ATOP THE OPERATING SYSTEM

Conventional wisdom recognizes that hardware is a commodity, operating systems are a commodity, even database management systems will soon be a commodity. And customers wouldn't have it any other way: Look at the success of IBM's PC against the record of its RT PC; look at UNIX or OS/2 against any number of possibly "better" but proprietary operating environments; look at the blossoming of SQL. Customers don't like to feel locked in.

At the same time, everyone knows, customers want value-included "solutions" -- full-featured applications with embedded knowledge (whether in LISP or COBOL). Yet in this age of mission-critical systems, software is no longer overhead; it is production equipment. A user company's "solution" is its means of differentiating itself -- hardly something it wants to pick up off a shelf. How can the computer industry reconcile customers' technology-averse need for total solutions with their strategic need for differentiated solutions? The tools that solve this problem will themselves become standards, and help the application builders (vendors or end-users) who use them to integrate their systems with other people's. Builders who ignore the emerging standards will be forced to build stand-alone total solutions from scratch -- an unattractive proposition in tomorrow's marketplace.

Suddenly the most lucrative business for vendors (and the greatest demand among customers) is shifting back, from applications to richer, more powerful but easier-to-use tools for systems development. We're not just talking CASE for traditional mainframe-oriented systems, either (see box, page 4); we're talking tools for developing groupware for intelligent workstations (and people), tools for linking existing applications, tools for modeling businesses and building systems tailored to execute their unique operations. We're talking about tools that will let users build their own applications.

The new application systems they build will perform many of the same functions as the old ones, yet they will control and embody more closely how a company actually operates. They will reflect not just internal flows of repetitive data, but also channels of responsibility and communication. They will handle not just recurring events, but also exceptions, and they will

**FORUM DETAILS ON PAGE 26**
change more easily when the corporate structure changes (as they must, because they embody it). They will deal with rich data, text, decision rules -- the medium of strategy, design, differentiation, positioning...rather than just with numbers -- the measure of costs, production, and sales.

A NEW SYSTEMS ARCHITECTURE

Common wisdom about the system of the future is correct: It will be small but extremely powerful. It will have high-bandwidth access to other machines and data. Most important, there will be many of them working in concert, sharing data and tasks transparently and acting as agents for the people who use them. Rather than buy arbitrary "applications" with functions determined by a vendor, users will be able to assemble their own sets of cooperating application modules tailored to their own unique needs. Moreover, not only will it be impossible to say where one application ends and another begins, so will it become impossible to say when one version ends and another begins, as software will evolve continuously.

In systems as in companies, the whole is not merely the sum of the parts; it is the transformation of the parts into something broader and more abstract. The next development in computing -- fostered by connectivity, distributed database management, and artificial intelligence -- will be the integration of parts and the whole, the individual tasks and the unifying structure.

Traditionally, computers accomplished simple tasks we were already doing, although many of those have become too large to handle any other way: huge payrolls with complex tax calculations, airline reservations, credit-card validation, and all other kinds of transaction processing. Flexibility was the domain of small computers, which automated only small tasks -- word-processing, model-building, file management.

Largeness always implied rigidity. Why? Because in a large system there are so many ways that things can go wrong. (After all, why do we need Air Traffic Control given the existence of published airline schedules?) One of the main goals of a traditional large-system designer is to minimize complexity so as to minimize possible errors. With a more flexible, coordinated, distributed, locally controlled system, it's possible to accommodate local variances and exceptions both in structure and in behavior. To render this idea in systemspeak: Before the advent of flexible interprocess communications and data exchange, distribution of data and function generally required centrally controlled, compatible systems. With better techniques for communication and data representation, there's no reason we can't work easily across environments. The whole will retain the flexibility and variety of its parts, and will reflect their structures rather than imposing a rigid one of its own.

Nuts and bolts: Code and data

To build such an environment from scratch would be relatively simple. (God could build the universe in a week because he did not have to deal with history.) But in fact any system must accommodate both existing and future applications from a variety of vendors, to say nothing of operating systems and existing in-house and purchased applications with their attendant data and data structures.
The quickest way to achieve such coexistence (and cooperation) involves an extra layer on top of the operating system and under or around the applications -- and typically considerable overhead. A better but hard to retrofit approach is to use common constructs across and within applications, a sort of "dynamic command exchange" (as in Microsoft's Dynamic Data Exchange).

The basic tools are several: Glue for binding existing applications together into meta-applications (retrofit) or for cooperative processing so that applications on the same or different machines can share data and tasks; user-task routing so that tasks can be assigned appropriately in real-time (i.e., this goes to Alice because the amount exceeds Juan's purchasing authority); rich data structures (text, images, real-world models) for many purposes including compound documents and easier application development; reusable code modules to save work and ensure consistency and quality; a common database and network management underneath (table, page 14).

Where and how

Much of the work on platforms for this next generation of applications is happening in the personal computing world, which now mostly lacks rich development tools of any kind. The pc community is the home of much of the new development as systems power is distributed to end-users. This makes sense -- personal computers are where computing meets people and where people are linked to each other.

Depending on your perspective, the mainframe will be replaced by the pc or the pc will become a mainframe. Precisely because the pc world is less constrained by history, it may more willingly accept a new generation of tools that inherits the simplicity of the pc and the power of the mainframe -- a style able to serve a new user base of business strategists and end-users.

The tools divide into two major categories: Those that make it easier to program, and those that make it easier to model the real world.

Such tools are on the way: Last summer Bill Gates published an article in Byte proposing a generalized, cross-application macro language. Last month, Hewlett-Packard announced NewWave, a product; and Peter Miller and Mitch Kapor announced ON Technology, a company (although it has changed its short-term focus since; see page 24). Last fall IBM announced that REXX, the procedures language for the VM operating system, would be part of its Systems Application Architecture, functioning across all IBM/SAA operating environments. Mansfield Software is selling a PC version of REXX from Quercus; The Whitewater Group's Actor, now in its second release, is starting to get favorable reviews and better yet, users. Coordination Technology, Action Technologies and FCMC are working on groupware development environments.

All these vendors have a common goal: They're looking to foster the next generation of applications. The tools they sell divide into two major categories: Those that make it easier to program, and those that make it easier to model the real world. The programming tools are designed to overcome the difficulties of working with windows and other graphical objects, interprocess communications, heterogeneous environments, cooperating software modules; the design tools are helpful in modeling group interactions, rich data structures, mission-critical software with strategic rather than logistical significance.

Release 1.0 30 December 1987
CASE IN CONTEXT

Just as in the pc world, mainframe vendors and customers are beginning to realize that database managers are commodities and applications confer competitive advantage. Thus mainframe database vendors such as Cullinet are repositioning themselves as CASE vendors, providing tools to support application development, installation and customization for any dbms. IBM is working on tools for DB2 and will soon announce a DB2-based information repository that will become the industry standard. This is IBM in its traditional role, supplying the underlying systems software for other vendors' applications -- in this case, CASE tools. DEC too is redoubling its CASE efforts.

Traditional, mainframe CASE tools are targeted to building large, centralized systems. They exist to handle the complexity that results from sheer size and number of interacting processes, rather than the intrinsic complexity of real-time or calculation-intensive applications, which are mostly better hand-coded. CASE tools encourage system-builders to take a modular approach, beginning with a high-level description of the processes to be automated on the one hand, and a set of related data elements and reports and screen displays (forms, menus, prompts) on the other. These are all rigorously described according to a particular syntax and methodology, and stored in a central database -- the integrated resource directory system (or some such name). Then, ideally, an application generator can automatically produce a bug-free, complete, integrated application that produces the desired system as defined by the inputs and outputs the user selects. (For further details, see Release 1.0, 32 December 1986.)

Theoretically, the system-builder never touches the application itself; he merely creates or alters the specs. The directory system and the associated tools and rules thus ensure that any application is correctly implemented (if not designed) and that any changes are made consistently. Multiple applications can be created from the same set of data definitions, and forms and reports can be edited and respecified. If correctly used, CASE can lead to remarkable gains in productivity and error reduction. For traditional transaction-processing, database-based applications, you can't beat CASE.

However, traditional CASE tools still generate mostly procedural COBOL applications that execute on a mainframe and communicate in screenfuls or panels (cf. the Query Management Facility in IBM's OS/2 Extended database manager). They don't deal with the new application paradigms explored here -- group coordination, rich data structures and graphics, heterogeneous environments, cross-system communication (save via dumb terminals), exception-handling. They fail at building truly interactive, fine-grained systems. There's no reason a set of traditional CASE tools couldn't be enhanced with the capabilities described here; and there's no reason that CASE tools couldn't generate (slightly enriched) traditional applications to run in the OS/2 environment. The gap is not technology but culture.

Call it CASE or call it new-age development tools. As in other areas, the mainframe and micro approaches will eventually merge.

