these terms should be used with explicit reference to a domain -- "portable across all Intel-architecture machines" or "interoperability within the SAA environment" or "interoperable between UNIX and OS/2 systems" or "compatible with any SQL database" or "open to access but not to copies" or "portable with a week or so of tweaking." How many "portable" programs can be run on an Apollo? A PC with 256K? An Apple //? A washing machine? Explicit references help keep people honest. (And coherent: When people say a database or operating system is "portable," they generally don't mean that at all: they mean that it has been implemented with great expense and trouble on a variety of machines so that *applications written for it* are portable.)

ics for a variety of systems. We've heard of a lot of these, especially ones that will allow you to turn the same specs into Windows or Mac implementations, but none are yet in wide use. CASE tools in general, in combination with machine-specific compilers for the source code they generate, are an effective way to achieve portability in the sense of easy movement of applications across platforms. Or they can simply be written with portability in mind.

Existing applications

The secondary, portability-based form of interoperability -- making the same application run anywhere -- can be achieved either by re-engineering the application wholesale for execution in another environment (easier if it's portable), or by emulating some part of the original environment within the new one, either with software or with a co-processor. (This includes DOS compatibility boxes and even, in some sense, the wasteful use of OS/2 on a 386, when it's the operating system holding the applications back rather than vice versa.)

Both these approaches have their trade-offs. The cost of re-engineering is high per application, but provides the greatest performance benefits if that single application is to be used frequently. On the other hand, if you're going to be running a variety of applications infrequently, the initial cost of emulation is high, but there is no variable cost per application (assuming that a given application runs at all).

Worst is the certainty that an application within an emulated environment will enjoy few of the benefits of that environment (except perhaps a more powerful processor emasculated by emulation overhead), whereas a rewritten application may actually be improved somewhat in the process of rewriting.

Trade-offs

Obviously, none of these approaches easily allows for an application to be optimized for any specific environment. An environment, of course, includes not just raw performance but the peculiar characteristics that enliven any operating environment, such as a user interface or access to database tools or local subroutine and driver libraries (which will become an increasingly important asset, along with class libraries).

And here we come to the fundamental problem of the current obsession with interoperability as portability: It militates against optimization not just across systems and vendors, but across time -- in other words, it retards progress. The Mac, maintaining compatibility with the system announced in 1984, has failed to bring forth much fundamentally new software after its initial blossoming of graphics-intensive packages (which continues but may be on the wane). OS/2, by maintaining partial compatibility with DOS, has saddled us with an archaic file system, although we see signs of innovation both in applications designed to support the new multi-user networked environment of LAN Manager and to replace the file system with shells and front-ends. NeXT, with a totally new environment, is likely to provoke the largest outpouring of truly new software, as (we hope) will ON Technology's software platform. (See also the article on groupware, page 16, for some thoughts on appropriate applications for networked environments.)

Finally, although it is an old system, UNIX may be born again as a new platform that will support new applications; this transformation will be based on the addition of a new user interface (or several!) and a new marketplace of users and vendors who will treat it as a new system, rather than a home for existing scientific/engineering applications.

Recursion to the future

In the future, the specifics will change but the issues will remain constant, as operating environments offer an increasing amount of high-level functionality. Even as environment vendors attempt to differentiate themselves with superior functionality, users and software developers will have to measure those features against the sacrifice of portability -- much as ISVs have done recently to use the Mac's graphical interface. As Sybase has shown, you can fairly easily port a UNIX application onto the NeXT machine -- but that's the equivalent of porting a DOS application onto a Mac. The DOS-on-Mac application lacks the Mac's excellent graphics; the UNIX-on-NeXT application will deny users the extensibility and built-in ability to use NextStep class libraries that native NextStep applications will offer.

In the future, then, portability and compatibility across machines and even operating systems will become irrelevant, in part as CASE tools and cross-compilers take care of implementation details. This level of interoperability will be achieved, but it won't be very interesting anyway. The new cross-platform interoperability issues will be information models, (graphical or logical) object interchange and the groupware equivalent of cross-database integrity (or distributed database); that is, can the completion (or complete backout) of a transaction be guaranteed across multiple discrete databases? And can the completion of a unit of work be guaranteed through a process involving multiple workers on several discrete systems across periods of several days? (If Juan requests a loan at the bank, will he get a response within the three days the ad promises even if loan officer Alice goes on vacation the second day?)

Interoperability and cooperative processing

At its highest level, interoperability is the antidote to the failure of compatibility, portability and scalability to live up to their promises. It's accommodation to the differences and advantages among systems rather than obliteration of them. If you keep avoiding differences you'll never move forward to the next (NeXT?) machine.

Interoperability (like airline safety) is a two-sided responsibility, because it requires the presence of both parties. It is made easier by openness, yet another vaporword. Because interoperability is a result, not a technology, it can be achieved in many ways -- starting with compatibility and portability (innate or through recompilation). Those are conditions that foster interoperability.

Other measures include real-time intermediary facilities, such as protocol converters and gateways, and special-purpose languages such as SQL, which enable limited-domain interoperability: One application or database management system can call on another for data specified in SQL or slightly more ambitious schemes such as Apple/Network Innovations' CL/1 language. With the appropriate interpreter or database on the target server, they en-

able a client to initiate processing on an incompatible system -- that is, do cooperative processing.

Most notably, the features of IBM's Systems Application Architecture support interoperability: Common languages, databases, application and user interfaces, as well as CASE tools, communications facilities and subroutine libraries. SAA is an implicit acknowledgement that interoperability depends not just on operating systems but on an entire operating environment. A fundamental element of SAA, the APPC/LU 6.2 protocols, for example, manages transactions across heterogeneous systems -- or cooperative processing.

Cooperative processing can occur within a single machine or across a wide-area network (as long as appropriate facilities are in place). The vision underlying cooperative processing is that systems should *not* be homogeneous: Certain locations, certain architectures, certain facilities are better suited to certain tasks than others, and so the work should be distributed accordingly. Data should be centralized or kept close to the user, *depending*; processing should happen locally or at an optimized database, inferencing or number-crunching system, *depending*; and so forth. Cooperative processing supports optimization rather than least common denominators.

To the user, however, cooperative processing is invisible; he doesn't need to know that his local application is communicating with another, separate application. To the application developer, cooperative processing may or may not be visible. If he uses a database management system that automatically transfers requests for remote data to another dbms, he may not have to deal with the cooperative processing issues himself since they are already handled by the dbms. Any client-server architecture is cooperative processing; terminal-host interaction is not because a single entity (the host) controls both systems. In essence, cooperative processing is interactive processing where both parties are computer systems (vs. interactive where one party is a person).

One of the more interesting implementations of cooperative processing is X Window, the MIT windowing standard, which consists of a client attached to an application that calls on a server (generally resident on the user's workstation in an arrangement/terminology that many people find confusing) to handle the display of the application screen output in a window on the user's display. The X Window server knows its local environment intimately and can optimize display of the data an X Window client sends it, and can handle input from multiple X Window clients (on separate servers or from multiple applications on the same server). See page 12 for more on window managers.

Openness in context

The confusion over all these vaporvirtues explains in part the confusion over openness, which is a prime determinant of the ease with which interoperability can be achieved.

