POWER THROUGH SCRIPTING

Forget the killer app; look for the killer lib! Someday people will be using libraries of application components (objects) instead of applications. The only difference they'll notice is greater ease of use and integration. This is pretty far off, but the underlying technology is moving in that direction as application builders attempt to define "events" -- interfaces to their applications to make them scriptable. "Events" are a sort of precursor to objects -- defined procedures and data without a class hierarchy.

Scripting -- automating the execution of interactive applications -- is attracting attention with the recent announcement of UserLand's Frontier and Apple's Open Scripting architecture, a somewhat backhanded endorsement by Apple of Frontier. HP's NewWave deserves credit for having had similar facilities in operation for some time.

This newsletter puts scripting in perspective and considers the issues: What is necessary on the part of the applications to be scripted? How can it be made easier for users? In order to build user scripts we need not only powerful languages such as Frontier and HP NewWave's task language but also scriptable applications that adhere to standards such as Apple's Open Scripting or NewWave. For users, we also need scripting tools such as Apple's Eager, ObjectVision and mail tools.

Forget programming and new killer apps; what users really want now is a way to get the full power out of their existing applications. To do this, they need tools to do two things: to program, or script, their applications to do routine tasks automatically, and to use the combined power of multiple applications. Users are now discovering that interactivity amounts to baby-
sitting. They don't want rich interaction with their computers all day; they
want to give the computer instructions to handle routines and exceptions, and
and do the important stuff that only humans can do: Motivate, negotiate,
persuade, inspire, teach, schmooze at the water cooler (or the fax ma-
chine). And they want one application to drive another, instead of having to
make the switch themselves from one to another.

The reason for a language to automate applications rather than an end-user
language such as BASIC is that applications have already done much of the
necessary work -- defining the kinds of data and functions users want to
manipulate: i.e., Western region sales, the memo to Alice about Juan's promo-
tion, the skills section in a resume, the first paragraph of each chapter,
making an appointment, the sequence of steps in following up on a sales call.
What are the associated data items, the associated actions?

But what is frequently lacking in applications is clear, published ways (such
as Apple events or NewWave commands) of expressing the functions and objects
managed by each application, so that outside agents (scripts, users, other
applications) can control their execution. Once these are explicitly estab-
lished, the script-builder has some hope of describing how he wants them per-
formed, in a language that is neither formal English nor computer jargon, but
application jargon -- a set of defined terms related to the business he un-
derstands, meaningful both to the person and to a variety of applications.

Technically, what we're describing is object-oriented nirvana, but there are
ways to approach there from here without getting immediately to the paradise
promised by PenPoint and Patriot. What Apple calls events, for example, are
going close to objects without the underlying model: They are defined ac-
tions on defined kinds of data. As applications become class libraries, the
distinction between script languages that drive applications (Frontier) and
script languages that manipulate objects (presumably AppleScript controlling
Apple events) will disappear. (In a sense, we're talking about passive ob-
jects with central rather than local control.)

Without interapplication communication, you miss the major
benefit of multi-tasking: You can run more than one applica-
tion at a time separately, but you can't run them together, to
get the greater-than-the-sum value of joint operation. That's
why commercial application developers, typically developing one
application at a time, flocked to Windows, while systems inte-
grators and corporate in-house developers, trying to integrate
applications and build systems that could cycle data through
more than one function at a time, were quicker to appreciate
the value of OS/2 and its more powerful, robust multi-tasking.

The story so far...

During the Eighties we discovered direct manipulation and interactivity.
With a word-processor you could move chunks of text around, paste in boiler-
plate, create footnotes. With a spreadsheet, you could do even more --
select cells, add them together, build formulas. Desktop publishing lets
you move pieces of text around on the screen, select fonts and type styles
from a menu, and so on. Calendar programs let you make appointments by
typing them into the right place, and HyperCard lets you go from card to
card by clicking on buttons. (And you can create new buttons by copying old
ones.) The trend reached a height with GUIs. Want to throw something out?
Simply put it in the trash. Send it to a friend? Put it in the outbox, and
select his picture from a gallery of images of your friends' faces.

Meanwhile, frequent users were discovering macros. Rather than repeat a
routine sequence of keystrokes for using 1-2-3, for example, a user could
write a macro. Clicking a HyperCard button might do something a little more
interesting than finding another card: run a little script, for example, and
people were writing those little scripts. They were also building log-on
scripts with communications programs and using recorders to capture a key-
stroke sequence and play it again and again. The problem with the recorders
is that they're dumb; it's hard to make them do things conditionally, or to
recognize different objects: You don't really want cell 24G; you want sales
for the Western region, regardless of how the spreadsheet layout changes.

Even macros work only within a single application. Right now, we have bur-
geoning standards for data exchange, but not much more. Object linking and
embedding, cutting and pasting, publishing and subscribing, involve data
transfer and application launching (so the user can do something), not gen-
eral interapplication communication.

As for scripting, DOS has its batch programming language and REXX to do OS
functions and launch applications, and an increasing number of applications
come with their own macro languages to run, but there's no standard cross-
application tool. Nor can there be until the applications follow some stan-
dards and publish their interfaces. The only serious DOS/Windows-based
scripting environment, HP's NewWave, hasn't yet gained critical mass. The
Macintosh, with its wonderful user interface, has had no scripting tool un-
til now, and few Mac applications have macro languages, given the difficulty
of building scripts for inherently graphical applications.

Let's get real

The first thing to understand is that there won't be any global user pro-
gramming language, just as there's no global mechanic's tool. We have ham-
mers, screwdrivers, pliers, saws.... In business, we have people who know
purchasing, people who know car design, people who can write legal briefs
and people who do numbers. Yes, they all speak English, but have you ever
heard Alice the lawyer trying to talk about work to Juan the artist at a
cocktail party? ("Hey, lady, I'm just a runtime litigator myself!")

No one of the tools described here could be -- or could even grow to be --
the answer to everything by itself. You don't need to solve everything all
at once, but you do need to let all the solutions communicate. Thus, we see
a growing collection of mostly complementary functions that can work togeth-
er. Alice the lawyer and Juan the artist may have some problems in common.

The ultimate goal is not English or any natural language; just try to ex-
plain a network system or a tax rule without using technical terms. Techni-
cal terms exist because the things they describe exist. What we're reaching
for, however, is making the things in the computer relate to the user's
world -- whether it's tax rules, loan criteria or sales regions -- and not
to computer elements such as files, records and disk volumes. While the
interapplication communication languages and application commands and data definitions manage the actual functionality and the use of the applications' power and data, the other tools described here (ObjectVision, Eager, mail tools) deliver that power to the user.

But beware a misleading term: The scripting languages are not glue, which sticks applications together. Rather, they are two-part systems, with the scripting tools playing the role of the sticks in Tinkertoys™. The sticks allow you to connect arbitrary nodes flexibly -- but only if the nodes are built with the right holes, i.e., application events the script can use.

Imagine each application as a factory. You have to go from plant to plant to use each application. If you insist, you may take some data with you, but it has to be in containers to be moved. Direct manipulation gave us the ability to go into a factory with our data and operate the machines and data in each room. User programmability gives us the ability to set the machines up to do a job while we're away. Object-orientation and modularity of functions and multi-tasking put all the equipment at a single site (working from the same engine to save energy and space), so we don't need to go from plant to plant. We can use any machine at any time, with all our data in the same place; we can also sequence the machines as appropriate.

