TEXT IN ALL ITS GLORY

Most of this issue is about objects in some sense or other: object-oriented databases, object-oriented text management, model- or object-based expert systems and so forth. Most object-oriented systems are sold to engineers with Sun workstations who produce hard-to-handle engineering diagrams rife with cross-references, structural dependencies and multiple versions. But before we launch into the heavy-duty stuff -- engineering information management systems and industrial-strength object-oriented databases -- we'll start with something simple and familiar: text.

At least that's how we think of text. We all use it, and most of us can manage a letter or a memo on a pc. Yet, in a very simple way, even a word-processor treats text as objects: There are paragraphs and sentences (you can select one to move or delete), generally defined by periods or carriage returns. More sophisticated systems know about columns and tables, headers and footnotes. Layout systems with rules handle headlines and other text items as discrete, identifiable objects. But even as a writer builds his text from words into paragraphs, sections and chapters, a composition system reassembles it into lines and pages, text and graphics blocks.

There are semantic objects and then there are visual objects. What makes it confusing is that they don't map into each other: A lengthy footnote may flow onto a second page; a sentence in the text may refer to a particular illustration that the art editor moves from page to page to balance the visual effects.

Dealing with all these elements is complicated, requiring skilled people -- or object-oriented tools. As more and more text goes on-line, in potentially reusable chunks, users will want to manipulate it automatically. Layout systems will embody aesthetic rules. People will move beyond mail-merge to automatic text generation, just as they will move beyond reusing code from a library to reusing code with a code-generator -- an application that assembles chunks of code (or text).

CLOSED -- AND PROUD OF IT! PAGE 31
As long as humans are in the loop, it will be handy to translate terse database information into more discursive sales pitches, letters of response, contracts and wills, financial planning proposals, severance agreements and other "personalized" documents. Rather than insert data into a template, we will create documents from components. There's a flavor of hypertext (see Release 1.0, 87-9), but there's a need for something more automatic based on identification of text objects -- in other words, something intermediate between dealing with the text's meaning (which is generally too ambitious for now) and dealing with it merely as strings of characters.

This would sound esoteric but for the federal government's recent CALS (for Computer-Aided Logistics and Support) standard, which is basically a directive to all DoD and many civil agency vendors to submit proposals, documentation and other texts in so-called Mil-M-28001 format. This format defines a document format composed of precisely identified text objects and uses a language called SGML (for Standard Generalized Markup Language).

To do this, the first order of business is to create systems to define and manage text objects, using SGML. At a time when most text-oriented vendors are fighting over the pitch of the bells and whistles on their WYSIWIG systems or the speed of their indexing algorithms, a few companies are working on these problems. Among them are Avalanche Development (which automatically recognizes text objects such as tables, headlines and sequential page numbers in existing texts for ex post facto tagging of objects; see Release 1.0, 87-10), and two companies described below, SoftQuad and Datalogics, with word-processing systems designed to assist in the creation of object-oriented text.

Text objects: How do they work?

Think of object-oriented text as akin to databases, with content separate from selection and display. These are the components: the rules defining objects (data elements) and their relationships (the structure of a database); and the specific contents of the objects, supplied by the user. A resulting document is akin to a database report (the output of an application), with the document elements selected by criteria and versions, the report's structure and sequence governed by a Document Type Definition (DTD) or template, and its appearance specified in a format table, for example: "Footnotes must be at the end of each chapter, in 10-point italic type, separated..."

In the old-fashioned world, of course, all these components were intermingled (just as data once resided with their applications rather than in a separate database). Documents included embedded, hard-to-change formatting commands and could be restructured only by hand with cut, copy and move commands. Outline processors were only the first step towards the rich, malleable structures we can build and manipulate today.

--------

1 Anyone with serious interest in these issues should immediately sign up for The Seybold Report on Publishing Systems, (213) 457-5850 -- and should probably order some back issues too.
SGML: How does it work?

The language of choice for specifying text objects is SGML, a burgeoning standard. Like SQL in some respects, SGML is a data-oriented language. It identifies things; it doesn't contain procedural commands (while SQL has a few) and is independent of OS, hardware, display, everything. It simply describes the text objects (data) and leaves it up to the local system how to organize and display or print them. Notions of structure and relationships are explicit (nesting, sequence, cross-references), and can thus be easily changed. These structural relationships are persistent if mutable, while decisions on where to place footnotes, how to combine illustration and captions, where to locate table of contents or index, are implementation questions decided by applications -- the combination of a specific document's structure, content selection and formatting commands. SGML merely identifies the chunks of data.

Why we care

While these notions are obviously of commercial interest for vendors to the government, their implications are far broader. They allow authors to identify the objects in their documents without regard to specific design rules, and allows editors/publishers to apply those rules without resorting to hand markup. And those rules can be changed globally in a flash. Change your art director or get a new color printer? You can automatically change all the subheads to display in royal blue. You can automatically impose rules specifying the relative placement of figures and captions; require at least three lines of text (or an entire paragraph) after a subhead and before a new page; start chapters 16 lines from the top of a new page; etc. SGML makes layout, as well as typesetting, easier to manage.

As noted, you can look at text in two ways: The text objects themselves -- chapters, documents, titles, subheads, tables, boxes, footnotes, cross-references (unique, identifiable text markers and arbitrarily many pointers to each) -- and the objects they are mapped into: pages, lines, columns, words, characters and combinations of them such as widows and orphans, word breaks. A structured, object-oriented system makes automatic mapping from one to another and back possible. Text objects become even more interesting when they are even more tightly defined, as in <drug> in a pharmaceutical manual -- where it might guide special formatting for each reference to a drug.

Getting close to content

But form isn't all. Designation of text objects is also important for a large number of content-oriented document management tasks such as indexing and creation of hypertext (see Release 1.0, 88-9). Do you want to weight words in headings and subheads more heavily in constructing a weighted index (where word-frequency is used to assign text items to a list of topics)? Or do you want to search for citations naming J. Fred Bloggs (text references are of no use)? Do you want to create hypertext links to chapters with the word "earring" in the title? Or suppose you want to create a document that includes subsections and chapters assembled from texts floating around your system. They include cross-references. You want it to come out correctly paginated and with the sections and cross-references correctly numbered. To do this you need a system that recognizes the text components, including
subsections and their components -- headings, footnotes, captions, illustrations -- and can model their behavior without having to load them into memory and examine them one by one.

The object-oriented approach allows computers to see words in part as librarians and other scholars see them -- in context. A word casually mentioned mid-paragraph is "worth" less than the same word in a headline. That same word in a caption indicates something about the drawing it annotates; the cross-reference to it on page 49 has meaning, too.

The tools

SGML is a syntax for describing and identifying, or tagging, text objects and the document structures they comprise. There are two main functions required of an SGML word-processor: computer-assisted tagging and validation. These both rely on a parser, a sort of tree-creation tool that assesses the syntactic state of a piece of text at any point. Because it knows what is allowable at any point, the parser-based tool can present a user with a menu of possible text objects. It performs the tagging automatically when he selects one. Moreover, as he moves the cursor through the text, it moves around the virtual structure of the document, with the possibilities for object insertions constantly changing. (A parser is actually the first half of a compiler; it understands the syntax of the words before the translator takes over.)

The parser is driven both by the general rules of SGML -- sections compose chapters, paragraphs compose sections, titles come at the beginning -- and specific, arbitrary rules as delineated in a Document Type Definition (DTD). Rules range from specifying that a document must begin with a title and subtitle, to constraints on the number of chapters and so forth. The user can use pre-defined SGML text objects -- titles, subheads, levels of emphasis, boxes, footnotes, cross-references (unique, identifiable text markers and arbitrarily many pointers to each) -- and add his own, such as special kinds of boxes, quotations, citations, etc.

At the end of the process, a document can be validated: Not only are all its parts correct, but they "total" correctly, following the rules of a specific DTD -- one title, several chapters, an index. Does every object have both a beginning and an end? (Most parsers automatically end a paragraph whenever a new paragraph or new section begins but other kinds of objects need explicit endings.) Does the document include specified elements such as indices, captions for illustrations, etc.? You could even program it for further refinements: Is every figure referenced at least once in the text? Is the number of objects of various kinds correct? What's the ratio of text to tables? Are subheads evenly spaced to break up chapters? "Correct" answers to these questions don't ensure quality, but inconsistent answers certainly raise doubts.

SoftQuad's Author/Editor and Datalogics' WriterStation

Some of these capabilities already exist in individual publishing systems such as Interleaf's or Scribe's, but output from one can't easily be interpreted by another. Moreover, how many writers do you know who can afford an Interleaf system? (If so, they're probably not full-time authors.)
Two companies offer SGML-parser-based word-processors: SoftQuad and Datalogics. SoftQuad, in Toronto, was co-founded by publishing consultant (and current CEO) Yuri Rubinsky in 1985. Its major business is a composition system called SoftQuad Publishing Software, an enhanced version of AT&T's Documenter's Workbench (also on UNIX systems). Datalogics, in Chicago, has been selling electronic publishing products and services since 1967. It numbers the best among its customers: financial printers such as Bowne and Chas. P. Young; legal publishers such as Commerce Clearing House, BNA and West; documentation mavens such as Boeing, Ford, GE and Grumman; and database publishers such as Standard & Poor's and Moody's.

SoftQuad's Author/Editor was originally conceived as a front-end for SoftQuad's publishing system, but was realized as a generic system for creation of SGML documents. It uses the power of a Mac to provide real-time formatting display and feedback to a user as well as a speedy underlying interactive SGML parser that assists users in tagging and validating their documents. Datalogics' WriterStation, by contrast, is a more sober-looking but equally powerful tool that can operate standalone or hooked up to a VAX running other Datalogics software.

Both companies' products do interactive syntax checking; you can also turn off the checking and validate an entire document at the end of a session based on a specified Document Type Definition. Text or figures from outside can be incorporated by file name; commonly used phrases or formats can be included as entities (represented as icons in A/E). Users can define their own text objects (entities) for special treatment: Everything from a standard WARNING box with fancy borders for use in DoD documents or a chemistry lab text, to embedded processing instructions that will automatically load an SQL interpreter and fetch updated data (assuming that a target system is set up to do so at runtime). And it's all just an ASCII file at the end, suitable for transfer to any SGML-parsing publishing system. Using simple codes, these files can generate output from a low-end typesetting system; at full strength, they can do awesome things.

