Patricia Seybold Group

Editor-in-Chief
Michael A. Goulde

INSIDE

OPEN INFORMATION SYSTEMS

Guide to Unix and Other Open Systems

Vol. 8, No. 4 • ISSN: 1058-4161 • April 1993

Unisys ASD Framework

Meeting the Challenges of Software Development in the 1990s

By John R. Rymer

IN BRIEF: The popularity of rightsizing and outsourcing in corporate information systems is bad news for corporate software development staffs. The message is that business managers have not only lost their patience with late, over-budget, and inadequate software, but that they are also making serious changes. To meet these new demands, corporate developers must reduce the cost of software development while increasing its speed, develop software that can be changed, closely link software development to business planning, and migrate from host-based development platforms to Unix servers and distributed platforms. ASD Framework, Unisys's strategic CASE environment, seeks to meet these goals without forcing developers through wrenching change, and, at the same time, borrows key concepts from object-oriented software development. Over time, the Framework will help current corporate developers migrate to full object-oriented development while adhering to open-systems standards. Thus, ASD Framework illustrates the future direction that productive CASE technology can and should take. *Report begins on page 3.*

Accelerating the Standards Process

Is COSE on the Right Track?

THE GROUP OF SIX vendors developing the Common Open Software Environment (COSE) is taking a novel approach to developing vendor-neutral specifications. Discussed in more detail on page 24, COSE will be a specification written by a small group of self-selected vendors, which, although not under the control of any one vendor, is not open to general industry input either. No users or application software developers are involved. The COSE approach may be valid when contrasted to other methods for establishing specifications that would, we hope, become standards.

One is the Microsoft method: Develop a draft specification and circulate it for comment among a small, select group. Weigh the group's feedback against your strategic objectives, and use or don't use it to develop a final specification. The process isn't really open, since it isn't open to all, but it isn't completely closed either, since Microsoft does solicit outside input. What it certainly isn't is vendor neutral.

Another is the consensus method: Develop a specification through an extensive series of meetings of special interest groups (SIGs) and technical review committees which discuss the relative merits of requirements, proposals, drafts, and amendments. Continue this process until specifications are ready for circulation and comment by any interested party. After review and comment, additional revisions are made and commented upon. Balloting often occurs at various stages in the process. The result is a consensus on the part of as many parties as may be interested that the final specification is a satisfactory compromise.

The COSE process is different from both of these processes. Although it is not being initiated by a single vendor, it is not open to all interested parties, either. At present, we don't know to what extent consensus among the six parties will be sought, let alone what efforts will be made to seek input, not to mention consensus from other interested parties in the industry.

The six COSE vendors claim 70 percent market share for desktop Unix. Should the industry abandon the consensus model for evolving specifications in favor of a market share model?

The benefits of the COSE model are the same as those that Microsoft claims justify its process for developing specifications. Consensus processes are too slow to keep pace with technology and the need to get technology to market. So, by sacrificing consensus in favor of a more autocratic model, specifications can be developed faster, and products based on those specifications can be delivered to customers in a more timely manner.

Customers get caught in the middle of the single vendor, multivendor, and vendor-neutral approaches. The single-vendor approach is pragmatic, practical, and painless. But it carries the traditional risk of vendor lock-in and limited choice. The multivendor approach has moderate risk of lock-in or limited choice. The vendor-neutral approach has the least risk of lock-in and the maximum choice, but it is also the least efficient, practical, or pragmatic.

Perhaps there is a middle ground. What both the Microsoft and multivendor models offer is an accelerated process for developing specifications where none exists. If a multivendor group developed the basis for a specification, then circulated its work throughout the industry, and used the resulting feedback as the basis for a final specification, then its work should result in broader industry support. The next step would be the same as what COSE has done: Submit the specification to X/Open to be considered for adoption into the Common Application Environment. This final step is one that even Microsoft should consider.

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 1058-4161) 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.

Unisys ASD Framework

Meeting the Challenges of Software Development in the 1990s

Introduction: The Software Crisis

The Crisis in Corporate Software Development

Corporate software development is in crisis. Corporate development shops can't reliably build the right information systems within useful time frames and at a reasonable cost to support business objectives. The software they do build is too difficult and expensive to modify as business requirements change. The sign of this breakdown is the fact that software maintenance accounts for upwards of 70 percent of most corporate MIS budgets. There's little hope in sight for a reduction.