Different tools for different folks

The scripting tools described below fall into two camps. There are builder-user languages such as Frontier and NewWave's agent language, which let experts build scripts for users, and user tools, which help users build scripts for their own use, including NewWave in its recorder mode. Obviously, this distinction is clearer in theory than in practice. (And separately, there are programming languages, such as Visual Basic.)

The scripting languages are a means to operate applications in combination, using one application (seemingly) to drive another. But whatever the common language or at least syntax to manipulate applications, how do users get at it? Not by learning Frontier or REXX, normally. Instead, there are lots of possible interactive front-ends to such a language that can help a user build scripts rather than write them from scratch: forms (typical for database applications), programming by example (recording with editing, as in NewWave, or with sequence-recognition such as in Apple's Eager), flowcharts (ObjectVision and AllClear), and rules (Beyond Mail and WiJít). All these tools let the user program at one remove, by using the capabilities of existing applications (or in the case of ObjectVision, a database server).

They range widely in the extent of user expertise required and can be used at different levels; they have different approaches to different problems. But they all exemplify the trend to users' gaining more control over and power from their applications by explicitly representing what they want rather than by interacting continually.

For scripting to work, it needs applications with clear, defined interfaces and clear, defined data elements -- events, procedures and objects, to use
object-oriented terminology, which is unavoidable in this context. Events are powerful programming constructs. The more functionality they expose, the more power outside agents have to control things. To the extent that the applications allow and support it, Frontier and NewWave can do more than just drive them; they can examine the state of the objects they are managing -- how many paragraphs, what field names are available, where do certain strings occur, what's going on (be more specific), is the value for house price greater than the highest offer? -- and take appropriate actions to the extent that the script-builder specifies. Of course, there's the trade-off between power and details: How much do you really want to control?

Raw materials for the tools

Thus Frontier, NewWave and other scripting tools depend on the kindness of application vendors -- or on the support of platform leaders, such as User-Land has just won from Apple. While it's premature to hope for standards, application developers can get busy, and publish their own commands, following standard syntaxes such as Frontier's, NewWave's or REXX's. Why not?

How graphics gum up the works

Everyone knows that GUIs are good (possibly even environment-friendly, since they're natural). But making a system "fuzzy" also makes it harder to manipulate indirectly, through commands from other applications or over time. All this is a fancy way of saying that most GUI environments and applications lack a solid scripting language. You can record movements, but what if the screen layout has changed? You want screen-independence. This problem is most egregious on the Mac -- which is easy to use but almost impossible to program. Within Windows, many programs at least have a DOS heritage (for better or worse) and some vestigial text orientation.

In the old text-based applications, things were straightforward: Users typed commands or filled in forms, and there was a fairly linear sequence of commands, parameters, and interaction to follow. With graphics-based programs, life gets much more complex. The UI and the functionality get mixed together. There are lots of ways to make something happen: Use the mouse, select something, move it somewhere; or use a sequence of keystrokes; click on something; make a menu selection; etc. All these get translated, ultimately, into explicit events, but (unlike Apple events) they're low-level: for example, "mouse-up," which can have hundreds of meanings depending on the context. (Add a whole new set of these for pen-based gestures.)

(For more technical details, see page 8.)

---

1 A related problem is that of people who can't interact through a GUI because of physical impairment; they can't discern images on the screen, move a mouse or manipulate a keyboard. A text/command-based system can read its content to the user and interpret commands delivered in a variety of ways, while a GUI system typically mixes up interface and function inextricably, making it inaccessible to people who can't handle a particular interface. Thus the text-oriented, explicit commands and object definitions required by scripting tools are a way for them to gain the power other users get with direct manipulation of less explicitly defined objects.
THE ROLE OF THE MAC

One of the subtler positioning pitches Apple is making for System 7.0 is that it's ideal for workgroup computing. In part, that's because of specific technical features, such as easy file-sharing (as much as security concerns allow) and networking. But a greater factor is less technical; simply, it's the homogeneity of the environment. What other vendors try to achieve broadly through standards bodies (although only half-heartedly, because the benefits must be shared), Apple achieves by controlling fully a narrow share of the pie. The virtues of this are fully apparent this year, when there seems to be utter confusion in the other sectors of the pie -- DOS-Windows-OS/2-LAN Manager-NetWare-and-the UNICES. (HP's Bill Crow notes that you can use NewWave to unify these disparate environments, but that just gives you the rim of most of the rest of the pie.)

Thus some interesting plumbing and groupware tools are appearing in the Mac world first (Pharos Whirlwind; Instant Update, Release 1.0, 4-91; Aspects, the Farallon line, etc.), since they are easy to build there. It's easier to build cross-workstation tools, since you can trust that the workstations on either side will have similar capabilities, whatever their individual configurations. You have to deal only with the primary issues, without the added complexity of variety. Instant Update, for example, is likely to be much harder to implement meaningfully for the Intel world.

More significant for scripting and end-user programming is Apple's Open Scripting architecture, an umbrella recently raised to cover the transition from the current procedural world and UserLand's IAC toolkit and its RPEs to Apple's Apple events and Object Model. The Object Model is a comprehensive class hierarchy that will contain all the core Apple events -- your basic OS, wp, database, spreadsheet, mail, and other common functions and objects. It will also be extensible (through proper object-oriented inheritance and subclassing) to cover any new applications implemented as class libraries.

Until the announcement of Open Scripting last week, developers were caught in-between, reluctant to work with Frontier to build RPEs for their applications and then have to abandon it all for the Object Model and Apple's promised AppleScript scripting language, still a long way off. But now the companies have informally agreed that each will support both approaches. AppleScript will evangelize Frontier for now, and later on Frontier will support the Object Model and Apple events as they evolve. (And if Winer does his job well and makes RPEs pervasive, Apple won't leave their users in the lurch, although it's reluctant to guarantee that AppleScript will use RPEs.) In effect, Apple has said, "Go ahead and use Frontier. It works."

Aldus PageMaker and QuickMail and QuicKeys from CE Software will support RPEs. Significant applications offering or promising Apple event support soon include Excel, MacWrite Pro and Resolve from Claris, Deneba's Canvas, SuperMac PixelPaint and Acius 4D. Of course, any application can support both.
USERLAND'S FRONTIER

Into the breach has come UserLand with Frontier, a scripting language for the Mac. While Apple was promising its Object Model (below), UserLand founder Dave Winer was working on a more near-term scheme for a scripting tool that could apply to the current range of applications or one step beyond. Instead of full objects following a common object model or even just Apple events (individual objects without the model), all Frontier requires is that an application define some RPEs (Remote Procedure Events), using UserLand's IAC (interapplication communication) toolkit, which works both within System 6 and System 7. RPEs are easier to define and carry less baggage than full-blown objects or Apple events, but they leave more work up to the script writer, since he can't merely say, for example, "format paragraph 4," but has to define how to look for a paragraph, and for the fourth one, select it, and then go into the formatting details.

Of course, some word-processors do already define paragraphs, either explicitly or as something separated from the preceding text by a return character. But it's not a universal object yet that fits in the Object Model Apple is proposing. In fact, Winer asserts and we agree, it's probably too early for a global object model. Vigorous competition among approaches will foster improvements even if the Mac world settles on Apple's in the end.