Aside from that, they are just regular word-processors, with the usual facilities for text manipulation and spell-checkers, search-and-replace (for tags as well as text) and the like. They offer an automatic smart outlining feature by allowing you to see headlines, subheads or paragraphs or other specified text objects only, and allows you to move them around easily. Text objects can be nested inside each other. Writers who think structurally rather than word by word will appreciate them immediately; those who don't will find they help them to do so.

A/E costs $450 and comes with a set of standard DTDs including three from the Association of American Publishers, plus its own "Smallbook" DTD. The CALS DTD costs $500 extra, including a one-year update program as the government's wheels churn. WriterStation costs $1500 per seat, with volume discounts; it sells DTDs separately for about $300 each.

For another $995 more, you can buy SoftQuad's RulesBuilder, which parses DTDs just as A/E parses SGML documents. That is, you could build a DTD with a word-processor, but RulesBuilder guides you as you do it. Thus you can specify your own set of rules: "This is how we write business plans at Churn M. Aout Consulting," or "Here's the format for your article for Vanity Pressure."

Release 1.0 30 December 1988
WANG'S FREESTYLE

The most interesting product at Comdex was Wang's Freestyle, an image-based, pc-based communications system. It's simple, straightforward, easy to learn and to use. It exploits Wang's expertise at imaging -- but doesn't yet use the rest of the power of the computer it lives on. Wait for a toolkit late this spring. For now, pricing alone ($2000 extra per networked pc) will limit its market to less than critical mass.

Freestyle is intuitive and simple because it employs the principle of direct manipulation: Here's a piece of paper (a screen image captured from anything you can display on your pc with Freestyle in the background); do what you want with it. You can staple it to one or more other Freestyle pages, type on it or mark it up with a stylus, voice-annotate it in sync with the markup, duplicate it, throw it away, or send it to other people on a network (Wang VS for now, but others soon) who have Freestyle or any Group III fax.

They will need to have Freestyle themselves to hear your sound annotations, but they can look at your documents with any fax receiver. The sound annotation, however, isn't linked to any particular part of the page but rather to the entire page, so that you get to watch and hear as the stylus moves from point to point on the page: "This number here, I think it's suspicious... We got the figures from Juan last week, but, well, you know... [A long silence; the ghostly pen starts to circle another number.] This one here, well, actually, it's probably okay, but it depends [the pen moves again] on these figures from Alice; they had bad weather last week. I don't know if they'll make it up this week or if it's gone forever. If we could fix that..." We're not sure we wanted to know all that; we just wanted to point at a single figure and get the commentary for that number only, rather than a blow-by-blow of the whole page. (See also our comments on Forum Systems' PC/Forum; Release 1.0, 88-6.)

Moreover, we don't want to have to wield stapler and mailbox every time we ship out the week's numbers; we want to have the system do it automatically. We want to have it automatically graph the results, highlight the exceptions (10 percent or more out of line with the budget), and list the locations where the exceptions occurred, so that we can annotate them one by one. And when we send the report out, we want it live, so that each recipient can use 1-2-3 (or whatever) to do his own extrapolations of disaster.

Versus NewWave

Perhaps the best way to describe Freestyle is to compare it to another innovative object-oriented platform, H-P's NewWave. Both deal with objects and both eventually will allow you to deal with individual data elements as objects, but both started at too high a level. NewWave treats an entire file (linked to an application) as an object; Freestyle treats a single screen as an object -- although Freestyle gives you the added benefit of the image-markup capabilities. Both offer toolkits to enable intrepid end-users and far-thinking software vendors to customize their products to take advantage of either and open them up to data-element interaction with Freestyle or NewWave. But because Freestyle is design-centered around image and NewWave around code, we suspect NewWave will be the system of choice for agents, scripts and other extended programs, while Freestyle will benefit most from allowing its users to annotate live data rather than captured images.
As the SGML examples above show, the world needs more data types than are dreamt of in a standard relational database. It also needs better ways to manipulate them. But, as with any religion, there are degrees of observance and orthodoxy. (For earlier discussions of object-oriented databases, see Release 1.0, 87-8 and 88-9.)

You can have object-oriented programming without object-oriented databases, the standard modus operandi for people currently using C++, Objective-C, Smalltalk or other object-oriented languages, or even people building object-oriented applications in traditional languages. They build their object-oriented applications the usual way, and data is either stored with the application or called from a regular database with SQL or other data manipulation syntax that the objects know how to use.

Some object-oriented applications create their own file/storage structures, but it's stretching to call these databases since they're unique to the object-oriented application that creates them, and they rarely deal with traditional database issues: concurrency, distribution, integrity, to say nothing of transactions or configuration management. These structures are built for the application that uses them, not for yet to-be-specified queries and applications that might want the data they contain.

On middle ground, there are the databases that store objects. They can get the objects for you on demand, and they deliver them as objects, not as data fields that need to be assembled by the object-oriented application to come alive. These systems "hold" objects fine, but they're indifferent to their content. The methods of the objects are stored with them, but the database doesn't support inheritance or the notion that objects may contain or otherwise refer to each other. Those relationships are represented, if at all, by tables of relationships or within an application masquerading as a database. This approach can be very effective, to be sure, and is the method applied by TeamOne and Sherpa, below, both of which are object-oriented data management applications built on top of relational databases.

At the extreme, of course, an object-oriented database can be represented by a very unnormalized relational database (although relational purists would consider that an oxymoron). One of the theses behind relationalism is that all data is represented only once, and data sets (or objects) are assembled dynamically through dynamic joins across tables based on their values -- at some expense in performance. A denormalized database is so to speak "pre-joined" for performance, but it suffers from redundancy and complexity.

To offset the complexity and maintain flexibility and performance, an object-oriented database needs very clever algorithms for caching (holding data in memory) and clustering (storing data elements likely to be used simultaneously close to each other physically). These depend both on the relationships of the various objects to each other -- is-a, is-a-kind-of, is-a-part-of, is-a-value-of, and their inverses. For example, Juan, Alice and Hannah are employees. A boss is a kind of employee. Juan is Alice's boss, and Alice is in turn Hannah's boss. Their group, yield management, is part of the revenue-enhancement department. Sometimes Alice works on projects for the marketing-practices group, which complicates things further...

Release 1.0
30 December 1988
Or consider engineering drawings. Change subassembly XYJ11, and the change will ripple through the system. Some designs may have parts that are incompatible with the new version of XYJ11. Bills of materials have to be updated, too. The factory foreman has to be notified to reprogram the NC machines. You can't easily represent these designs and interdependencies without a richer data structure that "understands" these interrelationships, which include not just "is a part of," but also "fits into," "sits on top of," "is connected to," and other such relationships.

In addition, an object-oriented database needs to handle concurrency, transaction management, integrity, distribution of data across heterogeneous machines and operating environments, development tools and all the other issues that make a relational database more than just an SQL interpreter.

A rationale for objects

So when do we need which kind of solution? Up to now, with no object-oriented databases available anyway, the answer has been simple. Use what you've got. With the advent of DB2 and the other relational databases on the one hand, and pc-based databases (growing into servers) on the other, a lot more data is being stored, and it's likely to include things more complex than accounting transactions and payroll records (bad enough!). Text and engineering drawings are going on-line, as are models of workgroups (as in the example above) for use in project/process management.

In short form, objects should be used for things when only one or two have the exact same structure. When you can identify them by regular data fields, then you need a database that stores objects, and you can use an object-oriented application with data-fetching facilities. For example, do you want to make one interesting query, and then do the same thing to each item found? (Find all 1000 employees in the revenue-enhancement department, and give them raises.) But when you are likely to search for and manipulate objects in terms of their relationships with other objects, then you need an object-oriented database. For example, do you want to find the model that contains three rivets attached to a widget, and modify it? Are you dealing with a structure or a model of something, or with the many individual things (instances) that it represents? Can you store the structure in an application, and the data in a database? Or are the structures so large and complex that they need to be managed and reconfigured by a system that understands them?

Of course, you probably care about both, but where is the more interesting information: In the values within the data elements, or in the relationships among the data elements?

To summarize:

- A traditional database holds data. Its content is primarily values and strings in the data fields.

- A database with objects holds objects. Its content is the objects. Their content is data and interrelationships that are visible only to an application that can interpret them.

- An object-oriented database holds and manipulates objects. It sees and uses the objects' internal structure and interrelationships to do so.
DATABASES: INTERBASE

While the supply side of the database business is getting extremely crowded at either end -- vanilla SQL systems, and one-of-a-kind object-oriented systems -- the middle ground seems to have only one serious player: Interbase and its product (sic) InterBase. Interbase, you may recall, is the company that Ashton-Tate invested in last spring. (It was a minority investment with provisions for extending it to a majority investment.)

Interbase is a well-grounded company that knows its place; its InterBase works fine for complex data that doesn't need to be handled by a full-fledged object-oriented database. (This is the same model likely to be used by Oracle and RTI, among others.) The company was founded in 1984 by president James Starkey and five others, who had worked on databases at DEC and Seed, a maker of databases for VAXes and other minis. Interbase is distinguished by the quality of its product and by the deal with Ashton-Tate which that product enabled it to win. The company now has 50 customers and a close relationship with Cognos as well as A-T.

The product is distinguished by three factors:

One, InterBase can handle SQL, but it has its own internal data representation language. This provides greater flexibility and lies behind InterBase's ability to read Dbase code easily (since it has a generic facility for data language parsing).

Two, as a result of its mini heritage, InterBase is focused on peer-to-peer rather than client-server architectures, so that it can more effectively handle data distributed across heterogeneous systems. For communication among compatible systems, InterBase uses a simple, high-bandwidth client/server connection, but for communication to different systems it uses a hardware-independent message protocol that's translated by InterBase environment-specific systems at either end. These connections are set up automatically at log-on time. Data stored, in, say, the VAX double floating-point-precision format can be transformed into what's needed on a pc. This can be a touchy problem in something as simple as translating data from a 68000 to a 386-based system.