Release 1.0 30 December 1987
Different tools for different goals

Of course, just as you can build anything (including OS/2) on top of DOS if you're willing to try hard enough or to bypass DOS, so can you hack together a "next-generation" application with any of these systems. The question is, What is supported and generalized? What is made easy or automated? What is made routine so that the builder can focus on the unique parts of the application rather than the generic ones? Secondarily, how much of the existing application/OS suite can you salvage? How do the various applications and the various tools work together?¹

The systems described below range from glorified memory-resident keystroke recorders to Whitewater's total development environment. (Many others are left out; new tools and repositioned old tools are everywhere.) They embody different visions of the organization whose work they automate: Some focus on interaction among applications, others on interaction among people, some on automating tasks and others on managing transactions among people. Some work by task, others by user, while still others work by following a piece of virtual paper through its lifecycle.

The trouble with DOS

The big problem now for developers of new-generation cooperating applications is DOS: What if you're trying to talk to a machine while it's doing something else? Do you want to let an agent take over your pc while you're trying to get some work done?

Developers and users are still on their own when trying to get applications to work together, or provide support for people working together.

This notion still reigns: one person/one machine/one application. There is no need for coordination of anything. Database management systems are considered applications, and different varieties (with their own file structures) abound. Application developers invent arbitrary file structures for their applications; many people, both developers and hypervisor vendors, have come up with their own memory managers (e.g. Microsoft's Windows 386 and Quarterdeck's DESQview; see Release 1.0, 2 December 1986).

But that model no longer works. Even the hypervisors divide a 386 up into separate virtual machines, each separately loaded and unaware of the other's existence, while the new architecture demands a single environment (possibly across different machines) managing and coordinating concurrent processes. Sometimes reluctantly, most vendors are working on the assumption that somehow they'll cooperate with OS/2, just in order to get a standard approach to multi-tasking and interprocess communications. OS/2 Extended, from IBM, will go even a little further, providing a common database and communications facilities, but in its initial release (with a single-user database model) it still does not go very far. Developers and users are still on their own when trying to get applications to work together or provide support for people working together.

¹By analogy: Don't give the user a hammer and boards to build his house, but don't give him a finished house, either. Let him design his house, but then build it automatically with pre-fab parts. And make sure the plumbing in the bathroom uses the same kind of pipes as those in the kitchen.

Release 1.0

30 December 1987
The operating system traditionally deals with the least common denominator and performs basic functions that are common to all applications, such as handling hardware internals and I/O. On larger systems it handles memory management and hardware, while on smaller systems (pcs) such issues are left to the application. In fact, there's also a higher level of problems that get solved over and over, from systems issues to complete application modules, from laying out displays and managing user interaction to trading data and sharing files to sharing work, arranging schedules and trips, and creating implicit models of a business -- divisions, employees, locations, taxes, etc. But each developer solves them in his own unique way. Customers want to make use of these sophisticated solutions, but since none of them solves everything, customers want these solutions to work together. These are the recurring problems the new-generation tools will solve.

SYSTEMS CONTENT: MODELING THE REAL WORLD

Although the initial impetus for the creation of Macintosh and Windows was a common interface, the goal for the future also includes a common back-end (or program interface) and common rich data structures so that applications can communicate and share data and tasks, avoiding redundancy of data and programming effort. While builder/users want the ability to embed their own knowledge and processes into the systems they build, they don't necessarily want their tools to be content-free; they just want them customizable.

As it happens, the difficulty of coding for these same more complex environments is so great that it has fostered the use of object-oriented development systems to do a lot of the low-level work. The graphical primitives necessary are fairly standard, but tedious to create over and over again. Graphical objects come in fewer varieties and are easier to represent than real-world models -- which is why the object-oriented environments are offering class libraries of graphical objects before getting around to doing the same for higher-level objects such as companies, purchase orders, etc.

The benefits of these common objects are not just speed of development but consistency across applications, ease of maintenance/enhancement, and the like. (The disadvantage, generally, is that object-oriented programs execute more slowly than traditional ones.) Long run, if enough people use PPI's ICpak classes or the ON Technology platform's objects, applications from different independent vendors will work together nicely, sharing not just data but also tasks. In fact, in truly object-oriented programming, there are no applications, just communicating objects. The programmer builds objects rather than discrete applications; collections of interacting objects perform the functions that would be considered an application.

Indeed, much of the value of these products lies in the work they save the user, not just in coding, but in design of higher-level elements -- or, as defined in objectspeak, class libraries. The most common class libraries so far contain graphical objects, which are relatively generic, although they also form a large part of the entire repertoire of many CAD packages (or, as Softsel's Dave Wagman calls it, word-processing for engineers). And, of

2These two environments borrowed the look and feel of Xerox PARC's Smalltalk, but not the programming techniques that made it easy to implement.
REUSABLE CODE:
SUBROUTINES, MACROS, TEMPLATES, CLASSES AND INTERFACES

The holy grail of application development is reusable, enhanceable code, a notion that has been around since the time of Ada and before. Reuse of code saves development time, ensures quality and consistency, and even saves memory in the case of dynamically linked subroutines. Subroutines, macros, templates, classes and interfaces are all examples of reusable code, distinguished mostly by how easily they can be reconfigured to suit a user's particular needs.

Subroutines are independent named code modules. The benefits include smaller programs because frequently executed procedures need be written out in full only once, and called multiple times within a single application. Within a newer environment such as OS/2, with a dynamic linked library, they can be loaded dynamically, at runtime, and shared across multiple programs.

Macros began as a kind of programming shorthand that was expanded at compile time by a macro assembler or pre-processor (linked "in-line") into the normal language of an application or operating system. The benefit here is that a programmer can define a complex piece of code once and reuse it as necessary.

New macros. However, since the early days of 1-2-3 the term macro has taken on a new meaning (which purists call scripts), as a special kind of interpretive procedure that can be called by a user as well as a programmer. It consists of a series of commands interpreted by a resident application or operating environment. User-friendly programs allow a user to create macros by recording his actions, and then edit the listing. Macros can also be created in response to a series of prompts, or can be written from scratch by an experienced user.

Templates are code structures with variables/parameters left empty or easily changed. User-friendly templates provide a front-end that prompts the user for appropriate responses that the system uses to customize the template to create the desired application.

Classes are pre-configured generic objects that a system builder can use as is ("inheriting" their behavior and attributes) or reconfigure or specialize (overriding the inheritance) to suit the user's needs. They are designed to be reused, and are structured to accommodate multiple data types, new procedures, and the like, "encapsulating" such details within a forgiving structure.

User interfaces are a vaguer notion than the rest of the items here. A "standard" user interface is generally implemented as a library of reusable subroutines and templates for user-interface constructs -- including command interpreters, menus, prompts, graphical objects and other intermediaries -- that transform an operating system or application into something intelligible to the typical end-user. An operating system plus an interface that is used around the applications it supports is called an operating environment (such as Windows and the Macintosh environment).
course, most office automation packages include text objects -- pages, words, and the like, even though they don't think of it that way. Desktop publishing packages also know about paragraphs, boxes, captions, titles, cross-references, titles, headers and footers and the like.

From the tools vendor's point of view too, content is a valuable product. While languages may not be copyrightable, and the systems built with those languages are owned by their builders, tools-vendors can make substantial revenues from sales of reusable code with embedded knowledge -- libraries of run-time subroutines, class libraries, application templates, interface modules, report forms, business models, and the like.

Object-oriented models

Objects take on various levels of granularity. At one extreme is the Macintosh Finder or Hewlett-Packard's NewWave, which treat any set of data (typically a file or an extract thereof) and its associated application as an object. Beyond that, New Wave relies on the applications themselves, and it includes a script/macro language to connect applications together. At the other extreme is The Whitewater Group's Actor, a full-fledged object-oriented programming environment designed for application development rather than manipulation of existing applications.

The fundamental division in approach between traditional development models and object-oriented ones boils down to where control resides: Is it with the application and its control flow, or with the data elements, which operate autonomously in accordance with their defined behavior (procedures)? (For more on object-oriented programming, see Release 1.0, 24 March.)

SYSTEMS GLUE: BUILDING META-APPLICATIONS

Aside from the operating system and higher-level data structures, there are all those existing applications to worry about, to say nothing of all those DOS users. Accordingly, many tools vendors are building hybrid languages designed to manipulate existing applications and help glue them together. Such hybrids languages are a cross between procedure (shell) languages that operate within an operating system environment, and application languages that can talk directly to applications, understanding their file structures and coordinating their interaction. For example, such a glue language could be used to write a meta-application that would run one application, take some data from it, use it in another application, examine certain results, and, according to those results, take further appropriate action. Or it might queue a certain job onto a certain person's job list, assign a room and notify participants of a meeting, etc. This assumes some operating system support, a multi-user database management system, and other amenities.

The lore of languages

Traditional high-level languages allow their users to manipulate data and to address the operating system for access to hardware (storage, displays, other input and output devices). The OS takes care of memory management and other physical issues (except in DOS).
Procedures (or shell) languages allow programmers and systems operators to manage their operating environments -- JCL (Job Control Language), Exec and REXX on IBM mainframes, and the nameless set of commands that comes with DOS on the pcs. They allow the user to address the OS and applications directly -- loading applications, manipulating system resources, etc. Sequences of such commands can be stored as batch files, for interpretation and execution on demand by the shell interpreter.

These languages can manipulate applications on a high level, and will gain greater precision of control to the extent that application builders use the same syntax within applications. They will gain power by becoming standards.

