Patricia Seybold Group

Editor-in-Chief
Michael A. Goulde

OPEN INFORMATION SYSTEMS

Guide to Unix and Other Open Systems

Vol. 7, No. 10 • ISSN: 1058-4161 • October 1992

Galaxy from Visix

Application Portability Breakthrough?

By Stanley Dohlberg

IN BRIEF: Application developers face the daunting challenge of building applications for deployment in a world of heterogeneous architectures, operating systems, networks, and user interfaces. Playing it safe by oversimplifying the options can leave an ISV choking on the dust of the opposition, or it can position an end-user company at a competitive disadvantage. Galaxy, from Visix Software, Incorporated, could simplify the developer's dilemma and move choices about how to deploy applications to where the decision belongs—with the user. *Report begins on page 3.*

Flexibility

The Hallmark of Open Information Systems

IN OUR NEVER-ENDING pursuit of interoperability, portability, and manageability, we often overlook the less obvious but perhaps more valuable benefit of open systems: flexibility. Flexibility derives from modular architectures that use standard interface definitions to describe how the various pieces of a system work together. Flexibility means that components, both hardware and software, can be quickly and easily mixed, matched, and configured by using products that adhere to standards. Flexibility also means that applications can be developed rapidly and deployed across a variety of architectures.

The importance of flexibility in today's business environment was recently highlighted by Thomas Stewart in his *Fortune* article, "Brace for Japan's Hot New Strategy" (*Fortune Magazine*, September 21, 1992, p. 63). Stewart points out that, while many U.S. companies have finally caught up to Japanese competitors on quality, they are now missing the boat when it comes to understanding the importance of flexibility. He says the value of flexibility is that a company which quickly understands and reacts to changes in the market, is able to use the same production facilities to build a variety of products, can reduce the cost and latency of switching a line from one product to another, and can maintain healthy margins with short production runs will have a distinct competitive advantage.

For example, Toshiba, responding to market pressures, now generally builds laptops in lots of 20 and can make money on lots as small as 10. It has a network that links office, engineering, and factory operations and provides "just in time" information as well as just in time manufacturing. Since laptop computer product life cycles are measured in months, it is critically important to get faster feedback from the marketplace so that sales and distribution can be as flexible as manufacturing. Flexibility can also save money by matching production to demand.

The article also cites Kao Corporation, Japan's largest soap and cosmetics company, which is known for its flexible distribution system driven by an information system that allows Kao to deliver goods within 24 hours to any of 280,000 outlets, even though the average order is just seven items. The mission of Kao's information system is to maximize the entire company's ability to quickly respond to demand. It does this by collecting a massive amount of data from the marketplace and distributing it to employees in all areas of the company. A single information system links everything: sales, shipping, manufacturing, production, purchasing, accounting, R&D, marketing, hundreds of cash registers, and thousands of salesmen's hand-held computers. Brand managers see daily sales, stock, and production figures and make adjustments to a competitor's moves within a day. The system virtually eliminates the lag between an event in the market and the arrival of the news at Kao.

Flexibility is the essence of a customer-driven approach, requiring that all areas of a company work in concert at being responsive to a specific customer's requirements, as opposed to developing products for a statistical profile of broad market segments. Individual customers become a market segment unto themselves, and the companies that can rapidly deliver on this almost unlimited variety of requirements will be the ones which are able to be the most flexible. They will also be the most successful.

The flexibility that companies like Kao, Toshiba, General Electric, and Motorola are striving to achieve can only be accomplished with an information infrastructure that is easy to adapt, change, and extend. Open systems are technically appealing because of their promised simplicity. But, from a business perspective, the appeal of open systems is that they are ideally suited for supporting the new economics, which says that the organization should strive not for economy of scale, but for economy of scope.

OPEN
INFORMATION
SYSTEMS

Editor-in-Chief
Michael A. Goulde

MCI:
MGoulde
Internet:
mgoulde@mcimail.com

Publisher
PATRICIA B. SEYBOLD
Analysts and Editors
JUDITH R. DAVIS
DAVID S. MARSHAK
RONNI T. MARSHAK
JOHN R. RYMER
ANDREW D. WOLFE, JR.
News Editor
DAVID S. MARSHAK
Art Director
LAURINDA P. O'CONNOR
Sales Director
PHYLLIS GUILIANO
Circulation Manager
DEBORAH A. HAY
Customer Service Manager
DONALD K. BAILLARGEON

Patricia Seybold Group
148 State Street, 7th Floor,
Boston, Massachusetts 02109
Telephone: (617) 742-5200 or
(800) 826-2424
Fax: (617) 742-1028
MCI: PSOCG
Internet: psocg@mcimail.com
TELEX: 6503122583

Open Information Systems (ISSN 0890-4685) is published monthly for $495 (US), $507 (Canada), and $519 (Foreign) per year by Patricia Seybold Group, 148 State Street, 7th Floor, Boston, MA 02109. Second-class postage permit at Boston, MA and additional mailing offices.

POSTMASTER: Send address changes to *Open Information Systems*, 148 State Street, 7th Floor, Boston, MA 02109.

FEATURED REPORT: BY STANLEY DOHLBERG

Galaxy from Visix

Application Portability Breakthrough?

The Holy Grail of Portability

Application portability—the Holy Grail of the open systems movement—goes to the heart of the open systems proposition. As achievable as portability is for single user applications, it is elusive for complex applications. As platform price/performance continues to improve at a steady rate, users want to move applications to the "platform du jour." But today's distributed, graphical applications are too closely tied to the specifics of operating systems, file systems, distributed computing mechanisms, and graphical user interfaces to allow easy porting. Besides, advanced distributed computing "enableware" lies at the heart of many system vendors' value-added to otherwise commodity product lines, working against portability and raising the specter of proprietary lock-in.

Developers Are Weighed Down by Platform Proliferation

Even though most ISVs and in-house application developers have become expert at isolating platform-dependent code, many are choking on the proliferation of new system platforms. Microsoft alone has discussed a major new system software plan in each of the past three years. With Windows NT still over six months from first release, Microsoft is already publicly discussing its successor, an object-oriented operating system code named "Cairo." IBM and Apple are working on Pink for delivery three years from now. And, of course, versions of Unix continue to proliferate and compete with incompatible application programming interfaces (APIs).

Application Developers Want to Build Applications, Not Portability Tools

In order to survive in this difficult environment, many application developers end up investing valuable resources in building and maintaining their own complex and expensive porting tool environments. Understandably, they would prefer to focus on building best-of-breed applications in their market segments.

Visix Software Proposes Galaxy as the Answer

Specialization as the Industry Matures

This difficult environment invites a solution. Visix Software, Incorporated, of Reston, Virginia, has accepted the invitation with a platform for the development of portable, distributed applications called Galaxy. Visix is best known for its Looking Glass desktop managers for the Unix operating system. But, while Visix was publicly slugging it out for market share in that commodity market, it was quietly putting significant development resources into solving the complex application development and portability problem by building Galaxy.

Visix calls Galaxy an "application environment" because it is designed as a complete development and run-time environment that insulates developers from the uniqueness of operating systems, networks, user interfaces and file systems. Visix is convinced that portability across diverse operating environments, even for complex, graphical, distributed applications, *can* be accomplished—if the application development environment is comprehensive and extensible and if the API is functionally rich and clean. Interestingly, this is not a case of a company deciding after the fact to market its tools. The Galaxy environment was the original conception, and the Looking Glass product line was the first set of products built on early versions of Galaxy. Some of the attributes of Looking Glass, such as run-time-selectable look-and-feel, high performance, and rapid cross-platform porting, demonstrate benefits derived from having been built even on early versions of Galaxy.

An API Cast in the Multiplatform Crucible

Development of Galaxy began six years ago, three years before the 1989 spin-off of Visix from American Management Systems. For the last three years, the Visix engineering staff has been broadening and deepening the Galaxy API. The Galaxy architects garnered valuable practical experience from the Visix experience as an ISV whose first products were merchandised through a gaggle of system vendors and across machine architectures and operating systems. The Visix development team experienced firsthand some of the pains of porting and maintaining multiple versions in multiple national languages across multiple platforms, and they used this experience to shape the depth and breadth of the Galaxy API.

Mac Admirers in Unix Clothing

The management at Visix has been inspired by the success of the Macintosh in many ways. Nearly two-thirds of the development staff consider themselves former or current Macintosh application developers, and a peek into virtually every developer's office reveals a current Macintosh system right next to a PC-clone and any one of several Unix workstations. Visix management believes that the Macintosh's high level of success is substantially due to the completeness and functionality of the Mac Developer's Toolbox. The reasoning is that the Mac Toolbox offers application developers a very good development environment with well-defined, well-supported methods for application presentation and behavior. The fact that most developers could do what they needed within the supported methods naturally led to cross-application and cross-vendor "drivability" compliance. Developers could choose to work outside the preferred methods of the Mac Toolbox, but those who did were penalized twice. The first penalty was increased cost and longer time-to-market due to having to hand-code at low levels. The second penalty was even costlier. As critical mass developed around adherence to convention on the Macintosh desktop, deviant applications were often rejected by the consumer. The lessons from this parable have guided the philosophy behind Galaxy in its own lengthy path to the market.