When a platform such as the PC is so open (whether purposely or not) that it has become a standard, portability is easy. The target platform -- hardware configuration, database, operating system -- is so ubiquitous that any application built for it seems portable. In this context, openness means open to emulation: Anyone can imitate the interfaces the open product

presents to programmers and users (whether by reverse engineering or licensed or unlicensed copying), and replace the open system with his own.

But when openness means open to anyone's changes -- as was the situation with UNIX for years -- then portability diminishes, because seemingly identical platforms in reality are not. The same thing has started to happen to DOS, despite the pc market's attention to binary compatibility at the hardware level, because so many vendors have been "enhancing" DOS (which desperately needed it). Unfortunately, most of these enhancements are not interoperable or compatible, and have created a variety of nonstandard platforms that make it difficult to build PC software -- a situation Apple avoided by keeping the Mac closed. Do you support EGA or VGA? Do you need or support LIM-spec memory? Can you work within DESQview? What happens when you combine three TSRs? Can you run this off a 3Com network? Can you trade data with 1-2-3?

Openness of the third kind

By contrast, openness of the third kind is a matter of published, fully documented, stable interfaces that support interaction with other systems -- that is, interoperability. In essence, such open systems are a standard owned or controlled by a single party. When a vendor openly publishes the interfaces to his product -- the data structures it uses, the calls it accepts, the calls it makes -- other vendors or users can easily build complementary systems to interact with it or enhance it. (Of course, they can also build systems that emulate it, which in turn will foster even more complementary systems...and before you know it a clone market will be born.) Such openness fosters the extensions and enhancements that can create new standards (although if you get too many, as above, the result is chaos rather than a single new standard).

The goal, then, is to balance openness with a reasonably complete implementation, so that only geniuses feel called upon to improve upon it -- whereas anyone could improve on DOS, dBASE, UNIX or other standards that now look woefully incomplete. Obviously, this is in part a matter of timing. What looked complete in 1986 now looks pitiful.

A grand unified theory of vaporvirtue

The basic fallacy behind the current fascination with vaporvirtues is the notion that there will one day be a standard that we can all adhere to. (See **Release 1.0**, 88-6.) The underlying question is, "Who will win?" Rephrase that as "Who will live forever?" and its fundamental wrongheadedness becomes clear. Some standards live longer than others, that's all.

* * * *

JUAN AND ALICE IN THE U.K.

JUAN: They gave me a blank look when I asked for an org chart.
ALICE: Well of course they did. We call them organograms here.

SOFTWARE ACROSS SYSTEMS: HUNTER'S PROMISE OF PORTABILITY

While we're on the subject of portability, let's consider Hunter Systems, a start-up whose immediate goal is to help vendors and users port DOS applications to UNIX environments, thereby making compatibility less of an issue. When he founded Hunter Systems two years ago with money from his share in Hunter & Ready, a real-time operating system company, Colin Hunter expected to solicit venture capital funding the following year. As it happens, customers such as Motorola and Sony liked his concept so much that the company has sustained itself on upfront contracts. Shipments of XDOS, a two-part conversion tool, are starting now to OEMs, with end-users getting their piece early next year. Rather than promise the moon of automatic portability, Hunter provides conversion assistance with a tool that makes porting almost painless, and a distribution scheme that makes sense. The result is applications that can run concurrently and usually much faster, based on performance of the target machine. UNIX system vendors of all persuasions can now offset a competitive disadvantage by providing access to DOS applications and focus on getting new, competitive-advantage UNIX applications.

Half the problem of portability is getting software to run on a variety of machines. The other half is getting the right versions to the right customers, without carrying (or forcing resellers to carry) excessive amounts of inventory. Hunter Systems addresses both problems by providing two different products -- a software analysis tool, the XDOS Analyzer, which prepares an application for conversion, providing an application-specific "key file" for use in the conversion process; and a machine-specific compiler, XDOS, which can be used separately by OEMS, software vendors or end-users to recompile the application plus its key file into a new machine-specific version of the application.

That means that hardware vendors, resellers or customers themselves can create machine-specific versions of applications as needed -- and as licensed. Typically, Hunter hopes, hardware vendors will sell XDOS to their customers, who can then purchase software and key files from software vendors or resellers. Alternatively, the hardware vendor can also manage the distribution of the applications to save customers the trouble of conversion, which is about as complex a process -- to the user -- as installation.

XDOS can't do much without a copy of the original application and the corresponding key file, so by controlling distribution of the application (to say nothing of the key file, which Hunter would also prefer to handle), application vendors can control distribution of the software as effectively as they do now. (They simply avoid the burden of creating copies for specific UNIX machines. The number of copies is the same, but they aren't tied to a particular UNIX machine until the last possible moment at a reseller or customer site.)

Pricing, distribution, terms and conditions are up to the individual vendors and resellers involved. The legal issue is not what it appears, since copyright law (some shrink-wrap licenses to the contrary) specifically permits users to make a (single) backup copy and to modify software to execute on their individual machines. The key files come from Hunter at this point, but the user buys his own copy of the application binary. Even though Ashton-Tate among others has said it has no objection to Hunter's distribution of key files (for about $50 each), this venture won't be a success

without the enthusiastic participation of the application vendors one way or another -- if only to promote the availability of their software under UNIX and to promise to support it there, and ideally to take on the creation and distribution of key files.

Motorola and Sony have already signed up as OEMs; the trick will be to get the Top-10 software vendors to sign up. Theoretically, they should jump at the chance, but they're mostly noncommittal as yet: One software vendor we talked to said that "of course no vendor is going to admit that they're using a conversion routine to do a port. Nothing's as good as hand-coding." But that misses the point: The Hunter approach is not automatic; it is computer-*aided*. Essentially, XDOS is a compiler that takes as input a hand-built .exe application file and a key file hand-built with the help of the Analyzer, a decompilation tool.

Yes, software vendors *ought* to be delighted, since Hunter gives them a relatively painless way of extending their reach, while it gives hardware vendors access to a larger base of software that their customers want. Nonetheless, application vendors dislike anything they perceive as distancing them from their customers. Hunter is talking to a number of brand-name software vendors, but the best it can eke out for the record as yet is this comment from dBASE product manager Eric Kim: "Although we have not committed to specific A-T products at this time, we consider XDOS to be an excellent method for addressing the broad range of hardware configurations that make up the UNIX market." In general, a large vendor who liked the idea would probably prefer to use the key files and XDOS in-house and sell the results in a shrink-wrap, even if it meant missing a few machines it could otherwise reach. That might not make sense, but it's reality -- though it could change with proper leadership. For smaller vendors, however, wide availability through XDOS could be a competitive advantage, both with customers and with distributors who won't carry inventory for several UNIX machines but could sell applications and key files to owners of XDOS machines.

Where it works

Unlike most conversion systems we read about (but never see in use), the Hunter system is not magic. It involves a fair amount of work, and so far it converts DOS programs only to a variety of 68000-based UNIX environments, with the 88000 and the 386 promised. That's not a broad range, but it's credible -- and serves the most enticing immediate market. Hunter says that a DOS-to-VMS system is likely; a DOS-to-OS/2 system would address a less pressing need (since both environments are already based on the same machine architecture). The Analyzer's key files are not machine-specific, and could ultimately be used to convert programs to any platform for which Hunter builds an XDOS converter. XDOS holds the promise of letting UNIX applications port easily from one UNIX platform to another, theoretically obviating the Archer/OSF fuss and creating a virtual binary standard. But for now XDOS reads DOS .exe and .com files only, not UNIX coff or a.out files.