On the Frontier

Frontier is a language in which builder-users can build scripts with interfaces for end-users -- which the end-users can then use by clicking on buttons, filling in forms and selecting values. It also lets applications communicate. It has no keystroke recorder, working only with defined RPEs that it sequences and controls with branches, loops and conditionals. It uses a fairly simple C-like syntax, and is easily extensible from RPEs to Apple events. It has about 50 commands that work within the Mac OS, for things like copying, deleting and moving files, creating dialogue boxes, buttons, menus and the like, as well as creating aliases, a new System 7 feature. It lets a builder-user build menus for end-users, thus building interactive scripts. For example, it could list selected kinds or collections of files in a dialogue box for the user to select from. The user's selection would provide a parameter to a file-deleting script.

Frontier is not just a language, but a development environment with a runtime component and an outliner so you can see the overall structure of a script. This is no surprise; it reflects the mind of author Dave ThinkTank Winer, a visual thinker who sees structure everywhere. Frontier also has its own "object database" which stores variables used in the scripts, so that you can see what's going on and make changes effective across several scripts. The objects the database holds can be everything from numbers and dates to entire files, other scripts, or other tables. Basically, it's a way of keeping track of the objects scripts are likely to share or reuse. They exist outside the applications that created them. And of course a script in turn can manipulate values in the table.

Extending the frontier

The value of Frontier depends on wide use of Apple's Open architecture (RPE or Apple events). It and IAC and Apple events are an integral part of Sys-
tem 7, so that UserLand's timing is perfect -- a hard-earned coincidence! (NewWave, by contrast, was early to market with its agent task language; some successor to DDE looks to play a similar role for NewWave.) Meanwhile, AppleScript is still a long way off, and HyperTalk is a scripting language for HyperCard only, although it could be extended. For now, HyperCard's commands are not RPEs, let alone Apple events.

Winer's goal is to broaden Frontier to be a scripting language for a broad range of Mac applications -- thereby occupying the same position on the Mac that Microsoft wants to carve out for its Object Basic. Frontier also plays an interesting role for Apple in somewhat the same way Claris does: It holds the promise of interapplication communication across platforms -- where it will run into NewWave, and of course eventually Microsoft (unless it can gains Microsoft's support where NewWave couldn't). Although UserLand is concentrating on the Mac for now (a full platform), Winer is something of an evangelist and would like nothing better than to see RPEs and Frontier implemented everywhere. The scripting tool is fairly platform-independent, and could run Intel-based RPEs as easily as Mac-based ones. Says Winer: "OS/2 has adequate plumbing and multi-tasking; DOS has no plumbing. And of course UNIX has the plumbing and a culture of scripting, but it needs something like Frontier that normal people can use."

HP AND NEWWAVE: OLD KID ON ANOTHER BLOCK

Frontier is the new kid on the Mac block, but HP's NewWave was first in the whole neighborhood. Its DOS/Windows environment is an object-based system, built on Windows, that treats applications and associated data files as objects at a minimum, and provides a spec whereby applications can become NewWave-aware (basically, more modular and object-oriented). It can also handle DDE and OLE objects, giving it connectivity to a lot of non-NewWave aware applications, although not at the same level.

NewWave includes a recorder capability that can record, edit and play back the keystroke sequences even of non-compliant applications, including clipboard data transfer. To the extent that applications are NewWave-aware, the NewWave agent task language can manipulate them and their commands in a more precise and concise manner, just as Frontier can do with RPEs and Apple events. Compliant applications implement the NewWave architecture with two modules, the action processor and the command processor. The action processor watches the user's actions (within Windows) and translates them into commands; the command processor, the innards of the application, takes the commands and follows them. Thus, the action processor is interface-dependent, while the commands are interface-independent. This architecture means that a NewWave script deals only with commands, not with interface events -- and that it's easy to change the interface of an application.

At a minimum, NewWave records and plays back on the level of "select object, open it," rather than combining those actions into "open object." But if you are writing a script from scratch or editing it, or if the application has a smart action processor, you can express it in the more succinct form.

At this point, NewWave has the scripting tool that the Windows environment lacks -- but it still lacks a broad range of scriptable applications. The most notable NewWave application is Excel, whose macro language is also

Release 1.0 31 May 1991
usable by NewWave as NewWave commands. Also, Asymetrix Toolbook will support NewWave, providing an object-oriented interface to NewWave’s language.

Meanwhile, once an application has defined interfaces for DDE, it is a relatively simple job to make those applications NewWave-aware -- defining the interfaces as NewWave commands. (They're not DDE exactly, since the communication protocol, managed by the OMF, is different.)

NewWave also manages the linking of data and functions into objects, so that when data is transferred, the functions go with it. OLE does this too, but it does it at the application-to-application level, so that if the link target moves, an object is mailed, or an application is upgraded there may be problems. NewWave's Object Management Facility manages this process automatically, which makes it much easier to use in practice.

REXX -- SCRIPTING FOR DOS AND OS/2

Since a general scripting language depends on application interfaces it can use, there is no analogue in the DOS-OS/2 world. The likeliest prospect is REXX, a general-purpose operating system scripting language descended from a similar mainframe language developed years ago by IBM’s Michael Cowlishaw (see Release 1.0, 12-87). It is widely used in the VM mainframe world, but it has gained less momentum in the pc world, where BASIC and the DOS batch language hold sway.

Nonetheless, Quercus Systems has made a business out of a version of REXX for DOS, Personal REXX, developed by Charles Daney. (For a while, Mansfield Software Group was distributing it, but Quercus is about to resume marketing it directly.) Like OS/2, REXX is superior to many alternatives on its merits, but it has lost the marketing war so far. Personal REXX is an application-scripting and batch language with a good ability to communicate with applications and parse strings (which is what you have to do if applications don't supply nice, tidy objects). It's simple and clean, say developers who use it and love its power. On the other hand, it has no native text editor; and it offers no particular Windows support, although there will be a Windows version later this year from Quercus.

REXX is also now getting a push from IBM, which has bundled its own REXX/2 with OS/2 and is generating some interest from application software vendors who would like both to save the work of writing their own macro language tool, and the benefits of sharing a common syntax with a semi-standard. IBM is also publicly demoing Object REXX, which supports PM and an object model.
The end-user is more comfortable dealing with high-level concepts, such as those in his applications, but he wants the control he gets with objects....

Scripting on the spectrum

The lowest, but still useful, form of scripting is simple keystroke recording. Then there's interpreting commands from keystrokes or input events such as mouse movements. You can generate sequences of commands either through such event interpretation, or by writing a script, and you can then refine the script with conditionals and other logical controls. Finally, if the application defines its data -- or you can go in and grab it -- you can manipulate the application's data through your own commands. (This is a procedural way of looking at it; in an object-oriented world, you would add new methods to the object hierarchy. Then you or other objects could use those methods to manipulate the objects in question.)

There's a reason developing scripts is tough: You have to hit precisely the right trade-off between abstraction and control. The benefit of an application or object is that it takes care of the details you don't care about, and gives you an easy way to specify those that you do. It should also have sensible defaults. There's the difference between, say, "apply paragraph format," and a sequence of formatting commands. (We'll explore this next month in a discussion of several text tools and SGML.) This depends not just on the commands, but defining the objects that the commands work on. I.e., can you select a paragraph, or just a sequence of characters in a file? And is it a normal paragraph, or a caption or a special inset quote?

What did you mean by THAT?