The Galaxy Mission

The Galaxy mission has three key elements:

- Build and deliver a high performance, cross-platform operating environment that insulates the application developer from the excessive, and growing, number of options in the run-time environment. The first part of the mission is accomplished by delivering a comprehensive, platform-independent API.

- Institutionalize distributed computing by designing the Galaxy API to make it possible to build distributed applications or nondistributed applications with the same basic techniques. The second part of the mission is addressed through a set of object-oriented class managers and standard components that comprise the API and that service the application at run-time.

- Deliver to developers a set of integrated tools and functionality that enables building complex applications with little or no costly handwork that might compromise application portability. The third part of the mission is accomplished through the breadth and depth of the API and by offering a comprehensive set of tools, along with the ability to incorporate additional tools with the documented "software backplane" interface.

The overriding principle of this effort is that Galaxy should enable developers working on complex distributed applications to experience the equivalent of working with the Mac Toolbox on single-user applications. Galaxy offers broad areas of functionality and is very complete, particularly in the graphics area. Some of the most sophisticated graphics features

The Galaxy Mission

are based on the needs of CAD developers looking for a Windows product that could be used to expand the market. Visix acknowledges that most developers will not use more than 40-50 percent of the total functionality and that the industry-wide acceptance would hinge on the cross-market capacity of the API.

Visix has put the bulk of its resources behind the Galaxy mission—well over 75 percent of its engineering staff is dedicated to building the Galaxy environment and working closely with the early adopters.

Beyond GUI Builders

A critical part of the portability problem has been associated with graphical user interfaces, hence the market interest in a range of hosting and porting tools that present applications to users in their choice of look-and-feel. With Galaxy, Visix took the development of a portable user interface as the starting point for addressing the overall application portability problem. Early Galaxy customers evaluated portable GUI builders before selecting Galaxy, and each found that the depth and breadth of the Galaxy API far exceeded the goals, no less the deliverables, of other potential portability enablers, particularly for platform dependencies related to client/server application development. In Galaxy, the user interface development and run-time functionality is estimated to be less than 20 percent of the total functionality contained in the 1.5 million lines of Galaxy code, yet the user interface builder and run-time capability are on par with market leaders.

Galaxy Support for Distributed Computing

Galaxy Applications Are Architecturally Distributed

In designing Galaxy, Visix took the approach of tackling a very large problem by defining a set of programming interfaces for the complete set of functions needed to solve the problem. Visix then selectively went beyond simply providing programming interfaces to providing full-depth functionality under those interfaces. In effect, this approach takes responsibility for characterizing the entire application environment, even though it does not provide the entire environment.

The Galaxy environment offers inherent support for distributed applications through the client/server architecture of the API and the run-time environment. The Galaxy Distributed Application Services (DAS) server is designed to be able to interface to the array of established and emerging communications protocols, messaging systems, and distributed object management systems, along with offering its own interapplication communication and directory services. To deepen the support for distributed computing, Visix plans a variety of means for supporting interoperation with a broad set of interapplication communication mechanisms, such as Dynamic Data Exchange (DDE), Object Linking and Embedding (OLE), ToolTalk, and Distributed Computing Environment (DCE).

DAS Directory System

The DAS server enables Galaxy applications to communicate with each other in peer-to-peer or client/server relationships, as well as with non-Galaxy applications. (See Illustration 1.) The DAS server uses a registration-oriented approach to match the attributes of service requests with the attributes of registered services. Applications register their available services and their operations with the DAS server. Services are then requested by operation, not by service name. When an operation is requested, the DAS server locates and directly connects the application to the most appropriate service based on the requested operations. Galaxy defines a service location and naming scheme that is unique to the Galaxy environment. The interfaces to this scheme are fully documented and can be replaced with the X.500 directory services of DCE when X.500 achieves a position as a standard in the marketplace or by the Object Management Group's (OMG) Object Request Broker. In the current state of the market, no network-wide registration mechanism is dominant, so the Galaxy DAS provides a registration facility. In fact, Visix would prefer to rely on a standard such as the OMG Common Object Request Broker Architecture (CORBA) model when it is accepted in the marketplace.

Galaxy Support for Distributed Computing

Galaxy Component Overview

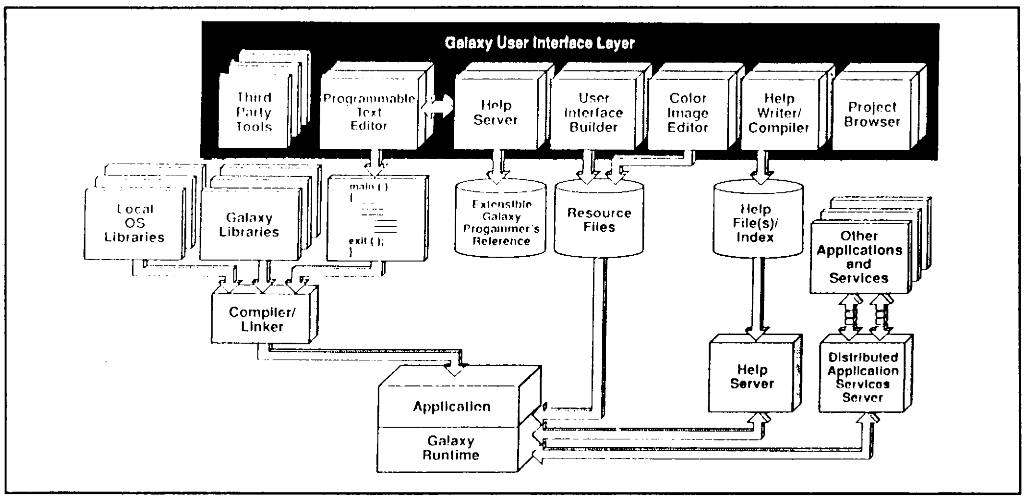


Illustration 1. The Galaxy environment defines a broad set of functional components addressing the application development process, the distribution of application function between clients and servers as well as peer-to-peer, and the integration of diverse applications over the network.

Galaxy Interoperation with External Distributed Computing Mechanisms

Interoperation with applications built on distribution mechanisms—such as OSF's DCE, SunSoft's ToolTalk or HP's SoftBench messaging systems, OMG's CORBA, and Microsoft's OLE model—will require the addition of drivers or additional protocols to the set of protocols already supported in the lowest layer of the DAS. DAS, as currently defined, operates as a general mechanism that will locate services, establish connections, get attributes, and request that the services perform needed operations. The enhancements to the DAS might also be made at higher layers, depending on the model of the external service, possibly including special purpose Galaxy APIs to front-end the external services with Galaxy code that supports the registration of the operations offered by applications running externally to the Galaxy environment. Interestingly, Visix management sees DAS interoperation with OLE as the first priority of the many possibilities.

CORBA and DCE Compliance and Interoperability

Through DAS, Galaxy could interface to DCE-based applications by supporting access to Galaxy-based applications from DCE-based applications and by supporting access to DCE-based server applications through support for the DCE Distributed file system in the file system abstraction provided by Galaxy's Operating System interface. Visix has been active with OMG since the early days and is well-informed about the technical breakthroughs being achieved there. ORBs can exist as external applications to the DAS services. Internally, the Galaxy environment has been built in compliance with relevant OMG specifications.

The Galaxy API

The API and libraries that implement the API are at the heart of Galaxy. The functionality of the API falls into three major groupings:

- Network Interface, which hides network and messaging protocols, directory services, distribution mechanisms, and global help
- Operating System interface, which masks variety in file systems, memory management, sound, timers, and event management

- Window System interface, which provides a single view of the multiplicity of ways that color, cursors, drawing, fonts, imaging, printing, geometry, and windows are managed

File System Abstraction Increases Programmer Flexibility

The Galaxy File System Manager, like all of the Galaxy Function Managers, provides the developer with an idealized programming interface that supports cross-platform deployment. The Galaxy File System abstraction actually increases the functionality of some of the platforms by, for example, removing the restriction that Windows and Unix place on simultaneous open files. The Galaxy File System Manager has no limit on open files, and it provides expanded functionality to the application through a file-swapping mechanism that operates almost like virtual memory. The Galaxy File System Manager allows the developer to write an application with unlimited open files, and Galaxy then keeps track of and manages the actual opening and closing of files required to operate within the limits of the system platform, without involving the application.

Standard Components: High-Level Objects Pre-Built with Galaxy API

The Galaxy API includes interfaces to a set of standard components, which consists of a set of Standard Choosers and Standard Items. These standard components are essentially "superwidgets" that are so commonly used that Visix took the trouble to provide them to increase developer productivity. As with all the user interface components, the standard components are actually object classes, not widgets, so they can be subclassed and modified to suit particular needs.

