THE GILBANE REPORT™
on Open Information & Document Systems

Vol. 1, No. 6
January/February 1994

Publisher:
Publishing Technology Management, Inc.
Applelink: PTM
(617) 643-8855

Editor:
Frank Gilbane
fgilbane@world.std.com
(617) 643-8855

Subscriptions:
Carolyn Fine
carolyn@world.std.com
(617) 643-8855

Design & Production:
Catherine Maccora
(617) 241-7816

Associate Editor:
Chip Canty
ccanty@world.std.com
(617) 265-6263

Contributing Editor:
Rebecca Hansen
MCI Mail: 4724078
(617) 859-9540

Document-Centered User Interfaces & Object-Oriented Programming — How Will They Affect You?

It may not seem these two topics have much to do with each other at first, but together they will have a profound affect on how we use computers in everyday business environments. You can get a glimpse of where this is all headed by looking at some existing document management software and operating system features. In this issue Keith Dawson explains why these two trends are important, how they relate to each other, and how we can expect their convergence to affect our development and use of software for business applications.

This issue also sets the stage for a future article analyzing the compound document architectures that will be vying for dominance in the next generation document computing environments.

Anniversary Issue

It seems hard to believe, but with this issue we conclude our first year of publication. It has been an exciting year for us, and we look forward to another year of keeping you up-to-date on the issues and technology for managing documents and document information.

In our next (anniversary) issue we will look at how Andersen Consulting helped the State of Wisconsin Legislature reengineer their entire document workflow and management process, and put together a document management application that involved integrating multiple document management products.

We also will begin to report news items relevant to document systems and document management that we think will be of interest to our readers. You can be sure there will be a lot of activity in the coming year. Stay tuned!

Contents

Document-Centered User Interfaces & Object-Oriented Programming — *How Will They Affect You?*

Executive Summary

The Impetus for Change

- Despite desktop-based graphical user interfaces, computers are still too hard to use.
- Individual software products have become bloated with functions, such as spell-checking, that could be shared across applications.
- The number of distinct applications programming calls available today forces programmers to specialize and complicates applications development.
- Falling software prices and tight IS budgets also create incentives for greater programming efficiency.

How "Objects" and "Documents" Help

- With object-oriented programming (OOP), the amount of programming needed scales linearly (not geometrically) with the complexity of the project.
- OOP fights "API overload" by letting programmers build applications from a relatively small set of high-level objects.
- OOP and document-centered applications allow persons "farther down the skill pyramid" to take on tasks that only programmers can tackle today.

The Document and the Desktop

- In programs based on a desktop metaphor, users must understand how to navigate through computer's file system and how to relate files to applications.
- Spurred by innovations in publishing software, many vendors have begun adding support for various forms of "document-centered" functionality.
- The initiatives taken to date by operating system vendors are largely incompatible; such lack of standardization has made applications developers slow to support them.

Document-Centered User Interfaces

- Pen-based computers are beginning to expose users to document-centered user interfaces (DCUI's).
- Key features to be found in DCUI's include linking, cross-application "drag & drop" capability, automatic activation of applications, and location transparency.
- *App-lets*—mini-applications that "plug into" other programs—are a promising outgrowth of OOP and DCUI technology.

Implications for the Future

- An object management standard, CORBA, promises to help ensure that object-oriented systems can share applications and objects.
- OOP will force new applications developers and operating systems vendors to negotiate new relationships, as more core technology is absorbed into OS-level objects.

- DCUI's and OOP create new opportunities for IS departments to increase productivity and improve service.

Risks & Costs

- Planning for DCUI's may uncover ways of using existing document-oriented features in existing or installed products.
- Early object-oriented operating systems may lack the features needed to re-engineer applications around DCUI's.

Conclusions & Recommendations

- Be careful in committing to using a particular OOP environment. Study your own applications first so that you understand what resources are needed to transform them into object-oriented programs.
- Support standardization efforts such as COBRA which can help ensure the portability of applications and objects across platforms and operating systems.

The Document-Centered Model and the Object-Oriented Paradigm

In an exciting new trend in software design, application developers today are giving computer applications the "look and feel" of books, notebooks, memos and other documents.

The goal behind this movement is to make computers easier to use. But discussing *document-centered user interfaces (DCUI's)* is hard without also addressing object-orient programming (OOP), a parallel trend whose goal is to make computers easier to program.

For many reasons the computer industry is moving to the object-oriented model, the programming context in which document-centered interfaces are being developed and deployed. Soon these two threads will meet in end-user computer environments. Applications will become customizable and tailorable by any user in ways that today would require hours, days, or weeks of concentrated, expert programming effort, and developers will have access to even higher-level facilities from which to construct robust new applications quickly.

These developments will have both obvious and not-so-obvious implications for managers of information systems.

"… in a few years users will have to deal with multimedia documents in relational and object databases on file systems that span continents."

Background: The Impetus for Change

Why is the industry both adopting a document-centered applications metaphor and an object-oriented programming model? Let us outline a number of problems that characterize the worlds of computer users and developers today, and look briefly at the ways the trend to objects helps alleviate them.

Computers Are Still Hard to Use

Despite windows, icons, drag-and-drop, and all the other trappings of modern graphical user interfaces (GUIs), computers still are not simple enough to use. The desktop is showing its age.

As today's reigning paradigm, it is certainly a large improvement over cryptic commands and codes. But in a few years users will have to deal with multimedia documents in relational and object databases on file systems that span continents. The "desktop" is too

small, too narrowly focused to present us with an adequate view of the data universe through which we will all need to navigate.

Applications are Ballooning

Applications today are chock-full of features that the average user neither needs nor understands. The recent commoditization of software—and the degree to which vendors must fight for shelf space in a frighteningly narrow sales channel—force developers to cram all the features that *any* user may want into the software that *every* customer has to buy.

As a result the "footprints" of applications have grown—both in the amount of disk space they consume upon installation and in the amount of memory they need to run. Vendors can assume little about users' computing environments. Under today's model, code that is needed by many different programs to handle common problems must be duplicated in, and delivered and installed with, each application.

The result is not only disk- and memory-glut, but more costly applications that take longer to develop and to test. User frustration goes up as duplicate features proliferate, many with very different user interfaces. Training burdens increase. Computer users, having invested in learning one set of capabilities, become specialized and focused on one environment, reducing their flexibility and that of their enterprise.

"API overload also forces programmers to become increasingly specialized, which in turn reduces managers' flexibility in assigning staff to different programming projects."

API Overload

The term *applications programming interface (API)* ideally refers only to the set of programming calls supplied for a particular system.[1] Application programmers on most major operating systems today must contend with up to 5,000 to 7,000 separate API calls. This number rises as operating systems become more powerful, so unless we adopt a new development paradigm, we can expect it to climb soon to the neighborhood of 10,000 to 15,000 calls.

Even today the sheer number of API calls represents an unwieldy burden—it makes cross-platform development increasingly difficult and expensive, and stretches out product development schedules. "API overload" also forces programmers to become increasingly specialized, which in turn reduces managers' flexibility in assigning staff to different programming projects.

Economic Factors

Non-technical factors, too, put pressure on applications developers to adopt more cost-efficient development methodologies.

For commercial software developers, one is the price that users are willing to pay for software. The price of software today is in free-fall. Partly this is because hardware, too, is cheaper, which influences what users think software "should" cost. Partly, too, it owes to the rough-and-tumble competitiveness of the PC software market, which, to give just one example, has now driven the street price of PC database software to under $100.