Macro languages

Macro languages are a widespread means of enhancing any development system. The original mainframe macro languages were a cryptic programmer's shorthand, while the first few macro languages for pc applications were little more than keystroke recorders/interpreters. As time passed, both mainframe and pc vendors made their macro languages more flexible and usable, with facilities for branching, dynamic linking, recovery from error conditions, interactivity, and the like. With these capabilities, macro languages have become powerful, but they are still limited to the capabilities of the application they control. They can't work across applications, nor does the syntax of one application's macro language match another's -- a confusing, wasteful state of affairs.

Recorders

Meanwhile, some generic keystroke recorders and richer products such as Direct Technology's Automator mi and Alpha Software's Advanced Keyworks stay memory-resident and can work across applications, but they don't know anything about the applications they work with. Vendors use terms such as "robot" and "automate" to describe their systems, which essentially are and do just that: They automate particular sequences and act as robots, carrying out the rote instructions the builder/user gives them. These systems have limited knowledge of the world around them; they simply do certain tasks in response to the screen, to keyboard input, and to the system clock. Change the screen layout and you have to reprogram the robot. The vision of these automators is essentially one-machine: Each robot sits on its own machine and does its thing in response to local cues. If there's any notion of data structures or other machines or other people on a network, it resides only in the mind of the developer.

Hybrid vigor

One step beyond all these languages is a new breed, which combines the best features of all of them. They know something about the individual applications they're controlling, and they know how to coordinate across applications. These are procedure languages not with intelligence but with knowledge -- they know about data structures, about certain popular applications, and about communications. Their "language" is higher-level and more abstract than the systems above, because they can translate a user's command into the precise actions needed to carry it out in a given environment.
THE DATABASE UNDERNEATH

The basic difference between a development environment and an operating environment is that the environment sticks around after system development to offer execution facilities to the ultimate end-user. Along with an interface and shell, the most widespread such persistent structure is the traditional database management system, which includes both programming tools and a database manager to manage data for the applications. Many other applications and development environments have an underlying dbms that may not be visible to the user, that can either store application elements (code modules, screen descriptions) or data or both. Among the new-platform environments, these include Action Technologies' Conversation Manager and the database underlying Staffware.

The standard next-generation operating system/environment will include a standard database management system, to store simple data in a standard way. The obvious standard will be the database engine of IBM's OS/2 Extended Edition, and the many clones thereof. Microsoft already has an unannounced deal (Apple has one that's announced) with Sybase for such a system. Oracle has announced a joint marketing agreement with Novell, so we now have three teams: IBM, Microsoft-3Com-Sybase, and Novell-Oracle. (Of course, Microsoft is also on IBM's team, and is selling ammunition to Novell-Oracle as the supplier of OS/2.) Where does Lotus fit in, with Lotus/DBMS? Perhaps on the IBM team, but that's not yet clear. Then there's Ashton-Tate, Ingres, Migent, and all the other dbms contenders. IBM will be happy to help them as long as they support OS/2E, and are content to play supporting roles by providing interfaces, development tools, and other "surfaces."

However, a standard relational dbms is incapable of supporting the kinds of rich data structures the next-generation applications will need: not just text and graphics but also other data structures -- spreadsheets, hypertext, indexes, links to other applications and triggers (see Release 1.0, 10 August and 25 November). So while vendors such as Action Technologies and FCMC Inc. are moving towards replacing their internal proprietary databases with hooks for a customer-supplied standard database, other vendors such as ON Technology and Productivity Products International are working on developing their own new object-oriented data structures which will include procedures to extract data from standard databases.

These hybrid languages can do everything, addressing the OS as well as applications, and manipulating data as well as files. At this point, there are several major hybrid language offerings, all as yet immature: IBM's REXX, Hewlett-Packard's NewWave agent task language, and Microsoft's prospective MacroBASIC (not yet formally named). Each of these can now manipulate applications on a high level, and will gain greater precision of control to the extent that application builders use the same syntax within applications. In other words, they will gain power from becoming standards.
SUPPORTING GROUPWARE

A final set of tools/languages is devoted to a new kind of application model -- groupware. Its languages handle communications, message-routing, transactions, directory management. Traditional pc applications focused on one task at a time; traditional mainframe software emulated an assembly line. Groupware combines tasks in sets organized variously according to the developer's vision of administrative organization: data/forms, agents, work centers, or transactions. The forms-centered approach assumes the existence of a centralized or distributed database, and manages the paper flow through an organization, presenting the appropriate data to the appropriate people involved in the proper sequence (FCMC's Staffware). The agent approach divides the work among people, and assigns to each his or her proper role (HP's NewWave). Much of each user's work may be carried out by the system, acting as an "agent" for that person. The work center approach looks at combinations of tasks that are performed in concert by departments, or work groups (Coordination Technology's BCOS/Orgware). The transaction model recognizes that the most interesting transaction is one not yet completed, and maintains careful status and history records of all work items, open and closed (Action Technology's Conversation Manager).

Groupware organizes tasks according to the developer's vision of administrative organization: data/forms, agents, work centers, or transactions.

Each approach can ultimately perform the same functions, but the different flavors both reflect and influence a corporate culture. Underlying facilities include parsing data, switching according to parameters contained in the data, message formats and handling, text management, resource allocation -- and the old standbys, networking, communications and database management.

As these systems get more complex, they build their own files -- for storing parameters, subroutines, rules and the like. Staffware, for example, maintains a directory of its users, with passwords, roles, and privileges as well as the database for the applications it manages. It also stores the procedures and rules that control the data flows: Who gets what, under what conditions? What is the time allotted for each step? What reports are necessary to report on the action (number of complaints handled, letters sent, orders processed, etc.). Changing a system is no longer a matter of editing a macro listing and searching for side-effects, but of changing a value or a procedure in a database that describes the meta-application.

Sitting in front of this data structure is a friendly interface that describes what the routines do, allowing an unsophisticated user to point and select, without necessarily understanding the low-level intricacies of what he's doing. Such a system lets the user/builder deal on a conceptual level with the functions he wants to manage. The issue is not what these systems can do, but in what they already do, in a reusable but customizable way.

---

3Note that this is roughly akin to how software might be priced in a world without diskettes: by information content, by user, by site/machine, or by transaction.
THE ROLE OF FORMS

If spreadsheets are the electronic representation of financial management and budgeting, documents the representation of writing, and relational tables the embodiment of database management, then forms are the embodiment of administration and management -- the fundamentals of a company's work. Forms receive and deliver data. Forms also attach to spreadsheets, documents, and table-oriented reports, and control their disposition. Forms, of course, take many forms: Expense reports, purchase orders, requisition slips, phone messages, buck slips, pink slips, etc.

Forms are also the front-end to many data structures: database files, worksheet files, and other more application-specific structures. Many groupware systems are built around the concept of forms, although future versions may abandon it in favor of things more intrinsically electronic (and impossible on paper). While automator systems typically look for strings on a screen, forms-based systems can look more meaningfully for certain values in certain fields regardless of display details, and control actions according to those parameters. Thus we can move from transferring data through pipes to "intelligent pipes," with branches, valves, and sluices in the middle, that manage and guide the flow of information at a higher level.

SOME NAGGING QUESTIONS

The range of tools explored here is broad and their potential is exciting, but not everything will proceed smoothly. For starters, now more than ever we need standards underpinning the customized/customizable parts, so that code can be reused, so that modules can work together with common protocols, so that data can retain its richness as it passes from place to place. Some vendors seem to be going their own way in a market where consistency and predictability should be the norm.

Will mass-market applications vendors adopt these new tools and languages so that their power can be applied on a broad scale? Will each vendor give up his own slight optimizations for the common good, much as Apple has forced the Macintosh community to do at another level? Otherwise all this good stuff will be limited to a much smaller locus of in-house applications and its potential will be wasted.

Will customers perceive the value of the goodies vendors will dazzle them with? Vendors must do a good job of marketing -- but customers must realize that some of the value of these system lies in the changes they will bring, as well as in automation of existing procedures.

Meanwhile, it's disturbing to hear some vendors talk about "developers" in a product area designed expressly to empower users, who are the people best qualified to implement systems to carry out their own unique tactics and strategies. Some vendors are still looking at their work from the wrong vantage point, asking, "How can we extend the operating system?" (our original title for this article, we confess). Instead, they should be asking, "How can we make computers valuable to people?" Operating systems assume people know which application they want to use and how; the new generation will comprehend the user's goals at a higher level and choose the right sequence of application modules to fulfill those goals.

Release 1.0

30 December 1987
PLAYER PROFILES: TEXT

The systems limned here (table on next page) are loosely grouped as follows:

Advanced automators: Advanced Keyworks and Automator mi
Hybrid languages: REXX, MacroBASIC, Lotus LEAF
Groupware tools: Staffware, BCOS/Orgware, Conversation Manager
Object-oriented environments: HP NewWave, Actor/StageManager, VIGI/Objective C, ON Technology platform
Macintosh environments: HyperCard, MacApp

Like most real-world objects, many of them share characteristics of several categories.