How XDOS works

The original software, a DOS application with or without graphics, is run through the Analyzer, a powerful tool that parses and analyzes the structure of the application and creates a giant directed graph of its execution flow and data structures. (It deals with object code only, and is thus language-

independent.) This mathematically represented graph lets XDOS optimize across the entire program rather than simply translate granular commands and code sequences piece by piece. "Analysis" is not a batch process, since the Analyzer discovers anomalies that must be assessed and resolved by an experienced programmer. The process generally takes a couple of weeks for a program the size of, say, 1-2-3. *The question is not whether XDOS "works," because it will always work given an effective key file. The question rather is how hard it is to build the key file, and how much extra a good programmer can put into the key file to enhance the program as it is converted for its new environment.*

Some of the anomalies are flat-out bugs; others are quirks peculiar to DOS software that should be eliminated by the conversion process. One of the most notable, for example, is that many DOS packages, designed for single-user and single-tasking environments, are very "active" and continually poll the keyboard to check if the user is typing anything. In a single-user, single-task system this behavior makes sense, but in a multi-user or multi-task environment it wastes resources that could be put to better use. Instead of checking with the keyboard frequently, the converted software waits in suspended animation for interrupts passed on by the operating system.

As for memory management, the conversion process eliminates a program's swapping of its own code, but if the application's logic limits it to handling 640K of data, that isn't automatically flagged. (A clever programmer could engineer it out by putting appropriate data into the key file.) Likewise, some software may still display applications in a 25-line window, although a programmer could remove that restriction too through the key file. (Does the program say, "Display this on the bottom line," or "Display this on line 25"? The latter kind of specificity, repeated throughout a program, dramatically hinders conversion/enhancement.) These are just examples; the point is that converting software so that it will work is trivial compared to converting it so that it will perform well.

Graphics and I/O routines are handled somewhat differently with an environment-specific runtime library in each version of XDOS that's linked to the new application during conversion. In addition, write calls to the video RAM buffer in the original are retained and handled by a virtual video buffer, so that read calls to the buffer can be handled from that same virtual buffer. Thus the application gets the maximum benefit from its new graphics environment and still responds like the old one.

Once the application-specific key file has been created, the basic work is done: After this, everything is automatic. The DOS .exe file plus the key file are compiled automatically by XDOS, the second half of the Hunter system. The key file is application-specific; XDOS is machine-specific, and together they produce a new machine-specific version of the application, obviating the problem of the current hundreds of versions of UNIX.

Hunter Systems knows its place

In a world of vaporwords and vague, grandiose ambitions, Hunter Systems is a delightful surprise. It has a single set of products that accomplish a single, well-defined task. Yes, they will continue to grow and improve, but for now the company knows what its business is and how its products are to be used. It has carefully figured out the appropriate distribution channels

and discerned the role of and benefits to each player in the cycle. Yet for all that, the company is hardly austere or unimaginative: The fundamental technology within the XDOS Analyzer is the equal of some of the most powerful reverse-engineering CASE tools around.

Is this just a temporary market until people like us get our act together and build portable code? one of the software vendors wonders. The specific DOS-to-UNIX use of the Analyzer surely is, but it's probably the highest-value task to which it could be applied right now. Hunter knows that value lies in defining a problem to be solved as much as in its solutions. Other years and other specific solutions lie ahead. *(Hunter will be exhibiting in the Motorola booth at Comdex.)*

SOFTWARE ACROSS SYSTEMS: THE INTERFACE WARS

Even as we come to some standards for operating systems, the war is shifting to user interfaces. Here too crackpot theories abound and technical information is scarce. Perhaps the most important distinction is between specs, shells and toolkits. None of these three is an application interface: The work of creating that is still an exercise left to the developer, although his task may be made easier by the existence of specs, shells and toolkits. Specs are the specifications for how the interface should look and behave; shells are interfaces to an operating system which provide users access to system resources and may or may not follow those specs; and toolkits are programming systems that provide widgets (reusable interface objects) and tools to make interfaces for applications. To illustrate: A spec tells a developer how a menu should look; a shell contains some menus that launch various operating system functions; and a toolkit contains sample menus that can be turned into working menus with the developer's addition of a few parameters and options.

And then there's X Window, an implementation standard that UNIX systems generally use to manage resources (data files and applications) anywhere across a network. X Window isn't an interface at all, but a set of defined calls that applications can use to display graphical windows remotely or locally on a variety of workstations/terminals that support it.

In the DOS world, there are a variety of DOS shells (to be covered at length in a future issue) and a variety of interfaces, but few published specs or toolkits, except for Actor, a sort of Windows toolkit from the Whitewater Group. In the OS/2 world, Presentation Manager is a given -- not that it won't change slightly. (The conspiracy theory that Metaphor is an alternative to PM doesn't wash; Metaphor's "interface" is about representation of data structures and relationships and visual programming, not just the selection of icons on a screen to represent system resources.)

In the UNIX world, things are far less settled -- and more interesting. For starters, the Open Software Foundation is pledged to come up with its entry by year-end, although it has also promised it will use whatever "standard" X/Open supports. Founding OSF member IBM, meanwhile, has just bought rights to NextStep, more than just a pretty interface since it includes an interface toolkit and powerful programming environment (Objective-C and application widgets or objects) as well, and has made no promise to sell whatever

OSF comes up with[1]. Hewlett-Packard, by contrast, has said it will adopt the OSF standards and has already stopped or redirected work on a couple of projects in conflict with OSF's direction.

Presenting on UNIX

Microsoft and Hewlett-Packard are working together to create versions of Presentation Manager for UNIX (although the positioning is a little shaky given HP's commitment to OSF and Microsoft's commitment to OS/2). HP could make fine use of this, since New Wave is based on Windows/PM and it plans to extend NewWave to UNIX; it would be nice if both versions could use the same interface. Microsoft supports it grudgingly. (Why make UNIX look any better than it has to? seems to be the general attitude. But on the other hand: Let's have PM all over the place!)

In fact, the project will have several implementations. The first is CXI (Common X Interface), available today, a set of existing facilities bundled together under that name and offered to OSF in the interface sweepstakes. CXI consists of a Presentation Manager style guide for behavior (feel) and two implementations of the PM look -- one like the OS/2 original and one with thickness, so that active buttons appear slightly raised (with a light source in the upper left corner). Selected buttons look pushed-in, while inactive buttons look flat. This is basically an X Window server with a Presentation Manager interface, running under UNIX.[2] As with any X system, the application code can run locally or across a network. CXI enables UNIX developers to build applications with PM-style interfaces using widgets created with XLIB calls, and lets them run it on any X server machine.