Three, InterBase is uniquely capable of handling "foreign" data types such as text, graphics, arrays (soon) and other objects. To do this, it uses attached procedures that reside in the database on each host system and not in the applications. When a particular item is loaded, a library of functions for handling it comes with it (much as with H-P's NewWave, which, however, relies on traditional DOS, OS/2 or UNIX file systems).\(^2\) Take, for example, images. Each image has a unique ID as part of a record. It may be the only image per record, or one of several. It can be found by querying for the related name, data, claim amount, etc., in that record. But it's a discrete object; there's no internal structural information that the database manager...

\(^2\)In fact, the conjunction of InterBase and NewWave makes us think how nice it would be to have an installable file manager based not on a traditional file system but on a content-based database manager that would treat data and applications as linked objects -- sort of a combination of NewWave, InterBase, a text-retrieval program and any OS file manager you care to replace. (More on this notion next month in our essay on access managers.)

Release 1.0 30 December 1988
needs to know about (although some information may have been taken from it by a data-entry clerk who entered the name and other fields in the record). When the image is loaded, InterBase loads the code to display it for the particular machine that called it (assuming that capability is available).

For example

For example, with InterBase, you can easily retrieve names and addresses and other data, and use the data to match each record to any of a set of boiler-plate letters. With a mail-merge application you could create a flood of letters of appropriate sternness warning people with details of the amounts owing that their legs will be broken if they don’t pay up.

But InterBase has its limits vis-a-vis a truly object-oriented database. Consider another kind of images, engineering designs. Each can be displayed in full by InterBase, but many of them consist of collections of related objects that can’t be manipulated separately by InterBase. It sees text as a file rather than as a collection of objects. You could certainly identify each chunk of text as a certain kind of object by indicating that information in a specified field on each record, but the database itself (as opposed to an application) wouldn’t maintain those relationships.

InterBase has also developed a graphical front-end called Pictor, with multiple windows, cursor selections and pop-up menus. It’s not as elegant as some of the best tools on the Mac or the Sun, but it has the virtue of running across a broad spectrum of machines. The system also offers the usual array of forms-builders, a 4GL (called SDML, similar to DEC’s Datatrieve), and report writers.

...and Ashton-Tate

Interbase’s agreement with Ashton-Tate should help the company extend its reach into retail distribution, much as the A-T/Sybase deal is doing for Sybase. It’s still unclear how A-T intends to use InterBase, since it is committed to exclusive use of Sybase for any OS/2 server system for an unspecified period that may last several years. With its peer-to-peer capabilities, InterBase would make a fine server for any A-T front-end.

For now, A-T talks specifically about using InterBase on a "workstation." Perhaps that’s a reference to the system’s graphics capability, which needs a workstation to be used to advantage. Regardless, it’s a pity A-T doesn’t have total freedom to use InterBase in any way it chooses, since that would give it a back-end as well as a front-end strategy for OS/2. Perhaps A-T will use InterBase -- which runs on VAXes, Suns and Apollos so far, with OS/2 and HP Spectrum versions promised -- as the basis of its promised DEC system as well. InterBase’s internal OSRI language is a superset of DEC’s DSRI, used in its standard database Rdb.

Meanwhile, Interbase’s 50 customers are primarily people interested in the sorts of unpredictable, rich-data applications that will proliferate in years ahead, including most of the major aerospace companies and the Software Productivity Consortium. Theirs are applications that generate revenues rather than cut costs. They are not about the business (revenues, accounting transactions, and the like) but what the business is about -- documents, images, diagrams.

Release 1.0 30 December 1988
"The purpose of engineering is to design products, but the product of engineering is documentation."

-- a Sherpa brochure

As noted, in many cases an object-oriented application sitting on top of a traditional database can handle many of the simpler cases where object-oriented data management is needed. In other cases, and in the long run, many of those applications themselves may rely on an object-oriented database, regardless of where their instance data is stored. Although database companies compete vigorously (in the SQL market and separately, soon, in the object-oriented market), each of the systems that use these databases has its own orientation, its own positioning, its own design center.

Among them are three Silicon Valley companies selling design data management systems of three different flavors: TeamOne, EDA Systems and Sherpa Corporation. Their competition is mainly in-house efforts by end-user customers, as well as data management systems embedded in or closely linked to design tools such as Automation Technology Products' Cimplex, SDRC's DMCS and Matra's Euclid/Datavision, and documentation management tools such as DocuGraphix' Manufacturing Operation Documentation System.

Although these data management systems are occasionally sold standalone, in practice they are linked to their vendors' other products -- and other tools vendors are reluctant to support them. One other independent vendor not covered here is Atherton Technology (see Release 1.0, 88-5) which compares most closely to EDA Systems; however, Atherton is focused on CASE, while EDA sells to electronics design tool users and vendors. Likewise, the hardware vendors' own efforts, such as Sun NSE, DEC's EDCS, the UNIX source code control system and most notably Apollo's DSEE (Domain Software Engineering Environment), provide many of these functions, but are generally considered too vendor-specific and limited in functionality. They manage objects as files, and don't provide the database functionality -- integrity, reports, query tools -- of the database-based systems.

One electronics-oriented vendor told us haughtily: Well! Our problem is very different from software engineers', because we have many more types of objects -- circuits, transistors, capacitors, voltages -- and all they have is files of code.

But he's wrong -- which is why there's a need for a more general approach than all these market-specific application answers. What looks like code (or like a file) to the naked eye in fact has a distinct personality. Code comes in modules: Subroutines that can be called only by certain other functions, data definitions, user interface objects, standard modules for frequently performed functions and the like. We might as well say that all engineering drawings are collections of vectors and data. The whole point of the object-oriented paradigm is to get inside these structures so that we can manage (and reuse) the individual components that make up the deliverables specified by users or customers.

Thus, while none of the three vendors to follow uses a full-fledged object-oriented database management system, their products provide clear evidence of the need for one. Most of the object-oriented database vendors we've
talked to have visited several of these companies in search of feedback and guidance for their own development efforts. What they have learned is the importance of performance, the importance of clean programming interfaces, and the importance of all the features database users now take for granted -- tools, integrity, transaction management, distribution of data across heterogeneous platforms, multi-user support. (The third-party tool vendors who would use an object-oriented database foundation typically consider user interfaces their own province, a center for value-added. Remember the illusory promise of the common interface: The functionality of each tool still varies. You still have to learn each tool, although you do not need to relearn such basic functions as save, copy, load, delete, time-stamp.)

Also, general experience seems to be that less is more: Customers don't want to disrupt their current operations to absorb a new data-management system, however powerful. Most customers already have a variety of tools and approaches in place, which makes for delicate integration and installation problems. In the end, this is the problem that object-oriented databases should solve, by providing underlying database management for all of them in a way generic enough to encompass a wide variety of front-ends.

Oddly, the smaller customers -- with limited budgets, a limited diversity of existing tools and strong, centralized management -- can most easily afford to make the commitment to a single unifying standard. Large companies may have a bigger problem, but it would cost far more to solve it because of all the investment they have already sunk in existing systems and people trained to use them. Larger companies will be buying retrofit technology for years.

Vendors will succeed in this market to the extent that their solutions can fit neatly around existing products, or to the extent that they can get third parties to use their products as the foundation of their tools.

Of the companies described here, EDA Systems is going the foundation route, while Sherpa and TeamOne are attempting to insinuate themselves into existing environments. All three serve as monitors for the process and for each application's management of data, but they do not manage the application data itself. They work as extensions to the file system, requiring users to work through them rather than directly with the file system. EDA Systems inserts itself into the user's environment more than the other two, but at the cost of substantial installation and configuration efforts.

SHERPA CORPORATION: GROUPWARE FOR ENGINEERS

While most engineering databases (including TeamOne, below) focus on the management of design objects -- code modules, documentation, component schematics and versions thereof -- Sherpa Corporation deals with the management of the design cycle -- seeing to it that items pass through their proper review cycles and that the right people (and only the right people) pass on them. In fact, Sherpa is an excellent example of task-specific groupware, with a focus on process rather than content.

Sherpa was originally conceived as a far more ambitious company called Cadtec -- one that would manage both the work cycle and the content of electronic design work. However, it gave up on that idea in 1984 to concentrate
on a more manageable subset of that problem. ("Everybody had a doughnut with their system in the middle," says president Steve Schopbach, the only founder who is left. "Now we fit around everybody else's systems.") Sherpa has a total of $10 million in funding from Sevin Rosen, NEA, Fostin, Century IV and Abingworth and annual revenues around $3 million.

Sherpa's Design Management System uses an object-oriented front-end, and stores its objects as records in a traditional relational database -- Ingres, as it happens. The Sherpa software is in fact an application based on Ingres that includes a capability for building triggers (a feature it could have gotten almost free if Sybase had been around when development started) and a set of pre-defined objects with pre-defined relationships. The objects are the projects, people, tasks and files that comprise projects, with associated access privileges (depending not just on the objects themselves but the stage of the work process, since items may be changed by certain people only at certain points in the cycle) and procedures.

The content of the work is regular old files that are tracked by the Sherpa system: Each time a file is touched -- read, modified or stored -- Sherpa's DMS approves the transaction, records it, and (optionally) executes a stored procedures -- perhaps notifying someone that a piece of work is ready for approval, or creating a new file for a related document. Although DMS is an application, it takes over an entire user's system by encrypting all files and restricting access to them unless a user enters properly through the Sherpa environment.

You don't need an object-oriented language to create objects; the system is written in C. On the other hand, it provides no support to users for building new classes of objects, although there's reasonable flexibility in adding new kinds of triggers or subclasses of objects if you use Sherpa's toolset (for $15,000). The system also comes with a nice suite of Ingres-based query and reporting tools.