ADVANCED AUTOMATORS

Alpha Software's Advanced Keyworks and Direct's Automator mi

"WHILE YOU WERE OUT," says the ad, mimicking a handwritten pink message slip time-stamped 4:07 am, "Your PC hot-keyed your 3270 emulator, logged on to Tokyo (making 2 reconnections when the line went down), extracted yesterday's figures, typed them into Lotus, compared them with your forecast and printed out a graph." All this took 10 to 100 lines of Automator mi code -- plus the facilities of a number of other available applications.

Automator mi, $1995 from Direct Technology, and Advanced Keyworks, $299 from Alpha Software, are the simplest of the tools described here. Unlike most, they can be used by genuine novices. Both have a record mode (Direct calls it "LEarn") which captures keystrokes, supplemented by a friendly prompt-and-select interface for creating mini-applications with branches, pauses, waits for events or specified times and dates, and so forth. However, they lose their touch fairly quickly once you try to do anything complex (and the man who demonstrated Automator to us froze up when we asked about its more advanced features). They offer more power than the DOS command language because they can manipulate applications (with the user's recorded instructions), but they can't understand them.

Unlike competitors such as Advanced Keyworks and Software Recording's Autopilot, Automator can operate in background during 3270 terminal emulation, thereby effectively interacting with a mainframe (rather than just across pc applications), using whatever pc-to-mainframe programs the user may have installed. ("mi" stands for "mainframe interface.) The compiled Automator mi application (written in editable Automator Control Language, a Pascal-like language with the addition of "wait" and "whenever," triggered by hot-keys among other things) requires a runtime module ($200 or less) to run. By contrast, Advanced Keyworks routines require the original or a runtime, which starts at $25 and decreases with volume.

Customers have used these keystroke products not just for automating tasks such as communications and automatic back-up, but also for applications such as training (using branching to provide appropriate follow-on questions),

Release 1.0

30 December 1987
## PLAYER PROFILES: DATA

<table>
<thead>
<tr>
<th><strong>Language</strong></th>
<th><strong>Data Structure</strong></th>
<th><strong>Underlying OS</strong></th>
<th><strong>Price or Avail'ity</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Alpha Software</td>
<td>Keywords</td>
<td>DOS [OS/2]</td>
<td>$299</td>
</tr>
<tr>
<td>Advanced Keyworks</td>
<td>Prog. Lang.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Direct</td>
<td>Automator</td>
<td>DOS [OS/2]</td>
<td>$1995</td>
</tr>
<tr>
<td>Automator m1</td>
<td>Control Lang.</td>
<td></td>
<td>&lt;$200 rt</td>
</tr>
<tr>
<td>Mansfield</td>
<td>REXX</td>
<td>DOS [OS/2]</td>
<td>$125</td>
</tr>
<tr>
<td>Personal REXX</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IBM</td>
<td>REXX</td>
<td>OS/2E</td>
<td>1989?</td>
</tr>
<tr>
<td>REXX for OS/2</td>
<td></td>
<td>dbms</td>
<td>bundled</td>
</tr>
<tr>
<td>Microsoft</td>
<td>BASIC</td>
<td>OS/2</td>
<td>bundled</td>
</tr>
<tr>
<td>MacroBASIC</td>
<td></td>
<td></td>
<td>mid-88</td>
</tr>
<tr>
<td>Lotus</td>
<td>LEAF</td>
<td>Lotus/DBMS</td>
<td>bundled</td>
</tr>
<tr>
<td>LEAF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FCMC</td>
<td>script</td>
<td>proprietary</td>
<td>UNIX [DOS]</td>
</tr>
<tr>
<td>Staffware</td>
<td></td>
<td></td>
<td>$1200 up</td>
</tr>
<tr>
<td>Coord. Tech</td>
<td>proprietary</td>
<td>SQL</td>
<td>OS/2</td>
</tr>
<tr>
<td>BCOS/Orgware</td>
<td></td>
<td></td>
<td>mid-89</td>
</tr>
<tr>
<td>Action Tech.</td>
<td>proprietary</td>
<td>B*-tree or</td>
<td>DOS [OS/2]</td>
</tr>
<tr>
<td>Conversation Mgr</td>
<td></td>
<td>user choice</td>
<td>in 1988</td>
</tr>
<tr>
<td>HP</td>
<td>Agent Task</td>
<td>object manager,</td>
<td>DOS [OS/2, UNIX, MPE-XL]</td>
</tr>
<tr>
<td>NewWave</td>
<td>Lang.</td>
<td>app-dep</td>
<td>$895; rt negotiable</td>
</tr>
<tr>
<td>Whitewater</td>
<td>proprietary</td>
<td>application-</td>
<td>Windows/DOS</td>
</tr>
<tr>
<td>Actor/[StageMgr]</td>
<td></td>
<td>dependent</td>
<td>[$995]</td>
</tr>
<tr>
<td>Obj. C/VICI</td>
<td></td>
<td></td>
<td>DOS, UNIX</td>
</tr>
<tr>
<td>ON Technology</td>
<td>new platform</td>
<td>object-</td>
<td>Mac, OS/2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>oriented</td>
<td>1989 sample app</td>
</tr>
<tr>
<td>Apple</td>
<td>HyperCard</td>
<td>HyperTalk</td>
<td>Mac</td>
</tr>
<tr>
<td>MacApp</td>
<td></td>
<td>flat files</td>
<td>$50</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Obj. Pascal</td>
<td>Mac</td>
</tr>
<tr>
<td></td>
<td></td>
<td>app-dep</td>
<td>$325</td>
</tr>
</tbody>
</table>

**Notes:** This information is subject to change! Most of these systems are unique in one or two categories, but rely on accepted standards elsewhere. In particular, all but Staffware and the Mac and ON platforms will use whatever database the builder/user already has; some will expect that database to be DB2/OS/2E-compatible. Pricing is usually negotiable, especially for large configurations or multiple runtime copies. Versions in brackets will be available soon. (rt = runtime)

---

Release 1.0

30 December 1987
demos, and context-sensitive help. Advanced Keyworks users include Eastman Kodak, Merrill Lynch, and Hyatt (host access). Automator users include IBM Europe (for demos), Coopers & Lybrand (computer-based training), Ernst & Whinney (downloading from multiple hosts).

Advanced Keyworks is the fourth release of a product that first hit the market in 1985 at under $100. The new version garnered the name "Advanced" instead of "IV" because of its programmability. Shipping since May, it had sold 5000 copies by the end of November.

Automator mi has just entered the U.S. market, backed by a vigorous ad campaign and a six-person support operation funded by Direct Technology Ltd. of the U.K. Direct began life as a West Coast company that built pc-compatible terminals; that company eventually disappeared, but its U.K. subsidiary lived on, in part because its personnel were so clever at hacking together effective demos. The tool they used was the precursor of Automator mi. The company has now sold 20,000 copies of the development system, including 3000 in the U.S. (Runtime systems, many of them on site licenses, are impossible to count.)

See also our discussion of Tempo, an excellent macro language for the Macintosh, in our 26 January 1987 issue.

HYBRID LANGUAGES

IBM's and Quercus' REXX -- the SAA standard

The primary appeal of REXX, a command language for the VM/CMS operating system, is that it is part of IBM's Systems Application Architecture and will be sold in OS/2 and System/3X versions by IBM as a common language across systems and applications. With the volume of information in IBM's announcement of Systems Application Architecture, the mention of REXX was generally missed. For the moment, REXX appeals primarily to converted VM users -- a significant but limited market, big enough to support Personal REXX vendor Mansfield and developer Quercus, but not big enough to turn the language into a standard without an IBM version for the PC.

In the IBM/SAA world, REXX will be the language to unify applications across systems, and to manage cooperative processing. It should eventually grow to be the de facto macro language of many applications, each with its own extensions. REXX is a handy tool for linking together existing applications. Unlike, say, Automator or Keyworks, it does not offer a "friendly" menu system that makes developing macros almost foolproof, but it's far more powerful in the hands of a capable user/builder who understands his applications. Unlike most traditional languages, REXX manipulates words easily and uses untyped data, which lets it interact comfortably with a multiplicity of applications and foreign syntaxes. When it encounters unfamiliar terms, it does not blow up but simply passes them to the resident application for execution -- which is usually the right thing to do.

REXX was created within IBM U.K. by Michael Cowlishaw, and quickly adopted by other VM/CMS programmers in lieu of existing VM command languages Exec and Exec2, which were harder to use. It is also the macro language, for example, of the VM editor, XEDIT. Its simple but powerful syntax has proven

Release 1.0
30 December 1987
extremely popular with VM users. By now, estimates Kearney, VM REXX has close to 200,000 users (reportedly including the person who wrote a 40-line REXX program that masqueraded as a Christmas card and clogged IBM's internal network earlier this month). These users form a ready market of mainframe-oriented people both for the third-party version from Quercus and for IBM's own PS/2-OS/2 version, which should be out by 1989 (following IBM's policy of implementing announced SAA directions within two years).