Likewise, at one extreme, you can say "Do payroll." At the other, you can type in a sequence of letters, move the cursor down two lines, type a number.... Somewhere in the middle is the relevant level: "Select the table of employees; enter hours worked; change any pay rates if necessary; calculate overtime; transfer the results to the accounting program; credit payroll and debit cash; [do a lot of things too complex and scary to list here with taxes]; issue paychecks."
This (somewhat sanitized) is the level at which you can change the business process by changing events rather than having to wrestle with code. But does the computer recognize what's happening? It certainly can't by monitoring keystrokes alone. In fact, it doesn't have much chance of figuring this out unless the program defines these activities; Apple calls them events; HP calls them commands; the object-oriented world would call them rudimentary methods. In the server world, they are the equivalent of stored procedures. That is, they are defined, named functions that take parameters and that can be activated by a request from the "outside."

How they work, the order in which they accept parameters, how they handle variables, the way they fit together, all vary widely, which is why it's necessary to have standards to make them usable. The second reason for standards is that it's convenient for all applications to use the command "print" rather than some using "type," others "r" and yet others "pf" or whatever. There's obviously no way that each application will have the same set of commands -- a spreadsheet will have calculations and a calendar program will have appointments -- but if they have common functions, such as hiding details/outlining, or adding annotations/footnotes, it's handy if they use the same verbs. How the levels of detail are determined or the annotations are displayed may vary, but the concepts are the same.

OBJECTS FOR SCRIPTS: ALPHABET LIB

Much of the work involved in making scripting tools a success will depend not on the scripting tools themselves but on the application developers -- and on the development of standards across applications that will foster interapplication communication. The drive to interapplication communication and client-server architectures has started developers thinking about publishing their APIs (application programming interfaces) so that others can use them, and thus the trend is in the right direction.

The trick is to separate the user interface from the underlying functionality. Ironically, it's the "simple" functions that are the most trouble to discern and define. "Complex" commands, such as "do a fast-Fourier transform" are likely to be explicit anyway. Oddly, developers have worked hard to make their applications interactive, and may have put the interface and the functionality too close together. (Those who wrote with a view to porting, and kept the interface code separate from the functions, turn out to be better off even when they stay on the same platform. Score one more point for modularity.)

"Scriptability" is not all or nothing; a script can use as many functions as an application will offer, and beyond that a script can try to get in there and do things by itself. But the better the application interfaces, the less work the scripting tool needs to do. All it does is supply syntax and a control sequence, and possibly maintenance of variables, message dispatching, etc. In fact, there's no need to wait for standards for application vendors to do at least the first part of the task, which is to build their applications with clearly defined functions/events.

Scripting started as a procedural notion, but it works better and better as you move from applications and commands on data, to clear, defined actions on clear, defined data elements -- procedures and objects, in short. We had clear, defined commands to some extent before the age of interactivity, when
the only way to drive applications was through a command line, but neither
the commands nor the data were necessarily very usable from the outside.
More recently, graphical user interfaces sometimes mixed up the user’s ac-
tions with the application’s functionality, making the problem even worse.

With that in mind, here are some of the approaches towards defining applica-
tion interfaces for potential use by scripting tools (and any other outside
agents).

Objects and servers

Objects and application servers are ideal for scripting, and from the out-
side at least they look like the same thing. One feature of an object is
the precise definition of its capabilities, or methods, while an application
server has defined procedures that the client can request. The only way to
create and modify and manipulate each instance of an object or the data on a
server is through these defined methods. Thus, because you know exactly
what has been done to the data, you also know its "state." (In this view,
many database front-ends are macro languages for a database server.)

Eventually end-user scripting tools will end up with a variety of inter-
faces, suited to a variety of tasks. Underneath, they will drive both plain
object-oriented languages, and ones that take their objects from applica-
tions. Of course, the distinction between the environment and the applica-
tions will diminish. In the end, there will only be shared application ob-
jects and their graphical screen representations. Of course, this is the
kind of environment GO has built, and that Patriot intends to build, right
from the start.

Apple’s Object Model and Apple events

In the long run, Apple wants all Mac applications to adhere to its Object
Model architecture, basically a class hierarchy within which any particular
application would technically be a class library. (Apple is a member of the
Object Management Group and will probably support the OMG’s Object Request
Broker technology for communicating with other object models; see Release
1.0, 3-91.) The commands that will manipulate the objects are "Apple
events." Currently, there’s a core of about 40 such events defined; they’re
the standard items you see on most Mac menus -- open, close, delete, print,
select, and data-access routines such as the Data Access Language -- which
can be useful in scripts as well as interapplication communication. They
can act on a variety of user interface objects and text objects as well as
plain data defined by an external user or application or script. You can
use these events procedurally without the full Object Model.

The Object Model will impose (or offer, depending on your point of view) a
single class hierarchy, so that, for example, "paragraph" will be a common
item from application to application. That refers to the identity of a
block of text as a paragraph, not to its (default) format, which can vary
from application to application (and user to user); it’s like defining a
spreadsheet cell, with width, size, and other parameters independently de-
termined. In addition, the Object Model will offer (as does object-oriented
programming in general) elimination of redundant code, consistency from
application to application with variations due to choice rather than acci-
dent, and faster programming.

Release 1.0 31 May 1991
Thus, it would be very easy to build powerful, high-level scripts, consisting of objects' messages to each other. And Apple indeed also had its own vision of a scripting language, AppleScript, that would use the full Object Model as its foundation. But unfortunately most applications don't yet implement the Object Model; in fact, no well-known commercial ones do as yet, although Adobe's PhotoShop and others soon will. Apple's MacApp is an application framework and development environment that implements the model and helps programmers build compliant applications/class libraries.

UserLand's Remote Procedure Events

In the meantime, UserLand has defined Remote Procedure Events and a syntax for them. These are methods (verbs) that can act on an application's data, but without the full-scale Object Model hierarchy. They use the Apple interapplication communication protocol, a procedural protocol for how applications send and receive messages. In their ultimate form, RPEs will evolve into object-oriented methods, arranged in an object hierarchy.

"Supporting" either RPEs or Apple events means that an application lists the verbs it has defined and the kinds of data they act on in a listing in the resources section of its code file. Then an outside agent -- Frontier or other tools -- can use IAC and those events to manipulate that application. To "support" Apple events, an application must use a reasonable proportion of the 40-odd core events for common functions, and may add its own.

Publish and subscribe

For what it's worth, publish and subscribe" does not use IAC. The process uses shared data files -- and thus is quite different from Microsoft's DDE and even from OLE, which is generally one-to-one rather than one-to-many. An application "publishes" or defines and makes available, a set of data (ultimately, an object) that may be frequently updated. "Subscribing" application files contain a link that finds the latest version of the data and uses it to update its own information.

Dynamic Data Exchange

Dynamic Data Exchange, Microsoft's primary protocol for interapplication communication, is basically a protocol for one application to talk to another using a small set of ten expressions focused on defining and transferring data rather than executing application-specific procedures. For example, "send over the block of text highlighted on the screen, with formatting commands embedded." Or, "monitor the news feed for articles containing these words in the titles, and send them over when you receive them." DDE requires that both applications be operating. Moreover, it doesn't send results back to the calling script or application, but to the clipboard. This makes it better for one-time exchanges than for an interactive script which makes decisions on the basis of the answers it gets (and would have to keep looking at the clipboard). For example, a script might run different routines depending on where each paragraph ends on a page.

In addition, DDE has the execute command, which can pass a fixed sequence of commands and parameters to another application; it just can't interact easily. The interfaces defined by applications that use DDE could be a good
starter set for building the kinds of verbs one might use in a macro or interapplication communication language.