By the book

Cleverly, Sherpa has created QuickStart, a set of application templates that sit on top of Sherpa DMS and assist customers in getting up and running quickly. QuickStart does 80 percent; the customer adds the remaining 20 percent with the help of Sherpa's tools and possibly some consulting at $4000 a week. (The first ten days of training are free.) The system starts at under $50,000 for a maximum of eight users, net of hardware from DEC, HP or Sun. The first QuickStart application handles document flow; future ones will address configuration management and engineering change management. The document flow application manages a review cycle: Design files pass through a series of stages (levels) and get "promoted" when the requisite people (identified by job title and project) approve them. There are facilities for senior managers to override the process, and default assumptions that the process starts with the file's creator having sole authority over it.

3In the same way, many DOS access managers are separate applications that create a shadow file system that a user can talk to instead of DOS, storing extended file names and attributes in a database that monitors file activities and intercepts application calls to load or store files. Underneath, however, there's still that same old DOS directory. See our next issue for more on this topic.

Release 1.0

30 December 1988
his data until he moves it into the cycle. Access rights change as the file moves through the process, and sets of files can be grouped together.

Configuration management will deal more with the relationships between various files representing different components of a design (components, assemblies, documentation) rather than its lifecycle as it's created; engineering change deals with it after it's created and is used and modified between design and production groups.

Sherpa has been shipping since 1986 and has over 30 customers, including GE, Combustion Engineering, Schlumberger, Sikorsky and Hughes Aircraft. DMS first appeared on VAXes and now also runs on Suns and the HP 3000 under HPUX. HP is helping Sherpa to market the system to its large base of engineering and manufacturing customers. President Schopbach considers the company's edge to be its full-service approach, including training and consulting to help customers use its products to full advantage.

TEAMONE: A NOVEL APPROACH

TeamOne sells a database designed for configuration management and general engineering data management. The system is an object-oriented application built on top of InterBase. However, TeamOne doesn't take the traditional (in this context, anyway) approach of storing objects as records in tables, but rather of using InterBase as a low-level network page-server to handle the issues of heterogeneous platforms and cross-system file access.

TeamOne was founded in 1987 by a team of people who were building an engineering information management system for CAE, the engineering design automation company subsequently acquired by Tektronix (and just recently sold to Mentor Graphics along with the rest of Tek's computer-aided engineering unit). "We were working on a large joint development project [at CAE/Tektronix with DEC] for engineering data management tools -- version control, configuration control... It became apparent to us that software engineers needed this just as much as hardware engineers," says co-founder and vp of R&D Dave Hoffman. He and two others subsequently left Tek to found TeamOne.

TeamOne is initially addressing the software-engineering market with a UNIX-based system optimized for managing software development data -- source files in all their multiple versions, and (later on) associated documentation. The product is just starting to ship, with beta sites at Data General and Cadence, who are evaluating it for internal use and resale.

Like Sherpa DMS, TeamOne manages files, with no intimate knowledge of the applications it supports. And like Sherpa, it is focused on group rather than individual productivity. But the flavor is totally different. Whereas DMS interacts directly with users, controls file use and requires installation, training and configuration efforts, TeamOne sits virtually "underneath" the UNIX file system, extending it rather than managing it. For better or worse, TeamOne leaves far more in the hands of the user, who is expected to know his way around UNIX (as any software developer probably does), while Sherpa keeps the user away from the file system and within his application environment.

TeamOne creates a sort of shadow directory that automatically manages multiple versions of code files in a project repository. For example, when an
engineer modifies a file, TeamOne automatically creates a copy for him that he can store without altering the baseline or checkpoint configuration he took it from. When he (or his manager) is satisfied with his work, TeamOne will merge it into the baseline. The merge process is semi-automated: TeamOne understands about code modules and certain code syntax and can merge non-conflicting modules or lines of code and report on conflicts -- but it can't currently recognize possible side-effects or other more subtle problems such as one module calling a subroutine in another that's been deleted. That part still requires human intervention! TeamOne's basic contribution to the process is the maintenance of clean baselines and multiple versions of changed files, with automatic access to the latest versions unless someone specifically requests otherwise.

Behind the scenes

TeamOne operates invisibly so that users feel they are dealing with the regular UNIX file system and applications needn't change, but it offers an SQL-based query tool for managers who need project information. It has menu picks for standard reports about versions, configurations and user activity.

TeamOne maintains its information about these versions as objects with relationships; each software project is represented by a configured root directory, or an object tree with multiple nodes and multiple versions of many nodes. (The actual file contents are stored in a UNIX file system.) But rather than store the file description objects as records, TeamOne maintains their rich interrelationships in an object-oriented structure that uses InterBase to manage the physical storage on pages where it clusters the objects. In other words, TeamOne handles part of what a "real" object-oriented database does -- physical clustering and caching of objects. But it borrows from InterBase its ability to operate in heterogeneous environments across networks.

The premise behind TeamOne is that the other guys are too ambitious: Users don't want to change their modus operandi (as is required by Sun's own Network Software environment, which offers some of the same capabilities) and are unlikely to adopt a new environment that requires them to reconfigure their existing software or retrain their existing employees. TeamOne fits around the existing environment but doesn't disturb it, providing a few extra services but requiring no changes in behavior (unless you were accustomed to fooling with files you shouldn't have had access to).

TeamOne is also nondisruptive in its pricing, at $2500 (list) per seat. The company will sell mostly with telemarketing, to customers who know the problem TeamOne solves and can install and evaluate the product themselves.
EDA SYSTEMS: A FOUNDATION FOR DESIGN

While the two systems described above act as front- or back-ends for file management, EDA Systems is taking on the more ambitious task of providing a foundation for the design management and applications themselves. EDA's Electronic Design Management System represents not just data files, but also the tools that create them, as objects. Because its foundation is so malleable, EDMS's character depends more on how it's used than on its own characteristics.

EDA Systems was founded in 1986 by Lucio Lanza and Jean Brouwers, both from Daisy Systems. EDA is a leader in the CAD Framework Initiative (thank goodness they didn't name it the Open CAD Foundation!), a group working to create standards for data formats and structures across a wide range of CAD systems. The group also includes Motorola, Hewlett-Packard, DEC and MCC.

Nothing would please EDA more, of course, than to be a standard foundation for many of these vendors' tools. Unlike its competitors with engineering databases (ATP, Matra, SDRC, Mentor Graphics) it sells no tools itself, thus maintaining a certain neutrality (although of course it is now teaming up with tool vendors such as DEC and Silvar-Lisco who do compete with other tool vendors). EDA sells primarily to end-user groups at the major electronics manufacturers and ASIC vendors such as Motorola and GE Solid State, and to major CAD and computer manufacturers who build customized engineering environments for large customers. (DEC, for example, will be using EDMS in its systems integration efforts.)

EDA Systems' EDMS is a high-end object-oriented system that manages metadata about users, tools and data in a design environment. It encapsulates the tools the user selects, associating each with the proper procedures and the relevant data files and structures. Using a graphical front-end called WORKShell, the user can browse his domain through project view and process view windows, which display respectively the data files associated with a project (a tree showing the files and their multiple versions as extra nodes where appropriate), or the sequence of activities (tools) required to complete a project. The user can click on the items he wants, and select individual options from pop-up menus customized for each application he uses. But once he enters each tool, he's in that environment. The tools themselves are not modified, unless a vendor has used EDA's tools for constructing rather than encapsulating CAD applications. These are CADbase, a data-management tool that overlaps EDMS but lacks some of its generic features and has instead a set of data elements specific to electronic CAD such as nets and pins. CADview provides interface building blocks.

EDMS costs from $7500 to $15,000, but no one buys just one. From management's perspective, EDMS performs the usual functions of managing group work -- file access and integrity, version management, review cycles and the like. Because it is so extensible, it can do almost anything a user-builder cares to program in.

While EDMS manages objects representing files stored by an operating system, CADbase stores actual engineering data. In the long run, systems like these would benefit from the capabilities of a true object-oriented database. EDA and its competitors are prime targets for the object-oriented database vendors described below.

Release 1.0

30 December 1988
FACTORY OF THE FUTURE IN THE U.K.

Dynamic design applies to more than just electronics and machinery. The following example, a factory-management system, points out the utility of the object-oriented approach in the large. Although you could build a single factory-automation system more efficiently the traditional way, the value of Kewill's system will lie in its easy configurability, both at installation and as the factors shaping each factory's operations change over time. Although the system one does not yet exist, which might normally disqualify it in favor of others closer to actual use, it's of special interest because of its venue: Right outside London.

The papers are full of the revitalization of industry in the U.K. Venture capital is starting to flow; industrialists are feted rather than dismissed as tradesmen; smart people are going into business. When we visited before we got the uncomfortable feeling that in this "nation of shopkeepers" even entrepreneurs were content with very little: Their goal was to own a small business, not to build a big one. This time, attitudes had changed. Entrepreneurs dream bigger dreams. Little companies don't just find markets; they aspire to create markets. Of course, our evidence is mostly anecdotal, albeit bolstered by studies from outfits such as Investors in Industry (III), a merchant bank/VC firm in London. Herewith, one such anecdote:

Kewill Systems is an old-line consulting firm in Walton-on-Thames that for years helped small businesses set up and manage their manufacturing facilities. In the late Seventies, Kewill developed a micro-based inventory-management package that it ultimately sold more broadly, garnering an installed base of 1400 users, including 150 in the United States. Revenues from consulting and product sales totaled $20 million in 1988. The product has broadened beyond inventory control to Micross CAPM (Computer Aided Production Management), including a full suite of functions such as scheduling, purchasing and sales management, MRP, data collection, printing of shop tickets, etc.

Yet, having started early, the company finds itself behind in technology in a world of networked micros and SQL databases. So Kewill is planning to leapfrog that entire hardware shift and lead the move to new software platforms, using an object-oriented substrate to create a modular, uniquely flexible factory automation system that will give it and its customers an edge in the years ahead.

Look out for the small guy

Kewill's approach is an interesting test of the premise behind object-oriented programming -- the reusability of code to create a profusion of similar but unique systems each tailored to a customer's needs. Large companies can afford to build manufacturing systems from scratch; small companies have always had to shape their operations to fit around a rigid system such as Micross.