Personal REXX is a PC-DOS version of REXX that was ported by Charles Daney (a.k.a. Quercus Systems of Saratoga, CA) and has been distributed by Mansfield Software Group of Storrs, CT, since the fall of 1987. The company has sold about 10,000 copies, including many to users of its other product, KEDIT, a pc version of XEDIT. Sales have prospered since the SAA announcement, says Mansfield president Kevin Kearney, especially among people who no longer need to worry about "getting left high and dry by a small company" if they adopt REXX. Developer Charles Daney has attended the Microsoft OS/2 training and intends to produce an OS/2 version before too long.

Microsoft's MacroBASIC -- basically a shoo-in

Sometime in 1988 Microsoft will be offering its MacroBASIC, also a product as yet announced as a direction only. (At least there's a clear model for PC REXX on the mainframe.) MacroBASIC, an extended version of BASIC, is being designed to perform the same sorts of functions as REXX, providing users and systems builders a way to communicate with both the OS and the applications. Microsoft hopes that many application builders will adopt some of the MacroBASIC syntax in their own macro languages, so as to make things simpler. Each application's macro language would then be a superset of MacroBASIC, fostering easy communication and coordination -- "dynamic command exchange" as well as the Dynamic Data Exchange that Microsoft currently offers within Windows and OS/2.

Recently, Microsoft's Bill Gates has been writing and talking in public about MacroBASIC, in effect encouraging people to adopt the BASIC syntax where possible (and to use Microsoft's BASIC compilers and interpreters, to be sure). QuickBASIC 4.0 was a completely new product from its predecessors, developed starting in 1984 with this role in mind. Gates and Co. clearly recognize the need for some such language, although they have also recognized the impossibility of getting the world to standardize on a single language: There's just too much variety in the kinds of things programs and applications are designed to accomplish. For the moment, Microsoft is working on a Presentation Manager version of BASIC, and canny users will note that Excel's macro language has a lot in common with BASIC. (For internal application development, the company uses C with informal object-oriented extensions.)

Just as Microsoft and IBM will offer their database managers and communications facilities as extensions to OS/2, so will each have its own shell language (although both will use SQL for data manipulation). Which will prevail? Naturally, IBM will have the greater share among true-blue accounts familiar with REXX on mainframes, while Microsoft will sell to the clone users and pc-environment types. Both companies haven't yet formulated their marketing plans, but we expect both of them to include these languages as part of the OS rather than as separately priced add-ons. Will there be any...
other offerings? We expect Quercus to flourish with its Personal REXX (for people who don't buy OS/2E from IBM), and Borland will no doubt provide its own shell BASIC, just like Oracle/Novell in the database/communications world. With second sources, these two "standards" should be quite enough.

Lotus and LEAF -- off the main branch...or on the leading branch?

A third seeming standard is Lotus's LEAF (for Lotus Enhanced Application Facility). However, Lotus is careful to position LEAF as a multi-application language rather than as a cross-application language. LEAF will enable builder/users to hook together and use the facilities of its own suite of new Lotus applications (1-2-3 Release 3 and beyond), or applications specifically written by others to conform. (This will of course include a full-scale SQL-oriented database that will at least give the Lotus application suite that much of a hook into the "standard" world.) The sorts of batch processes in its new graphics products are an example (see Release 1.0, 21 September, page 3).

By way of perspective: 1-2-3 became a standard -- and de facto an environment -- in a world where the most interesting data on pcs was numbers and formulas. The locus of interesting data has now changed to data in databases, where IBM sets the standard (with SQL). However, in the future text will gain in importance, and LEAF or its Lotus successor will likely prosper to the extent that it focuses on text manipulation, an area where Lotus is leading the way with Agenda and its Information Services division (while Microsoft's success with Word is based on strength with graphics).

GROUPWARE TOOLS

FGMC's Staffware -- the paper-shuffler

Staffware was built in 1983-84 by Tony Kobine, an Englishman who also wrote Finar, a well-regarded financial planner before the days of spreadsheets. Staffware is sold by Unisys as Procedure Manager, with a total of 60 installations so far, and will soon also be available from Panasonic and ICL under other names. (The Unisys folks are extremely enthusiastic about Staffware and want to move it to other machines in their line. Current prices on UNIX machines range from $1200 for 8 to 12 people to $5000 and up for 64 or more.) Staffware hasn't gotten the recognition it deserves in part because the concept of groupware is just gaining recognition, and in part because the necessary platform -- groups using networks -- has just started to proliferate.

Staffware is a development and runtime environment devoted to creating lifecycles for pieces of paper. That is, the developer focuses on the paper, rather than the workstation, in deciding what happens. In fact, the system is built around a proprietary database into which the electronic forms provide differing views depending on the viewer and the actions to be taken. But unlike a plain old database, Staffware includes routing logic, a lot of date and time arithmetic, and e-mail facilities that allow it to monitor and control the progress of the work in process.

Release 1.0

30 December 1987
For example, take a customer's order for 500 copies of "How to Code for Success." Juan takes the order (by phone or from a mailed-in purchase order\(^4\)) and enters it into a form on the screen. A "normal" application sits behind that form, and does the usual work -- checking the data for simple errors, calculating prices, looking up the customer's address and credit limit, etc.

But Staffware does more than that. The results of the credit check determine who should receive the form next: Alice in credit or Terry in order processing. If the order is more than 10 percent over the credit limit, Alice must approve it, based on a credit history that the system sends to her along with the current order. Terry, by contrast, gets only the order and shipment information that will enable him to fill the order -- along with a message that the company has only 502 copies of stock. Does he want to requisition more from the central warehouse? If he says yes, he'll get the appropriate form to do that, too. If the central warehouse is low on stock, the system then sends a message to the company's other warehouses to see if they have surpluses.

Obviously, this didn't happen automatically the first time. Someone painstakingly used Staffware to set it up, using a combination of Staffware's proprietary database, its e-mail facilities, and a prompted scripting language. For each step and branch of a process, the user defines a recipient, a form with data to display or enter, conditions, etc. The system checks to make sure that none of the branches ends in limbo, and prompts the builder/user to set deadlines for the end-users to act on the forms they receive. (If they don’t act, they or their supervisors can be notified by a form.)

Currently, Staffware operates under UNIX, with several MS-DOS versions in the works. Staffware is essentially a multi-user script-building system (with a knowledge of the underlying data and multiple, cooperating agents). In another sense, it's a sophisticated switching system, with rules, stored parameters, and a good sense of timing. While the systems above can respond to an event and in that sense "wait" for something, Staffware is much more active. If something scheduled doesn't happen, it wants to find out why: Who has the ball?

Staffware's major flaws are its reliance on a proprietary database and a certain clumsiness of interface, even in its forthcoming pc versions. Kobine says he considers the current version of Staffware "a mature prototype" and has bought the technology (but not rights to the current product) back from its owners, U.K.-based FCMC plc. He will continue to do business in the U.S. and Asia as Berkeley-based FCMC Inc. Now talking to prospective marketing and development partners, Kobine is itching to get started on a new, improved OS/2 version that will be built around a standard SQL database and standard communications protocols including X.400 for e-mail and X.12 for electronic data interchange. He has visions of using Staffware or its successor as an intelligent switcher across wide-area networks -- a system that could manage inter-company distribution or purchasing, for example, or a good back-end to a public network facility.

\(--------

\(^4\) Of course, eventually the customer will enter the order himself, and transmit it electronically.
Coordination Technology -- the work is in the links

While NewWave sees the world in terms of agents and documents, and Staffware sees it as data-based forms, Coordination Technology (Trumbull, CT) sees it as a factory of autonomous work cells. Just like an agent, each work center knows its tasks, but a center is a virtual location rather than a virtual person or a virtual form. The center accomplishes a number of defined tasks and supports a number of defined roles, and coordinates with other centers. The fundamental philosophy here (Coordination, not Control Technology) is that work cells are interacting but autonomous units that can be built by their own "users"...although of course a user company could force its employees to use centrally designed "Organizationware," or scripts.

The CT platform consists of BCOS (for Business Coordination Operating System) and Organizationware. BCOS attempts to present to builder/users a higher-level vision of their systems, so that they can think in terms of centers, roles, tasks, etc., rather than files, applications and programs. Orgware is to BCOS as templates are to 1-2-3: generic templates and scripts for creating centers and tasks that can easily be customized or extended by users in a simple language with about 30 verbs. The Orgware, some of which will come with each system, should be about all that half the customers ever need, says president Roger Moody (55, formerly with IBM, AT&T, and two start-ups). Others may build their own. The interface isn't all there yet, Moody acknowledges, but the goal is to offer the same kinds of helpful forms and prompts as supplied with Automator, Advanced Keyworks and Staffware.

Coordination Technology is the creation of Anatol Holt, 60, a longtime explorer of the field who brought the concept (and term) Petri nets to the United States.\(^5\) Two years ago he left ITT (where he worked with the founders of PPI, below) to found Coordination Technology, and has garnered $4 million in funding from E.S. Jacobs & Co. The company doesn't expect to ship in volume until mid-1989 (with an OS/2 foundation), but it is talking now to commercial alpha and beta sites for installations starting in March. In addition to OS/2, BCOS will assume the presence of a standard SQL-style database, plus standard networking software and mass-market productivity tools such as 1-2-3. Special-purpose applications could be built around other PC software. Pricing has not yet been set. CTI also has a letter of intent with DEC to enter into a development and remarketing agreement for a VMS/VAXstation version.