DDE, however, doesn't have an industrial-strength scripting language. That is, there's no independent script to call DDE functions, but rather another application. A scripting tool is needed to organize and control the sequence of events, manage the variables, and so forth -- all the things that Frontier, NewWave or REXX can manage, and that Object Basic will manage. Any conditionals or complex operations are a function of the individual application's macro language. (Eventually, too, the communications underpinnings of DDE will have to be strengthened.)

Microsoft's Open Tools strategy and its object architecture will likely extend DDE into the foundation of a richer scripting language, based on the use of common terms for functions and objects, and a standard class hierarchy (cf. Apple's Object Model).

Object Linking and Embedding

Object linking and embedding uses Dynamic Data Exchange, either to establish a persistent link to data that gets updated, or to embed an object into another document or data file. In a document, the imported object would typically be displayed or printed as part of the document; in a spreadsheet such as Excel, it would typically be represented as an annotation of some kind in a layer separate from the worksheet. The code to display or print an object comes with it to the document that it is embedded in, but the application commands to edit or modify it stay separate and can be found when the object is opened, rather than just printed or displayed.

In the case of linking, the target object has to be available, but its application need not be open, when the calling object needs it to print or display. (That way, it's assured of getting the latest possible version.) But if the user clicks on the object, it automatically loads the application that created it (if it can find it), so that the user can, for example, calculate on a spreadsheet embedded in a wp document.

Linking is generally used when the information is likely to be updated and the user wants to link to the latest version of it. The application support required is similar to that for embedding, except that the linked-in data keeps an independent existence and must be found each time the receiving application needs it. (Unfortunately, there's no foolproof way yet to ensure that it or its application will be there or that the receiving application can find it, if it has been moved, a task that is managed by NewWave's Object Management Facility or Apple's Finder for Publish and Subscribe, within limits.)
POWER FOR THE USER

The tools we describe here illustrate some of the front-ends that will become more and more powerful as they get better back-ends -- scripting languages and events. To call them scripting tools may be a stretch, but they are ways for users to make it easy to explain what they want from applications.

EAGER -- NOT JUST FRIENDLY, BUT HELPFUL

But what if the user doesn’t know what he wants? He knows his work is sort of repetitive, but he’s not sure how he would tell a tool what to do. Besides, it’s not exactly repetitive; the numbers are different each time, or there are criteria he can recognize but not explain. Not only does he not know how to express conditionals, he doesn’t even know how to express the conditions themselves.

Just as NewWave and some other systems have tools that can turn a series of low-level events and contexts into higher-level command sequences, so could a clever tool recognize a repetitive sequence of commands as a single task (with variations) worth storing and repeating. Apple’s Allen Cypher has built such a tool, called Eager. For now, it’s limited to the commands in HyperCard, and it’s just a research project, but the company has started to publicize it widely.

Eager is a combination action-recorder and pattern-recognizer with carefully developed ergonomics developed with extensive testing on actual users. That, and the difference between keystroke recognition and event recognition, differentiate it from Key Watch, which something like it for the pc.

The principle is simple: Eager is a process that watches the user’s actions and looks for patterns. Because it knows about HyperCard events, it’s looking not for identical low-level actions, but for series of similar high-level events involving HyperCard commands such as selecting several cards, using a particular button and entering different parameters following some pattern, or selecting all but the first word in the second field of a succession of cards and pasting each in order onto another card next to numbered buttons. When it thinks it has found a pattern, it doesn’t simply take control away from the user, nor does it ask if the pattern is correct: "Are you selecting all but the first word in the second field of a succession of cards and...?" The user himself may not have an abstraction or generalization of the task in his head; he simply knows what he is doing. (One example cited by Allen Cypher is the task of duplicating each card in a sequence. First you duplicate the first card, A, to get A,A; then you duplicate the second card, B, to get A,A,B,B; then you duplicate C to get A,A,B,B,C,C. The system, however, is actually duplicating every second card)

2. There are three papers about it: for the CHI conference, just held in New Orleans, and for forthcoming workshops on Human-computer interaction and visual languages in Moscow (August) and Kobe, Japan (October) respectively. Also, Dave Nagel showed a video of it at the recent Demo '91 conference. For copies of the papers, call Allen Cypher at Apple, listed on the resources page.
-- cards 1, 3, 5, 7... -- because by the time it gets to each card its number has changed. Try explaining this to a user -- or even to a reader of the paper!

Instead, Eager simply highlights the user’s expected next action -- for example, highlighting the last four words of five on the second line of the third card in the sequence. If the user indeed selects them himself, Eager takes that as confirmation. If he does something else, Eager knows that it guessed wrong. (The user does not have to stop his work to say yes or no.)

Eager’s approach minimizes the user’s work, but it doesn’t take the power away from him. It lets him assent to the system’s actions with a minimum of effort; then lets him click on Eager if he wants Eager to take over.

If the user clicks on Eager, represented as a green cat, Eager goes ahead and finishes the job. When you match patterns in this way, it may be hard to know where to stop, so Eager always stops at the end of the lowest group in the hierarchy; i.e., if you were deleting all the files in a folder, you might not want Eager to continue and delete files in other folders!

This system does have fundamental limitations, however, which could only be improved with a significantly more powerful expert system/pattern-matcher underneath. It doesn’t recognize conditional actions very well, for example, nor can it recognize high-level patterns. The trick is figuring out what the user has selected, whether he’s adding a line or creating a new paragraph, starting a new task because he’s finished the first one or simply performing a related subtask, starting a new page because he’s within 12 lines of the bottom, or because he’s beginning a new section. In some cases, Eager may just have to ask in order to translate an inexplicit sequence of actions into something more explicit. An advanced version, of course, could also let a user edit the script and add logic.

With a small amount of work, Eager’s pattern-recognition capabilities could be broadly applied to any application that uses Apple events, RPEs or other defined and visible command sequences. We’re looking forward to it!

OBJECTVISION -- NO OBJECTS, BUT NICE TREES

ObjectVision is one of the nicest database front-end tools around. Rather than a friendly query tool or an object-oriented interface that offers or supports hooks into a database, ObjectVision is a true application-development tool, generating scripts to run the Paradox engine or a number of other database managers, other applications (through DDE) or in some cases acting on the data directly (easy to do since it’s so well-defined).

For example, you want to automate the process of responding to someone’s request for back issues of your newsletter Release 2.0. (It’s desktop-published, but slow in catching on to trends.) An operator takes the call and selects the script from a menu. The system first asks if the caller is a subscriber; if he is and remembers his subscriber number, the operator types that in. Otherwise, the operator could type in the caller’s name, hoping to find the subscriber record that way. Give or take an initial, that should work. Now the rest of the screen automatically fills in. A box pops up to remind the operator to ask whether the caller’s data are still
correct. Perhaps there's a second box that appears if the subscription is about to expire, or already has -- and the system can branch into another application, receiving credit card information for a subscription payment over the phone.

But to get back to the main procedure, ordering the back issues...

The point of this example is that even a simple application can be quite complex, with little branches, subtasks and the like. Imagine the code to do all this, with look-ups and go-tos and if...elses. But the steps are nothing that the user-builder can't understand in principle; in fact, organizing and managing such tasks are the basics of his job. ObjectVision makes it easy for the builder-user to understand and design the process he intended by displaying the relationships and sequences of the tasks visually.