The Standard Choosers, which include a Color Chooser, Command Chooser (see Illustration 2), File Chooser, and Font Chooser, offer the developer a consistent Galaxy-compliant means of presenting these common functions to the user. The Standard Items include the Ruler Item and the Palette Item, among others. The detailed elements of the Standard Components adjust to the active look-and-feel of the client, ensuring that buttons, list boxes, and other visual elements are always compliant with the user's environment.

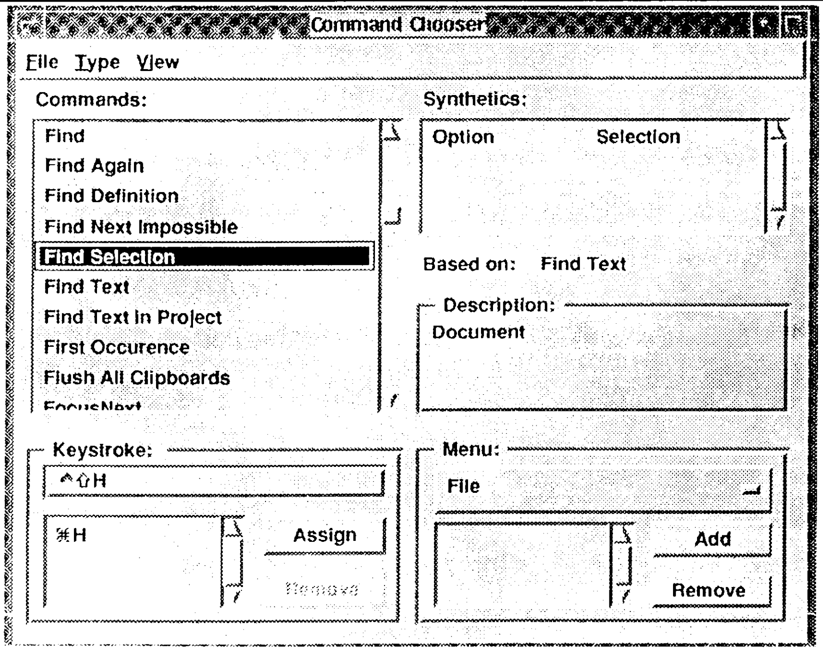
Selected Function Class Managers

The set of Galaxy Class Managers is exceptionally broad (see Illustration 3). Each Function Class Manager is itself a true object-oriented class and incorporates features that can dramatically improve developer productivity and application flexibility. Two examples are:

- The Spring Manager, which provides a high-level visual interface for designing the layout of windows using the concept of *springs and struts*. With the Spring Manager, the developer can completely avoid the difficulty of programming how the user interface objects change when windows are resized or reimplemented under a different look-and-feel. The Spring Manager also provides a feature called "natural sizing," which can automatically resize an entire window based on the minimum size required to contain its contents. It can also resize individual objects based on the text string labels, which has particular benefit in managing internationalization issues related to local language conversion.

- The Command Manager, which supports the separation of application code from user interface code by providing a set of objects that implement the actions of the application in conjunction with the Galaxy Class Manager attribute binding mechanism. Every action-oriented user interface command generates a command to the Command Manager, which sends the command to the application for action. The Class Manager attribute binding mechanism handles the translation from command object references to specific application methods that are required to implement the actions of the application. This powerful Class Manager, in conjunction with the Command Chooser standard component, greatly simplifies the development of the user interface to the application command set and supports customization of the interface by users. The Command Manager also allows the application to be driven externally through DAS, since messages to the Command Manager can come from any source, not just interface selections.

The Galaxy Command Chooser



Commands:
Find
Find Again
Find Definition
Find Next Impossible
Find Selection
Find Text
Find Text in Project
First Occurence
Flush All Clipboards
FocusNext

Synthetics:
Option Selection

Based on: Find Text

Description:
Document

Keystroke: ^⇧H
⌘H

Menu: File

Illustration 2. The Galaxy Command Chooser offers "superwidget" functionality to the application developer. The developer loads the commands into the Command Manaager, brings up the Command Chooser, and interactively defines the keystroke shortcuts for the commands. The Command Chooser is accessible to the user to support modification of the command interface.

The Galaxy Sound Model

As part of Galaxy's goal to far exceed today's mainstream platforms, the company offers industrial-strength support for sound input and output through a sound abstraction layer in the API. Galaxy supports a broad range of sampling rates and encoding schemes, and it even monitors output to ensure that it meets the output device specifications. The Galaxy audio driver operates multithreaded within Galaxy.

The Galaxy Imaging and Color Model

The Galaxy color model is 48 bits deep, to ensure that the limiting factor is in all cases the device or the application program, not Galaxy. When an application asks for a color rendering of an image, the Galaxy color and imaging models are smart enough to select the closest match from a 24-bit, 16 million color chart, but they also support dithering, either in color or in gray scale on less capable or noncolor displays. Galaxy currently supports several color models and will be adding support for Pantone colors in 1992.

The Galaxy drawing model is based on Display PostScript technology. Display PostScript offers the highest level of correspondence possible between display images and printed images without involving the developer in the details of image resolution. The Display Postscript imaging model is implemented in Galaxy through API calls, not PostScript calls. The Galaxy Display PostScript capability was built from the ground-up based on the published specifications for PostScript, and it has been demonstrated to deliver WYSIWYG rendering that is among the best on the market. The Galaxy rendering capability has the intelligence to work with fonts on the local machine to find a match between the requested font and the fonts available, based either on a default priority scheme or one that has been dictated by the developer. Galaxy is designed to leverage the functionality on the host machine. Thus, if Adobe Type Manager is installed, it will use that to draw and scale fonts; if not, it will bit-map them.

Galaxy Managers' Functional Classification

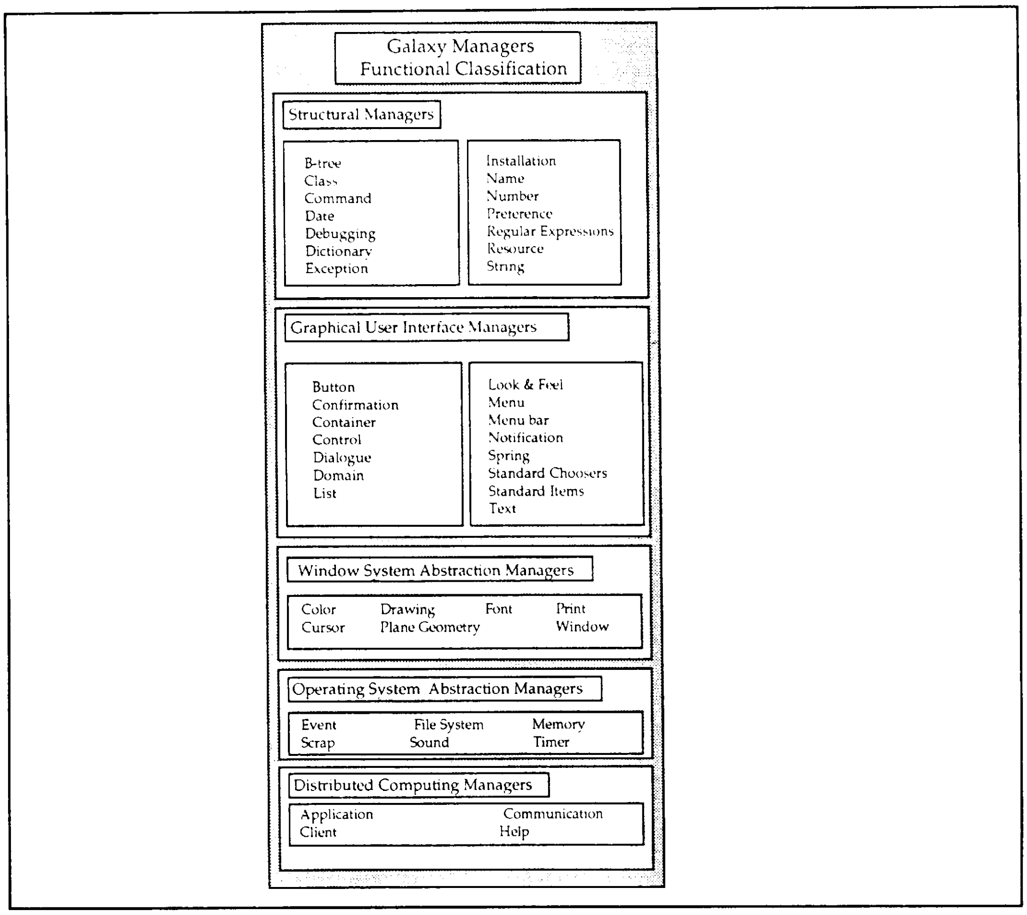


Illustration 3. This functional manager grouping illustrates how the class managers relate to the three major system platform abstractions (operating system, network, and windowing), and the granularity of support for building look-and-feel-independent applications with Galaxy.