The new system, nicknamed Group Super Product, will be far broader and will come in modules that work together but don't require each other, except for the foundation relational database. Part of the challenge will be to provide for easy transition from Micross and competitors' products, with the capability to transfer data files and some application logic.
The new system is still a couple of years out, with funding from Kewill itself and from a £2-million Department of Trade & Industry grant aimed at enhancing small-business manufacturing productivity (that is, the productivity of Kewill's customers, not that of Kewill itself). Thus, it's hard to discuss the product itself in any depth, since everything is subject to change -- and to the achievement of Kewill's goals.

The basic thrust of GSP to represent everything in a factory -- raw materials, inventories, equipment, work-in-process, people -- as objects, with defined, modifiable behavior, using Objective-C. It will run with a standard SQL (avoiding proprietary enhancements) database that provides storage for objects as records. Functions the system will perform include inventory management, machine and task scheduling (with the ability to adjust dynamically when a machine breaks down or a new high-priority order comes in), cost and schedule estimation, cost accounting (with input for a customer's financial accounting system), order tracking, sales management, personnel management, strategic information (decision and alert support) and a set of rules for acting on that information (being developed by consortium partners Liverpool Polytechnic and the University of Manchester). Design and engineering functions are not part of the system.

Kewill has hired Sharon Burgmeier, an American who ran software development for Convex, to manage the project, and is hiring staffers mostly from universities (so they won't have COBOL habits). Although we're generally skeptical, we think that Kewill, with an installed base of 1400 real-live manufacturing customers, has a better chance of success than most.

None of those customers has asked for an object-oriented manufacturing system, although many of them have asked for the benefits that GSP will bring: greater flexibility, dynamic reconfiguration, coverage of more functions, vendor independence. Foundation platforms will be POSIX/UNIX and OS/2 and SQL and communication software from British Telecom. Kewill will work closely with a couple of Moog Controls, a Ministry of Defense high-tech firm; S&J Perfumes (continuous manufacturing) and Erskine Controls, a small maker of metal parts, to keep GSP from straying too far from earth. III has an investment in Kewill as a lender; interestingly, its VC division wasn't really aware of the company. Kewill is the kind of company that doesn't interest VCs; for that very reason, its use of high technology is likely to be successful.
A RATIONALE FOR OBJECT-ORIENTED DATABASES

So why do we need true object-oriented database management systems if these object-oriented front-ends work so effectively? The truth is, they have severe limitations. They store metadata about the data, but the data itself remains in files that are linked by an application that manages those cross-references. There's an extra layer in there, and the data isn't stored with reference to those relationships. What's missing is performance, and concurrency controls and integrity constraints that are maintained within the database, not by an external application. Moreover, none of these systems stores data at a finer level than a file, while engineering and design depend on the use, reuse, assembly and interaction of fine-grained components.

Performance

When you think of performance, you may think of queries, but we're not talking just about engineers loading models. Frequently, engineering projects use subcontractors, so you want to be able to excise an object cleanly, and then reinsert it with no side-effects. (Otherwise you get tires that don't fit the rims of wheels or parts for subassemblies that have been rejected.)

Consider the execution of a short sequence of commands such as this: "Find all the parts that use this screw and enlarge the screw hole by 0.3 cm, calculating the new stress tolerances that will result. Then generate a report of all parts where the screw holes will now be too close together (within 0.4 cm for one kind of metal and 0.3 cm for another), and print it, along with a list of the subassemblies each of those parts is used in. Finally, create new bills of materials for each subassembly affected."

An engineer frequently needs to roam around a design space, and doesn't like waiting. A relational database holding large objects can load a single design into memory, but once the objects he needs are outside that virtual space, performance drops dramatically. An object-oriented database doesn't just help you find things one at a time; it lets you traverse quickly (and transparently) from object to object through their cross-references. Moreover, an object-oriented database gives you the same programming interface to deal both with the large objects that a relational database can handle and the subcomponents that comprise the larger objects.

How does the database get performance? It needs to know such things as how to "traverse" references across objects. Objects may contain direct references to each other, such as the elements that constitute Version 23 of Part JY4X11. The object-oriented database will store those parts close to each other (in a cluster), and may cache them all even if the user asks for only one of them. It optimizes its structure based on relationships and actual application behavior (or at the direction of a clever programmer who thinks he knows better). Some of the optimization is based on the data's structure; further gains may be achieved by compiling applications for fast access -- if the database provides the user with appropriate tools.

---

4Data records contain data, and may be joined on the basis of the data. For example, in a relational database you can find a customer's address by matching the name from the invoice with the name from the address file, which requires a long search through all the possibilities or (for better performance) a denormalized "pre-join."

Release 1.0 30 December 1988
On the PC, most databases made quite an advance when they moved from file-level locking (as offered by dBASE III for example) to record-level locking (long a staple in Paradox). But how can you lock an object when it refers to hundreds of other objects that may or may not be affected by changes to it? Ultimately, the database can't decide, but it must provide facilities to carry out whatever level of locking is required by users or applications.

Transaction management

For example, two engineers who work closely on a particular object may decide they'd like to share it but lock others from access to it. They define the boundaries of what they're working on, and leave the rest of it clear for access and modification by others. Their design group designates a broader domain to be locked from other design groups, and uses the database facilities to define which other objects may be affected by the changes it makes -- all the systems that use that object as a component, for example.

You may want to define which kinds of operations can be performed, and which are locked out -- an extension of traditional databases, which may allow reads but not writes. For example, it might be okay to change the shape but not the total weight of a certain component, or its internal configuration but not the choice of which other components it's connected to. These constraints, while not strictly part of the database, can be maintained by it, rather than relegated to applications where they may be bypassed by other applications.

This includes the notion of nested transactions, where small or short transactions are completed, with results recorded in the database, but the database keeps a larger, longer transaction open, maintaining the system in a state of potential, partial inconsistency. For example, the two close-working engineers may have agreed on their portion of the system, but have not yet reconciled it with the work done by other members of the team. They need both their original and current versions maintained so that they can perform that higher-level reconciliation. (Moreover, they wouldn't want their changes aborted and rolled back if there was a general failure, but would rather want to recover from where they left off, even though that would leave the system in an inconsistent state.)

Other traditional database issues include operation across heterogeneous platforms, networking, development tools (such as optimizing compilers for data access), and general reliability.

THE FOUR CONTENDERS

In general, when innovative companies position themselves and define their markets correctly, they end up with no competition. But every once in a while that doesn't hold quite true. Relational databases is one such example; object-oriented databases is about to be another. All four companies are going after the same customer set -- white-collar software and hardware engineering and design organizations -- with varying degrees of interest in follow-on business in documentation management and commercial, white-collar-and-tie applications.

Release 1.0

30 December 1988
There's some pressure from investors for these companies to merge (if only to combine forces, let alone to eliminate competition). Yet whatever theoretical efficiencies might result, the market will be best served by a variety of offerings from which it can foster (through competition) and ultimately select the best one or two. Combining forces at this early stage would create internal redundancies and deprive the market of the trials and errors that will be needed to spur progress. It's too early in the cycle for consolidation, however inevitable it may be in the long run.

And there are distinctions, as we limn below. For starters, a friend asked us, does it make any difference whether an OODB company is West- or East-Coast? There's a slightly more electronic and relational flavor to the West Coast contingent, while the East Coast has more mechanical engineers and academics, but other differences prevail.

The original: Ontologic

Ontologic was founded as Mosaic Technologies in 1982 and was transformed into an object-oriented database company in 1985 (see Release 1.0, 87-8). It built its system around two proprietary object-oriented languages, COP (for C Object Preprocessor) and TDL (Type Definition Language), at a time when C++ did not have the apparent lead it has now. The effort to build both the database and the associated languages, with compilers, debuggers and tools, took more time and money than expected and produced a slow, single-user product that has gained 55 beta sites but only one production user. That's both good news and bad news, since it has left Ontologic free to change its strategy dramatically with minimal disruption to customers, who seem almost indifferent to the news. (They probably considered it inevitable.)

Ontologic has bitten the bullet and announced that it will convert everything to C++. It has pared its employee roster to 17 people from 39, with few design engineers left (although exact counts vary). As ceo Si Lyle (promoted from his position as de facto ceo when Bob Berkowitz left) points out, the company now has far less work to do since all it's building is a database system. It can abandon the 75 percent of its effort that went into the COP and TDL compilers and debuggers. The abandonment of COP greatly eases the porting issues, since there's no more need to build separate languages for each target system, and the database logic, written in C and C++, is relatively machine-independent and will rely on other people's compilers for portability.

Much of what Ontologic spent the last few years on -- the language-based tools -- has been deemed worthless by the marketplace, whatever its intrinsic capabilities, because the market wants to use C++. The overhaul leaves Ontologic with a name in a new marketplace, some customers who don't really have anywhere else to go, and a lot of experience in the market. It also knows -- although it has yet to solve -- most of the technical problems. The database expertise it put into its COP compiler must now be moved into a runtime system and programming protocols that can be used from within a C++ application or environment -- with a little more effort from the programmer than would have been the case previously. But Ontologic makes a point of being language-independent -- the result of painful experience! -- so that it can work with Smalltalk, Ada and other languages. The final big challenge is building a distributed system -- one Ontologic did not meet with the original Vbase.

Release 1.0
30 December 1988
Most customers are willing to wait and see. These people are risk-prone types in the first place, given their willingness to sign up for a strange new language. Most of them are glad Ontologic will now support C++, however painful the transition. Meanwhile, the company is going back to a couple of prospects who turned them down cold when they were working with COP, but expressed some interest in the general concept. Ontologic hopes to build a core group of five to ten paying customers who will get the alpha version due out this spring, and help Ontologic ship a revenue-generating version next fall.

The company seemed amazingly cheerful when we visited. This, however, is clearly the last chance for a company that will have lost the cumulative $26 million raised since it was founded. If it doesn't build a viable product this time, it will be facing competition that can.