**Envisioning ObjectVision**

Note that it is not necessarily the builder-user's job to build or define the objects he uses (although ObjectVision can do so); those are held within a database, and typically defined through Paradox or Btrieve or dBASE. The user can select what he needs from a listing of fields displayed in a nice friendly way by ObjectVision. He can also call other applications through DDE. For example, if a customer called asking for back issues, but didn't know which ones he wanted, there could be a branch to a text query tool that could search the online file of issues by keywords, titles or a variety of other criteria to find exactly what the caller wanted.

This is the kind of set-up that would normally have taken a real, cola-drinking programmer to create -- but it probably would never have been built. Put it this way: We haven't done it yet, but maybe with ObjectVision we'll get around to it.

Under the covers, ObjectVision is attaching little event-driven decision-tree scripts to each field -- thus making them into intelligent "objects." It is a fine example of the sort of hybrid tool that will enable users to leverage the power of their applications, combining not just control over the structure of a program through trees but the modern traditional use of forms to build database access. When there's more than one thing to do, it makes sense to take more than one approach -- and to combine them elegantly.

**ALLCLEAR -- THINKING TOOL**

For working in the other direction, there's AllClear from Clear Software. If you don't necessarily think visually, or if you have a sequence of steps already laid out (in a manual, say), you can derive their structure by putting a carefully worded (not coded, but following some minor rules of syntax) version of the steps through the AllClear parser, which will generate a tidy flow chart. Forgot to specify what happens if the subscriber wants the back issues sent to another address (but doesn't want to change his subscriber record)? AllClear (or ObjectVision) will probably show you not the loophole, but the hole in the loop.

AllClear isn't a code-producing tool, but it's a valuable thinking tool -- and technology that may well be incorporated by some canny toolmakers. (See Release 1.0, 10-90.)
MAIL TOOLS

One way to get users to program or script is to make it so easy, as Eager and ObjectVision do, that you've lured them into it before they realize what they're doing. Two examples are Beyond's Beyond Mail and Agility/D&B's WiJit. (They get mentioned a lot because they're the only commercial ones announced so far, but even they aren't yet shipping.)

Rather than explain what they do (see Release 1.0, 11-90 and 10-89 for that), let's just consider them as scripting tools. Both have the two components -- a control language and events, or a way of communicating with other applications -- although they're presented somewhat differently.

Fundamentally, both allow the user to define sets of rules and conditions to execute a variety of procedures in response to events or information -- mail messages -- from the outside.

In addition, they provide support for building defined interfaces to applications so that users can then construct their own applications using those defined interfaces. Each can define specified information that the tool can parse, such as the presence of text strings in fields, or fields in a full-fledged form. These mail messages are the equivalent of events, although for now they're typically defined by the vendor as interfaces to public data services or by an in-house builder, rather than by the applications or services themselves. (Of course, the events can come from another user as well as from an application.) A programmer or builder-user also builds the events/commands that the user or his automated agent can send back to an application, with parameters filled in. This is the real work; after this, it's easy for the end-user. Certainly, to the extent that applications provide easy-to-use, published commands, it will become easier to use Beyond Mail and WiJit to control them. (WiJit, a Windows tool, will use DDE from the start, but it is not yet clear when or in what form its new owner, D&B Software, will ship the product.)

These mail tools are a means for users to control and automate interaction with external applications, although they don't quite qualify as scripting tools proper. Their function is not just an information filter, but potentially a powerful way of letting users define and implement sequences of actions in response to information. While there will be a number of "user-scripting" tools, these products' initial positioning as mail tools make them less frightening to individuals, and their client-server architecture allows users to interact easily with other users and applications, extending the user's control easily beyond the desktop.

MICROSOFT: BASIC AND BEYOND

In the Intel pc world, besides NewWave for Windows, there are a batch language for DOS, Personal REXX, numerous keystroke recorders (many with logic and editing capabilities), macro languages for a variety of applications, and lots of end-user programming tools, but no clear, visible analogue to Frontier (which of course itself will only start shipping this fall).

To fill gaping hole, Microsoft is now promising an object architecture for Windows (and presumably beyond) which will serve as the basis of a class
hierarchy that will help applications/class libraries communicate, and support a scripting language, in the future. But for the moment, the closest Microsoft comes to a global standard for interapplication communication is Dynamic Data Exchange. No one as yet has proposed a language built around DDE, probably because it's not really suited for that. However, most full Windows applications (as opposed to applications that run under Windows) will support both DDE and OLE, useful for creating compound documents. Many application-specific macro languages also use DDE, so there is a de facto way to get some multi-application scripting, but no common tool for it. It's likely that current DDE commands specified by applications may form the foundation of Microsoft Basic events in the future.

Microsoft is now moving towards such a goal. Visual Basic and WordBasic are both early in a sequence that will culminate with Object Basic, a global object-oriented language, of which they will be a subset, says Microsoft. For now, Visual Basic is what Microsoft calls "event-oriented," meaning that it handles events such as mouse actions, although you can also build custom controls for everything from pen gestures to voice input or various data feeds. Fundamentally, Visual Basic is a BASIC-like language for building applications with all the accoutrements of the Windows interface. You can manipulate Windows objects, files and the like, and launch applications, but it's not designed as an application macro language. It communicates with other applications through DDE but does not support OLE at this point, since its focus is creating applications, not data files that could store objects.

BASIC As Simple-as It Can-be

Visual Basic is a programming language or developers or very savvy users that combines BASIC syntax, user interface development tools and event management, rather than a meta-macro language.

All the programmer has to do is write event-handlers -- the functionality that the user or other outside agent calls through the interface. He doesn't need to worry about intricacies of interpreting user interface interactions or building elegant UI objects. A runtime module handles the message dispatching; all the developer does is attach the functionality, written in a BASIC-like language. "The programming language underneath it is BASIC, but the efficient, high-level architecture is not BASIC's at all," says aptly named product manager Tom Button. The programming environment makes heavy use of forms and menus and an object palette to generate interface objects instead of requiring the user to write a lot of procedural code. While a script language builds a program by manipulating application commands, Visual Basic works on the data directly, with its own commands.

WordBasic

Meanwhile, WordBasic is an embedded macro language for Word that has no particular relationship to Visual Basic beyond common heritage and the BASIC syntax and a future common parent, Object Basic. It is designed to automate Word, and understands basic word-processing functionality, but it's still mostly a procedural automation tool. It lets you do such things as integrate faxing and e-mail, run mail-merge and do other text-oriented tasks.

WordBasic allows the creation of Word-oriented macros, and does so automatically by interpreting keystroke sequences, treating them as the underlying
commands (like what NewWave calls an action processor). These sequences, now at a higher level than just keystrokes, can be edited by a builder-user to incorporate loops, conditionals and other logic.

In the long run these languages will evolve to join a set of object-oriented Object Basic languages: one to script multiple applications for end-users, several single-application macro languages, and a visual object-oriented development tool descended from Visual Basic.

RELEASE 2.0: TOASTER FAX

About a year ago (see Release 1.0, 2-90) we wrote about Interfax, which offers a fax-response service used by Ziff-Davis, among others. Now two of the founders of the company, sales type Phil Sih and wizard Dave Schwaderer, have left to launch a new fax-related venture, DBC (for Daniel Burnham Court, where Sih lives). This time the offering is a product, not a service. The user can buy a box for $1795 and manage the entire process himself. The appeal is the same as that of a pc -- local control, defined costs, and so on. (Actually, the server can be anywhere, since you talk to it by fax and phone.)