For in-house IS departments, the cost drivers are shrinking budgets and increasing demands from users for support. Users—especially those who know enough about PC software to be dangerously naïve about the true costs of customizing applications—don't want to wait weeks or months for access to new features. Competing demands on IS departments to maintain legacy code or to re-engineer aging but mission-critical applications have to be reconciled with end-user demands and "downsized" budgets.

[1]To confuse matters, the term "API" is also sometimes applied to a single call from such a set.

How "Objects" and "Documents" Help

But why objects? How does the use of OOP address these problems? And what is the nature of this synergy between object technology and "documents" and/or DCUI's?

"Once low-level objects exist… development effort scales linearly with system size, not geometrically…"

Let's look first at objects. Depending on the context, the word "object" can refer to a component at any level, from the very basic (*e.g.*, an integer) to the large-scale (a document part, a whole document, or a network server). *Object-oriented development* refers to a process of analysis, design, and programming centered on the use of objects in all phases of a project.

Object orientation promises efficiencies at many levels. Once low-level objects exist, programmers can use them to construct higher-level objects. Building each successive level requires about the same effort as the previous one. In other words, development effort scales *linearly* with system size, not geometrically as at present.

Document-centered interfaces, on the other hand, represent the whip-end of the movement to object orientation— a paradigm for presenting to the end user a perspective on the power and flexibility built into object-oriented applications. Let's keep that in mind as we review some of the ways in which object orientation addresses the problems enumerated above:

Death of the mega-application

Document-centered computing lets us break the huge, category-spanning applications we know today into smaller functional pieces. There are a number of ways in which this modularity can help craft smaller, more focused applications for end-users. (See "Applets" below, for example.)

New options for software packaging and pricing

Such modularity allows us to re-invent ways of delivering software to users; doing so promises to help us overcome our current software pricing dilemma. OOP encourages layered software architectures and standardization at the boundaries between layers. A more standards-based world encourages innovation and helps smaller, entrepreneurial developers both (1) by enabling them to attach code to add functionality to more established software products and (2) by encouraging the proliferation of software products aimed at smaller "niche" markets.

A way out of "API overload"

Instead of worrying about 10,000 API calls in the C language, a programmer in the future may only have to deal with a few hundred higher-level objects. This promises to greatly improve the productivity both of individual programmers and programming teams. Some object-oriented development environments also run on or across different hardware platforms, further reducing the complexity presented to application developers.

Objects model the real world

To create a business application in object-oriented fashion, one begins by identifying the "people, places, and things" that are important to the application. Each is then modeled by software objects as the application is developed. Keeping such a close tie between business analysis and programming simplifies the task of re-engineering applications, especially as businesses move critical applications and databases off legacy mainframe systems and onto more modern technology.

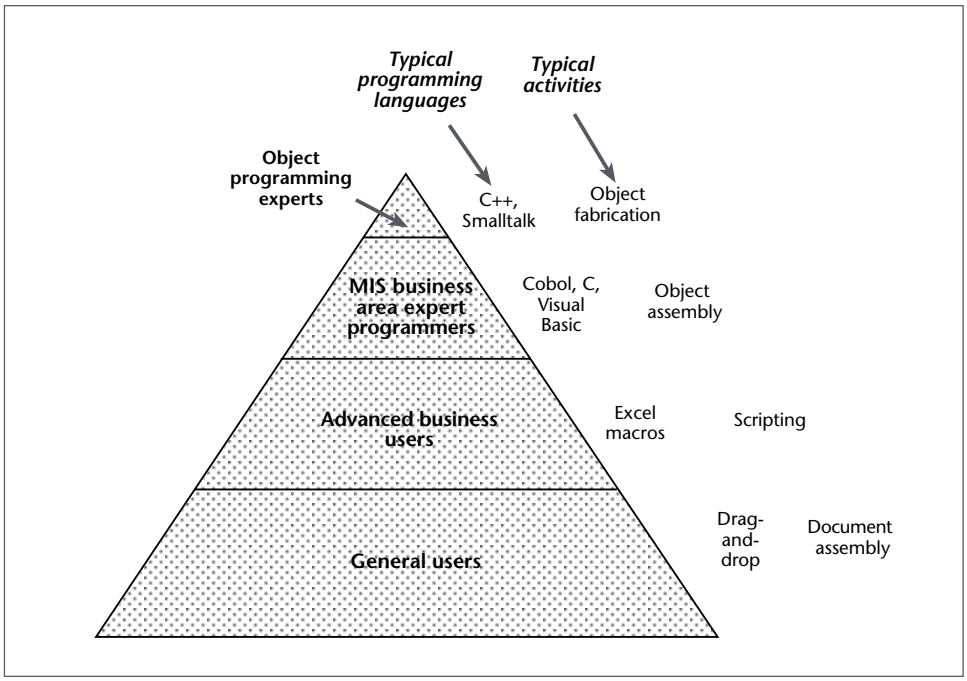


Figure 1

Programming expertise in an organization

"Objects hide the details of how they work, presenting instead a simple and consistent interface to the outside world."

Helping dissimilar systems work together

Objects hide the details of how they work, presenting instead a simple and consistent interface to the outside world. Other objects use these interfaces to access the services that the objects provide. As standards emerge that describe how objects communicate (see below, "Standards: OMG and CORBA"), it becomes easier for systems with different software and hardware to cooperate in distributed systems.

Making the best use of programming skills

Object technology promises to reduce demands on business application programmers by allowing people farther down the skill pyramid (see Figure 1) to build their own applications using higher-level parts.

IS departments are struggling to meet user demand for new and enhanced applications while maintaining legacy code, re-engineering applications or introducing client/server computing models. Once object technology gives these departments the tools and training they need to produce objects and to combine them into useful applications, several things can happen: (1) programmers who are not object experts can build applications by piping together high-level objects; (2) sophisticated end-users can build their own applications using objects distributed by IS; and (3) the skills of the experts can be focused more narrowly on lower-level code or more demanding applications;

OOP has also encouraged the development of visually oriented tools with which one can build applications without traditional programming. Examples of such tools include Oberon Software's *SynchroWorks* and DigiTalk's *Parts*.

For all these reasons, many firms have decided already that the benefits of adopting OOP outweigh the costs of re-designing and re-building applications.

As the 1980's saw a shift from monolithic computer systems to client/server computing, the 1990's will see a further shift to distributed object-based computing and development paradigms. (See Figure 2) Vendors of operating systems are working to incorporate object-oriented extensions into their environments. Some are even developing new object-oriented operating systems from the ground up. (See box, "Leaders in the Move to OOP.")