The rest of the field

Ontologic certainly fits the role of the pioneer penalized for being first. Arrayed against it are three new start-ups, to say nothing of veteran Servio Logic, a company that is rarely seen in the market these four are after because of its identification with Smalltalk and its absence (until recently) on any workstation platform.

All three new companies have sprung up in the last few months, in part responding to the stir created when Object Design founder and CEO Tom Atwood (who left Ontologic in 1986 after strategy disagreements) started canvassing for funding and people last winter. Several West Coast VCs, including Mayfield and Menlo Ventures, considered funding him and brought in a couple of consultants to assess the deal and the market. In the end, the deal fell apart, and all parties involved have signed general releases (absolving each other of any liabilities).

Atwood went back east and assembled a different team there. Meanwhile, the West Coast consultants felt they had discovered an exciting opportunity -- not for the technology per se but for its use in the engineering database market. They pursued the idea (but not the expression!) with such vigor that they ended up persuading Menlo and Mayfield to fund their own company, Objectivity. ("Drew [Wade] and I were called in to round out the team, and discovered we were a team," says Objectivity's Bob Field.) Finally, Kee Ong, system architect at Relational Technology Inc., concluded that RTI's success with its existing relational engine (and its ties to RTI founder Mike Stonebraker and his work at Berkeley on Postgres, an object-oriented front-end to Ingres) would keep it from staking out new ground in a from-scratch object-oriented database. He left last summer to start the last (so far) of the bunch -- Object Sciences.

The second original: Object Design

Object Design has the most high- (or exotic-)tech flavor, with a roster of veterans from the major first generation of object-oriented databases: Gordon Landis, chief architect of the database component of Ontologic's Vbase (laid off this fall); Dan Weinreb, chief designer of Statice at Symbolics (see Release 1.0, 88-4); and Jack Orenstein from CCA, where he worked on Probe, its object-oriented database; as well as a couple of advisors from
Founder Atwood has retained Heidrick & Struggles to find a CEO, but the company seems to be doing fine for now without one. Instead, it is getting consulting advice part-time from Cooper & Raburn, a VC firm that also provides a sort of home-care consulting service to start-up outpatients. (Partners are Vern Raburn of Microsoft, Lotus and Symantec; and Phil Cooper of Palladian and Bachman, among others.)

While Ontologic is pulling back to conserve money and other resources, Object Design is taking the most ambitious approach of all, operating on the premise that a raw object-oriented database needs a development environment to amount to anything -- and to generate a life-sustaining revenue stream. So while Ontologic will rely on customers and third parties to build the tools to use its system (leading competitors to wonder out loud what it’s bringing to the party), Object Design will ship a suite of C++-based tools to support design and development of large-scale database-intensive applications as a core part of its offering.

Atwood’s team intends to provide a fully distributed system with support for nested transactions, so that user groups can establish and enforce their own protocols for concurrent access to their objects. In addition, the system will sport its own C++ environment and toolset, with graphical browsers, a friendly query system and the like. These may come from third parties, as Object Design considers its strength to be the substrate.

The West Coast

The two West Coast companies have clear roots in two separate communities: engineering and relational database. Of all the companies, Objectivity is most clearly focused on a market rather than a technology, and has designers who fully appreciate the problems they will encounter -- both practical and intellectual. CEO Bob Field comes from National CSS, Tymnet (founding VP marketing), Zilog (VP sales & marketing) and Ungermann-Bass (de facto founding VP marketing) and six years in venture at Bessemer. Chief technical officer Drew Wade was formerly with Daisy Systems as group VP of electrical design automation. They have already hired several brand-name design engineers from the electronic and mechanical engineering communities, with a couple more in the wings.

Those announced are Wing Cheng, working on multi-user services (concurrency, distribution, transaction management, networking), from Rolm/IBM, with a PhD thesis on distributed database; Larry Lai, working on user interface and the type manager, from Cadence, where he managed a design group and wrote the object-oriented interface to Cadence’s database for integrated circuit design, with a PhD thesis on functional testing of digital systems from Carnegie-Mellon; and Leon Guzenda, working on object and storage management, from ICL, where he was project manager for IDMS in a joint effort with Cullinet long ago, and ATP, where he was head of the database group and director of the application environment. Backing up Lai is Peter Moore, a PhD candidate at Berkeley and co-author of Oct, Berkeley’s second-generation CAD system.

5"Object-oriented" CEOs seem to be in short supply: Ontologic promoted Si Lyle from within after an unsuccessful search and Object Sciences has an acting CEO. Only Objectivity is run by a certified "entrepreneur," Bob Field, whose track record includes several start-ups but no objects or databases. In fact, the database world in general doesn’t suffer from an abundance of good management. Please send your resumes!

Release 1.0

30 December 1988
database in use at MCC and elsewhere. Backing up Guzenda is Jeff Yang, a computer-science PhD formerly with Bell Labs and Esvel, now part of Cullinet, where he worked on its SQL VAX database.

Objectivity prefers to keep quiet about about its strategy, except that it will ultimately move beyond UNIX to VMS and OS/2. Nonetheless, its basic personality is clear from the people on the payroll; all are shareholders.

The last for now...

Object Sciences is the youngest start-up of them all, headquartered in Redwood City in room-for-growth quarters (sublet from the Singapore Technology Corporation, an investor) across an inlet from the future world headquarters of Oracle Corporation. That’s only fitting, since founder Kee Ong spent a few brief weeks at Oracle before joining RTI, and investor and acting ceo Mike Seashols was chief sales/marketeer at both Oracle and RTI, following a common database market pattern. The flavor at Object Sciences therefore tends far more to database issues, but the company hasn’t yet completed enough hiring to make its focus clear. As a rough cut, we suspect that Object Sciences may focus a little more on general issues of heterogeneous systems and networking, while Objectivity worries more about specific engineering application support. Besides Ong and Seashols, board members include Steve Gall, representing backer TA Associates; and Mark Leslie, founder of Synapse (where Ong once worked) and now ceo of Rugged Systems, a seller of ruggedized VAXes.

...but not forever

None of the companies is willing to talk dates for the record, but we hope to be able to offer a nice panel at our Forum in the spring of 1990. All of them are using C++ with varying degrees of enthusiasm, and will offer their systems on standard UNIX platforms -- probably on Suns to start.

When are we going to hear of a system targeted at users of Objective-C or Eiffel? Although these systems are theoretically language-independent, especially Ontologic’s, in practice there are lots of little factors that favor the language they are developed in, C++.

The object-oriented environment

Beyond that, why wrestle with traditional file systems and operating systems at all? Why not extend the object-oriented database to manage the total environment -- data, tools, applications? In a fully object-oriented environment, the applications are attached to the data anyway. We’re a long way from that, but it’s a worthy goal to move towards. In a sense, we’re approaching that in another way with installable file systems such as what Microsoft promises for OS/2 (see our next issue), and with NextStep and the environment promised by ON Technology. In the long run, such smart systems will manage the execution of code as well as the fetching of data, storing data and performing procedures wherever it’s most practical.

But don’t hold your breath.

Release 1.0

30 December 1988
Soon, the experts said, expert systems will use artificial intelligence techniques to diagnose failures in equipment: cars, electronic gear, factory machines, computers. It's happening, but only slowly.

As we all learned in AI school, a diagnostic expert system uses empirical rules of thumb rarely based on any fundamental understanding of the situation they concern. "If the window wobbles, then the warp is off the woof. Conclusion: Try wiggling the widget." Such a system may or may not be effective and complete, but it's generally hard to build and difficult to validate. Its quality depends on the quality of the "expert" supplying the rules. Hence the slowness of business to adopt diagnostic expert systems.

A number of people, however, are developing tools that reason from "first principles" (a term coined in this context by MIT's Randy Davis) to build diagnostic systems based on a model that can be validated against the actual product under investigation. All you need is a model that matches the product in question -- an increasingly likely occurrence in this world of automated design (see pages 7 to 24). In the future, we can imagine such a diagnostic system as just one more application built around an object-oriented database of product models and components.

Even when rules of thumb work fine, they aren't extensible unless they're based on a solid understanding of the domain they address. By contrast, a model-based diagnostic system can be extended to encompass new models or variations with predictable results. In fact, since the expert system is based on the design of the product rather than a maintenance person's expertise, it can be built along with the product (or changed as the product is updated) without waiting for technicians to gain experience with it. (Experience is harder and harder to come by these days as product cycles get shorter and technicians' salaries get higher.)

The model-based "rules" are about structure and dynamics, rather than specific examples. ("Traditional" AI says, "If A, then B." The modeling kind says, "A has a C relationship with B, so if you change A this way, then B will change that way.")

It's a question of understanding chemistry and nutrition, vs. knowing a collection of recipes. Can you replace wheat with rye? With egg? With tomato paste? There may be no specific rules, but a model of the domain will suggest the answers. (Do earnings go up because time passes? Or because a positive return on equity generates a larger equity base on which to earn a return?) If you reason from a basic understanding of the domain you're diagnosing, you're likely to come up with a better system, and one that's more consistent.

The thigh-bone is connected to the shin-bone

In essence, the user constructs a functional model of the system to be examined, breaking it down into components and subcomponents until the limit -- the smallest unit that can be replaced or easily fixed in the field -- is reached. But it models its construction -- the parts and the connections among them -- rather than the overall function of the system. Each is defined by its (measurable) inputs and outputs. Then when a failure or mis-
behavior is detected, the fault can eventually be isolated to the object that produces incorrect outputs despite receiving correct inputs.

Each model of each kind of equipment needs a specially tailored expert system, and expert systems just aren't that easy to build -- claims that they require no programming to the contrary. True, with just a few (tens or hundreds) rules of thumb, expert systems can automatically make decisions, figure out what's wrong with a car, or rate a credit applicant. But figuring out the rules to put into the systems -- the process of so-called "knowledge engineering" -- is just as exacting a job as programming; the difference is that the result is more intelligible and easier to change when the rules change.