Internationalization: Another Dimension of Portability

Galaxy has been designed to facilitate all aspects of portability, including deployment of the application in local languages. Internationalization capability is built into Galaxy through the internal wide-character representation based on the ISO 10646 standard character format. Galaxy functions internally handle the translation from the internal character set to the characters appropriate to the application—for example, 8-bit for ASCII, 16-bit for Unicode, and 32-bit for Kanji—without involving the application.

The Galaxy API: Clean Inside and Out

Visix actually built all of the Galaxy development tools by working with the same documented Galaxy API that any Galaxy developer would have access to. This commitment was made and adhered to in order to ensure that developers could extend any functions and tools that need to be customized to the purpose at hand through documented procedures for subclassing and extending the Galaxy product.

Application programs written in the Galaxy environment have direct access to all of the portable Galaxy API calls. In fact, Visix has committed to using only fully exposed and documented API calls between architectural components in the Galaxy environment.

The Galaxy API

C++ API Offers Developers Common Methodology to Develop and Extend

The Galaxy API is written in C, but it is very object oriented. For example, even initializing a button generates calls strictly to specific class methods. For developers working in C, extensions can be made in C; for developers working in C++, Visix will offer a C++ version of the API in the fourth quarter of 1992. The C++-based API will offer a class hierarchy based on C++ mechanisms. Visix is offering the alternative C++ API because those customers working in C++ want to be able to apply C++ concepts and operators to extending Galaxy.

Hiding the Data Structures Is Key to Longevity and Extensibility

A significant challenge to Visix's success with the Galaxy product is getting developers to work within the portability and extensibility boundaries of the API. Toward that end, the Galaxy programming environment offers no direct exposure of the data structures underlying user interface objects nor of Galaxy components. Instead, developers use documented function calls to get and set values for all objects, such as buttons. This restriction is key to avoiding the downside of developers using direct references to internal data structures and compromising the portability of the application, and to the forward compatibility that Visix is committed to providing in future releases of Galaxy.

Galaxy Architectural Issues

Multilayered Abstraction: Toward Developer Utopia

The layered Galaxy architecture builds on the three major API abstractions discussed above, which present the application and internal Galaxy functions with ideal interfaces to the network, operating system, and windowing services available on various system platforms. (See Illustration 4.) This architecture enables developers to write portable applications with high levels of functionality without the constraints inherent in any one platform. In fact, Visix believes it has not only achieved platform independence but has actually crafted a more capable development environment than is available with any of the native toolboxes. Within the layered Galaxy architecture, the application can call layered functions or the major system abstractions directly. When applications call layered functions, calls between internal Galaxy functions use documented API calls to ensure that the application developer can decide how to best get the functionality required. While the application cannot operate beyond the limits of the features of a particular platform, the developer is not limited by the characteristics of a target platform when writing the application.

Galaxy: Master of the Look-and-Feel Universe

Galaxy transcends the GUI wars with an API that offers the developer one abstraction from which Motif, OpenLook, Windows/CUA, or Macintosh look-and-feel can be derived. Unless custom objects are created within the application, the application and the developer have no awareness of the look-and-feel under which the application is presented to the user. Galaxy not only translates from one look-and-feel to another, but also manages compliance by transforming the windows and moving user interface objects to comply with locations on the screen per the style guides for each look-and-feel.

Developers Write to the Ideal Interface, Galaxy Handles the Mundane

An important architectural goal for Galaxy is to far exceed the functional needs of existing applications and mainstream devices in order to offer room for incorporating new technologies easily in the future. Visix believes that, by fully supporting the advanced capabilities future applications will require, applications developed with Galaxy will be suited to the needs of the next-generation mainstream user. For example, the Galaxy imaging model is based on the assumption of infinite resolution, unlimited colors, and scalable fonts in the target printer and display. The developer works with the imaging and color interface without concern for the particular limitations of today's printers and displays. Galaxy handles matching the output of the application to the highest capability of the output devices on the fly, using a variety of techniques, without explicit knowledge about the device on the part of the application.

Galaxy Architectural Issues

Galaxy Architecture

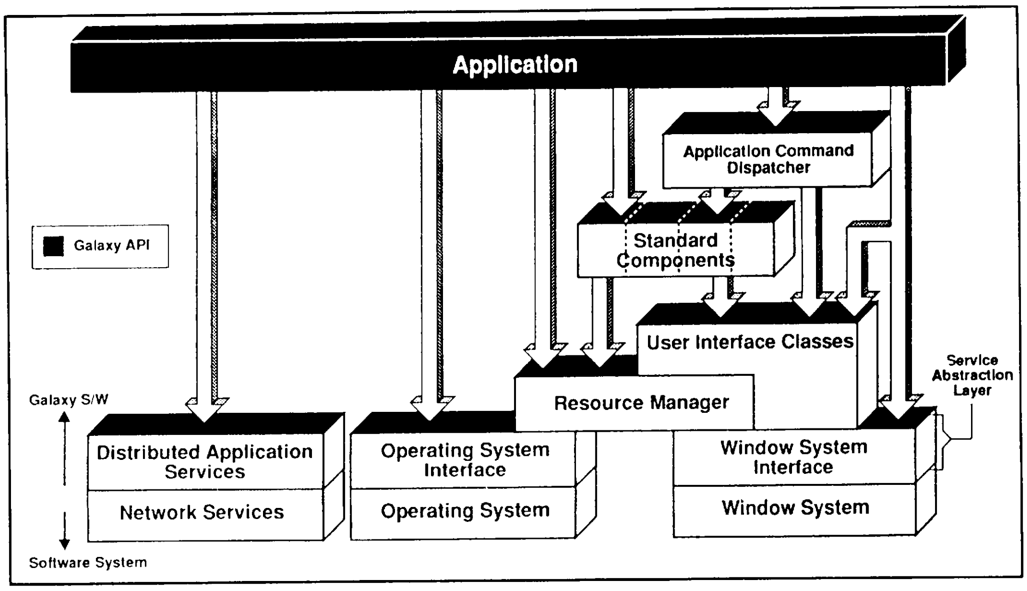


Illustration 4. The Galaxy Architecture exposes all of the interfaces directly to the application or offers layered interfaces to facilitate development. All calls between components are made with published API calls.

The Galaxy Porting Challenge

A major issue for buyers of Galaxy is the ability of Visix to meet the commitment to continually superset the functionality of the commercially important platforms and technologies as the marketplace evolves. This issue is fundamental to the acceptance of Galaxy. The Galaxy architecture directly addresses this critical issue by designing multiple layers of abstraction into each of the major "nonportable" interfaces to native system services. Two layers, the porting layer and the compatibility layer, insulate the portable layers from the uniqueness of each platform, lessening the downstream porting efforts to new and different systems. Both the porting and the compatibility layers are considered off-limits to developers, because writing applications that use this code would compromise portability. The porting layer operates closest to the particular platform and sets a sequence of flags that either implement or disable native operating system services. The compatibility layer runs above the porting layer, and it ensures that full POSIX and ANSI compatibility is available, in case the native platform does not offer full POSIX or ANSI C compliance.

For example, the Distributed Application Services Server (DAS) is built of three layers. The Distributed Services Registry is the top layer; the vremote layer, which implements remote operation, is in the middle; and the vcomm layer, which is the only layer that must be ported to each system platform to accommodate variations in implementation of connection-oriented or connectionless communications protocols, is on the bottom.

Application Portability Is Strongly Encouraged

In the interests of steering developers toward portability, not only have the interfaces to all library components been scrubbed of any details about their internal implementation, but some additional limitations have been placed on developers. These limitations have been conceived to avoid the problem of developing applications with assumptions about native functionality that, in fact, are not generally available. For example, developers cannot subclass the file system to modify it for memory-mapped file I/O because many OSs do not support it. However, although Galaxy does not preclude the developer from using memory-mapped I/O, developers are made aware that this will negatively impact an application's portability.

Galaxy Architectural Issues

Galaxy Class Hierarchy Is "Real World"

The Galaxy class hierarchy of function managers has been designed to meet what the Galaxy team considers "real world" requirements for significant distinctions among classes. As a result of this philosophical constraint, the class hierarchy may not be as deep as theoretical class hierarchies, but it has been designed to be readily visualized, understood, and used effectively by the average C programmer. (See Illustration 5.)

All of the managers in the Galaxy API, except for the Memory Manager, are actually Class Managers within Galaxy, offering full subclassing and extensibility based on inheritance of the class attributes and methods. The Memory Manager could not be classed or subclassed because the overall Class Manager needs memory management in order to function.

Performance

The architectural approach in Galaxy is to isolate work in the workstation client and to stay off the network in order to get the fastest performance possible. For example, the Communications Manager function in the DAS always selects the fastest interprocess communication mechanism available, including using shared memory for local operations, if shared memory is supported in the operating system.