Sometime in 1990 will come PMX, a toolkit targeted primarily at OS/2 developers who want to run the same applications in both UNIX and OS/2 without rewriting their PM interface code. PMX will let them preserve the interface calls and recompile only the hardware and OS-dependent code of their OS/2 applications when moving to UNIX (using Hunter's XDOS, perhaps?). PMX is a much more ambitious project than CXI, since PM's graphics model is fundamentally different from UNIX conventions. PMX and CXI widgets will look and behave alike, but their underlying technology will be different. (For starters, PM assumes local control of the screen. A user will be able to run a local PMX application under UNIX along with an X server that can run remote or local X application, but he cannot run a remote PMX application because PM's imaging model assumes local control.) PMX consists of the PM applica-

---

[1] *IBM may look unfocused to smaller companies where direction is set by one person. Within IBM, the SAA and UNIX camps are fervent believers in their own strategy, so it's hard to talk about "IBM's strategy." It has two. Within each, it is toying with a several interfaces, waiting to see which takes off. Remember, this is the company that gave us the PS/2 and the RT PC, the 8100, the 4300 and the 9370, etc. IBM believes in a competitive world -- because it owns one.*

[2] *By contrast, Control Data U.K. subsidiary Systime offers PC-XVision, which enables DOS and soon OS/2 users to host an X Window window on their systems under Windows, with a PM version due out soon. Needless to say, you need the appropriate communications facilities; the PM window is in fact a UNIX session running remotely and displaying on the OS/2 PC used as a display terminal. Users of PC-XVision could run CXI X-based applications, but their server is running under DOS or OS/2.*

A WINDOW ON X WINDOW

X Window is a standard (freely available from MIT's X Window Consortium and through OEMs) for implementing window management either locally or across networks. It has become the foundation of much of the interface work in progress, although there are few applications for it as yet. (Any UNIX text application can already use X Window through an application called xterm that comes with the X Window tape.) X comes in two parts: a set of XLIB calls that an X-based application can use to communicate with any X Window server, and the servers themselves, which are implemented by vendors for specific machines and graphics environments. (Consider XLIB the equivalent of an SQL for display functions, and the X Window server the dbms that can execute them. Multiple applications can share the X server, appearing in multiple windows on the display that it controls.)

What renders X Window so confusing to the casual eye is that the client XLIB calls can come from anywhere, but the X Window server generally sits on a terminal or workstation where the display functions are executed. A number of vendors are offering terminals that provide graphics-rich X Window servers. They include Acer Counterpoint (8086, low-end, MCGA, under $1000, with a custom version of Locus Computing's Xsight) and Bridge-3Com spin-through Network Computing Devices (68000, high-resolution, $2-3000). These two systems devote all their resources to display and do not actually execute UNIX applications, which remain on a remote system and use the terminals only for display. (See also page 6.)

A second confusing item is that X is a "window manager," but the definition of window is very broad. Windows can be nested inside other windows ad infinitum, and they can take on arbitrary shapes (such as buttons, boxes containing icons, and so forth). In fact, X Window is a display manager that interprets a display as a set of areas (windows) that can accept or display data -- text, graphics or mouse events. Its appearance depends on the capabilities of the system hosting the X Window server, plus the application that executes them through the XLIB calls. These applications may use XLIB directly, or via reusable widgets built with toolkits that use the X Window intrinsics -- graphical objects, fonts, sounds and functions such as handling mouse events, clipping, sizing and arranging objects and linking objects to executable code.

Sun has had an alternative to X Window, NeWS or Network-Extensible Window System, but it has recognized the emergence of a standard and is building a new, compatible system called NeWS/X11.

tion programming interface recoded from assembler to C, and implemented on a variety of UNIX systems. Each one will require a specific implementation, unlike those for CXI, which can take advantage of X wherever it appears.

Within two years the project will reach a third phase, of interest primarily to Hewlett-Packard, with a UNIX version of NewWave wrapped around both CXI

and PMX. Objects built under NewWave with this toolkit will be transferrable from UNIX to OS/2 within NewWave. ("Hey!" says UNIX Alice to OS/2 Juan, "send me that diagram showing how the health club will be laid out.")

On the Sun/AT&T side, there's an interface spec for Open Look and the beginnings of a shell, plus two toolkits in development, one for NeWS/X11, Sun's merger of its window system with X, and one for X Window itself from AT&T. AT&T's Open Look X toolkit is scheduled to ship in the first quarter, while Sun has promised two: View2 or SunView2, based mostly on X, for developers porting from or familiar with Sun's existing SunView toolkit, in the second quarter; and the NDE toolkit, a new toolkit based on NeWS and PostScript, for NeWS developers and licensees, in the third quarter. (SunView is an interface-cum-implementation, whereas NeWS works at a lower level, like X, providing window and network management but no interface.)

Hello UNIX world!

Shells for UNIX play a bigger role than DOS shells do in the pc world because UNIX users have tended to be sophisticated programmers rather than end-users of applications and also because much of the work accomplished on UNIX systems is done by concatenating modular operating system facilities rather than by running a discrete, separate application. Traditional shells for UNIX (the Bourne or C Shell) are closer to programming languages than to interfaces.

While we wait for Open Look toolkits and the OSF- and X/Open-blessed new look & feel standards for UNIX, several vendors have gone ahead and provided graphical shells for UNIX. Note that these are shells for UNIX itself rather than tools with which to build UNIX application interfaces. Accordingly, they're easy to use and install, but they don't fundamentally change the applications "within" them; they just make them easier to get to.

"User-friendly" shells for UNIX include Directory Shell from American Management Systems and the Navigator from ParcPlace Systems. Both help users manage their files until they can get into an application, and provide substantial assistance for systems administrators as well.

Directory Shell (a mostly OEM product listing from $495 to thousands for mainframes) has a character-based "graphical" interface: That is, it displays directory trees on UNIX terminals, but it doesn't have the icons and pretty things that distinguish the new generation of UNIX systems; on the other hand, it does help the user through a variety of UNIX systems administration tasks as well as help him launch applications. This spring AMS will release another product, Looking Glass (starting at $595), with icons and all the requisite widgets as well as the systems administration functions of Directory Shell. Directory Shell customers include Arix, Harris Corp., Honeywell Federal Systems and, yes, Softsel, now in the UNIX market.

The Navigator has bit-map graphics (see **Release 1.0**, 88-9), and provides not just a navigational tree for end-users to get at their files but also a graphical representation for system administrators of everything on a system -- users and resources as icons, permissions shown in dialogue boxes, etc. It was built using a Smalltalk-based toolkit (the Navigation Framework) that can also be used to build application interfaces. The Navigator for UNIX has been licensed to Ardent; other vendors are considering the use of Framework to build customized interfaces for their machines or applications.

Both ParcPlace and AMS, as well as IXI (below), promise that they will support whatever becomes the interface standard, but none of them wants to lobby for any particular one as yet. Of course, that standard will probably already have its own UNIX shell and toolkit vendors, but other implementations are welcome to compete.

X.desktop on top of X Window

Finally, there's X.desktop^{tm} from IXI of Cambridge, England, a shell for OEMs to resell with an object-oriented customization toolkit attached. Either they or their more sophisticated customers can use the toolkit to add file types and icons, change system or local behavior and otherwise create custom environments for small customer sets or within a company.

Because it runs as an X Window application, X.desktop is easy to install on any UNIX systems with an X server. X.desktop represents files and other system objects as icons. It can group files into folders (mapped into UNIX directories), rename, copy, move or link them, launch applications, and so on. A naive user can use the system as is; a sophisticated user can add his own icons and specify their behavior, or change the system's behavior overall. For example, you can specify that if you drag one icon onto another, the first should run on the data in the second, start up a printer, or whatever is appropriate.

X.desktop manages a list of all icon types and their associated behavior. For starters, each icon represents a file -- system, program or data -- or system resource such as mail and printers. X.desktop looks for identifying extensions or ownerships or type indications when a new file is created so that it can properly link it to an object class. Does that sound object-oriented? It is -- and you can create new icons that share the behavior of their parents, or that override with their own individual rules. (Rules include display instructions and what an object does do in reaction to a variety of mouse actions, what happens when one is dropped onto another -- fore example, an application executes using the data contained in the file dropped onto it -- and appropriate error messages.)