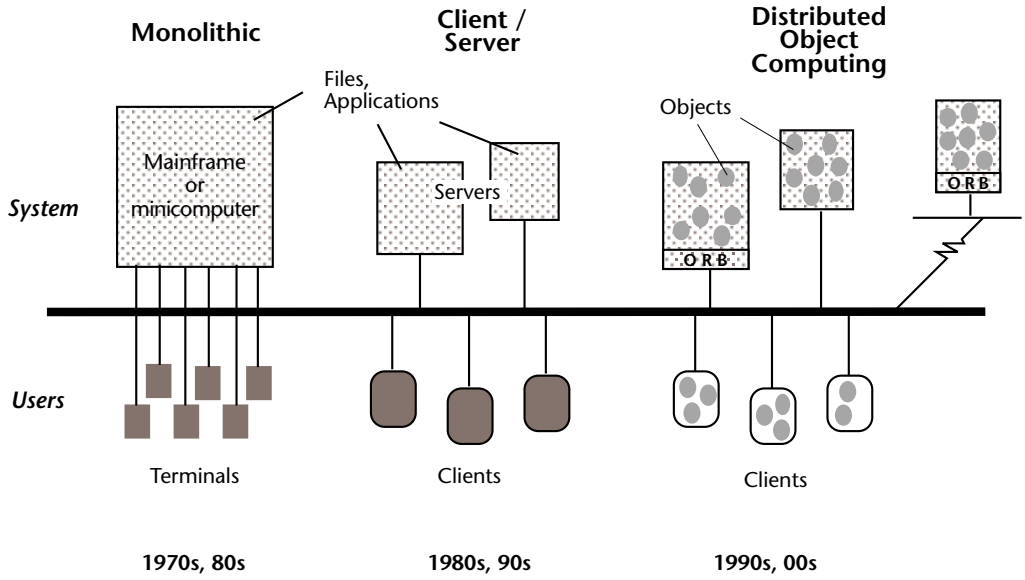


Figure 2. The movement toward distributed object computing

The Document and the Desktop

In this context, the movement towards document-centered interfaces is a manifestation of the shift to object orientation—specifically, the part that is visible to end-users.

Before studying DCUI's and their implications, let us first review the role of documents in the current desktop-oriented U/I paradigm.

"Under the desktop model, applications eventually become feature-heavy and evolve into mega-applications."

Documents and Applications

The desktop metaphor pioneered by Xerox PARC and commercialized by Apple dominates user interfaces today. Under this model, the *document* is the repository of the user's work. (The term "document" is itself a metaphor borrowed from the physical world.) A document can represent something that looks like a traditional book or report; but it could also represent five pictures, a stack of index cards, a multi-dimensional spreadsheet, or a film clip with audio.

In this model, both computer users and developers think first of *applications,* then documents. Applications are what users buy to get their work done. As a rule, different applications cooperate and intercommunicate in only rudimentary ways. Applications know how to display, edit, and print only certain kinds of documents.

Applications have native forms in which to store documents, with some ability to import and export documents from and to the formats of other applications. The native form is usually the most efficient, compact, or fine-grained storage format.

Under the desktop model, applications eventually become feature-heavy and evolve into *mega-applications*. With each new release, vendors competing for market share feel compelled to add new features and to make their applications more aware about other resources or limitations that may characterize a particular computing environment or problem domain.

As this happens, common facilities that may be needed in many problem spaces are duplicated in each application. Spelling checkers provide a good example: a different one

may be included in each and every word processor, presentation application, drawing package, etc., that users install on their systems. In addition, vendors begin adding features to undercut competition from vendors with essentially dissimilar products. Word processors, for example, grow page-layout features; page-layout programs sprout both drawing and word-processing functions; presentation programs add all of the above, plus outlining, audio-visual functions, and multi-megabytes of clip art and "clip media."

The Desktop Operating Environment

While the document may be central to the user's concerns, it is applications and files that are central to the desktop view. A weak point of all existing desktop user interfaces is that at some point they require users to navigate through some representation of the file system to locate a document or application to launch.

Most of the methods for doing so require some degree of awareness from the user of the document's location in the underlying file system.

Any exceptions to this rule result from establishing explicit associations between applications and documents (files). Various schemes are used in desktop environments today to accomplish this. Microsoft Windows relies on three-character file-name extensions and a table of associations between such extensions and various applications. The ultimate responsibility for filling out this table, and ensuring that it stays up-to-date, resides with the end-user. The icon used to represent either an application or a document is also left for the user to assign.

On the Macintosh a more rigorous association of icon, document type, and application is tracked by means of four-character *type* and *creator* codes associated with each file in the file system. These codes are not normally visible to the user; however, when they get out of sync, the user may be faced with blank icons and no easy way to identify to what application a particular document belongs.

Cracks in the Desktop

Some of the earliest challenges to a strict desktop metaphor came from electronic publishing applications such as Ventura Publisher.

Single-application features that resemble linking and embedding were introduced as early as the mid-80's.[2] The first version of Ventura Publisher (1986), which ran under the DR/GEM windowing environment, supported links to external word-processor files. Aldus PageMaker by the time of version 3.0 (1988) allowed the user to link to graphics files. By version 4.0 (1990) Aldus had extended support to linking or embedding graphics or text files, with optional updating if the external files changed. Microsoft Excel by the time of version 2.0 (1989) supported linking to external spreadsheet files.

Interleaf 5, released in 1991, introduced as *Active Documents* what we would now call adaptive dynamic object linking, whereby external events could "call" documents and tell them to change themselves via a Lisp-based scripting language. Other publishing system vendors too, including ArborText, Xyvision, Frame/Datalogics, and SoftQuad, have provided scripting capability for linking to third party products and incorporating distributed data into documents.

[2]If we consider cut-and-paste as the simplest form of embedding, we can go back before 1980 to the Apple Lisa and the Xerox Star.

Document-Centered Features Today

Figure 3 illustrates the development of features supporting a document-centered world-view.