IMPROVING GRAPHICS PERFORMANCE. To achieve high graphics performance, Galaxy drawing routines are implemented at the lowest level possible to replace the functionality of the graphical toolboxes underlying the look-and-feels replaced by Galaxy, including:

- GDI primitives level for the MS Windows
- QuickDraw level for the Macintosh
- Xlib level for Unix environments

Galaxy graphics are quite lean and efficient. One early developer calculated that compiling all the look-and-feels into an application for all target platforms would enlarge the application by only 40 KB.

GALAXY IMPLEMENTS SLOT-BASED CLONING SUBCLASSING. Galaxy supports a subclassing methodology called *slot-based cloning*. The slot-based cloning methodology subclasses objects through the duplication of pointers to class data or to reentrant code. This approach compares with other methods of subclassing objects which require the replication of all the class data as part of creating the subclass. This design decision is one of many made in the Galaxy program to try to offer high levels of functionality *with* high performance, which is certainly an issue of concern with a highly layered and modular architecture. In fact, the slot-based cloning methodology implemented when the developer subclasses a Galaxy object or class manager is implemented throughout the Galaxy internals. The Galaxy team is required to write reentrant code for all Galaxy components, and this constraint is rigorously enforced within the Galaxy development effort.

Multithreading Support

Galaxy internally supports fully asynchronous communications with virtually unlimited multiple outstanding threads. For example, in theory, an application could batch 1,000 vectors and have error messages returned asynchronously by having the vcomm layer of the DAS launch as many RPCs as needed to service the number of threads being managed by Galaxy. The multithreaded capability of Galaxy operates whether or not threads are supported in the host platform.

Galaxy Class Hierarchy

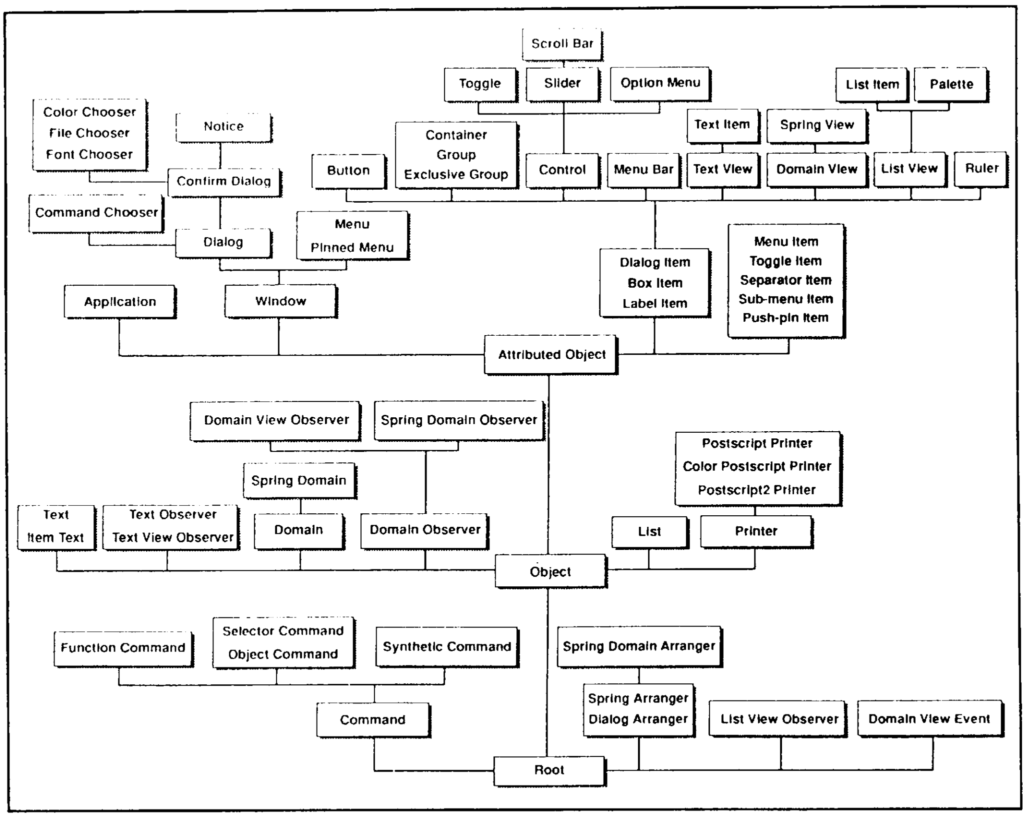


Illustration 5. The Galaxy Class Hierarchy illustrates the parentage of the class hierarchy implemented in the Galaxy system. This object-oriented scheme facilitates extending the features of Galaxy by supporting the subclassing of existing function managers as the basis for creating new class managers tailored to the needs of an organization or specialized application.

The Galaxy Tool Environment

Galaxy Tools Are Integrated Tools over a Software Backplane

The tools provided with Galaxy include the Programmable Text Editor, User Interface Builder/Resource Editor, Color Image Editor, Help Writer/Compiler, and Project Browser (see Illustration 1). These provide a nearly complete toolset for the development of complex applications, and they are integrated through the Galaxy "software backplane," which supports smooth switching between tools to maximize developer productivity. The Galaxy software backplane specification has been documented and is being made available to developers who want to incorporate additional tools into the builder environment.

Galaxy Supports High- or Low-Level Programming

Galaxy tools support working in high and low levels. Developers can work with the high-level tools or can access lower-level graphics functions as needed to build the graphics part of the applications. When the programmer needs to get to make calls specific to the windowing system, such as X11, Windows, or Macintosh, he or she is referred to a document that addresses how to write nonportable code. Visix believes that, if the programmer is knowingly going to build a nonportable application, there are ways to minimize the damage, and procedural recommendations have been documented for the programmer.

Debugger Support Included

The Galaxy development environment includes debugger support. In addition to the tools provided with Galaxy, all that many developers will require is an ANSI C compiler, though some users have supplemented the Galaxy tools with advanced object-oriented compilers such as Energize from Lucid, Incorporated (Menlo Park, California), which supports the development and debugging of complex applications.

Galaxy Programmable Text Editor

Application developers may choose to use any available text editor to develop an application in the Galaxy environment. However, the Programmable Text Editor included in Galaxy has been engineered with features that make it particularly useful to the professional developer. For example, the Editor stores images and fonts separately from the source text. This feature supports the mixing of images, drawings, and various font styles with source code, at the discretion of the programmer. Also, the Editor may be used to invoke the Galaxy online programmer's reference for detailed assistance in working with the Galaxy API. Because it is connected via the software backplane to the User Interface Builder, the Editor can also be used to edit resource files and user interface objects referenced in the application source code. The functionality of the Galaxy Text Editor is so rich that one early developer believes it could be used to build a powerful word processor in a few weeks.

Galaxy Project Browser: Project Management and Integration Utility

The Galaxy Project Browser tool will play a key role in providing a variety of views of the entire application, including components developed with third-party tools integrated through the Galaxy software backplane. The Project Browser can provide the developer with a variety of useful graphical views of the component pieces of the application, including the help files, source files, and resource files. The Browser keeps track of projects that inform the compiler what functions have changed; thus, only source code that has changed is recompiled. The Project Browser operates like a project data dictionary, enforcing basic integrity attributes.

The Galaxy User Interface Builder

The Galaxy User Interface Builder tool includes a resource editor that can be used to create or modify all types of resources. Extensibility is key to Galaxy. A developer can create new user interface objects without having to work with source code and low-level toolkit functionality that would be required with Motif or OpenLook because the user interface objects are actually object classes, not widgets. The programmer can subclass existing classes, override methods, and extend the class system. For example, the developer can pick the List class, subclass it and change its methods, and link it with libraries to build a new class, such as a class of ticker tapes for Wall Street. Some very complex objects are included in the object classes, such as a class of layered drawing objects that are rich enough to support the efficient development of a GIS system.

Galaxy includes the standard collection of widgets for each supported look-and-feel, and it includes the tools for developing within the prescribed style guides. Galaxy even includes some widgets for MS Windows that are not in the Windows SDK but are used by Microsoft and desired by developers, such as icon bars.

The Galaxy Resource Manager

The Galaxy Resource Manager is actually a B-tree database built on top of the File System Manager. The most outstanding aspect of the Resource Editor is the Spring and Strut feature that minimizes the work associated with relocating or resizing user interface objects. With conventional tools, developers are sometimes reluctant to create new dialog boxes because of the need to deal with complex resizing code. Springs and struts not only locate the object in the window with respect to the window but also provide for auto-resizing based on the size attributes of the contents of the object. This feature first appeared in the Looking Glass Advantage product, and, with additional refinements, it plays a key role in tackling the difficult problems inherent in taking applications cross-interface. The spring-and-strut methodology for building graphical user interfaces controls the relationships between user interface objects and the edges of the window, and the relationships among the user interface objects. Spring-and-strut location and relationship tools offer a very advanced visual description metaphor that ensures that the objects on the screen always maintain a useful location on the screen and with each other.