The system comes as a working shell with a set of default icons and behaviors that the builder-user can alter or extend to taste. Hardware vendors who want to enrich their systems can do so easily with X.desktop and an X server. However, X.desktop does not provide an easy way for application-builders to use its graphical objects, which are basically designed to be linked to files and file-level operations. For that, there are products such as CXI/PMX and Navigation Framework, above; Chips from Virtual Machine Corp., **Release 1.0**, 88-9; and the not-yet-shipping Open Look toolkits.

X.desktop customers include Locus Computing Corp., who will license X.desktop to certain customers of Xsight (its X Window implementation), BiiN, Olivetti (on products not yet announced), Compaq and Acorn Computer. A predecessor company, Torch Computers Ltd., had a license agreement with NeXT for some of what inspired the IXI technology, but NeXT has made no explicit use of either version in its final product. IXI was founded by Ray Anderson, a former Cambridge student who has worked for several other Cambridge companies including Acorn, Sinclair and Torch.

FREE ADVICE: BE SPECIFIC

It's an old story: "Where shall we go?" "Out." "What shall we eat?" "I don't know; anything, I guess." There's nothing more annoying -- or common -- than a user who doesn't know what he wants. That's why restaurants (and software programs) offer menus; why travel agencies package vacations; why florists suggest arrangements for birthdays, weddings, product shipments and other joyous or tragic occasions. As development tools get more powerful, it will be the selection of functions and product positioning rather than implementation that will distinguish the successful application company of the future. Examples of clear positioning and focus for products that incorporate technologies far more capable than the tasks they're used for include XDOS (page 8), Zeno (page 19) and DesignView (page 20) in this issue. Each solves a specific problem; users who need one of these know that they need it!

We recently spent half a day with a start-up that has built a wonderful, powerful, all-singing, all-dancing personal programming tool that lets an end-user do anything he wants. How to market this thing? Wouldn't users really like to program if only they could? But what can you really *do* with it? You can build your own environment, you can create little applications, you can do graphics, you can manage files, it's awesome. But how do they position it?

In the end, most users want task-specific software. The advice we offered this company was to imagine that someone had given them this product: They had not just spent two years painstakingly building it, but had instead received it in the mail. (Just like a free C compiler but easier to use, albeit less powerful.)

So what should they do with it? Should they resell it as is, or should they build something with it? "We could build PowerPoint in a week," one of them pointed out. Yes, but then they would have to document it, advertise it and compete against Microsoft, Symantec, Cricket, Adobe and Aldus. Timing matters! They could make a DOS shell, or a PC version of HyperCard. They could build a sales-management system for lead-tracking, or an automated hair-dresser's appointment book... In other words, what should they build? And how should they position and market it? That's the tough problem. Of course, they could sell the product as is and leave that problem (and the opportunities) to their customers.

Groupware: what's in it for us?

As it happens, we did have a specific piece of advice: Build a groupware-building tool. Why?

For starters, this company's stated purpose is to exploit the next generation of platforms (read: OS/2 with Presentation Manager). To our mind, the salient product of this new generation is LAN Manager. We've had a widespread graphical user interface since the Mac came out; we've had multitasking and multi-user on UNIX and minis. The new generation revolves around networks and the related customer base: office workers using networked workstations, people who share work with each other (that is, they're not clerks who interact only with a transaction-processing system, and they're not lone contributors who work on their own). Server-based tools to

monitor the execution, delegation and completion of the work performed by the network's various users will comprise a major network-specific application category that has no existing powerhouse to dislodge. The category is currently almost empty because it's new, not because it's worthless.

Groupware is nothing new, of course. COBOL whizzes were building groupware on mainframes long before we were born. What's different is primarily three things:

First, groupware is now active rather than passive: It doesn't just *let* you share information; it actively reminds you what information to share (or forms to fill in, or items to review, or reports to complete).

Second, the groupware logic is separate -- reconfigurable independent of the applications and users it manages just as, say, database application logic is separate from the data and data structures it manages. This makes it easier for the groupware tool-user to focus on the particular part of the system he's concerned with, without getting all mixed up in the content of the work or the execution of the applications themselves. That is, groupware lets you describe what should happen -- who does what when, concentrating on the sequence (with conditional branches and cross-dependencies, etc.) without getting mired in the details of each step. In the same way, a factory management system worries about moving the work-in-process along, independent of the functioning of each work cell and machine tool along the way. (From the techie's perspective, groupware is the eighth layer on top of the seven-layer networking model -- physical network, communication session, presentation, application, etc.) The separation of task flow and content can be seen most clearly in a variety of image-based systems, where it's clear that the software is not dealing with the images directly, but merely controlling the movement of the images and some related data from worker to worker through a process such as loan approval, insurance claim evaluation or newspaper production.

Third, there will be tools to build it, so that people who understand the work flows don't need to understand computers as well. Those tools will comprise the new groupware category; many of the systems they build will be classified as applications such as (group) software engineering, (group) publishing systems, (group) sales-lead management systems, etc.

What it does...

So, this mystical tool needs to know enough about applications to launch them properly. It probably has some of the flavor of NewWave, in that it treats data and the associated application as a single object, but where NewWave typically thinks of its scripts as agents for *individual users*, groupware is its own agent (that's why it's active) that acts on behalf of the group (or perhaps the group leader). Although technically it could be built some other way, logically the groupware system resides on the server and monitors the activity of all the users. It is probably built around a database management system, but it appears to users mostly as mail messages or system menus.

To be effective, a groupware tool should provide an easy, intuitive way for the user-builder to model the workgroups, with each individual's name, roles, functions, relationships to other individuals, authorities and re-

sponsibilities; the applications, data and other resources required; and the work flows, who does what and in what sequences, including what happens when a person or necessary information is missing or when unusual events occur.

The power of the groupware tool lies in its ability to help users model their work processes and to implement them effectively. The tool should let users specify and automatically implement in the resulting system the answers to questions such as: What determines the sequences of steps to be followed? How much time is allowed between steps? Can several people perform certain functions when others are unavailable? Since most of the actual work is probably carried out by a number of network-resident applications, what are those applications and how can they be invoked? (A good groupware tool will probably have some knowledge of dBASE, 1-2-3, Paradox and a variety of word-processors.) What text should the system display at various points? (The system may have defaults: "<NAME>, you have <NUMBER> items left in your work queue," or "Here are the unfinished tasks, listed by priority.") How much choice is left up to the individual users? Can users bid for tasks or ask each other to hurry up? Does the system ask the users questions, or is it smart enough to make routing decisions depending on the data they enter into the applications they're working with?

Both the groupware tool and the groupware application accomplish real work -- one in building the system and the other in making it run smoothly. Just as a factory transport system accomplishes real work even though it never directly works on the goods it carries, groupware plays a vital part in the office "factory" by moving the goods through the office production lines.

(Note: Used loosely, "groupware" can also refer to a broad range of information-sharing tools, e-mail systems, and plain old multi-user database transaction systems designed for use by several people. The difference is that these systems manage the information but don't actively monitor the flow of work from person to person or ensure its completion. For more on this, see our June 1988 issue.)