DBC’s MessagePost is a fax administrator in a box that both the fax owner and the owner’s customers (information receivers) speak to via their telephone and fax machine. In essence, it’s a fax server, which stores fax information and manages it. Both the owner (who gives it the faxes to distribute) and the owner’s customers are clients.

MessagePost, which contains a 286 pc and 40 megabytes of storage (currently) talks to the outside world only through the phone line. Even the owner/user simply dials it up with a password and either pushes telephone buttons or sends it faxes (data entry) to manage it. It doesn’t perform any complex tasks -- which is why people should find it usable and useful.

From the perspective of the owner’s customers, it functions as a fax server for users who follow its simple instructions. The caller uses the handset on his fax machine to call MessagePost, using a number listed in an ad or brochure or yellow pages listing, or perhaps after getting instructions from an answering machine receiving after-hours calls. MessagePost takes the call and leads the user (by recorded voice) through a set of questions that the user answers by pressing the telephone buttons on the fax machine; it faxes him a directory of items available if he needs one. Then, following or bypassing the instructions, the user dials the numbers of the items he wants and receives them by fax. MessagePost uses the answers/numbers to select the documents (loaded by the MessagePost owner) to transmit to the user’s fax machine.

Thus, the user doesn’t need to sit and listen to a long list of items available, and can keep the directory for reference for the next time he calls. He can also get an updated list at any time -- and the initial voice message can let him know when he should do so: "Because of circumstances beyond our control, we have added a large number of important advisories for owners of our exciting MegaMIPS line of high-quality pcs. Please press OOPS [6667] to receive the most recent listing of trouble-shooting tips available."

Release 1.0 31 May 1991
I’ll have my machine call your machine...

We see the greatest use for MessagePost for distributing common, changing information -- catalogues, directions to a party, product literature, newsletters (free ones, anyway), product documentation and updates, instructions for ordering, restaurant menus for delivery service (the specials -- and the prices!?? -- change daily), press releases, forms, schedules, etc. Since the information-receiver initiates the call and his machine stays on the line to receive the faxed material, he pays the phone bill. You could also extend this to paid information by using a 900 line, or setting up account numbers linked to telephone numbers, but that’s in the future.

Moreover, with a little reprogramming by DEC, MessagePost could also be used for fax store-and-forward. For now there’s one password, for the administrator, and all the items stored are freely available to anyone who knows their numbers. (Of course, you could have unlisted numbers -- 123 for Juan’s messages, 124 for Alice’s, and so forth -- but security is minimal at this point.) One can easily imagine additional features, such as an automatic accounting and billing system, customized faxes, and so forth. There is a pc inside there, after all.

No muss, no fuss

MessagePost could be an add-on to a pc, but Sih believes (and we agree) that it will sell better -- for now -- as a black box. As far as the user is concerned, he says, "There could be squirrels in there." There are similar systems with more features selling for thousands of dollars more, but people are leery of fancy equipment. Give them a simple machine and they’ll use it and ask for more; give them a fancy one, and it won’t get used. Right now the issue is saving labor and getting information promptly to (potential) customers; time enough for enterprise integration later on. It wouldn’t be cost-effective for a larger organization, but for a small business it’s the next handiest thing after the fax machine itself.

Fax services have caught on, but they’re still mostly the province of the big guys -- just like computers before the pc era. Buying a MessagePost is a limited chunk of cash upfront, and it requires less management overhead than signing up with a service. Thus, as they have done with pcs, many people will probably migrate to inhouse fax response. Once the preserve of magazines and large companies, it can now be a do-it-yourself proposition.

GOODBYE LORI! WELCOME ROBYN!

We’re delighted to announce that Robyn Sturm has joined us from Gralla Publications to handle circulation and fulfillment -- that means seeing that you get our issues and that we get our checks! She will be running our database, dealing with our mailing house, and answering any questions you may have about your subscription (as opposed to the content of the newsletter). She replaces Lori Mariani, who liked Tucson so much after several PC Forums that she’s moving there. Good Luck, Lori! And good luck, Robyn!

Release 1.0 31 May 1991
RESOURCES & PHONE NUMBERS

Roger Heinen, Apple, (408) 974-3511
Allen Cypher, Apple, (408) 974-5415; fax, (408) 974-0234; cypher@apple.com or CYPH ER on AppleLink
Chuck Digate, Beyond, (617) 621-0095
Kathleen Garvey, Richard Schwartz, Borland, (408) 438-8400
Vadim Yasinovsky, Clear Software, (617) 965-6755
John Landry, DB Software, (508) 370-5800
Phil Sih, DBC, (415) 776-6227; fax, (415) 776-0615
Bill Crow, HP, (408) 773-6345
Jay Tunkel, IBM, (407) 443-5955
Kevin Karney, Mansfield Software Group, (203) 429-8402
Mike Maples, Microsoft, (206) 882-8080
Robert Beech, Pharos Technologies, (513) 984-9273; fax, (513) 984-1623
Charles Daney, Quercus, (408) 867-REXX
Dave Winer, UserLand, (415) 325-5700

COMING SOON

- The electronic frontier.
- Text tools & SGML.
- Performance support.
- Upgrades: Aftermarket comes into its own.
- And much more... (If you know of any good examples of the categories listed above, please let us know.)

-------------------------

Release 1.0 is published 12 times a year by EDventure Holdings, 375 Park Ave., New York, NY 10152; (212) 758-3434. It covers pcs, software, CASE, groupware, text management, connectivity, artificial intelligence, intellectual property law. A companion publication, Rel-EAST, covers emerging technology markets in Central Europe and the Soviet Union. Editor & publisher: Esther Dyson; associate publisher: Daphne Kis; circulation & fulfillment manager: Robyn Sturm; executive secretary: Denise Dubois; editorial & marketing consultant: William M. Kutik. Copyright 1991, EDventure Holdings Inc. All rights reserved. No material in this publication may be reproduced without written permission; however, we gladly arrange for reprints or bulk purchases. Subscriptions cost $495 per year, $575 overseas.
## RELEASE 1.0 CALENDAR

<table>
<thead>
<tr>
<th>Date</th>
<th>Event Description</th>
<th>Location/Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>June 17-21</td>
<td>*International Computer Forum - Moscow. Sponsored by the International Computer Club. Call Levon Amdilyan, 7 (095) 921-09-02, or &quot;levon&quot; on MCI Mail at 439-1034; or Esther Dyson at 1 (212) 758-3434.</td>
<td></td>
</tr>
<tr>
<td>June 18-21</td>
<td>Videotex 91: Broadening the consumer market - Crystal City, VA. Sponsored by Videotex Industry Association. Call Debbie Tritle, (301) 495-4955.</td>
<td></td>
</tr>
<tr>
<td>July 2-4</td>
<td>*Machine Translation Summit III - Washington, DC. Sponsored by the Center for Machine Translation, Carnegie Mellon University. Call Jaime Carbonell, (412) 268-6591, e-mail: <a href="mailto:mtsummit@cs.cmu.edu">mtsummit@cs.cmu.edu</a>.</td>
<td></td>
</tr>
<tr>
<td>July 9-14</td>
<td>*PC Forum - Moscow. Organized by IDG World Expo and Information Computer Enterprise, USSR; co-sponsored by several USSR state committees. With a focus on the Republics, likely to</td>
<td></td>
</tr>
</tbody>
</table>
be more important as the Soviet government devolves power to them. Call Terence Coe, (508) 879-6700.