At least at the start, Coordination Technology sees its product as part of a support-intensive relationship, where the customer is prompted to understand its own internal dynamics, create and link the work centers, and finally select and customize the Orgware for each center. Will customers like this idea? Probably some will and others won't. Like any successful move to automation, the use of BCOS will require an understanding of the processes to be automated. Coordination Technology's goal is merely to make it easier to turn that understanding into an automated system -- and to work from a higher-level understanding than the traditional one of files, records, and other electronic elements.

---

\(^5\) Petri net theory, a type of applied combinatorial mathematics, can be used to model coordinating, concurrent processes, and to ensure that interactions reach closure.

Release 1.0 30 December 1987
Action Technologies' Conversation Manager -- animated transactions

We first wrote about The Coordinator in September 1986, comparing it with the MIT/Sloan School Information Lens project. In brief, The Coordinator focuses on "speech acts" -- the implicit commitments and requests people make -- without regard to content, building a transaction-oriented, time-sensitive system to coordinate people; by contrast, Information Lens is more focused on filtering messages and building a database derived from the content of the messages it handles. With Information Lens, you can filter out urgent messages, but what you do with them is up to you; The Coordinator wants you to respond to them or offer a reason why (a counteroffer). A project, Information Lens could be extended to do anything; a product, The Coordinator is tightly focused on managing transactions within a group of people working together and relying on each other's work. Commonly seen as an extended, intelligent e-mail package, the Coordinator in fact includes its own database for logging and keeping track of users' speech acts (the Conversation Manager).

Now Action Technologies has gone backwards and refined out some of The Coordinator's underlying facilities, notably the Message-Handling Service (MHS), about to become a de facto standard for transporting messages across a variety of environments, including Novell/NetWare and some others to be announced. While MHS will sell immediately in greater volumes (and be bundled with every shipment of Novell's NetWare from January on), Conversation Manager is more interesting in terms of higher-level representation of commercial activity.

Until now the Coordinator's selection of capabilities implicitly indicated that content belongs to the participants and will take care of itself if people can communicate effectively through explicit requests, commitments, etc. However, with customization (i.e., embedded knowledge) the tool can represent and handle recurring content through forms and prompts, which "let the participants put more of the content into the machinery," says Chauncey Bell, vp for strategic development. Currently, Action is showing the tool to OEMs as it continues with internal development for PC, OS/2 and Mac environments, due sometime in 1988.

The Conversation Manager maintains the "status" of conversations and determines what responses are required (who needs to answer whom, when a conversation can be deemed closed, etc.). As it happens, although the ANSI X.12 standard for Electronic Data Interchange lists some 500 kinds of transactions (orders, confirmations, invoices, bills, purchase orders, requests for proposals, etc.), they can all be expressed as specialized forms of the Conversation Manager's four basic speech acts -- requests, commitments/promises, assertions, and declarations. The Conversation Manager could also be used within a company (or industry) to implement special routines for internal communications -- for example, within our business, requests for sample issues (which also put the requester into our database of subscrip-

6 With about 20,000 nodes installed, The Coordinator ($995 per server, up to 50 people) is about to grow further with the recent signing of a number of still undisclosed OEM deals. Information Lens currently has about 20 regular users, including several at Xerox PARC (which co-funded the project). A second, more robust version of the project, which deals with more kinds of objects, is nearing completion. More on this when it happens...

Release 1.0

30 December 1987
tion prospects), address changes (which are circulated to both editorial and publishing people), and so forth. CM will come with its predefined conversation types, a customization language, and a tool for creation and parsing of forms, and can work with the builder/user's own database. Individuals can also use the CM locally to organize their own conversations.

While conversations are defined as transactions in the CM database, they are unlike most database transactions in that they are most interesting before they are reconciled: That is, most work revolves around concluding transactions rather than (in traditional systems) recording them once they are somehow or other completed. The file of open conversations constitutes the company's workload at any given moment. In fact, consider this analogy: An office is like a factory, where one of the goals is to turn inventory over quickly, and to keep its levels low. Open conversations are the workgroup's inventory, and response time (to bills, orders, complaints) is a measure of its turnover rate.

OBJECT-ORIENTED ENVIRONMENTS

Hewlett-Packard's NewWave -- agents abroad

Hewlett-Packard's NewWave was announced last month to considerable fanfare. NewWave is exciting for a number of reasons, including its provenance from a well-known, resource-rich company that is willing to spend considerable funds and effort to make its product a standard. Indeed, HP is hoping to license NewWave not just to independent software vendors but also to other hardware makers, including direct competitors (none yet announced, however).

HP's NewWave, as befits a company that sells more than just pcs, is designed ultimately to integrate systems running on a variety of HP hardware and standard operating systems including not just DOS and OS/2 but also UNIX and MPE-XL (HP's 3000/Spectrum environment). Written in C, it comprises a development environment for building systems around existing applications, and software agents (just like some other folks' software robots). Although lots of enhancements are planned, the first release of NewWave is basically a powerful, flexible script language and glue to manage existing applications, combined with a rich interface manager. Users can build agent scripts in a record mode, or use the Agent Task Language, which began as BASIC and then evolved, according to section manager Larry Lorren. (The company is keeping close tabs on the progress of Microsoft's MacroBASIC.)

Within NewWave, collections of data (that is, each DOS file) are treated as objects -- data plus the associated application, or "procedures," in object-speak. (An object manager, a set of relational tables with pointers to the appropriate data and program files, manages these objects and binds data with applications as necessary.) Thus if you wish to examine some data, you can either look at it "flat," or unstructured, in an Ascii file, or complete, in its rich form with the application included. And you can incorporate that data into an existing document. Moreover, NewWave can create hot links, between spreadsheet data and an associated graph, for example. The whole system seems built more around office automation than around actual transaction processing. Its richest feature is the interface -- an extensive use of Windows 2.0/Presentation Manager to build a helpful series of
menus, icons, and other meaningful graphical objects that can easily be manipulated by a user.

NewWave's initial approach is to build hybrid documents that incorporate information in multiple structures from multiple sources; there's little of the coordination notion here even though NewWave does include "agents." These agents are essentially scripts that can carry out assigned tasks; any cooperation is implicit. This gives NewWave the ability to manipulate and coordinate applications, but the system is basically an environment for managing applications rather than a system in which to build them.

Moreover, even the coordination doesn't offer much flexibility, although in the long run, HP says, they'll include "artificial intelligence." We're not sure what that means, although we suspect it may have something to do with the kind of branching and routing capabilities Staffware already offers, as well as a natural-language interface. "We'll add knowledge bases and world models, so the agents can be more ambiguous," says developer Bill Crow. "Our Agent Task Language is the muscles [in conjunction with the applications it drives]; we still need an AI brain."

Longer run, we expect to see a set of generic agents (i.e., application managers) that can be reused or customized for various kinds of tasks. Pricing is $895 for the developer's kit at retail (or whatever an OEM decides to charge), available in the first half of 1988. Runtime prices will probably start at $200 and decrease with volume.

**Productivity Products' Objective C -- connectivity classes**

In 1981 Tom Love, a development manager at ITT, was working on a software engineering environment and wanted to infuse it with coordination technology (lower-case). He hired Coordination Technology founder-to-be Tolly Holt. But ITT lost its enthusiasm and in 1982 Love left to go to Schlumberger, followed by another ITTer, Brad Cox. A year later the two started Productivity Products International, in Sandy Hook, CT, to build an object-oriented development environment rich with reusable "software ICs," as the company calls its object classes.

PPI's foundation product is Objective C, an enhanced version of the C language with object-oriented extensions, and an interactive development environment, VICI. Its class libraries (ICpaks) include graphical objects implementable under Sun Windows and X Window (with Windows/Presentation Manager and Macintosh versions on the way). Other software ICpaks include the "foundation" library, which contains collection classes (sets, lists, and other groups of data elements that can be manipulated as a whole), filers (for saving on objects on disk), and other low-level programming amenities.

Chairman Love and chief technical officer Cox have always retained a soft spot for coordination technology, and Cox is now at work on a new ICpak for Objective C -- connectivity classes. These are classes that contain methods for negotiating a network and talking to tasks and data on other, heterogeneous machines, and will help open up object-oriented programming to working across networks -- and make it a lot easier for people to develop for such environments, just as graphical objects make it easier to write port-
able graphics-based applications. Programmer's lore says that it takes one line of code to print "Hello world" in BASIC, 50 to 100 in a window within Windows ...and two within Windows with the help of Actor. With a connectivity class you could do it all in one line and print it out on your neighbor's incompatible machine, by sending a message to a printing object that is in fact a connectivity object which knows how to reach your neighbor's machine and print it out there.7 (A fundamental characteristic of classes and objects is that they hide their details.) Now try that in pure C!