NITTY-GRITTY EXPERT: RULES FOR REPORT CONVERSION

Artificial intelligence will succeed commercially when it becomes as exciting as the miracle plaque-removing ingredient in toothpaste. That is, its presence will be used to explain how a product achieves a particular benefit, but will not be considered a benefit in itself. Such is the case with Zeno, an exceedingly down-to-earth PC product that uses AI (in the form of a parser and syntax analyzer) to produce customized programs that translate mainframe-style reports into PC formats such as spreadsheet (.wks, etc.) or comma-delimited files. (Much to our amazement, we couldn't find any other product that does this in a general way.)

Although many PC users may not realize it, most mainframe screens and reports are generated without tabs or other formatting commands, as a structureless sequence of characters and spaces. Getting them into spreadsheets can be tedious work: You can retype them, you can edit them and insert formatting commands, or you can build little conversion programs tailored to each kind of report or screen you work with.

As a consultant to Marine Midland Bank (where he used to work) and other places, Michael Flinder of Michal Flinder & Associates watched his clients struggle with the problems of transferring mainframe-based production/ transaction data into PC-based analysis tools. Flinder's solution was Zeno -- an expert system that automatically builds a set of rules to run a conversion program just like the ones people now build by hand.

To use Zeno, you train it by example: You give it your file and use the first screen image or block of data as a prototype, pointing out the fixed and variable elements and selecting from a short set of options (fixed or variable, numeric or alphabetical, required or optional, etc.) as appropriate for each. Obviously, such a process need only be performed once for each kind of recurring report, as the rules are incorporated into a customized conversion program specific to that particular report format.

Zeno can handle tricky formats: For example, one beta customer we talked to, information systems manager Andy Jezioro at Dominion Bankshares, gets mortgage reports from a service bureau with four lines of data per customer. He needs just five fields out of a possible 20 in those four lines transferred to a single row per customer for a 1-2-3 spreadsheet. It took only five minutes to set up the program, carefully ignoring page numbers, headings and other red herrings that appear irregularly from the data's perspective. It now takes about 15 minutes to produce a 1000-line spreadsheet two or three times a month (vs. a horrific typing job subject to errors!).

Zeno isn't very smart: It can handle only files that adhere rigidly to a specified format -- but that's what computer reports are all about. In other words, Zeno is not a general-purpose program that can figure out the structure and formulas behind a table such as, say, Datacopy's OCR Format+ from Avalanche Development, nor an image-analyzer such as Calera's PagePro ($1000) which identifies fields on forms by precise location according to a user's template, but it can create a program to convert any regular report structure you can describe by example. Not fancy, but think of all the typing -- and mistyping -- it can save for $269!

NITTY GRITTY EXPERT: DRAWING BY THE RULES

Computer-aided design. Helps you design things, right? In fact, most PC-based CAD packages help you draw something after you have designed it. Mechanical engineers tend to use calculators and pencils to design things, and then CAD packages to record the results nicely. It's like using a spreadsheet because it lines up the rows and columns so evenly! But that's all most PC CAD packages can do. They're still in the direct-manipulation stage: What you draw is what you get. Now a new generation of packages applies rules or constraints to models, much as the new generation of page-layout packages applies rules to pages, so that the user's mind and not just his hands can be extended.

Such a new package is DesignView from Premise, a start-up founded in 1987 by MIT CAD laboratory manager (and former student) Jon Hirschtick. Its capabilities are not unique, but they have not hitherto been available on a PC (to our knowledge), even at $1895. DesignView, shipping this quarter, fits into a 640K PC (although it can use extended memory) and runs under Windows, which enables it to share information with a word-processor for creating reports or other documents. While automating design is the more difficult task, communicating the results is also necessary.

Where most CAD packages help you to draw exactly what you like with specified dimensions -- make this beam 6 feet long, for example -- DesignView lets you start with a rough sketch with specified constraints, i.e. ratios or dimensions. For example, beam A should be twice as long as beam B, and it must always join beam C at a 45° angle. DesignView itself then creates the drawing, to scale and in accordance with the specs. Change the ratio between A and B, and it will recalculate and redraw the design in seconds.

In short, DesignView allows you to do what-if? for mechanical (two-dimensional) models, using what computer scientists call constraint-based modeling. It captures both the final drawing and the design requirements, which can be easily changed to try out other solutions or measure the impact of a changed constraint.[3] DesignView's constraints can be geometric (a circle must be tangent to a certain line, for example), dimensional (size) or calculated (one block must be twice as wide as another, and a certain angle must be maintained). The system shows and understands construction lines (see across), which the user can use to guide himself and to communicate constraints to the system or to other viewers. Using the facilities of Windows' Dynamic Data Exchange, constraints can be linked to data in other DesignView drawings or spreadsheets stored in separate files.

DesignView is written in C, with an object-oriented approach to representing the design elements and the rules and constraints that govern their behavior

---

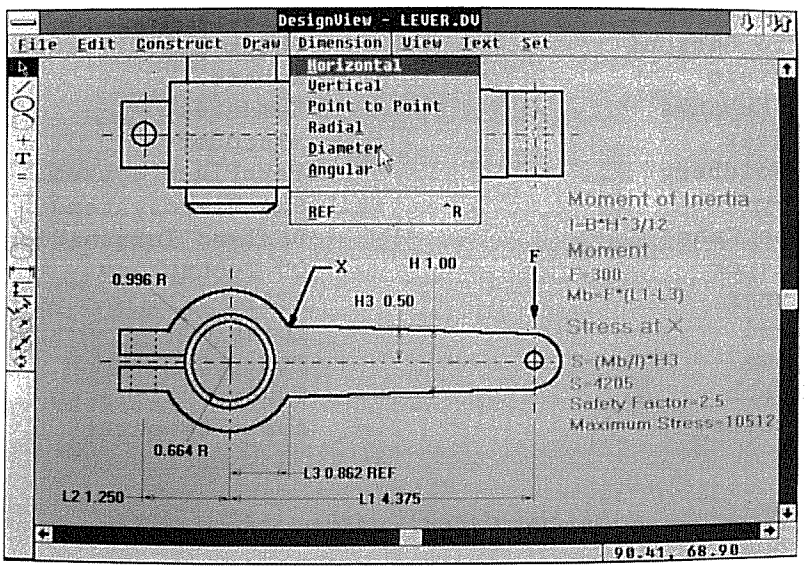
[3]*Some other design packages, mostly for CASE, also apply constraints or logic to drawings, but most of those have to do with objects and logical connections between them rather than with physical dimensions, or what Premise calls "dimension-driven variational geometry." In addition, some CAD packages use "procedural" rather than "declarative" constraints, so that dependencies are instantiated sequentially -- first draw A, then draw B one-half of A, then make C perpendicular -- rather than as a set of relationships that must be maintained. You can't change B and have it change A because A came first.*

and parameters. The system is entirely vector-based, and knows enough geometry and trigonometry to calculate relevant values and draw the resulting diagrams. The package can't handle 3D constraints explicitly, although a smart thinker can use it to handle certain problems involving extra dimensions, such as calculating weights and densities. You can specify, say, a moment of inertia and it will back-calculate to find a beam's cross-section.

In perspective

DesignView is part of the trend to using smart graphics (see **Release 1.0**, 88-9) that the computer actually understands, rather than images with meaning only to humans. Hirschtick, who has also worked for Computervision, built the product around ideas formulated by Ivan Sutherland and David Gossard, associate professor of mechanical engineering at MIT and director of its CAD Lab. But, he notes, "We had to solve some significant engineering and design problems to make those ideas happen" -- especially in 640K.