July 13-16  

July 14-19  
*AAAI conference* - Anaheim. Sponsored by American Association for Artificial Intelligence. Also includes Innovative Applications of AI. Call Carol Hamilton, (415) 328-3123.

July 15-18  
*Network computing conference and exposition* - Washington, DC. Sponsored by IDG World Expo Corporation. Call Brenda Cone, (800) 225-4698 or (508) 879-6700.

July 15-18  

July 17  

July 23-26  

July 28-Aug 2  

July 29-Aug 1  

August 5-8  
*International workshop on human-computer interaction* - Moscow. Sponsored by California State University and the International Centre for Scientific and Technical Information, Moscow. Contacts: Larry Press, (213) 475-6515, fax (213) 516-3664, e-mail lpress@venera.isi.edu; or Yuri Gornostaev, 7 (095) 198-72-41 or enir@iaeal.bitnet.

August 6-9  

August 11-13  
*GeoCon/91* - Cambridge, MA. Sponsored by Softletter. An international product showcase for European, Canadian, Asian and Latin American developers who seek U.S. publishing or partnership contacts. Call Jeff Tarter, (617) 924-3944.

August 14-16  

August 19-23  

August 19-23  
*SCO Forum91* - Santa Cruz, CA. Sponsored by The Santa Cruz Operation. Call Zee Zaballos, (408) 425-7222.

August 24-30  
*12th international joint conference on artificial intelligence* - Sydney, Australia. Sponsored by the National Committee on Artificial Intelligence and Expert Systems of the Australian Computer Society. Contact: Beverley Parrott, 61 (2) 3572600, fax 61 (2) 3572950.

September 4-5  
*Electronic Democracy conference* - Arlington, VA. Sponsored by Government Technology and Riley Information Services in


September 8-10 The Culpepper Forum - Atlanta. Sponsored by Culpepper and Associates. The fifth annual meeting of the software industry’s top sales and marketing executives. Call Doris Brinks, (404) 688-0616.


September 11-14 Software Publishers Association annual conference - Orlando. Sponsored by SPA. Call Ken Wasch, (202) 452-1600.


September 15-19 *EastEurOOPe '91 - Bratislava, Czecho-Slovakia. Sponsored by JOOP, ParcPlace, Xerox, Digitalk, Software Slusovice, Kan- celarske Stroje, others. With Adele Goldberg, Krysten Nygaard, Pierre Cointe. In English. New developments, new people. Contact: Augustin Mrazik or Peter Mikulecky, 42 (7) 724-826; fax, 42 (7) 725-882; e-mail: eeoop91@mff.uniba.cs.


September 25-27 *Second European conference on computer-supported cooperative work - Amsterdam. Knowledge workers and academics, unite! Organized by the Center for Innovation and Cooperative Technology of the University of Amsterdam. (The language of cooperation is English.) Call Mike Robinson or Liam Bannan, 31 (20) 525 1250/1225; fax, 31 (20) 525211; e-mail, Bannon-@learn.ucd.ie; or Charlie Grantham, 1 (415) 370-174; cegrant@well.sf.ca.us.

Sept 30-Oct 1 Virtual Reality conference - San Francisco. Sponsored by the Heckler Corporation. Call Marilyn Reed, (203) 226-6967 or (800) 635-5537.

Sept 30-Oct 4 *Seybold Conference - San Jose. The leading event in the computer publishing community. Sponsored by Seybold Seminars/Ziff. Call Kevin Howard or Beth Sadler, (213) 457-5850.


October 6-11 *OOPSLA '91 - Phoenix. Sponsored by ACM. Call John Richards, (914) 784-7731.

October 7-11 *Interop '91 - San Jose. Sponsored by Advanced Computing Environments/Ziff. Developments in internetworking and network computing. All equipment on display must work together by being connected to the show network. With Ellen Hancock, IBM Communication Systems. Call Wendy Gibson, (415) 941-3399.

Release 1.0 31 May 1991
October 15-17  *NetWorld '91 - Dallas. Sponsored by Bruno Blenheim. Call Annie Scully, (201) 569-8542 or (800) 444-EXPO.
October 17-21  USA Showcase '91 - Budapest. Co-sponsored by the Hungarian Ministry of Trade, the Hungarian Chamber of Commerce and the American Chamber of Commerce in Budapest. Call Jay Bowman at (713) 266-0610.
October 21-25  *Comdex - Las Vegas. So wonderful they couldn't wait until November? Whatever the reason.... Sponsored by the Interface Group. Call Elizabeth Moody or Dick Blouin, (617) 449-6600.
October 27-29  The Classic - Monterey, CA. Sponsored by the American Electronics Association, for cute companies and eager investors. Call Flo Lewis, (408) 987-4200.
November 4-7  ADAPSO fall management conference - San Francisco. Sponsored by ADAPSO. Call Shirley Price, (703) 284-5355.
November 7-9  1991 Nanotechnology conference - Palo Alto. Sponsored by Stanford University Department of Materials Science and Engineering, University of Tokyo Research Center for Advanced Science and Technology, Foresight Institute. For scientists and technologists working in fields leading toward molecular nanotechnology: complete three-dimensional structural control of materials at the molecular level. Call Chris Peterson, (415) 948-5830.
November 10-13  **Second East-West High-Tech Forum - Warsaw (Prague in 1992). Sponsored by EDventure Holdings. With a roster of serious-minded entrepreneurs and vendors from East and West. Don’t just come to listen to advice; come to mingle with the people making it happen. Call Daphne Kis, 1 (212) 758-3434 or fax (212) 832-1720; MCI Mail: EDventure, 443-1400.
November 19-21  PC Expo - Chicago. Sponsored by Bruno Blenheim. Call Steve Feher, (201) 569-8542 or (800) 444-EXPO.
December 2-4  **Alliance 91 - Tokyo, Japan. Sponsored by Harvard Business School Ass’n. Strategic alliances with Japanese companies. Call Mark Francis or Yasuhiro Mikamo, (415) 742-0757.
December 3-5  European Publishing conference - The Hague, Holland. Sponsored by Seybold Seminars. Contact: Laurel Brunner, 44 (323) 410561 or fax, 44 (323) 410279.
December 15-18  *Hypertext '91 - San Antonio, TX. Third international conference on hypertext. Sponsored by ACM. Call Janet Walker, (409) 845-0298, e-mail leggett@bush.tamu.edu.

Release 1.0 31 May 1991
1992

February 11-13  NetWorld 92 - Boston. Sponsored by Bruno Blenheim. Call Annie Scully or Mark Haviland, (800) 444-3976 or (201) 569-8542.

March 2-6  OpCon West - Santa Clara. The west coast session of Softletter's twice yearly conference for operations managers. Call Tom Stitt, (617) 924-3944.


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

*The asterisks indicate events we plan to attend. Lack of an asterisk is no indication of lack of merit.
RELEASE 1.0

SUBSCRIPTION FORM

Please enter my subscription to Release 1.0 at the rate of $495 per year in the U.S. and Canada. Overseas subscriptions are $575, airmail postage included. Payment must be enclosed. Multiple-copy rate on request. Satisfaction guaranteed or your money back.

Name ________________________________
Title ________________________________
Company ________________________________
Address _________________________________________
City __________________ State ______ Zip ______
Telephone _________________________________________

How did you hear about Release 1.0? ________________________________

☐ Payment enclosed.

☐ American Express # - - - - - - - - - Expires ______
Signature _________________________________________

☐ Please send me additional information on your multiple-copy rate.

Please fill in the information above and send 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