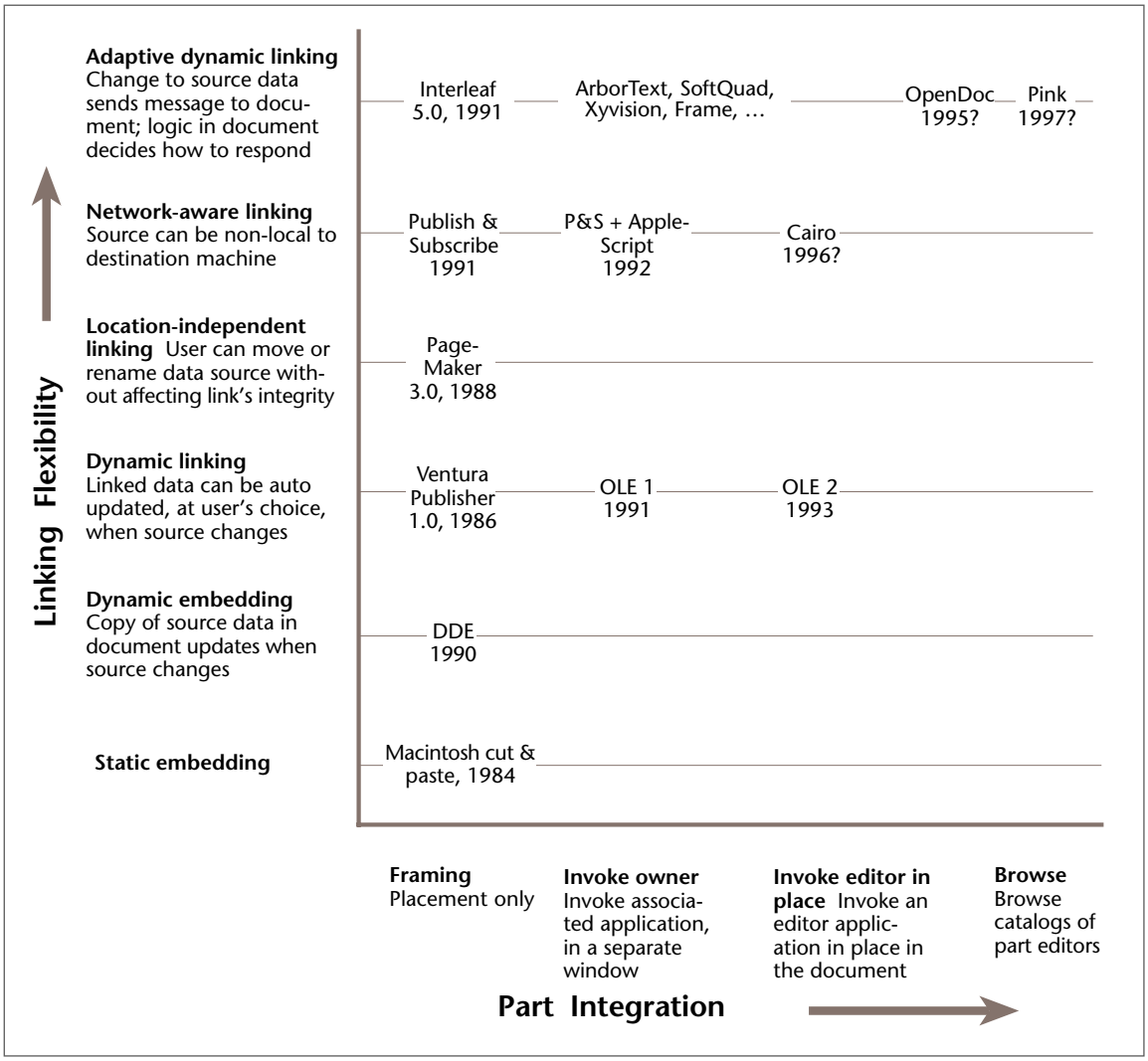


Figure 3. The history and outlook for features supporting document-centered interfaces

DDE and OLE

Microsoft introduced a simple flavor of OS-level linking support, called dynamic data exchange, in the Windows environment in 1990. Microsoft also supported DDE in Microsoft applications running on the Macintosh once Apple released System 7.0 in 1991. An application vendor using DDE could implement linking and embedding of one document or part inside another.

Microsoft's Object Linking and Embedding technology was introduced on Windows in 1991. OLE 1.0 provides for activating another application (in a separate window) from a part embedded in a document.

OLE 2.0, announced in May 1993, supports in-place activation of linked applications and drag-and-drop embedding. As such it is a platform on which most of the features of the document-centered vision could be implemented—at least on a single machine.

Publish and Subscribe

Apple's current network-wide linking technology is called "Publish and Subscribe" (P&S).

Neither DDE-based nor OLE-based embedding on the Macintosh, as implemented in Microsoft applications, is network-aware: the source application must be present on the same machine as the application that uses the linked/embedded information.

P&S, however, removes the dependency on a creating application. Once you *publish* an *edition* that represents some information from a document, any other document on the network can *subscribe* to the edition without recourse to the original application. Whenever you change the published information in the original document, the operating system updates both the edition and all documents that subscribe to it.

"... the first glimpses of truly document-centered user interfaces are coming courtesy of hand-held "pen" computing systems."

AppleScript

Apple's AppleScript, shipping since 1992, automates the invoking and coordinating of Macintosh applications across a network. Major components of the Apple environment, such as the desktop Finder, know how to respond to Apple Events. A number of third-party applications have been made scriptable as well. A third-party product called Userland Frontier allows users to knit together such scriptable applications in a straightforward way.

Visual Basic

Microsoft has announced its future direction for scripting— its Visual Basic language—will be extended to meet this need. By the time Windows 4.0 (Chicago) ships, users should be able to program its desktop and file functions using Visual Basic.

Application-Specific Plug-Ins

One early step on the path away from monolithic applications was the introduction of application-specific *plug-ins*.[3] To support plug-ins, the vendor rearranges the application architecture so that there is a hub to which the plug-ins can attach, and publishes the specs for the API that a valid plug-in must satisfy.

Vendors who support plug-ins ordinarily supply several with the product, then work with cooperating software developers who want to write others.

Dynamic Link Libraries

Microsoft's Windows environment supports dynamic link libraries, or DLLs. Apple has not released corresponding functionality, and the uncertainty over Apple's plans in this area have kept other vendors' solutions from becoming established. DLLs depend on the application knowing where to find the DLL (or on the user putting it in the right place). Things can break if the user moves or renames a file.

Document-Centered User Interfaces

As operating systems, especially, evolve to support OOP, some users begin to benefit from the sharing of data, even on desktop-oriented systems (see box, "Are People Using These Features?"). For most people, however, the first glimpses of truly document-centered user interfaces are coming courtesy of hand-held "pen" computing systems.

For an example of what document-centered computing might be like for users, let us, too, consider such a system: AT&T's PenPoint operating system and user environment.

A First Look: the PenPoint O/S

To use PenPoint today is to be almost completely unaware of applications and files in the traditional sense. The main interface to PenPoint— called the *Notebook*— looks like a

[3]Different vendors use different names for this capability. Examples include Quark Xpress's Xtensions, Adobe Photoshop's Plug-Ins, Aldus PageMaker's Additions, Deneba Canvas's Tools, Microsoft Word's Commands, and Microsoft Excel's Add-Ins.

document divided into sections and pages. A table of contents and margin tabs helps you get around. Selecting an object brings up whatever software component you need to work on it: spreadsheet, form, drawing software, etc. Select another object and the available tools change.

In such an environment, one never needs explicitly to put applications into the background, switch among running applications, or launch or quit applications. The applications simply seem to appear when needed (and go away when not).

Objects of any type can be embedded anywhere on a page. The user is never made aware of files or a file system. No desktop metaphor is provided—only the "document."

Beneath PenPoint's document-centered user interface lies an object-oriented operating system. Application developers use a C interface t the low-level objects provided by the operating system; users employ a document interface to the high-level objects provided by applications and "app-lets" (see below).

Linking

Among the key features to be found in document-centered interfaces are linking, "drag & drop" capability, activation and location transparency. Linking refers to the ability to place pointers to external data inside documents. (A related term is embedding.) The

continued on next page

ARE PEOPLE USING THESE FEATURES?

OLE 1.0:

Microsoft has used OLE 1.0 in its own applications on both Windows and Macintosh platforms. Microsoft Word for Windows ships with a number of app-lets: for equations, graphing, drawing, "word art," etc. A number of other vendors support OLE on Windows; any of their applications present on your machine will show up as OLE servers in the appropriate dialogs.

OLE 2.0:

Vendors agree that they will want to incorporate OLE 2.0 into their applications, but it is a big job to move from the application-centered to the document-centered approach. Most vendors are planning to incorporate OLE 2.0 functionality into upcoming major releases— that is, they are not rushing out special releases to support OLE. One reason for this go-slow approach is that there have been few target clients in which OLE-2.0-aware objects can be embedded. Among the recently available applications are Microsoft Excel 5.0 and Word 6.0, which are components of Microsoft's new Office bundle. Vendors believe they will not see increasing demand from their users for OLE 2.0 functionality— that is, for document-centeredness— until the well into 1994.

A more basic factor inhibiting the demand for OLE 2.0 is the lack of operating-system support for distributed objects. OLE 2.0 has no provisions for naming objects in a location-independent way, so a link can be broken if a user moves or renames a file. Distributed-object support is promised for Cairo (also known as Windows NT 2.0), slated to be available late in 1995.

Publish & Subscribe:

A good number of applications support P&S. Support for the capability is often a check-box item in Macintosh product reviews. But there is little evidence that users make use of it even though it works just fine. It is thought to be somewhat confusing, especially given that the major Microsoft applications support both P&S and linking/embedding (using DDE) off the same menus.

linked part inside the document can behave in a static or a dynamic fashion—that is, it may change when the source data changes, or it may not.[4]

To be useful across applications, linking mechanisms must be defined and supported by an operating system. Once this support is in place, applications can link to documents (or pieces of documents) that were created by other applications, as well as to data and documents that these applications created and control themselves.