A similar but more powerful product, Mechanical Advantage (including the SketchPad variational geometry module, named after Sutherland's original SketchPad program), is sold by Cognition of Billerica, MA -- whose founder, Bob Light, wrote his thesis with Gossard at MIT. But Mechanical Advantage costs from $7500 to $10,000, and requires a VAX or a Sun. HP has also just announced a similar product, the Variation Design Module, as part of its $7500 (on a Vectra PC) ME Series 10 CAD line. (The pricing algorithm is complex, but that's what it amounts to.) The system will need UNIX when it ships in January, but a DOS version is promised for April.



DesignView from Premise automates not just the drawing but the calculations (at right), which refer to elements in the model at left. The construction lines around the model represent the dimensional constraints. By contrast, most so-called design packages don't automate the design process, but the process of representating the design after the engineer has figured it out with pencil and paper (or calculator). DesignView allows mechanical engineers to join financial analysts (and HP) in asking what-if?

1989 PERSONAL COMPUTING FORUM: GET SET FOR THE NINETIES

The Personal Computing Forum will take place a month later than usual in 1989 (no nefarious reasons, just the availability of suitable hotel space), from *March 19 to 22*. We are moving back to our old haunts, Palm Springs, in search of good weather and morning alertness among West Coast attendees. We will mail invitations to subscribers as soon as we get back from Comdex, so don't worry yet. (If you get your registration back to us by year-end, you should have no problem getting in.) The Forum will be managed by associate publisher Daphne Kis, who has just joined us from Home Office Computing. Come meet her and circulation & fulfillment manager Lori Mariani at our Comdex booth, number 157 (Main Hall).

Please come and celebrate in Palm Springs!

Our preliminary roster of confirmed speakers includes:

Tanya Amochaev	Natural Language Incorporated
Gordon Bell	Ardent
John Seely Brown	Xerox PARC
Vittorio Cassoni	(back at) Olivetti
Bob Epstein	Sybase
Gordon Eubanks	Symantec
Bill Gates	Microsoft
Adele Goldberg	ParcPlace Systems
Bill Joy	Sun Microsystems
Philippe Kahn	Borland International
Mitch Kapor	ON Technology
Bob Kavner	AT&T
Jim Manzi	Lotus
Bill Lowe	IBM
John Roach	Tandy
John Sculley	Apple Computer

Note the dates: March 19 to 22!!

RESOURCES & PHONE NUMBERS

Pauline Lo Alker, **Acer Counterpoint**, (408) 434-0190
Jay Wettlaufer, **American Management Systems**, (703) 841-6021
Ed Esber, **Ashton-Tate**, (213) 538-7714
Bjarne Stroustrup, Paul Fillinich, **AT&T**, (201) 522-6664
Haviland Wright, **Avalanche Development**, (303) 449-5032
Joan Haber, Steve Dow, **Calera Recognition Systems**, (408) 986-8006
Bob Light, Robert Rescigno, **Cognition**, (508) 667-4800
Jim McNaul, **Datacopy (Xerox)**, (415) 965-7900
Sean Murphy, **Hewlett-Packard (design)**, (303) 229-4881
Nick Fowler, **Hewlett-Packard (interfaces)**, (503) 750-2242
Colin Hunter, **Hunter Systems**, (415) 965-2400
Bill Filip, Lee Reiswig, **IBM**, (914) 686-1900
Ray Anderson, **IXI Ltd.**, U.K. (0223) 462131
Ben Faul, Gerald Popek, **Locus Computing**, (213) 337-5112 or 670-6500
Dave Liddle, **Metaphor**, (415) 961-3600
Michael Flinder, **Michael Flinder & Asscociates**, (716) 693-0584
Paul Maritz, Mark Mackaman, **Microsoft**, (206) 882-8080
Judy Estrin, **Network Computing Devices**, (415) 694-0650
Steve Jobs, Conall Ryan, **NeXT**, (415) 424-0200
Alex Morrow, Henry Crouse, **Open Software Foundation**, (617) 621-8700 (note: this is a new number; the entity has moved to Cambridge)
Adele Goldberg, **ParcPlace Systems**, (415) 859-1000
Barbara Schultz, Peter Norton, **Peter Norton Computing**, (213) 319-2000
Jon Hirschtick, **Premise**, (617) 225-0422
Susan Morgan, **SoftView**, (805) 388-2626
Barry Folsom, **Sun Microsytems**, (415) 960-1300
Bob Epstein, **Sybase**, (415) 596-3500
Chris Holmes, **Systime**, U.K. (0532) 529292

COMING SOON...

- *Visual programming languages.*
- *Graphics standards.*
- *A database for document management.*
- *U.K. trip report.*
- *DOS shells and file managers.*
- *Active and passive objects.*
- *And much more... (If you know of any good examples of the categories listed above, please let us know.)*

RELEASE 1.0 CALENDAR

Some interesting events at Comdex:

Tuesday, November 15

8.30 am — *Software Horizons panel.* After the keynote panel, the rest of us: Fred Gibbons, Software Publishing; Bill Campbell, Claris; Mort Rosenthal, Corporate Software; Fernand Sarrat, IBM; Barry Folsom, Sun. Sarrat and Folsom are in charge of finding and caring for ISVs for IBM and Sun, respectively. They are particularly looking for exciting new applications, not warmed-over ports. Convention center, room N.

4.45 pm — *Microcomputer Managers Association meeting.* To discuss 486 standardization and other matters. Sponsored by MMA, a group of corporate buyers. Convention center, room M2-4.

Wednesday, November 16

Noon — *Hewlett-Packard and Microsoft announce CXI.* See page 12. At the Riviera Hotel.

And the rest of the year:

November 28-December 2 **International conference on fifth-generation computer systems** - Tokyo. Sponsored by the Institute for New Generation Computer Technology (ICOT). Contact: Ami Semba, (011) 813 456-3195, or Hideo Asiso, (011) 8144 631-141.

November 30-December 2 **Small business/home office conference** - Key Biscayne, FL. "Building effective channel strategies to reach the market," with Alan Hald, MicroAge; speakers from Apple, IBM, Sharp, Canon, Toshiba, Ameritech. Sponsored by CAP International. Contact: Kristin Fischer, (617) 982-9500.

December 5-6 **The personal computer outlook** - San Francisco. Mostly for investors, sponsored by Technologic Partners and Merrill Lynch, with Rod Canion, Bill Gates, Bob Kavner, Bill Lowe, Jim Manzi, Scott McNealy, Mike Moritz, John Roach and a host of others. Contact: Dick Shaffer, (212) 696-9330.

December 5-7 **Strategic issues forum** - Cambridge. With Eric Bush, Richard Carpenter, Rick Crandall, Michael Scott Morton, Patty Seybold, Pat Winston, Esther Dyson, others. Sponsored by Decision Support Technology. Contact: Donna Kacin, (617) 354-6400.

December 5-8 **CASExpo** - Anaheim. Sponsored by Arthur Young; chaired by Howard Yudkin, president of the Software Productivity Consortium. Contact: Rhoda Canter, (202) 956-6041.

December 5-8 **Document processing systems** - Santa Fe. Sponsored by ACM. With CD-ROM, hypertext, SGML, CALS, and object-oriented databases, documents are getting exciting! With Gerard Salton, Esther Dyson, among others. Contact: Rick Beach, (415) 494-4822.