The Whitewater Group's Actor and StageManager -- objects of creation

In 1984 two managers at Kriya Software left to work on an object-oriented language. In 1985 they bet on Windows, which as a programming environment provided an excellent opportunity for friendliness-added. Thus began The Whitewater Group (Evanston, IL) and its initial product, Actor (for further details see Release 1.0, 24 March). Actor takes the object-oriented paradigm far deeper than HP's NewWave, the closest Windows-oriented contender. While NewWave treats DOS files as objects with associated applications, the data structures and applications in those files are the same old standards. But applications and data can be a lot richer, and their creation much easier, with the use of object-oriented languages and development tools. The object-oriented approach lets one think about data instead of application flows -- the most natural way except for experienced programmers, who may find object-oriented programming confusing and not worth the effort for small applications.

While PPI considers its center of gravity to be its rich set of classes, Whitewater has focused on its rich object-oriented development environment. With its own language, Pascal-like in syntax, Actor requires more "reeducation" among its developers.8 Actor does not just enable people to develop for Windows, it lets them do so within Windows, with a set of interactive, graphically oriented tools including code browsers, debuggers, inspectors, and hierarchical representations of classes.

For the moment, Actor works only under/for DOS, and lacks much of the richness it will have (to say nothing of better performance once it gets memory relief). Whitewater's ultimate goal is to build an environment-independent development system -- StageManager. Builder/users will then have the option of implementing the system on a variety of choices, including OS/2 (first, in mid-1989), the MacIntosh, and possibly UNIX, VMS, etc. Just as an operating system has device drivers for peripherals, StageManager will have "device drivers" for many operating system functions and database calls, and better text-editing and graphics-editing. It will also have, developer Chuck Duff promises, a library of business-model objects.

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

7This may seem unfair because you're using painstakingly created reusable code, but that's the whole point.

8PPI's Objective C-oriented toolset sits at the middle of the spectrum between Actor and the C++ language from AT&T via Lifeboat and Oasys (Release 1.0, 24 March), but none of these three tools is likely to be comprehensible to your typical end-user. Actor is also Windows-specific, whereas C++ and Objective C are more general-purpose, although they rely on the familiar C language rather than Actor's Pascal-like language.

Release 1.0 30 December 1987
ON Technology -- show me!

It's not just windows and menu formats that are common from application to application but all kinds of data elements -- employees, balance sheets, corporate structures with divisions and P&L centers, products. So why not create rich data elements just as one creates graphical objects? That's a driving notion behind the platform planned by ON Technology, the company just founded by Mitch Kapor and Peter Miller (employee 1A) in Cambridge, MA.

However, since we described ON in our last issue (25 November), the company has changed its immediate corporate focus. Demonstrating unusual humility for this industry and an ability to change their minds quickly, Kapor and Miller have responded to "constituency concerns" among third parties about the breadth of their plan to create a rich, object-oriented development environment, and have decided on a depth-first attack instead. Rather than polish up a development environment with interface and documentation by late 1989, they hope to polish up an application, "thin but complete single vertical slice," for delivery in mid-1989, says Kapor. The slice will cut through everything -- object-oriented database, user interface, a rich knowledge-representation system and scripting facility. It will presumably demonstrate the validity of their vision, says Miller, but it will also be a self-supporting commercial venture.

As you might imagine, the application won't be another accounting package or spreadsheet but rather a service that will deliver information on a particular subject/industry yet to be chosen in a rich and useful way. Is this just a replay of Agenda with dial-up attached? Not quite. Rather than download text or data from a central database, the ON service will let users download objects into a local object-oriented environment from a larger-scale object-oriented database carefully built by subject-matter experts from a variety of sources. Take the idea of downloading data into a 1-2-3 template, and then imagine that instead you can download information into a system that knows, for example, how to combine parts to build a machine, or how to parse an income statement (not the same as matching row labels), or how to select relevant anecdotes to flesh out a speech outline (special price for politicians).

The fundamental premise behind ON is that there's value in real-world knowledge as well as in tools. In this case, there's valuable knowledge not just in data but in the structure and behavior of the data which can make it more useful and susceptible to manipulation by users. Some of the tools contain the knowledge, and others help people to represent it better. In the same way, the value of ON's contribution will depend not just on its ideas, which are terrific, but on their implementation -- as will be demonstrated by ON's first application. We're hopeful.

MACINTOSH ENVIRONMENTS

Apple's HyperCard and MacApp -- two ends of the spectrum

How does MacApp, the development tool for the Mac (see Release 1.0, 24 March) compare with HyperCard (10 August), which also includes sample applications? Both contain application modules that can be reused, customized,
etc., but there's a huge difference in power and capabilities on one hand, and in accessibility to regular people on the other hand. MacApp (with associated tools) is a full-fledged programmer's toolkit, a traditional language-oriented tool without rich graphics, although it does include Mac user interface primitives (menus, windows, and the like). By contrast, HyperCard can run procedures, and lets nonprogrammers create useful personal applications, but it's not meant for large-scale efforts. HyperCard is much more graphical and accessible, courtesy of its English-like HyperTalk.

But the real point about HyperCard -- and its great value -- is that it is leading us in the right direction towards accessibility, a direction mostly missed by all the other tools discussed here. Remember the old saw about the AT&T study that determined early this century that at current rates of growth every citizen of the U.S. would be a telephone operator by mid-century? Well, with the help of dial and Touch-Tone phones, that has come true. In the same way, with the help of HyperCard and other tools that follow its lead, every citizen of the U.S. will be a computer programmer by early next century. But few of them will know it.

* * * *

RELEASE 1.1: ERRONEOUS ADVISORS

In our October issue we referred to IBM's "DASD Advisor" as a product built with IBM's Expert Systems Environment. We meant, of course, IBM's DEFT, Diagnostic Expert for Final Test (of DASD), described in Release 1.0, 2 December 1986. There is a DASD Advisor, but it is sold by Boole & Babbage and was developed with Aion's ADS (Release 1.0, 26 January). We apologize for the error.

RELEASE 0.5: Overheard in the Valley

"Whatever he gets me will probably come from Fry's. It's the only place he feels comfortable shopping."

Release 1.0 30 December 1987
FORUM DETAILS

Forum invitations have been mailed to subscribers, who are entitled to two paid registrations per subscription as long as space is available. If you have not received yours and you would like to attend, please look around carefully and then call us for a copy. (We cannot handle registrations by phone.) We are already more than half filled up, so please hurry!

The Forum will take place February 21 to 24 in Naples, on the west coast of Florida (so that the sun sets over the water). The theme is "Worlds in Collision: From PC to Workstation." We will explore broadening definitions of personal computing, including some of the new approaches to development and new metaphors for groupware covered in this issue. Come and contribute!

Confirmed speakers include:

Victor Alhadeff  
Bob Berland  
Gordon A. Campbell  
Ben Rosen/Mike Swavely  
Vittorio Cassoni  
David Chapman  
Peter Coffee  
Michael Dell  
Bob Epstein  
Edward M. Esber  
Gordon Eubanks  
Robert Flast  
William Gates  
Philippe Kahn  
Jerry Kaplan  
Mitch Kapor/Peter Miller  
Barry Kotar  
Bill Krause  
Jim Manzi  
Mike Maple  
Scott McNealy  
Bob Orbach  
Vern Raburn  
John Roach  
Mort Rosenthal  
John Sculley/Larry Tesler  
Edward Tufte  
David S. Wagan  
Kenneth R. Waters  
Joyce Wrenn  
Haviland Wright

Egghead Discount Software  
IBM Application Systems  
Chips & Technologies  
Compaq Computer  
AT&T  
Cullinet  
Aerospace Corporation  
Dell Computers  
Sybase  
Ashton-Tate  
Symantec  
American Express  
Microsoft Corporation  
Borland International  
GO Corporation  
ON Technology Inc.  
Covia (United Airlines)  
3Com Corporation  
Lotus Development Corp.  
IBM Entry Systems  
Sun Microsystems  
47th Street Computer  
Cooper & Raburn  
Tandy Corporation  
Corporate Software  
Apple Computer  
Yale University  
Softsel Computer Products  
ComputerLand  
American Airlines  
Avalanche Development

In addition, there will be a special panel, "Beyond Numbers," on text-based applications, including e-mail, hypertext, and automated paper-shuffling (process management). In the afternoons you may attend parallel company presentations and demonstrations of products and vaporware by some of the speakers listed above and by other companies. Details next month.

Release 1.0  
30 December 1987
RESOURCES & PHONE NUMBERS

Chauncey Bell, Action Technology, (415) 654-4444
Richard Rabins, Alpha Software, (617) 229-2924, (800) 451-1018
Larry Tesler, Bill Atkinson, Apple Computer, (408) 996-1010
Tolly Holt, Roger Moody, Coordination Technology, (203) 268-4045
David Inbar, Direct Technology, (212) 475-2747
Tony Kobine, FCMC Inc., (415) 549-7100
Bill Crow, Larry Lorren, Hewlett-Packard, (408) 773-6284/6345
Eddie Currie, Kathy Ross, Lifeboat/Hudson, (914) 332-1875, (800) 847-7078
Conall Ryan, Ed Belove, Lotus Development, (617) 577-8500
Kevin Kearney, Mansfield Software, (203) 429-8402
Greg Lobdell, Bill Gates, Microsoft, (206) 882-8080
Peter Miller, Mitch Kapor, ON Technology, (617) 225-2545
Tom Love, Productivity Products International, (203) 426-1875
Charles Daney, Quercus, (408) 257-8440
Donel Moss, Jean-Francois Guilleux, Unisys, (215) 542-5594
Chuck Duff, Mark Achler, The Whitewater Group, (312) 491-2370

(A videotape of Chuck Duff’s background presentation on StageManager is available from the Boston Computer Society, (617) 367-8080.)