The Galaxy Run-Time Environment

The Galaxy run-time environment consists of three major components: the Help Server, the Distributed Application Services Server, and the Galaxy/CS User Interface Server, which will ship in 1993 as an add-on product.

Galaxy's Client/Server Implementation

Applications built in the Galaxy environment can be implemented in client/server mode by simply compiling with a different set of libraries. The Galaxy client/server (Galaxy C/S) stub libraries bound in with client applications (in the X Window sense of clients) generate RPC calls to the Galaxy User Interface Server (in the X Window sense of server), which manages access by multiple client applications to the user display.

The client/server dimension of Galaxy has particularly interesting potential because, with the PostScript imaging model, Galaxy can build "smart" servers (in the X Window sense) which could support low bandwidth X Window implementations that would offer high-performance, off-network imaging at the workstation. In the Galaxy model, the workstation-based server has knowledge of the resource files available to the application, enabling high performance as compared with the X11 model, which is an event-driven environment that sends information about user interface events with x and y coordinates that create hundreds of event calls to Xlib. The Galaxy client/server capability is planned to ship in the first half of 1993.

One of the most interesting components in Galaxy is the Galaxy Help System. The Galaxy Help System offers context-specific hypertext help capability to Galaxy *and* non-Galaxy applications. The Help Server offers the feature-rich graphics and text features of Galaxy, including font styles and sizes, hot links, and images. It even is designed to offer a less-intrusive version of Macintosh Balloon Help, called "live help," that automatically changes the context of a live help window to reflect the current context of the cursor. Galaxy includes translators for Microsoft RTF format help documents and Unix man (on-line command *man*ual) pages. The translators can be run in real-time as needed to display help to the local environment.

Galaxy Marketing and Business Issues

Galaxy Launch Plan: Build Credibility, Build Credibility, Build Credibility

The Galaxy launch plan is dependent on a ground swell of adoption in the target markets where application developers need to develop complex, graphical applications independent of operating system, network, file systems, and graphical user interfaces. Thus far, the launch of the product has occurred at a low level of visibility and with minimal information released through the normal channels of press and analysts. Visix has been operating with near-total emphasis on cultivating the early adopter ranks, on the assumption that a rich scatter diagram of diverse influential buyers will position Galaxy better than any claims or statements the company could make. In order to build the foundation of credibility to support a significant proposal from a small software company, the Visix marketing plan for Galaxy is to let the technical buyers within credible companies provide a groundswell of adoption by clearly staking their future plans on the Galaxy product. The adoption of Galaxy is clearly a significant bet-the-company strategy. In making the commitment to focus on developing applications instead of portability toolkits, adopters of Galaxy judge Visix as ready, willing, and able to continue to extend and maintain Galaxy as the interfaces to the underlying platforms evolve.

On the other hand, Visix has been trying to pique interest in Galaxy through the Unix trade press since last August. The approach has used intentionally cryptic advertising that links the name Galaxy with the tagline, "Galaxy Application Environment—Light Years Ahead."

Early Commitment Is Significant

Early adoption of Galaxy attests to the urgent need in the application development community for a light at the end of the tunnel for portability of complex applications. But it also says much about what Visix has achieved with Galaxy. Naturally, the early adopters of Galaxy are technically deep developers. The evaluation process conducted by the early adopters typically included detailed, in-depth architectural reviews that revealed a breadth and depth to Galaxy that eased the concerns about the ability of Visix to develop and extend the product as the marketplace evolves. The Galaxy launch, thus far, has been energized by strong word-of-mouth support from the technical community and concentrated somewhat in the "bleeding edge," early adopter end-user group on Wall Street. The most significant early contract for Galaxy is a multimillion-dollar deal inked in mid-September with Legent Corporation of Vienna, Virginia. Legent, a $600-million heavyweight in mainframe system management applications, aims to enter the OS/2, Unix, and MS Windows arenas with applications written with Galaxy. Legent apparently plans to use Galaxy primarily for new application development, but it will probably port some existing management applications to the Galaxy environment as well.

Channel Strategy for Galaxy

The channel strategy for Galaxy is initially focused exclusively on direct sales in order to develop proof-of-concept and to establish the highest-level value proposal possible across a range of user needs. In the future, indirect delivery could play an important role if the marketplace demands that system platforms offer some special support for Galaxy mechanisms, such as the DAS services. Merchandising to or through the industry consortia is not on the marketing plan at this time. The conviction runs deep at Visix that the direct developer-to-developer sell will more effectively establish the value of Galaxy than working through third-party forums or associations. We agree with this judgment.

Galaxy Delivery Progress Report

Visix has been delivering Beta and near-production-quality Galaxy code since February 1992 to an increasing sample of ISVs and end users. These early customers are porting existing applications, integrating tools through the Galaxy software backplane, and developing their next-generation applications for cross-platform delivery.

The release was upgraded to production-quality, controlled release in early summer. The full API has shipped to all of the controlled release customers. The technical evaluators and users of the early Galaxy software have uniformly found that the speed of response to problems and the turnaround on new features has exceeded expectations. Visix has even attracted an established training and consulting company to prepare an extensive training curriculum to assist programmers in learning and deploying Galaxy.

We expect to see the formal release of the Galaxy product in the fourth quarter of 1992, when the full set of tools will ship. The interest among qualified buyers has been extremely strong, and the company currently has devoted five salespeople and a senior sales manager to the early sales activity, in addition to the developers intensively supporting the controlled release customers.

Galaxy Pricing: A Bold Stroke

The published quantity price for Galaxy is $9,600 for a full development copy that includes the tools, the full API libraries, and the full set of documentation. Interestingly, Visix has made a bold stroke in pricing Galaxy by *not* charging for run-time copies. This strategy should help remove most ISV business objections and increase early penetration into the market.

Obstacles to Galaxy Adoption

The four biggest obstacles facing Visix in positioning Galaxy will be:

1. **Politics**—The unwillingness of the industry to concede that this fundamental layer of portability could not emerge from the many POSIX, IEEE, and X/Open committees that have been trying to define a set of interfaces for portable and distributed software. Unfortunately for the standards process, the richness of system software environments has grown dramatically while the standards processes have plodded, creating a usefulness gap that most in the industry concede.

2. **Homebrew Commitments**—The commitments already made to homebrew portability mechanisms, particularly by ISVs.

3. **Competition**—The competition today comes in two forms, portable GUI builders such as XVT from XVT Software and Neuron Data Open Interface from Neuron Data, and the nonportable but highly functional graphical development environment, NextStep from NeXT.

4. **Skepticism**—How can a small company solve this intractable problem and keep the commitment to extend and enhance the API and the class hierarchy to keep pace with changes in technology?

Summary and Conclusions

The industry has failed to offer developers of graphical distributed applications a viable model which addresses the multiplicity of interfaces and standards required for commercial success. The Galaxy mission statement is to liberate the developer from having to cope with a broad and changing landscape of interfaces to operating systems/ file systems, networks, and graphical user interfaces in the application development process and in the deployment process across networked systems. In liberating application developers from non-value-added drudgery, Galaxy allows developers to focus on excelling within their disciplines and broadens the range of platforms for distribution. Galaxy is significant because of its wide range of functionality; it frees the developer from most of the tasks of achieving portability.

Visix, with Galaxy, has brought forth a comprehensive architecture that could enable developers to build on any platform and deploy on any platform by recompiling and re-linking to the Galaxy libraries with the native compiler. The use of object-oriented concepts has enabled a consistent level of abstraction to be applied to the OS interface, file system interface, and network interface. The Galaxy product looks particularly timely in this period of new operating systems and next-generation operating system proposals.

Next month's *Open Information Systems* will address **Open Systems Strategies of the European Vendors** For reprint information on articles appearing in this issue, please contact Donald Baillargeon at (617) 742-5200, extension 117.

There is widespread FUD about which current or future applications will run on which current or future operating systems in hosted or non-hosted mode, with or without performance penalties. Galaxy cuts to the heart of this problem by offering ISVs and users a means to finesse the major system software variants already on the table and an architecture that appears to be extensible to cover future proposals for GUIs, networks, file systems, and OSs through a comprehensive class hierarchy and extensible object-oriented interface abstractions. The road to ubiquity for Galaxy might be steep, but Visix is known for technological vision and excellent engineering. The challenges lie in marketing, distribution, and, fittingly in a presidential election year, in industry politics. The success of Galaxy could be of great benefit to application developers and users. ◐

Open Systems: Analysis, Issues, & Opinions

FOCUS: NEXT COMPUTER

NeXTSTEP 3.0:

NeXT Goes Cross-Platform

NeXT Computer, Inc. has begun shipping NeXTSTEP 3.0 for its Motorola 680x0-based NeXTstations, and Beta releases will be available for Intel-based 486 and above personal computers beginning in the fourth quarter of 1992. General availability is expected in the first quarter of 1993. NeXTSTEP 3.0 represents a new direction in both strategy and technology for NeXT. It is a shift away from a proprietary hardware focus and a shift toward supplying a portable operating system.