The first-principles approach uses a model of the system under inspection to create an expert system that knows where to look for flaws based on the structure of the model. It's still "dumb": It doesn't understand what the equipment does or why it works, but it understands the flow activity through the system. It doesn't work by rules but with simulation and inference (i.e., simulation in reverse) applied to the model. That is, it simulates the model's operation backwards and forwards until it finds a conflict, relaxes each of the model's constraints in turn to determine where possible problems might be so it can advise the user to test for what caused a conflict.

It makes sure the rules governing testing and detection of faults are complete, validatable and logically consistent. Maybe your expert never checks the electromechano-optical widget because it's never given him any trouble, but he should. These systems make sure that every part is under suspicion.

An early practitioner

Approaching the problem this way is AI² of North Chelmsford, MA. Its product, IDEA (for Intelligent Diagnostic Expert Assistant), costs $10,000 and runs on a personal computer. The systems it develops cost only hundreds of dollars (depending on volume) and are cheap enough that vendors of expensive machines can afford to send one along with each unit (realizing a long-held goal of sending a caretaker with every machine).

AI² requires the user to think about his system methodically and build a model of it. Once the model is complete, the builder-user can further enhance it with whatever test and repair instructions might be helpful. The result is a runtime copy of IDEA that can be used to diagnose failures in a given model of equipment. Of course, in the real world, there may be multiple causes for a single failure, or internal components with inputs and outputs that are not measurable in the field. But, likewise, in the real world this doesn't matter: If you know which board has failed, you can replace it, and leave the details to the home facility which will have oscilloscopes, shake-and-kick artists and the like.

Moreover, by limiting the flexibility of its system, AI² limits the number of choices the user has to make, so that the design process ends up being as easy as filling in a series of forms. On the other hand, it would be a lot easier if the system were expanded to include graphics, so that the user could model the system graphically. (Depending on the screen used, the system can display illustrations to guide users in making tests or repairs, but these aren't "live." That is, they're images, not models.)

Release 1.0 30 December 1988
Obviously, IDEA builds systems that are less flexible and powerful than the best you could build by hand, but that's a reasonable trade-off. It's relatively simple to develop, and it works fine to the level of allowing an on-site person to find the part that needs replacement and even to fix it in many cases (with human expertise incorporated). AI²'s first customer, Medical Systems Support of Dallas, services medical equipment, and is planning to place a PC with a copy of IDEA with each CAT scanner it services, starting with the GE 9800 scanner from General Electric. AI² is now working with MSSI on the second system, for another brand of scanner, which should be relatively easy to construct now that they have developed the model for the GE system. (As a startup, AI² is still working closely with its customers, but IDEA has the potential to render that unnecessary.)

Although AI² works from first principles, it is an eminently practical-minded company. Its four founders (out of five employees) come from Wang, where they were field-service technicians assigned to a technology-transfer project with MIT, where they met advisor and board member Randy Davis. They have deliberately limited their focus to diagnostic systems, well aware that their one paradigm will cover a multitude of customers -- more than they can handle for the foreseeable future.
A WAR ABOUT WORDS: THE ASHTON-TATE SUIT

Ashton-Tate's recent lawsuit against Fox Software and The Santa Cruz Operation has merit, but the battle in the press is over the Dbase language -- an item part of the suit only by reference. In fact, there has been a strange campaign of obfuscation on both sides: Ashton-Tate evidently hopes that if it wins the suit the press will treat that as a vindication of its outside-court claims for the sanctity of its language. (We find it hard to believe that any company with competent legal advice could be so deterred.) Meanwhile, Fox and others are also focusing on the language issue, where A-T's claims are far weaker than on the look-and-feel claims.

Like Apple Computer before it, Ashton-Tate has launched a lawsuit over a product whose fertile years are over anyway (although its hopes are to extend protection beyond that, an outcome we think unlikely). While both the Mac interface and dBASE itself provided major competitive advantages to their owners for a time, the battle has now moved on to other arenas. Both products are still a source of cash flow, but neither provides defense against new competitive incursions. (See Release 1.0, 88-4, for a discussion of the Apple suit.)

Ethically and practically, it boils down to economics. (That's what this is all about, isn't it?) Does Fox benefit from its product's similarity to dBASE rather than from its inherent superiority? Its claims that FoxBASE has additional, different screens and runs faster are beside the point; it has copied dBASE screens and used its similarity to dBASE in advertising. We were stunned by Ashton-Tate's impressive press materials limning the similarities between its screens and those of FoxBASE. They seem as close as Twin's and Paperback Software's to 1-2-3's, and the marketing message is clear: FoxBase is an attempt to trade directly on the success of dBASE in the marketplace.

On the other hand, Fox is now about to abandon the old dBASE interface anyway in its new FoxPro product, demonstrated at Comdex and shipping soon) in favor of something closer to the interface on its Mac-based product, which uses the Dbase language but none of its look. Moreover, says Fox president Dave Fulton, he's tired of hearing his product lumped in with dBASE because FoxBASE is so much better. In other words, the similarity to dBASE might have helped in the past, but those days are over.

All this fits in with our belief that contrary to claims of impeded innovation, intellectual property protection encourages significant improvements in functionality, since minor ones aren't enough to overcome the retarding factor of a different interface. Fox should have had the courage to abandon the dBASE look and feel long ago. And Ashton-Tate should have sued Fox long ago...or not at all.

Countercharges

According to Fox's counterclaim, Ashton-Tate actively encouraged Fox's efforts, to the point of inviting it (among other third parties) to the 1986 Developers' Conference and featuring FoxBASE and FoxBASE+ "in the printed materials distributed by Ashton-Tate at the conference...as '100% source compatible with dBASE II...and dBASE III PLUS.'" It also considered buying Fox and took a close look at FoxBASE -- to the extent, Fox argues, that A-T

Release 1.0
30 December 1988
may have stolen property from Fox. (In fact, the list of items appears to be a variety of capabilities that dBASE should have had long ago, many of them implemented differently, rather than any unique functions. Few of them "look" or "feel" the same, since they are capabilities, not screens or lengthy command sequences.)

That being so, Ashton-Tate has shaky grounds for its new attitude. Legally it may have a case, since copyright (unlike trademarks and trade secrets) does not have to be enforced to remain in effect until the end of its specified life. On the other hand, you may not get much in the way of remedy for past infringement. So the courts may well rule in A-T's favor but probably won't award much in monetary damages, given the validity of Fox's counter-claims. In that case, the impact will be minor, since these are all old products (or why have we all been waiting so eagerly for dBASE IV?). Let's stop fighting over spilt milk when there's work to be done, and a new graphical interface to create.

A war about words

The real issue is what the court says about the more controversial aspect -- the status of the Dbase language. A-T is clearly hoping for a broader definition of exclusive rights to that language. Says Esber: "The courts are incrementally setting the boundaries of intellectual property protection. People are beginning to act as if there's no protection, so the bottom line is I can't lose [in court]. The issue is to get the court to define the law, not to shoot at everyone."

The specifics of the suit are tighter than the hoped-for ruling or Esber's public statements, says A-T counsel Stan Witkow, because his first charter is to win the case. That's why there's no mention of Nantucket Software, which makes a Dbase compiler. WordTech is partially protected because of a 1987 deal wherein it figuratively traded Harry "SQL" Wong to A-T in return for such protection. If A-T did win on the language issue, moreover, it wouldn't have to sue these firms anyway; they'd probably settle.

The status of the Dbase language is far more arguable than the dBASE look and feel, which precedent says is protectible. (The argument lies in what constitutes "look and feel" in any particular situation.) Part of the confusion stems from the word "interface," which refers both to a program's look, or the screens it presents to the user; and to its feel, the sequence of screens, also potentially protectible in toto, as well as the commands it understands and translates into machine language. The words themselves have no particular sequence (although there are certainly well-defined sequences which are possible and others that make no sense). Thus, while the code of any particular compiler is clearly protectible, the status of the commands it translates is less clear. A language, in a sense, is an interface, and a published interface is, by definition, public. You can talk to it, even if you can't imitate it.

But what constitutes "publishing" an interface? What's the implication of encouraging third parties to write applications in your language or compilers for it? A-T's Esber asserts that sale of its products includes a limited license to use the language in writing applications for an A-T interpreter [or compiler, now that there is one], not a general "publication" of it. We find that notion intriguing but unrealistic. Regardless, it's worth considering -- and it's certainly not morally outrageous.

Release 1.0 30 December 1988
Nonetheless, whatever the general status of a language, and whatever A-T’s particular legal situation, the putative owner of anything is responsible for consistency. In other words, the law should work as it does for trademarks: If the owner of a novel language wishes to declare it proprietary, fine. If the owner wishes to publish it freely and encourage others to use it, also fine; but then that owner has no right to turn around later and sue them for its use. As a practical business matter, proprietary language generally make little sense. But if a vendor might decide he can make more money by building customized applications with a secret, proprietary language, that’s fine.

In general, vendors have put languages into the public domain, so that users would use them and third parties would support them and everyone would turn to the language vendor for compilers and other tools. (It was A-T’s own dilatoriness in building a compiler for Dbase that led ultimately to much of this fuss.) In such cases, anyone should be free to interpret a language’s set of commands. Only the code that interprets the commands (and the documentation that explains how they work) may be protected.

All these issues are matters of degree, but reasonable people can come to the same conclusions in a large proportion of cases. Obviously, a command such as "get" should be fair game, whereas one such as "hairy-ape," which causes a program to draw a particular arrangement of pixels on a screen might not be. But that’s casuistry: It’s not the language that’s protected there, but the image of the hairy ape.

Just the beginning

In the future, we face a more interesting problem as languages become richer in content, with class libraries proliferating. While the basic syntax of an object-oriented language may be freely reused, what of its class libraries? Worse, whatever the ownership of those libraries, they are expressly created to be extended by builders of systems, who may in turn sell those systems to others. Will we start paying royalties on the basis of "object reuse content 78 percent"? Well??? Royalty accounting has always relied on good faith and careful accounting; now it’s going to rely on good faith and careful object-use auditing.

Of course, pricing takes magic. There’s a reason for those all-you-can-eat restaurants, for flat fees in hotels regardless of when you check in and check out, for health clubs that charge by the year and not by the visit. There’s a reason we all pay the same for our newspaper whether we use it to wrap fish, to find out about world affairs, or to make a killing in the stock market. It’s too difficult to keep records, and transaction costs might outweigh the value of the product or right transferred.