Finally, if linking and embedding capabilities are to be useful across the boundaries of platform and operating system, they must be based on widely adopted industry standards. A single application vendor can provide a mechanism by which its own applications can share data, but unless the mechanism becomes a standard other vendors cannot or will not use it.

"Drag & Drop"

Including part of one document in another should be natural and easy. "Dragging" a part from one document and "dropping" it in another is a more intuitive way of accomplishing this than cutting, copying, and pasting objects via an electronic "clipboard."

"... if linking and embedding capabilities are to be useful across the boundaries of platform and operating system, they must be based on widely adopted industry standards."

Many applications already support "drag & drop" between their own documents. In a document-centered world, however, you should be able to embed anything in anything else without restriction. Imagine dragging an icon from a desktop into a document as a quick way of including an entire file. Or dragging the icon for a directory as shorthand for including multiple files.

Activation

The code that provides access to a linked or embedded object becomes active when the user selects that object within a document. In the Macintosh or Windows worlds today this might mean launching another application. In the document-centered world to come the activation of a part would be both faster and more subtle.

The Microsoft (OLE 2.0) activation model is strictly hierarchical. In order to activate a graph imbedded in a chart in a text document, you must first activate the chart, then the graph. In the Apple (OpenDoc) activation model you could go directly to the graph and activate it.

Location transparency

A user-friendly, document-centered environment would shield the user from being concerned about files as well as applications. Documents, or parts of documents, could be stored in any way (or in any location) convenient to the operating system, provided that the user is given the means to find them as needed. Once assembled, each document would maintain its integrity even if the files that ultimately comprise it were moved or renamed.

All document-management, storage, and workflow solutions available today provide some degree of location transparency, if only an indexed method of finding files. Operating-system vendors such as Taligent and Novell, however, are busily incorporating location-transparency and even workflow features into their OS's.

"App-lets"

An especially exciting hybrid of OOP and the DCUI is the app-let: a mini-application that users can "plug into" other software applications.

[4]Some systems refer to cold, warm, and hot links. A cold link is unchanging. A warm link provides a warning if source data changes, allowing the user to update the linked part. A hot link updates the linked data automatically.

"Everyone who makes a living in the computer industry will be affected by the trends outlined here."

In the future world of document-centered computing, applications will be considerably smaller and simpler than they are today. Instead of buying mega-applications designed to serve all needs of all users, these small, plug-in modules can be installed to perform focused tasks in any application that needs the services they provide.

A word processor, for example, might provide only the basic editing and typographical functions, search-and-replace, and so on. Small add-on app-lets could be used as needed for spell checking, grammar checking, dictionary, thesaurus, drawing, charting, equation editing, etc. These app-lets could be used in the context of any document (and any application), so you would not need to carry around the baggage (in disk space and memory) of unneeded features. And the interface to your charting package, for example, would be the same wherever you used it.

App-lets are small pieces of code and probably won't cost much, so users should not exhibit great price resistance. Vendors do not yet, however, have a good business model for making money from app-let development and distribution. New pricing strategies and new distribution channels will have to evolve.[5]

Implications for the Future

Standards: OMG and CORBA

It would be better if the application could somehow "automagically" find all the resources it needs at runtime, without having to care which directory they are in or if some user moved or renamed them. One solution to this problem is provided by the *object request brokers* standardized by the CORBA specification developed by the Object Management Group (OMG).

Beginning in 1990 the OMG has been encouraging industry players to agree on standards governing the operation of objects in distributed-object-computing environments. Unlike the Open Software Foundation, OMG does not develop technology; instead it coordinates the adoption of standards in important areas of object computing. Its first influential effort was the Common Request Broker Architecture specification, commonly referred to as CORBA. It laid down the ways objects could communicate in a distributed environment, with communication mediated by an object request broker (ORB).

An ORB can help an object find another object it wants to use (for example, the code that knows how to do spell-checking), no matter where it may be located on a network. The ORB can pass messages between cooperating objects. And it can provide information about what interfaces an object supports.

Version 1 of the CORBA specification did not cover all the areas necessary to implement a functioning ORB. As a result vendors could and did implement compliant ORBs that nonetheless could not talk one to the other. The current version (1.2) of the spec addresses some of the shortcomings; the remaining ones are resolved in the 2.0 spec, now in draft form.[6]

Implications for Software Vendors

Everyone who makes a living in the computer industry will be affected by the trends outlined here. For application developers and IS managers the questions will be familiar ones: what companies and technologies will emerge as winners? What is the right point

[5]Network licensing technology provides one model that might be useful here—although it was developed to solve simpler problems in a simpler environment than the distributed object computing paradigm toward which the industry is evolving. Shareware and trial/demo applications might yield useful insights. (Perhaps the industry needs to evolve a pay-per-run scheme similar to cable TV's pay-per-view?)

[6]At recent ObjectWorld conference, several different (and competing) vendors demonstrated ORBs communicating together successfully.

at which to shift development effort to a new platform? How much compatibility will there be for existing applications in the new operating environments?

"For now it looks as if higher-level document functions are safer from being swallowed into the operating system."

Programmers will certainly have to learn new tools and new ways of working. The good news is that, once they have done so, productivity should soar, and cross-platform development should become much easier.

As noted above, the move to document-centered interfaces promises to upset established models for application packaging, pricing, distribution, and support. The coming changes could prove beneficial for vendors who are among the first in their markets to figure out how best to compete in this new realm of app-lets and distributed applications.

Concurrent with the trend to object orientation has been the migration into operating systems of functions once handled by standalone applications. Object technology provides tools that OS vendors can use to execute such migrations in a win-win partnership with independent software vendors (ISVs) who develop for that environment; but it cannot guarantee that the OS vendors will use these tools.

An old example is font technology. One could argue that font handling belongs in the operating system, and that Apple and Microsoft did the world a favor by implementing TrueType. But the action had broad ramifications in the publishing world. It forced Adobe to publish the PostScript Type 1 font specification, which had been held proprietary. And it led to upheaval in the markets for type fonts.

A more recent example, color management, provides a better story of OS-/ISV cooperation. Apple provided a simple form of color management in its operating system, and published the interfaces that an ISV expert in color science could use to replace the simple solution with a more comprehensive one. A cleanly defined object interface makes it easier to engineer this sort of mutually beneficial outcome as the OS vendor makes formerly arcane functions available to all.

Word Processing and Publishing Systems

Much of the expertise that used to be the domain of typesetting, word-processing, and graphics-arts application vendors has for many years been migrating down the food chain to end-users. Expect this trend to accelerate. Before long accessible and high-quality object-based composition services will be available from modern operating systems.

Apple has been developing—but has not yet released—QuickDraw GX, a graphics subsystem that incorporates significant composition expertise, including support for ligatures, expert kerning, even hung punctuation. QuickDraw GX embodies a sophisticated imaging model and supports high-level graphics functions as well. When released later this year, it will enable companies to quickly prototype and develop applications with integrated text, image, drawing, and/or design capabilities.[7]

Other functions less central to the traditional expertise of these industries—spell-checking, grammar-checking, thesaurus, equation, perhaps table handling—will become app-lets once widely adopted document-interface technologies are in use. The current vendors of the mega-applications can win by supplying these app-lets, sooner and better than the small players do.

For now it looks as if higher-level document functions are safer from being swallowed into the operating system. Vendors whose value-add is in page layout, "smart" document systems, and document-structure editors should less to fear.

[7]Apple says all its new software will be cross-platform, supporting at least Windows systems as well as the Macintosh.