With its entry into the Intel market, NeXT will now compete head on with the new generation of desktop operating systems, which includes Unix-based SunSoft Solaris 2.0, SCO OpenDesktop 2.0, USL Unix SVR4.2, and Microsoft's Windows NT. A tough set of competitors by any measure. NeXT's chances for success depend on users being willing to gain development productivity at the expense of application portability. Although NeXT plans to add on POSIX libraries next year, its primary goal in doing so is to meet FIPS procurement specifications, rather than to attract POSIX applications to its platform.

PORTABILITY VERSUS INTEROPERABILITY. Although based on a Mach kernel, NeXT is often criticized for being proprietary, since NeXTSTEP applications only run on the NeXTSTEP operating system. Applications written for other operating systems can be accessed by NeXT users through an X Window server that is available for NeXTSTEP, but other systems cannot run NeXTSTEP applications. NeXT emphasizes interoperability over portability, however, believing that the productivity advantage of the NeXTSTEP interface and development environments outweighs the sacrifice in application portability.

The lack of cross-platform portability has limited the number of applications that have been ported to NeXTSTEP. Because of its small installed base of about 50,000 machines and its unique programming interfaces, Independent Software Vendors (ISVs) have been slow to make the investment necessary to rewrite their applications. The resulting narrow range of 350 productivity applications from which to choose, compared with many thousands for other platforms, has been a barrier to wider adoption of the NeXT workstation. This chicken-and-egg syndrome will continue to plague NeXT, but the company hopes that bringing NeXTSTEP to Intel PCs will make it a more attractive platform for ISV development.

NeXTSTEP 3.0 Is a Shift in Focus

Apart from many feature enhancements in NeXTSTEP 3.0, the most significant aspect of this release is that portions of it have been rewritten in C from assembler, so it can be more easily ported to platforms other than the original Motorola 68K architecture. The first platform to be announced was Intel, but there is no reason to expect that NeXT will stop there.

By uncoupling its software from its hardware, NeXT gives buyers a choice of hardware vendors and access to a wider selection of hardware options. In this regard, its strategy is not unlike Sun's strategy with Solaris for Intel. However, unlike Sun and SPARC, NeXT has no vested interest in any particular processor architecture. In fact, the process of porting from the 68K to Intel forced NeXT to solve most of the problems it would face should it choose to take its operating system to another platform.

NeXTSTEP 3.0 Enhancements

NEW OBJECT KITS One of the keys to developer productivity in the NeXTSTEP environment is its object-oriented design approach. In addition to its object oriented application construction tool, Interface Builder, NeXTSTEP 3.0 contains four new object kits, which greatly facilitate application development. They are:

- *Database Kit (DBKit).* The DBKit provides objects necessary to build database applications, such as various controls, query tools, data display tools, and data types. It provides a consistent interface to SQL databases through "snap in" adapters for specific databases. Adapters for Sybase and Oracle are included, and other adapters will be made available.

The Sybase adapter is based on Sybase's DBlib, and the Oracle adapter includes Oracle SQL*Net and Call Level Interface. Adapters are being developed by third parties for Informix, Ingres, DB2, and Teradata. An SQL Access Group (SAG)-compliant driver will be able to provide access, using the same interface, to all SAG-supported databases.

- *3D Graphics Kit (3DKit).* The 3DKit enables developers to add three-dimensional graphics to new or existing NeXTSTEP applications. It is based on Pixar's RenderMan standard, and it includes both Interactive and PhotoRealistic RenderMan.

- *PhoneKit.* The PhoneKit supplies objects that support applications using both ISDN and regular telephone service for either voice or data transmission. Objects are provided that handle pickup, dialing, information transfer, and hang-up. It offers telephony hardware independence and provides for integration into NeXTSTEP applications.

- *IndexingKit.* The IndexingKit gives a developer tools that can be used to facilitate storing, indexing, and retrieving either text or record-based information in any application. It contains the B•Tree classes and includes some objects from Release 2.0 that have been put into the kit along with new objects.

INTEGRATED APPLESHARE AND NETWARE CLIENT. NeXTSTEP 3.0 now includes Novell NetWare client software fully integrated into the operating system. This is the result of a cooperative development effort between NeXT and Novell, and it is the first bundling of NetWare client software on a Unix-based platform.

NetWare client includes IPX protocols that enable NeXTSTEP users to turn their workstations into clients for NetWare servers. NeXTSTEP workstations can share files, printers, and other network services that are available on any NetWare server, including NetWare 2.2 and NetWare 3.11. One of the major benefits of this capability for users is the ability to access the Oracle Server for NetWare, which is becoming widely installed as a workgroup database server.

The set of NetWare APIs included will allow NeXTSTEP developers to build applications using NetWare services with the same interface used to build network applications using NFS and AppleShare. The AppleShare client capabilities included will give NeXTSTEP users access to AppleShare networks at the same time as other networks they are using. In addition, NeXT machines can now read Macintosh CD ROMs and Macintosh SCSI hard drives.

OBJECTS ARE NOW DISTRIBUTED. Previous versions of NeXTSTEP supported messaging between objects within a single NeXTSTEP application. Release 3.0 introduces Distributed Objects, which extends the same messaging model to include messaging between objects in different applications and across different computers on a network. This makes possible the creation of Object Links, a multimedia, hyperlinking system allowing dynamic sharing of information across applications.

OTHER ENHANCEMENTS. Release 3.0 introduces Display PostScript Level 2, including support for calibrated color output. Also included are imaging filters for faster printing and pattern support, and the Pantone color-matching system. Internationalization enhancements in NeXTSTEP allow users to set their systems for English, Spanish, French, German, Italian, Swedish, and Japanese. (Japanese is a separate product.) Unicode support will be included in a future release.

Object Links is new to Release 3.0, and it is similar in concept to Microsoft's DDE and Apple's Publish and Subscribe. It allows Release 3.0-compliant applications to share information through hot links.

Unveiling a Server Strategy

NeXT is a workstation company and makes no pretense that its machines should be used as servers. To fulfill that role, NeXT has entered into agreements with Data General (Westboro, Mass.), Auspex Systems (Santa Clara, Calif.), Solbourne Computer (Longmont, Colo.), Pyramid Technology (San Jose, Calif.), and the Teradata (El Segundo, Calif.) subsidiary of NCR. Through a variety of marketing and sales arrangements, NeXT customers will have a range of servers available to them for everything from file sharing to transaction processing.

An important component of NeXT's client/server strategy is the porting of its NetInfo network information tool to these vendors' platforms. NetInfo is a service consisting of protocols, library routines, application programs, and data that allows sharing of administrative information between computers on a network. Applications include UserManager, PrintManager, HostManager, NFS Manager, and NetInfoManager. They allow network administrators to administer systems from any point on the network. When a new node is connected, NetInfo provides automatic configuration of the new machine. With NetInfo on these new servers, the server acts just like another node on the NeXT network. Unfortunately, at the moment, NeXT has no plans to provide DME compatibility for NetInfo, although that would seem logical.

Meeting the Intel PC Challenge

Why would NeXT port its environment to Intel PC platforms? With an installed base of 100 million that is growing by many million each year, even a small percentage of that market is significant. More importantly, customers are far more likely to adopt NeXTSTEP if NeXT can reduce its image of being proprietary.

Porting to Intel PCs is no small feat. NeXTSTEP was designed with the Motorola 68K architecture as a central design point. However, since NeXTSTEP is based on the Mach microkernel, the amount of processor-specific code that had to be overhauled was relatively small. With most services running in nonprivileged user space, the porting task was much easier than it would have been had the NeXTSTEP architecture been more monolithic.

DEVICE DRIVER CHALLENGE. One of the challenges in developing system software for PCs is providing device drivers for the thousands of devices that can be configured on any PC. Driver development is costly, and, without support for a broad range of devices, the appeal for the operating system declines. Remember when OS/2 1.0 first shipped without a reasonable set of printer drivers? Microsoft has dealt with this problem in Windows by developing generic drivers for network interface cards, printers, and displays that have standard interface specifications which Microsoft publishes.

NeXT has tackled this problem in typical fashion. It has developed a DriverKit that is based on a new driver architecture. The architecture is object oriented, with several subclasses for each device type. At the highest level is a kernel data structure, below that is an object that knows how to talk to a class of devices, then a subclass for more specific category of devices, then a specific type of device, and then the code that is product specific. For example, a developer writing a driver for a network interface card can use the I/O class, the Network I/O subclass, and the Ethernet object in that subclass of the DriverKit. The only specific code that has to be written is for the particular ethernet card.

These classes consist of source code that is portable across all of the platforms that NeXT will support, as long as no processor specific structures are required. In the Ethernet example, the Ethernet object has to be written specifically for each processor, since byte ordering and buffer sizes vary.