1989

January 17-19 **Computer graphics New York** - New York City. Targeted to end-users. Sponsored by Exhibition Marketing and Management Co. Contact: David Small, (703) 893-4545.

January 23-27 **Improving productivity in EDP systems development** - Phoenix. Sponsored by Applied Computer Research. Contact: Gonzalo Verdugo, (602) 995-5929 or (800) 234-2ACR.

February 13-14 **LaST Frontier Conference** - Tempe, AZ. On software as intellectual property, with nine law professors thinking deeply, out loud; chaired by Milt Wessel (former counsel for Adapso). Sponsored by Arizona State University College of Law. Contact: Rosalind Pearlman, (602) 965-2124.

February 13-15 **The software re-engineering symposium** - Boston. Sponsored by Digital Consulting Inc. With Rich Currier, Panoramic; others. Contact: Dan Horgan, (508) 475-6990.

February 14-17 **Software development '89** - San Francisco Airport. Sponsored by Miller Freeman, with speakers including Bill Gates, Philippe Kah, Larry Tesler, Dick Gabriel, Terry Winograd, Bjarne Stroustrup, Ed Yourdon. Contact: KoAnn Tingley, (415) 995-2471.

February 28-March 2 **UniForum** - San Francisco. Keynote by Terry Lautenbach, senior vp and gm, IBM USA. Sponsored by /usr/group. Contact: Ed Palmer, (408) 986-8840.

March 6-10 **Fifth IEEE conference on artificial intelligence applications** - Miami. Contact: IEEE, (202) 371-1013, or Mark Fox, (412) 268-3832.

March 13-17 **Seybold Seminars '89** - San Francisco. The place to be published...er, seen. Contact: Kevin Howard, (213) 457-5850.

March 14-16 **Interface and World Congress on Computing** - New York City. Moved from Chicago, in search of more enthusiasm. Sponsored by Interface. Call Walt Heithaus, (617) 449-6600.

March 19-22 **EDventure Holdings Personal Computing Forum** - Palm Springs, CA. With the first annual EDwards for EDventurous performance. See page 22. Contact: Daphne Kis, (212) 758-3434.

March 28-30 **Fourth CD ROM conference** - Anaheim. Sponsored by Microsoft. Contact: Sherrie Eastman, (206) 867-3305.

March 28-30 **AAAI's Spring Symposium series** - Stanford, CA. AI in many guises, including manufacturing, language, software engineering, knowledge systems, planning and search. Sponsored by AAAI. Contact: Rick Skalsky, (415) 328-3123.

April 10-12	**The software re-engineering symposium** - San Francisco. Sponsored by Digital Consulting Inc. With Rich Currier, Panoramic; others. Contact: Dan Horgan, (508) 475-6990.
April 10-13	**Spring Comdex** - Chicago. Also including MACdex. Contact: Terry Catchpole at (617) 449-6600 or (800) 325-3330.
April 12-14	**International markup conference** - Gmunden, Austria. Sponsored by Graphic Communicatons Association. Contact: Marion Elledge at (703) 841-8160.
April 18-20	**Connect '89** - Boston. Sponsored by Cahners Exposition Group with Larry DeBoever and Dale Kutnick. For MIS managers and other victims. Contact: Dave Sell at (203) 964-0000.
April 30-May 4	**CHI '89: Conference on human factors in computing systems** - Austin. Sponsored by ACM/SIGCHI and a host of other groups. Contact: Claudia Raun, MCC, (512) 338-3798.
May 9-11	**Second international conference on computer-assisted learning** - Dallas. Sponsored by U of Texas' Computer Learning Research Center and other groups. Contact: Janet Harris, (214) 690-2204.
May 14-17	**Adapso Management Conference** - San Diego. Mingle with your peers (and the zoo's nearby just in case). Sponsored by Adapso, with the usual round of business-focused sessions, networking and speeches by luminaries. Contact: Sheila Wakefield, (703) 522-5055.
June 11-14	**Expert Communications '89** - San Francisco. Sponsored by Graphics Communications Association. Contact: Norman Scharpf, (703) 841-8160.
June 20-24	**PC Expo** - New York City. Sponsored by PC Expo. Contact: Steven Faher, (800) 444-EXPO.
July 17-21	**CASE 89** - London. Sponsored by Index Technology and a host of academic groups, including London's Imperial College. Contact: Elliot Chikofsky, (617) 494-8200.
July 31-August 4	**SIGGRAPH '89** - Boston. Sponsored by the Association For Computing Machinery. The annual festival for visual, graphical thinkers. Contact Cindy Stark, (312) 644-6610.
August 9-11	**Conference on object-oriented dbms applications** - Santa Clara, CA. Sponsored by Santa Clara University. Contact: Mohammed Ketabchi, (408) 554-2731 or mketabchi@scu.bitnet.
August 22-26	**IJCAI-89** - Detroit. The international version of AAAI. Sponsored by the American Association for Artificial Intelligence. Contact: Claudia Mazzetti, (415) 328-3123.

August 24-September 1 **Eleventh World Computer Congress** - San Francisco. With a focus on tools and application software this year; held in the U.S. for the first time in 24 years. Sponsored by 46 IFIP member societies. Call Nancy Dana, (303) 696-6100.

September 7-10 **Comtec '89** - Singapore. The third regional microcomputer exhbition. Organized by ITP Services and the Microcomputer Trade Association Singapore. With 71,000 vistors (28,000 professional, trade or business) in 1988. Contact: Yong Mee Hiong, Singapore 2913238; fax 2965384.

October 1-4 **Adapso Management Conference** - Orlando. Mingle with your peers (and Disneyworld's nearby just in case). Sponsored by Adapso, and always tremendously valuable. Contact: Sheila Wakefield, (703) 522-5055.

October 10-13 **Info 89** - New York City. Sponsored by Cahners Exposition Group. Contact: Frank Fazio, (203) 964-0000.

November 13-17 **Comdex** - Las Vegas. Also including MACdex. Contact: Terry Catchpole at (617) 449-6600 or (800) 325-3330.

Please let us know of any other events we should include. -- Denise DuBois

---

Release 1.0 is published 12 times a year by EDventure Holdings, 375 Park Ave., New York, NY 10152; (212) 758-3434. It covers exotic software, including CASE, groupware, text management, cooperative processing, connectivity and artificial intelligence. Editor & publisher: Esther Dyson; associate publisher: Daphne Kis; circulation & fulfillment manager: Lori Mariani; executive secretary: Denise DuBois; consulting editor and copy chief: Bill Kutik. *Subscriptions cost $395 per year, $475 overseas.*

SUBSCRIPTION FORM

Please enter my subscription to **Release 1.0** at the rate of $395 per year in the U.S. and Canada. Overseas subscriptions are $475, airmail postage included. Payment must be enclosed. Multiple-copy rates on request. Satisfaction guaranteed or your money back.

Name _____

Title _____

Company _____

Address _____

City _____ State _____ Zip _____

Telephone _____

How did you hear about Release 1.0? _____

_____

88-11

Please fill in the information above
and send with your check payable to:

EDventure Holdings Inc.
375 Park Avenue, Suite 2503
New York, NY 10152

If you have any questions, please call us at (212) 758-3434.

Daphne Kis
Associate Publisher