Because we’re dealing with electronic assets and the electronic world is digital, there’s an illusion that everything can be accurately measured, and ownership and value assigned. In fact, we can measure the bytes precisely -- but this is about intellectual property and its value in a market of human beings and commercial enterprises, where custom and contract law reign. Incentives and productive capabilities can be measured only in the aggregate and by their ultimate impact, which may not be separable from other factors. That being the case, let’s write our contracts carefully, define our property explicitly, and get on with the business of building software!
RELEASE 1.1: MORE FILE CONVERTERS

As soon as we said that Flinder Software's Zeno file-conversion software was unique so far as we knew (see Release 1.0, 88-11), two other such systems turned up, using slightly different technology to accomplish more or less the same end -- conversion and filtering of data including mainframe reports into formatted data suitable for use by databases or spreadsheets. One is CrossFile from Saxon Systems, which was shown at Comdex in the Softsel booth and started shipping last week ($150). The other is Extract, another new product on its way from Gnu Technologies of Los Angeles.

JOIN THE OPEN HOLE FOUNDATION (There's a hole in this story...)

We recently received a press release announcing the establishment of the Open Token Foundation. Whether tokens, software and other products should be open or closed is certainly open to discussion, but the case for open holes is clear.

It's time for the software community to rally round the open-hole movement and help stamp out the proprietary loopholes that threaten to close off further progress and provide competitive advantage to certain key players (you know who you are) who are taking advantage of proprietary loopholes.

Let's turn those nasty loopholes into generic, open holes! Other holy threats to openness include keyholes, mouse holes (a field unfairly dominated by Apple and Microsoft), foxholes (a database product), potholes, sinkholes, black holes, manholes, copy holes (for missing stories), peephole, holes-in-one, holes in the head, hellholes and rat holes, doughnut holes, Swiss-cheese holes. And then there are spurious holes, such as the sum of the parts, strongholes, handhole computers, hole-earth groups and pot-holeders.

Please join us holeheartedly to make the Open Hole foundation an open, value-free success. (You can sign up for our first committee meeting February 29 by filling out the attached form in triplicate -- mind the punched holes! -- and including a notarized statement from your attorney.)

---holes of the world, unite!

Release 1.0

30 December 1988
1989 PERSONAL COMPUTING FORUM: GET SET FOR THE NINETIES

The 1989 Personal Computing Forum will take place from March 19 to 22 in Palm Springs, in search of good weather and morning alertness among West Coast attendees. We’ll be starting our waiting list soon (space is limited), so please hurry! If you have lost (?! your registration form (two per subscriber; copies are fine), you can call us for another, but we can't take registrations by phone.

Please come and celebrate in Palm Springs!

Daphne Kis, associate publisher

Our preliminary roster of confirmed speakers includes:

Tania Amochaev  Natural Language Incorporated
Gordon Bell       Ardent
John Seely Brown  Xerox PARC
Jim Cannavino     (newly at) IBM Entry Systems
Vittorio Cassoni   (back at) Olivetti
John Dvorak       Himself (EDwards MC)
Bob Epstein        Sybase
Ed Esber          Ashton-Tate
Gordon Eubanks    Symantec
Bill Gates         Microsoft
Adele Goldberg    ParcPlace Systems
Bill Joy           Sun Microsystems
Philippe Kahn      Borland International
Mitch Kapor        ON Technology
Bob Kavner         AT&T
Jim Manzi          Lotus
John Roach         Tandy
John Sculley       Apple Computer

We are also arranging company/product/concept presentations by certain attendees. They may include (not all confirmed) Asymetrix, Calera, Chips & Technologies (Gordie Campbell), Coordination Technology, Hewlett-Packard (Bob Frankenberg, Bill Murphy; NewWave), Novell (Craig Burton), Phoenix (Neil Colvin), Premise, Traveling Software (Papillon), Saros, Stepstone (Objective-C and NextStep), Sun Microsystems (Wayne Rosing), Wang (Steve Levine, Freestyle).

Desert adventure! Stanford PhD in geophysics Colleen Barton (Larry Tesler’s wife) will lead a day-long field trip of natural sites and a geodetic station (where they measure fault activity) on Monday for companions, "children" over 12 (with a parent) and miscreant attendees. (Palm Springs sits near the San Andreas fault.)

Release 1.0

30 December 1988
RESOURCES & PHONE NUMBERS

Kevin Flood, AI², (508) 251-7400
Ed Esber, Stan Witkow, Ashton-Tate, (213) 538-7714, 538-7719
Kathleen Riley, Automation Technology Products, (408) 370-4000
Bruce Bourbon, Cadence, (408) 727-0264
Brian Groves, Ray Stochowiak, Datalogics, (312) 266-4444
Ray Taylor, Dave Fradin, DocuGraphix, (408) 446-9700
Isadore Katz, Tony Zingale, EDA Systems, (408) 986-9585
David Fulton, Fox Software, (419) 874-0162
Joshua Stern, Gnu Technologies, (213) 826-4149
Don DePalma, James Starkey, Interbase, (617) 275-3222
Sharon Burgmeier, Kewill Systems Plc., U.K. (0932) 248 328
or John Wicker, Micross (US distributor), (201) 808-9800
Dan Weinreb, Tom Atwood, Object Design, (617) 270-9797
Kee Ong, Mike Seashols, Object Sciences, (415) 637-9220
Drew Wade, Bob Field, Objectivity, (415) 688-8000
Si Lyle, Bob Martin, Ontologic, (508) 667-2383
Juan Tigar, Alice Haynes, Open Hole Foundation, (212) 758-3434
Leonard Simon, Saxon Systems, (213) 404-4030
Jere Brooks Hunter, SDRC, (513) 576-2400
Jacob Stein, Mike Connell, Servio Logic, (503) 644-4242
Steve Schopbach, Brian Stolle, Sherpa Corporation, (408) 433-0455
Yuri Rubinsky, SoftQuad, (416) 963-8337
Bob Epstein, Sybase, (415) 596-3500
Dave Hoffman, Kirk Norlin, TeamOne, (408) 986-9191
Steve Levine, Wang Labs, (508) 459-9000
Dave Miller, Dan Berkowitz, WordTech Systems, (415) 254-0900

COMING SOON...

- Visual programming languages.
- A composition engine.
- DOS shells and file managers.
- Status report on HP's NewWave.
- OSF's interface choice.
- Active and passive objects.
- And much more... (If you know of any good examples of the categories listed above, please let us know.)

Release 1.0 30 December 1988
January 17  
Fourth annual joint conference of New England software developers, distributors and legal counsel - Cambridge, MA. A non-intellectual look at intellectual property: software licenses and contracts covering outright sales, resale agreements, overseas marketing, unexpected (?) events such as bankruptcies. With a variety of academic and practicing experts. Sponsored by Franklin Pierce Law Center, MIT, others. Contact: Jeanne Moser, (603) 228-1541.

January 17-19  

January 17-19  

January 20-22  
Macworld expo - San Francisco. Sponsored by Contact:

January 23-27  
Improving productivity in EDP systems development - Phoenix. Sponsored by Applied Computer Research. Contact: David Wendland, (602) 995-5929 or (800) 234-2227.

January 26  

January 29-31  
Image scanning and processing - Monterey. Sponsored by Institute for Graphic Communication. A hot new area, with lots to discuss. Speakers from Apple, Calera, Kurzweil, Xerox. Contact: Lynn Bouthillier, (617) 891-1550.

January 30-February 2  
Goldman, Sachs technology investment symposium - New York City. Concurrent sessions with a total of 50 companies, plus speeches by Mitch Kapor and Will Zachmann. Contact your Goldman, Sachs rep or Andrew Krawitt, (212) 902-7771.

January 30-February 3  
Winter USENIX technical conference - San Diego. Keynote by AT&T's Bill O'Shea. Tutorials on C++, Eiffel and other object-oriented programming languages, as well as X Window, Andrew Toolkit, Mach (by NeXT's Avadis Tevanian) and many UNIX-specific topics. Contact USENIX at (213) 592-1381 or 592-3243.

January 31  
Software Publishing Corp. at the analysts' - New York City. Sponsored by the New York Society of Security Analysts. Contact: Lourdes Figueroa or Carol Morgan, (212) 344-8450 or (800) 248-0108.

February 1-2  
Businessland analysts' briefing - San Jose. With Dave Norman and newly promoted COO Jim Heisch. Includes a tour
of the Hayward distribution center. Contact: Beverley Bird, (408) 437-0400.

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


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


February 28-March 2 NetWorld - Boston. Keynote by James Bruce, vp of information systems at MIT. Sponsored by Novell. Contact: Ed Palmer, (408) 986-8840 or (800) 323-5155; or (312) 299-3131 for registration information.


March 6-10 Fifth IEEE conference on artificial intelligence applications - Miami. Keynote by Ray Kurzweil, plus Mark Fox, Ron Kaplan, Sanjay Mittal, Beau Shell, Mike Stonebraker, Len Tedesco (Ford), Esther Dyson, Ted Nelson, John Clippinger. Call IEEE, (202) 371-1013, or Mark Fox, (412) 268-3832.

March 9-10 COMPUTER ACCESS AND USE FOR DISABLED PERSONS - Decatur, GA (near Atlanta). A nuts-and-bolts workshop on a topic of increasing social and business impact. Do you know about recent government regulations on the topic, and how to implement qualifying products? Sponsored by Trace R&D Center (see Release 1.0, 87-6). Contact: Greg Vanderheiden, (608) 262-6966.

We have truncated this issue's calendar for production reasons. We will publish the full listings next month. Please let us know of any other events we should include. -- Denise DuBois

Release 1.0 30 December 1988
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? ________________________________

Please fill in the information above and send with your check payable to: EDventure Holdings Inc.
375 Park Avenue, Suite 2503
New York, NY 10152

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

Daphne Kis
Associate Publisher

Release 1.0
30 December 1988