This object-oriented architecture allows NeXT to build highly sophisticated drivers, including an SCSI driver that supports removable media. It also allows new types of device objects to be added very easily as subclasses, such as ISDN and token ring adapters. NeXT has been using the DriverKit internally to build its SCSI driver and others for the devices it will support in the standard distribution. It is making the DriverKit available to third parties, along with full documentation and sample code, so that drivers will be available when NeXTSTEP 486 Intel ships next year.

DISTRIBUTION CHALLENGE. Introducing a shrinkwrapped operating system for Intel PCs will present a major distribution challenge for NeXT. It will continue to concentrate on its direct sales efforts to land large multiple sales. Allowing customers the flexibility of selecting their preferred hardware supplier will help overcome many customers' past objections.

NeXT is also looking at OEM distribution possibilities but doesn't expect any initially. It would prefer to deal with the large, more stable companies with high-quality products, and those which have a systems perspective to their business rather than a "box seller" mentality. In the interim, it will be working closely with the hardware vendors to ensure compatibility and optimization of NeXTSTEP 486.

Making NeXTSTEP 486 a success will depend on generating demand, and normal PC channels will not be capable of this. NeXT would prefer to deal with value-added resellers (VARs) that are capable of supporting the hardware, networking, and software environment and that have value to add with their applications.

NeXTSTEP Continues to Gain Acceptance

NeXT continues to score successes in winning large accounts. Among the latest companies to select NeXTSTEP is Chrysler Financial (Southfield, Mich.), which will use 2,500 copies of NeXTSTEP 486 for a client/server retail auto financing application in over a hundred branches. Mobil Sales and Supply Corporation (Fairfax, Va.) will use 400 NeXTstations as traders' workstations, along with an energy trading system developed by a third-party developer, mc^2 (Westport, Conn.). McCaw Cellular Communications (Kirkland, Wash.) will employ several thousand NeXTstations over the next three to five years to deploy an application developed internally in a fraction of the time it would have taken in other environments.

The benefits of NeXTSTEP as an application development environment have been recognized for some time. The latest evidence comes from the Swiss Bank Corporation (London, U.K.), which estimated 50 - 100 percent productivity gains after switching to NeXTSTEP, as measured by function point analysis. With the introduction of NeXTSTEP 486, customers will be able to deploy those applications on 486-class personal computers from virtually any supplier, overcoming one of the large obstacles to widespread NeXTSTEP adoption.

Conclusions

NeXT has made a bold strategic move in its decision to work with other platforms. The quality of its implementation will go a long way in determining the success of this strategy. However, users will continue to be disappointed by the narrow range of shrinkwrapped application choices that run natively on NeXTSTEP. On the other hand, the NeXTSTEP development environment may attract creative new applications that more than offset the unavailability of other, more widely accepted applications. NeXT customers tend to be strong proponents for NeXTSTEP. The challenge NeXT faces is to sell interoperability along with ease of use, innovative applications, and rapid development to offset not having portability or the big applications. —*M. Goulde*

FOCUS: DOCUMENT MANAGEMENT

IDI Provides a Structured Database Model for Document Management

Treating Documents as Databases

For years now, we have been bemoaning the fact that you can't collaboratively work on documents using a database model. Let me elaborate. In a database table, only a single record is locked when it is being edited. All the rest of the records in that table are available for updating by other users. In a document, we have always believed that only the component that is being edited (say, a paragraph or section) should be locked from editing by others. The problem was that, in order to ensure that sort of component security, you needed to divide the document into the equivalent of records. And how do you take an unstructured stream of text bytes and impose a record-like structure? The most promising technology for doing just that is Standard Generalized Markup Language (SGML), fast becoming the standard for defining the structure of a document by tagging the different components within the contents. Once each piece of text is tagged, it can be identified as a separate object within the context of the entire document.

There are currently a variety of SGML editors and tools that will parse (verify that the document follows a valid structure) SGML documents, and new tools that will actually convert plain old textual documents into SQML documents.

Component Level Document Management Product

Information Dimensions Incorporated (IDI), makers of BasisPlus, a popular Unix and VMS-based document management system, has just announced a product that allows users to work on a document in just the way we've been waiting for. The Document Transformation (docXform) product includes technology for autotagging non-SGML documents into the SGML structure and for managing those documents at the component level. This means that you can check out, spell check, search, format convert, edit, etc., by component, such as chapter title, introduction, normal paragraph, etc. The product runs on Unix and VMS.

IDI has acquired two technologies that underlie the new product:

- FastTAG from Avalanche Development (Boulder, Colorado). FastTAG analyzes documents and infers structural information, autotagging the contents with SGML codes.
- Document parsing technology from Exoterica Corporation (Ottawa, Canada). The Exoterica technology includes a standard SGML parser that validates the autotagging done by FastTAG to make sure it conforms to the document type definition—an SGML equivalent to a data dictionary (DTD). IDI has also licensed Omnimark, Exoterica's fourth-generation text-programming language.

After a document has gone through the Avalanche and Exoterica technologies, IDI's docXform will take each SGML-tagged component and map it into the BasisPlus database.

This technology works with any scanned image, SGML document, or any word processing document (including PostScript files). At present, complex hierarchical documents, such as technical documentation publications, cannot be processed.

Benefits of Component Level Document Management

PRECISE SEARCHING. In an unstructured document, the best you can do is search for words or phrases. If keywords have been identified in a document header, you have a more precise search, but this is still a very work-intensive, manual process of specifying the keywords for each document. With a component model, you can limit searches to specific portions of documents. For example, you could choose to search only the summaries and introductions of a selected group of documents, based on the assumption that the significant topics covered in the body of the documents would be mentioned in one of these two components. This narrows the amount of text you need to search and thus improves performance. Another example might be that of searching through hundreds of resumes, trying to find someone who is familiar with current state regulations. In a traditional search, you could find all the resumes that mention the word "State," but that would also include people who live on State Street or who claimed skills on state-of-the-art word processors. With component searching, you could limit your search to employment history sections.

CONCURRENT COLLABORATIVE EDITING. Component document management supports the editing model we mentioned earlier. When you check out a specific component to edit, it is locked, but the other components of the document are still available for editing. Even more, if someone else requests that entire document or sections that include the locked component, he or she can access that portion. The locked component is read-only, but the individual can see the context of the entire document. This is very important. If I am working on an executive summary section, I need to be able to see the main contents of the document, even if I'm not able to edit it.

DECREASED NETWORK TRAFFIC. Since you can pull up a single component to edit, rather than an entire document, you reduce the amount of traffic on the network.

SOFT VERSION CONTROL. The system can tell which components have been changed from version to version. Therefore, the system saves only the deltas for each version. This saves storage space and also allows the system to keep even more accurate historical information by maintaining an audit trail on each component.

Working across Documents

An effective document database model not only needs to understand the logical structure of a single document, it must also understand the relationship of like component types across documents and across databases of documents. DocXform allows precise searches across documents, creating virtual documents out of the components chosen.

NEXT GENERATION PRODUCT. In spring '93, IDI will introduce the next evolution of its component document management product line, docXapi, a programmer toolkit that allows you to create hypertext virtual documents from componentized documents. The system will also create a virtual table of contents for the pieces it has put together. The toolkit supports development of graphical front ends, using tools like Visual Basic, Hypercard, Motif, etc. The target market for docXapi is graphical desktop environments. In addition, docXapi will support more complex cross-document, cross-database table precise searching.

Conclusion

It is probably apparent that we are pleased to see this technology become available. The ability to add structure with a minimum of pain to what was previously unstructured gives users new control over the information in their organizations, most of which is in standard office documents. The possibilities for integrating text components with other data types in database applications are endless. And, finally, we can simultaneously edit the same documents without clobbering each other's work! —*R. Marshak*

On Your Mark, Get Set, Go! ... to

Patricia Seybold Group's MARATHON WEEK

Take off with the Boston Marathon runners and cross the finish line better prepared to deploy open distributed computing throughout your organization.

Enter Marathon Week and Cross-Participate in 3 Simultaneous Forums!

- **Distributed Object Computing Technology Forum**—*How Can I Use Distributed Object Computing Technology Today to Build Mission-Critical Applications?*
- **Open Systems Forum**—*What Can I Do with Open Systems Today? What Lies Ahead?*
- **Seybold Executive Forum**—*Design a Real Business Solution That Implements Open Distributed Systems*

April 18 - April 23, 1993
Copley Marriott,
Boston, Massachusetts

As the technology boundaries begin to merge, so must our Forums. For the first time, Patricia Seybold Group will hold its three annual conferences all in the same week at the same hotel. Our goals are:

- To allow participants to focus on the topics that best meet their business needs. Participants may cross over to attend any sessions of interest.
- To enlarge the group of people with whom you can meet and share experiences.
- To explore how distributed object computing enables open systems.
- To allow participants to enhance their conference learning through business case-focused, hands-on applications of technology.

In addition to the three conferences, participants can visit the evening Technology Showcase, where leading vendors will demonstrate real business solutions.

For more information, call Deb Hay at (800) 826-2424