Document Management Vendors

Some functions common to document-management systems—such as the ability to locate documents by keyword—are also becoming more widely available as file systems become richer.

Document-management functions are included in OpenDoc, and are rumored to be in Taligent Pink (see box, "Leaders in the Move to OOP"). Novell plans to provide many traditional document- and storage-management functions as extensions to the NetWare OS. Novell has hinted at supplying workflow functions as well.

Implications for IS

As we pointed out earlier, document-centered interfaces and object-oriented environments and tools are converging; both developers and users will be manipulating document objects. The difference is that developers will be concerned with creating object methods and defining the bounds of the environment, while users will pick and choose the object methods that are most suited to the information content they need to create or use. (One can almost use "document" and "object" interchangeably in this context.)

Documents consist of objects—often complex objects created in different applications. Even our Report contains illustrations, charts, and other text objects imported from (or linked to) other applications. The process of assembling all these document objects is far from ideal. It is disruptive and frustrating to interrupt a thought process to open up another application, make a small change and then re-import, re-link, or re-copy and paste. A well-designed document oriented interface can make this process largely transparent.

"... this separation of skills is precisely why a document-centered interface is important."

The benefits to users of document systems are clear:

- ease of learning and use—the user interface is integrated and consistent
- increased productivity—no lost time moving back and forth between multiple files and different applications
- more accurate information in documents—it will be easier to ensure the information is up-to-date.

Is There A Downside?

Some are bound to have concerns about putting a variety of tools in front of a user who may lack the requisite skills to use them. A writer can easily get distracted with illustration or spreadsheet tools even though there is an artist responsible for creating graphics and an accountant in charge of the price list.

While we experience the temptation to get sidetracked by sexy features as much as anyone, we do not think this concern should change anyone's move toward a document-centered interface. This is a *management* problem, not something to be fixed by building limitations into products. It is not easy to sell products based on their *lack* of features. Products only need to provide management tools to facilitate users implementing restrictions where appropriate.

Also, this separation of skills is precisely why a document-centered interface is important. If authors and illustrators are collaborating on a document, you want them to do just that; collaborate on the document, not on different documents that then must be merged together.

How Will This Play Out?

All areas of document system technology (authoring, document management, etc.) are moving in this direction, so it is important to start thinking in these terms when planning

your information management strategies and evaluating products. Pay attention to how suppliers are planning to meet this need, and how they intend to position themselves to stay ahead in the changing competitive environment.

It is too early to tell whether one of the document oriented operating environments being developed will become dominant. It is not even clear at this point exactly what they will look like. The current battle between OpenDoc and OLE is valuable because it will expose both approaches to the level of scrutiny such an important ingredient of document computing needs.

It is our job, and yours, to state, loud and clear, what our needs are. As most of you are painfully aware, most applications and features of operating systems have been sufficient for simple documents (memos, etc.), but woefully under-powered for more complex documents, such as integrated technical information.

Risks and Costs

A document oriented interface is just that— an interface. By itself it will not solve your document management problems. There has to be a level of robustness behind the interface to deal with the kinds of objects that need to go into your documents. For example, a simple linking scheme might allow you to automatically update a spreadsheet in a memo—if both reside in the same folder or directory on your disk. The same approach may break down, however, when you need to incorporate information into documents from a CAD file or from a relational database on a wide area network. There is always the risk of failing to solve a critical business problem by implementing an underpowered solution.

Application suppliers must adjust to a rapidly changing business model and competitive environment at the same time. Not all of them will manage this successfully.

A real risk is that the emerging operating environments will not provide sufficient interoperability. This would make life expensive and complicated for the vast majority of businesses who need, and want, multiple environments. (Of course, what makes most sense for the consumer is not always what drives campaigns for market dominance. Much more is at stake here for vendors.)

Although most "experts" agree that document-centered interfaces are "better" than file–or application–based interfaces, it is always a challenge to convince users to abandon older interfaces with which they are comfortable (or at least familiar). Even changes for the better can be costly if made abruptly without careful planning.

Conclusions and Recommendations

Document-centered interfaces and object-oriented operating environments are coming, and they will be closely linked. Document-centered interfaces and application are already available, as is some middleware-level support.

Most important today is to recognize this trend and prepare to address it in future document management strategies. Any such strategy should include a document re-engineering exercise that is bound to result in a better match with business processes.

It was probably never a good idea to force information management strategies to fit into the arbitrary restrictions of file and applications, although we have had little choice when it came time to implement these strategies. While there may still be some implementation limitations, the time has come to seriously consider the impact of this trend on your long-term information-management architectures and methodology.

"You cannot be sure that the interoperability you need will be supported by the competing environments…"

In addition to incorporating this trend into your planning, you should:

- Review the technology you already have purchased (and paid for). In some cases, existing linking technology can provide substantial benefits, even if only partially implemented—and many companies have simply not paid any attention to it yet.
- Study what other current technology can do. Many new and established products have already begun to add the ability to manage document components as objects, even if distributed over a local- or wide-area network.
- Do not fully commit yourself just yet to a particular object-oriented operating environment. There is no reason to avoid making tentative plans, and to start a dialog with platform suppliers to find out whether their technology meets your needs. But assume that the specifications of these environments will change, and implementation schedules are bound to change, also.
- Make sure your supplier plans to provide an environment that is capable of dealing with the complexity of your documents. Educate suppliers now about the complexity of your future requirements and plans.
- Protect your investment in information and existing infrastructure. You cannot be sure that the interoperability you need will be supported by the competing environments, especially at first. The platform suppliers all seem to be interested in supporting certain critical vendor independent standards like SGML[8] and CORBA — you should forcefully encourage them to do so.

Keith Dawson

[8] The role of SGML in object-oriented environments with document-centric interfaces is surely critical, *e.g.*, as a vendor neutral way to interchange document structures and objects or parts, and protect user investment. We will cover this in a future issue when we look more closely at OpenDoc and OLE.

Leaders in the Move to OOP

The movement toward object-oriented programming on the desktop is being driven largely at the operating system level. Microsoft has the strongest cards, but two consortia—both powered by both Apple and IBM—are strong enough to mount a serious challenge.

The Big Three

Microsoft: Chicago and Cairo

Chicago is the development name for Windows 4.0. This operating system will be the first mainstream Windows implementation that does not rest on DOS; instead, it is built atop a true 32-bit multithreaded, multi-tasking virtual-memory operating system. Chicago is in early testing now and is scheduled to ship some time in 1994.

Architecturally Chicago is a cross between OS/2 2.1 and NT. Like OS/2 it is optimized for Intel-architecture CPUs and is not portable to symmetric multiprocessors or RISC machines. Like NT it is built on a layered, micro-kernel architecture model. (Some layers, in fact, are identical to the ones used in NT.)

Chicago uses OLE 2.0 to implement a Macintosh-like desktop that, like Apple's Finder, combines the functions of the current Windows File Manager and Program Manager. But this desktop will be a full OLE 2.0 application. It will cooperate with all other applications that speak OLE 2.0 and its functions will be fully scriptable from the Visual Basic language.

Cairo is the development name for Windows NT 2.0. Its scheduled availability is late in 1995. By this time Microsoft plans to introduce network awareness into its OLE technology, and the first use of such a feature could appear in the context of Cairo. Whether OLE will also move towards location independence— for example by playing in the CORBA-compliant domain— has not been announced.[9]

OpenDoc Consortium: OpenDoc

The OpenDoc Consortium's founding members include Apple, IBM, Borland, Novell, Xerox, Sun, Oracle, and WordPerfect. Membership is open to all, and OpenDoc source code will be freely available to all.