COMING SOON...

- Connectivity: Promises, promises.
- Parallel processing.
- Channels -- Micro and otherwise.
- Nitty-gritty experts: Are they intrinsically friendly?
- Airline experts.
- PC Forum program: Speaker profiles.
- And much more...

Release 1.0 is published 12 times a year by EDventure Holdings, 375 Park Ave., New York, NY 10152; (212) 758-3434. It covers the pc, software, CASE, groupware, text management and connectivity markets, and artificial intelligence. Editor & publisher: Esther Dyson; associate publisher: Sylvia Franklin; circulation manager: Hyacinth Frederick; copy chief & consulting editor: William M. Kutik. Copyright 1987, EDventure Holdings Inc. All rights reserved. No material in this publication may be reproduced without written permission; however, we will gladly arrange for reprints or bulk purchases. Subscriptions cost $395 per year, $475 overseas; multiple subscription rates on request.
January 11-12 Neural networks - Los Angeles. A commercial assessment of NN applications, with most of the major players, followed by a separate three-day course on the technology. Sponsored by the Institute for International Research. Contact: Russell Webb, (212) 883-1770 (note correct number, corrected from previous issue).


January 27 Philippe Kahn at the Boston Compute Society - Boston. Contact: Janet Cole and Pam Bybell at (617) 367-8080.

February 8-10 IFIP conference on computers and law - Santa Monica, CA. Issues that just won't go away: Copyright, contracts, taxation, computer crime, legislative actions. Sponsored by IFIP and Los Angeles County Bar Law and Technology section. Contact: Michael Krieger, (213) 208-2461.


February 21-24 ELEVENTH ANNUAL PERSONAL COMPUTING FORUM - Naples, FL. We moved it in search of variety and better weather. Registration forms have been mailed to subscribers (two per subscription). For further information, please see page 26 or call Forum director Sylvia Franklin at (212) 758-3434.


February 25-27 Workshop on technology and cooperative work - Tucson, AZ. Sponsored by Bell Communications Research and the University of Arizona. Contact: Robert Kraut, (201) 829-4513 or Jolene Galegher, (602) 621-7477.


March 3-5 ABCD visions '88 - Newport Beach, CA. Sponsored by abcd, the microcomputer industry association. Contact: Bernie Whalen, (312) 24-1818.
<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
<th>Location</th>
<th>Contact Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>March 7-10</td>
<td>IEEE conference on computer workstations - Santa Clara. Sponsored by IEEE. With Sun's Bill Joy, and sessions on distributed systems, computer-supported cooperative work, and OS/2. Contact: Pat Mantey (408) 429-2158 or Robin Williams, (408) 927-1842.</td>
<td>Santa Clara</td>
<td></td>
</tr>
<tr>
<td>March 7-11</td>
<td>Seybold Seminars '88 - Santa Clara. With Jonathan Seybold. The seminars of record in the electronic publishing business. Contact: Craig Cline or Lena Lillsunde at (213) 457-5850.</td>
<td>Santa Clara</td>
<td></td>
</tr>
<tr>
<td>March 8-10</td>
<td>Connect '88 - New York City. Sponsored by Cahners, with Datamation and the Gartner Group. A trade show on connectivity and integration directed at corporate end-users. Contact: Richard Molden, (203) 964-0000.</td>
<td>New York City</td>
<td></td>
</tr>
<tr>
<td>March 14-18</td>
<td>Artificial intelligence applications - San Diego. Sponsored by IEEE. Contact: Richard Greene, (301) 468-3210 (exhibits) or IEEE, 371-0101 (program) or Paul Harmon (415) 861-1660.</td>
<td>San Diego</td>
<td></td>
</tr>
<tr>
<td>March 16-18</td>
<td>Desktop presentations - San Jose. &quot;Computers reach for new media.&quot; Sponsored by CAP Interantional. Contact: Jean O'Toole, (617) 837-1341.</td>
<td>San Jose</td>
<td></td>
</tr>
<tr>
<td>March 16-23</td>
<td>Hannover Fair CeBIT - Hanover, West Germany. Contact: Donna Peterson Hyland, Hannover Fairs USA, (609) 987-1202.</td>
<td>Hanover, West Germany</td>
<td></td>
</tr>
<tr>
<td>March 20-23</td>
<td>ADAPSO SPRING CONFERENCE - Palm Desert, CA. Software and services vendors at the oasis. Contact: Sheila Wakefield, (703) 522-5055.</td>
<td>Palm Desert, CA</td>
<td></td>
</tr>
<tr>
<td>March 27-30</td>
<td>Software Publishers Association spring conference - Berkeley, CA. Contact: Jackie McDonald, (202) 452-1600.</td>
<td>Berkeley, CA</td>
<td></td>
</tr>
<tr>
<td>March 28-31</td>
<td>World Congress on Computing - Chicago. Interface Group's response to NCC. Contact: Jane Wemyss, (617) 449-0600.</td>
<td>Chicago</td>
<td></td>
</tr>
<tr>
<td>April 7-10</td>
<td>13th West Coast Computer Faire - San Francisco. Contact: Jason Chudnofsky at Interface Group, (617) 449-6600.</td>
<td>San Francisco</td>
<td></td>
</tr>
<tr>
<td>April 11-14</td>
<td>AIIM show - Chicago. Information and image management. Sponsored by Association for Information and Image Management. Contact: Sue Wolk or Betty Garrett, (301) 587-8202.</td>
<td>Chicago</td>
<td></td>
</tr>
</tbody>
</table>

Release 1.0 30 December 1987
<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>April 11-15</td>
<td>IEEE Tenth international conference on software engineering - Singapore. From an international perspective. Sponsored by IEEE and NCB Singapore. Contact: Tan Chin Nam or Lim Swee Say, (65) 772-0200.</td>
<td></td>
</tr>
<tr>
<td>April 19-21</td>
<td>CEPS/Spring '88 - Chicago. Corporate electronic publishing systems. Sponsored by Cahners and InterConsult. Contact: Mike Driscoll, (203) 964-0000, or Paula Wertman, (617) 547-0332.</td>
<td></td>
</tr>
<tr>
<td>April 27-29</td>
<td>Seybold Technology Forum - Cambridge, MA. &quot;Distributed network computing: A journey into the future.&quot; Sponsored by Patricia Seybold's Office Computing Group. Discussions ranging from communications protocols to computer-supported cooperative work. Contact: Catherine Cooper, (617) 742-5200.</td>
<td></td>
</tr>
<tr>
<td>May 3-6</td>
<td>CASExpo - Dallas. Managed by Arthur Young &amp; Co. Contact: Ken Burroughs, (703) 845-1657.</td>
<td></td>
</tr>
<tr>
<td>May 9-12</td>
<td>Comdex Spring - Atlanta. Peaches and PCs. Contact: Jane Wemyss at the Interface Group, (617) 449-6600.</td>
<td></td>
</tr>
<tr>
<td>June 6-8</td>
<td>Artificial intelligence in electronic publishing - San Jose. Sponsored by the Graphic Communications Association. Applying AI to design, content, process, etc. Contact: Marion Elledge, (703) 841-8160.</td>
<td></td>
</tr>
<tr>
<td>June 19-22</td>
<td>Congress VI - Paris. The world computing services industry gets together. Meet your potential partners or competitors abroad. Sponsored by national trade organizations, include our own Adapso. Contact: Phyllis Cockerham, (703) 522-5055, or Diana Kirby, London, (441) 405-2171.</td>
<td></td>
</tr>
<tr>
<td>July 12-15</td>
<td>CASE '88 - Cambridge, MA. Second international workshop on computer-aided software engineering. More academics and less hype than most CASE conferences, for better or worse. Sponsored by a number of academic institutions. Contact: Pamela Meyer, Index Technology (organizers), (617) 494-8200, x 454.</td>
<td></td>
</tr>
<tr>
<td>August 22-26</td>
<td>AAAI-88 - St. Paul, MN. The seventh annual. Sponsored by the American Association for Artificial Intelligence. Contact: Claudia Mazzetti, (415) 328-3123.</td>
<td></td>
</tr>
</tbody>
</table>

Release 1.0

30 December 1987
September 26-28  Second conference on computer-supported cooperative work - Portland, OR. Sponsored by ACM. Contact: Suzanne Sylvia, (617) 225-1860.


October 11-14  Info Show - New York City. Contact: Frank Fazio, Cahners Exposition Group, (203) 964-0000.

October 23-28  Monterey Classic - Monterey, CA. Contact: John Baumeister, (408) 987-4200.

Please let us know of any other events we should include. -- Anita Fowler

Release 1.0

30 December 1987
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? _________________________________________

SE 87-12

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.

Sylvia Franklin
Associate Publisher

**Release 1.0**
30 December 1987