OpenDoc represents technology mostly from Apple, with contributions by IBM and others, that previously went under Apple code names Amber, Exemplar, and Jedi.

One industry observer described OpenDoc as an "OLE 2.0 on steroids." OpenDoc provides a way to encapsulate OLE 2.0 objects, so that it can interoperate in a world containing the competing standard. Like OLE 2.0, OpenDoc features cross-platform availability and a scripting language (OpenDoc's roster of platforms is longer). OpenDoc goes beyond OLE 2.0 by providing document management, a consistent user interface, and certification of vendor implementations by an independent laboratory (the Component Integration Laboratory). OpenDoc also allows document parts to be any desired shape, while OLE 2.0 mandates rectangular parts. Most importantly, OpenDoc is network-aware and will support the CORBA spec for object communication.

Release of OpenDoc 1.0 is scheduled for mid-1994. "Release" in this context does not mean commercial availability; it means instead that final code is available to vendors to use in commercial products. Off-the-shelf products based on OpenDoc will probably appear by the end of 1994 from IBM (on AIX and OS/2 platforms) and from Apple.

Novell and WordPerfect are working on an OpenDoc implementation for the Intel/Windows platform.

[9]Microsoft has recently become considerably more active in the OMG, and may intend to push for standardization of OLE through OMG's Common Facilities Task Force.

Taligent: Pink

Taligent was started in 1992 with funding, staffing, and technology from Apple and IBM. Recently Hewlett Packard invested in a minority stake in the venture. Taligent's work on the operating system code-named Pink represents by far the most ambitious object-oriented operating system of any of those discussed here, and also the one that is farthest out in time. The company has rethought everything about the operating system and user environment in object terms.

In order to meet the competitive challenge of Microsoft's Cairo, Taligent plans to release some of its object technology in 1994 to its sponsors and to early-adopter developers. Delivery of the full operating environment to Apple, IBM, and HP may not happen until 1995. Taligent will not itself ship any commercial products; it is up to the sponsoring companies to incorporate the Pink technology into their product lines. IBM's announced plans for doing so are considerably clearer than Apple's or HP's. According to those plans IBM will be shipping products based on Pink in 1995.

Industry analysts who have seen the Pink technology believe that Taligent's from-scratch approach will result in a robust operating environment that could be in use well into the next century.

The Open-Systems Vendors

Digital: Object Broker

Digital delivered an Object Request Broker in 1991 before the OMG finalized the CORBA 1.0 spec. As a result Object Broker is not quite CORBA-compliant.

IBM: SOM / DSOM

IBM's object broker is CORBA-compliant and is based on IBM's System Object Model or SOM. This model was extended with network awareness; the result, Distributed SOM or DSOM, began shipping to OS/2 and AIX system developers in 4Q'93.

Hewlett Packard: ORB Plus

This CORBA-compliant ORB is scheduled to ship to system developers in mid-1994. It will include an Interface Repository, which will perforce be non-standard because OMG has not settled on a standard yet in this area.

Sun Microsystems: Project DOE

DOE stands for Distributed Objects Everywhere. This technology, being developed by the SunSoft subsidiary, is scheduled to ship to system developers by the end of 1994. The schedule has been called into question with the planned integration of NextStep technology.

Others

AT&T: PenPoint

PenPoint was developed by Go Corp. in the late 1980's for use on pen-based computers. It was available to developers in 1990.

It was originally thought that PenPoint would run on hardware from a number of vendors. But hardware development of pen computers lagged, and the market for the technology has not developed at the rate first anticipated. In 1993 Go Corp. was acquired by AT&T. Future deployment of PenPoint will therefore be limited to the hardware that AT&T produces (the Hobbit).

It is worth noting that although PenPoint sports a thoroughly document-centered interface, it is not necessarily intuitive to use. The user must internalize a number of gestures—expressive motions with the pen—and learn which to use in what contexts.

NeXT Inc.: NextStep

The NextStep environment is the most complete example today of an object-oriented operating system and development platform. But its interface is the traditional desktop-based, mouse-and-icon metaphor that was commercialized by Apple and pioneered by Xerox PARC (with roots going back to the SRI Augmentation Research Center of the 1960's).

NextStep originally ran on NeXT's own hardware, but in 1993 the company got out of the hardware business and announced a NextStep port to the Intel 80x86 environment. The Sun/NeXT deal assures that the NextStep environment will run under Sun's Solaris.

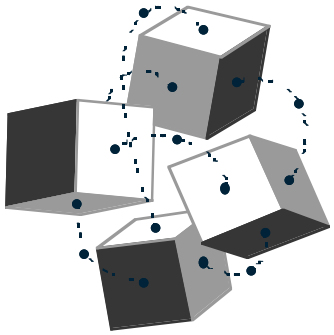
HyperDesk: ORBIX

HyperDesk was formed from technology developed by Data General. DG abandoned its large, ambitious program to develop an open platform for cooperating object development and sold the technology to Ascii Corp., which formed HyperDesk in 1991. The HyperDesk technology was central to the standards submission that resulted in the CORBA 1.0 spec.

Novell / HyperDesk

Novell and HyperDesk are working jointly on an ORB that will run in the Netware 3.x and 4.x environments as well as Unixware.

DOCU MATION '94



UPDATE

Because of the synergy between the Documation Conference and the topics covered in this report we will provide regular updates on the conference program and exposition.

Program Update

A program update was mailed this month. The good news is that there have been very few changes, and those have been additions to the program. We have already received dozens of proposals for speaking at next years conference. While we won't be making any decisions about next years speakers for a few more months, it wouldn't hurt to let us know if you have an interesting idea.

Exposition Update

So far close to 70 companies representing the leading suppliers of document management and document computing products and services have reserved booth space. We are expecting a few last minute entries as well. Even if you can't attend the conference you won't want to miss the exposition.

New Product Announcements

There are some exciting new products being announced at Documation. We can't tell you about them in advance, but you won't want to miss them. Many members of the press will be on hand, however, just in case you miss something while attending sessions.

User Group Meetings

The SGML Open Consortium will be holding a full day of meetings on Friday in addition to the activities they have planned during the conference. In addition EPSIG, The Southern California CALS User Group, the International SGML Users Group, and others are planning meetings for Friday.

Travel

Because of the highway damage due to last months earthquake, we recommend that you leave a little extra time for getting around. The roads between the airport and Century City were unaffected, however they have to handle an extra load for awhile. If you take a taxi from the airport, you can rent cars at the hotel.

We hope to see you all there!

CALENDAR OF EVENTS

Below is a selection of key events covering open information and document system issues. There are many other conferences and shows covering related topics. We will attempt to keep this list to those events that focus on areas most directly related to the areas covered in our report.

Documation '94. February 21-25, 1994, Los Angeles CA. The annual international event for document management applications and document computing. Call (703) 519-8160 or (617) 643-8855, Fax (703) 548-2867 or (617) 648-0678.

Intermedia. March 1-3, 1994. San Jose, CA. Multimedia and CD-ROM. Conference and Exhibition. Call (203) 352-8240, Fax (214) 245-8700.

Seybold Seminars '94. March 22-25, 1994. Boston, MA. The annual gathering of the computer publishing elite. Conference and Exhibition. Call (310) 457-8500, Fax (310) 457-8510.

OnLine Publishing '94. April 10-13, New York, NY. GCA conference on online publishing issues. Call (703) 519-8160, Fax (703) 548-2867.

AIIM. April 18-21, 1994, New York, NY. AIIM's annual show and conference focusing on imaging and storage and retrieval. Call (301) 587-8202.

Pen & Portable Computing. May 2-5, 1994, Boston, MA. Well, Documents need to be portable don't they? Sponsored by Boston University. Call (800) 733-3593, ext. 255, FAX (508) 649-2162.

EDD '94. May 10-12, 1994, Somerset, NJ. Bellcore's forum for discussion of issues relating to the exchange of technical information in electronic form. Call (201)829-4135, Fax (201)829-5883.

SGML Europe. May 15-19, 1994, Montreux, Switzerland. The European counterpart to the SGML '93 conference in the U.S. Call (703) 519-8160, Fax (703) 548-2867.

AIA Automated Technical Data Symposium & Exhibition. May 16-18, St. Louis, MO. The 9th biennial gathering of the Aerospace Industries Association group focused on managing technical data. This year's theme: Interactive Electronic Environments. Call (202) 371-8435, Fax (202) 371-8470.

Seybold Paris. June 8-10, 1994. Paris, France. Seybold's main European event. Conference and Exhibition. Call +44 (0)323 410561 , Fax +44 (0)323 410279.

Infobase '94. June 28-30, 1994. Salt Lake City, UT. Folio User Conference. Call (801) 344-3671, or (801) 344 3672.

International Conference on HyTime. July 24-27, 1994, Vancouver, BC Canada. New conference exploring applications of the ISO standard. Call (703) 519-8160, Fax (703) 548-2867.

CALS Europe '94. September 14-16, 1994, Paris, France. The annual pan-European conference on CALS technology and applications. Call (703) 578-0301 or +49 30 882 6656, Fax (703) 578-3386 or +49 30 883 8811.

Topics Covered In Previous Issues

Vol. 1, No. 1.

What The Report Will Cover & Why — An Introduction To "Open Document Systems", And A Description Of The Report's Objectives.

Imaging, Document & Information Management Systems — What's The Difference, And How Do You Know What You Need?

Vol. 1, No. 2.

SGML Open — Why SGML And Why A Consortium?

Document Query Languages — Why Is It So Hard To Ask A Simple Question?

Vol. 1, No. 3.

Document Management & Databases — What's The Relationship?

Vol. 1, No. 4.

Electronic Delivery — What Are The Implementation Issues For Corporate Applications?

Vol. 1, No. 5.

Multimedia Rights & Wrongs — What IS Managers Should Know About Copyrights In The Age Of Multimedia

Topics To Be Covered In Future Issues

The subjects listed below are some of the areas we will be covering, in no particular order. If you have an opinion about which topics you would like to see added or covered sooner rather than later, let us know.

Office Workflow Systems — Can They Handle Strategic Information, Or Are They For Casual Or *Ad Hoc* Use Only?

Documents As Interfaces — Is This An Option For Today? What Will The Future Bring?

SGML & Presentation Interchange — What Standards Are Available Or Appropriate? (DSSSL, OS/FOSI, HyTime, ODA, etc.)

Authoring Systems — Do You Need Different Kinds For Different Media?

"Middleware" — What Is This Layer Of Software In Between Operating Systems And Applications? Is It The New Proprietary Trap? What Does It Mean To Your Decisions About Document Systems?

ISO 9000 — What Kind Of Document Management System Do You Need To Meet This Quality System Standard?

The Airframe And Airline Industry's Strategy For Sharing Product Information — What Can You Learn From It?

New Drug Applications — What Document System Strategies Make Sense For The Pharmaceutical Industry?

Object & Relational Databases — Which Approach Is More Suited To Your Document Systems Needs?

Compound Document Architectures — Why Do We Need Them? Who Will Define Them? Will They Do What We Expect?

SGML Versus ODA — How Do They Differ? Is There A Reason To Have Both? What Can They Do? Which Approach Is Right For The Future?

Order Form

o Please start my subscription to: The Gilbane Report on Open Information & Document Systems (6 issues). Back issues available for $45. each.

U.S.A.: $225 Canada: $232 Foreign: $242

Additional copies and site licenses are available at reduced rates. Call for information.

Please send me additional information on:
o Consulting Services o Special Reports
o On-site CALS Strategic Planning Seminar
o Document Managment & Electronic Delivery Seminars

o My check for $ _____ is enclosed o Please bill me

o Please charge my credit card o MasterCard o Visa o American Express

Name as it appears on card _____ Number _____

Signature _____ Expiration date_____

Checks from Canada and elsewhere outside the U.S. should be made payable in U.S. dollars. Funds may be transferred directly to our bank: Baybank Boston NA, 175 Federal Street, Boston MA 02110, S.W. code BAYBUS33, into the account of Publishing Technology Management, Inc., number 1444-89-63. Please be sure to identify the name of the subscriber and the nature of the order if funds are transferred bank-to-bank.

Name _____ Title_____

Company _____ Department_____

Address_____

City _____ State _____ Zip_____ Country _____

Telephone_____ Fax _____ Email _____

Mail or fax this form to: Publishing Technology Management, Inc., 46 Lewis Avenue, Arlington MA 02174-3206
Fax: (617) 648-0678 • To order by phone call: (617) 643-8855

How To Find Out More About Companies Mentioned In This Issue

Apple Computer
20525 Mariani Avenue
Cupertino, CA 95014
(408) 996-1010

Aldus Corporation
411 First Avenue South
Seattle, WA 98104
(206) 622-5500

ArborText
1000 Victors Way #400
Ann Arbor, MI 48108
(313) 996 3566

Component Integration Laboratories
688 Fourth Avenue
San Francisco, CA 94118
(415) 750-8352

Datalogics, Inc.
441 West Huron St.
Chicago, IL 60610
(312) 266-3125

Frame Technology
1010 Rincon Circle
San Jose, CA 95131
(408) 433-3311

IBM Corporation
Old Orchard Road
Armonk, N.Y. 10504
(914) 765-1900

Interleaf
Prospect Place, 9 Hillside Ave.
Waltham, MA 02154
(617) 290-0710

Microsoft Corporation
One Microsoft Way
Redmond, WA
(206) 882-8080

Oberon Software
1 Cambridge Center
Cambridge, MA 02142
(617) 494-0990

Object Management Group
429 Old Connecticut Path
Framingham, MA 01701
(508) 820-4300

Oracle Corporation
500 Oracle Parkway
Redwood Shores, CA 94065
(415) 506-7000

SoftQuad, Inc.
56 Aberfoyle Crescent, Ste 810
Toronto, Ontario M8X 2W4
Canada
(416) 239-4801

SunSoft
2550 Garcia Avenue
Mountain View, CA 94043
(415) 960-1300

Xerox
XSoft Division
10200 Willow Creek Rd.
San Diego, CA
(619) 695-7700

Xyvision, Inc.
101 Edgewater Dr.
Wakefield, MA 01880
(617) 254-4100

WordPerfect
155 North Technology Way
Orem, UT 84057
(801) 228-9930

The Gilbane Report on Open Information & Document Systems is published 6 times a year.

The Gilbane Report is an independent publication offering objective analysis of technology and business issues. The report does not provide advertising, product reviews, testing or vendor recommendations. We do discuss particular pieces of product technology that are appropriate to the topic under analysis, and welcome product information and input from vendors.

Letters to the editor are encouraged and will be answered. Mail to Editor, The Gilbane Report, Publishing Technology Management, Inc., 46 Lewis Avenue, Arlington, MA 02174-3206, or fgilbane@world.std.com or APPLELINK:PTM

ISSN 1067-8719