MANAGEMENT IS SKEPTICAL. Corporate managers are growing increasingly skeptical about the payback from large, in-house software development staffs. In many corporations, corporate management is mandating that new software be developed on low-cost workstation and/or LAN platforms rather than on more expensive mainframes or minicomputers. This tactic, known as *rightsizing*, often also results in directives to move existing host-based applications to LANs to reduce their ongoing costs.

OUTSOURCING HIDES THE PROBLEM. In other corporations, management is actively farming out custom software development to service companies. This tactic has come to be called *outsourcing*. Often, corporate managers employ both rightsizing and outsourcing to reduce the overall costs of building and maintaining business information systems because they are dissatisfied with their internal MIS organizations.

THE BOTTOM LINE IS THE BOTTOM LINE. The growing popularity of rightsizing and outsourcing is bad news for corporate development shops. The strategic goal of both rightsizing and outsourcing is to reduce the costs of creating and maintaining information systems. Corporate management clearly has come to believe that the costs of information systems are too high—no matter how strategically important information is to their businesses.

This posture is a significant change for corporate software developers. For the first time in the history of the computer industry, corporate managers across the board are subjecting information systems to the same scrutiny as their other investments to support business objectives.

Corporate Software Development Addresses the Challenge

How has corporate software development arrived at this sorry state of affairs? Users have been investing an enormous amount of time and money in new software development techniques and technologies during the last decade. Chief among these have been computer-aided software engineering (CASE) tools that seek to improve the quality and maintainability of software. However, corporations have also invested in advanced technologies, such as object-oriented software development tools and expert systems, in an attempt to solve their development problems.

On the face of it, these investments have generated only paltry dividends. CASE tools and integrated environments have failed to produce the expected breakthroughs in the quality and cost of custom software. Expert systems have long since ceased to be a focus of corporate software development, and object-oriented tools are too new and unproved in mainstream development to be judged.

Introduction: The Software Crisis

And so, despite a great deal of effort, the software development crisis has worsened, not improved. Why? The primary reason is that no single new tool or technique can fix the structural problems of software development. Most attempts to solve the software crisis attack the problem piecemeal, when a comprehensive technology and management strategy is what's required.

How to Beat the Software Crisis?

In order to turn the situation around, corporate software development shops must fundamentally change the way they design, build, and deliver software. The cost of software development must be reduced at the same time that the speed of development is increased. Software must be developed that can be changed. Development must be closely linked to the business planning process. And software development must migrate from host-based development platforms to Unix servers and PC LAN servers.

Four General Requirements for Success

Link Software Development to Business Requirements

The most important change in current software development practices is to make business managers real participants in development. Software developers must throw software development open to scrutiny by business managers to regain their trust. Managers should be able to look at any project in any stage and be able to independently judge whether or not it is on track to satisfy their business needs.

MAINTAIN FOCUS. Current efforts to involve business managers are too circumscribed. Developers solicit the views and requirements of managers and end users through joint-development planning sessions. After these initial sessions, developers may show managers prototypes from time to time, but the process of development remains a mystery to the business managers who, ironically, fund it. Moreover, as the typical implementation project proceeds further and further from the initial requirements session, it becomes a less and less accurate representation of those requirements and the high-level design that proceeded from them. All too often, by the time the software is finished, it is out of step with the original design *and* with the changing business requirements.

ACCOUNTABILITY AND COOPERATION. Rapid prototyping, the current rage in the industry, is only a partial response to this problem. What's really needed is an accountable, cooperative approach to requirements gathering, functional specifications, application design, implementation, and maintenance that includes, but is not limited to, prototyping. To satisfy this requirement, developers must be able to represent their understanding of requirements, their designs, and even their code in terms that business managers can understand. The best way to do this is for developers and managers to identify the unique concepts that define their components and operations, and then to encapsulate these definitions in software objects. Then developers and managers will have a common language to discuss applications—the names and functions of the objects.

ONGOING DESIGN REVIEW. The ideal CASE environment, in this regard, will allow managers to critique requirements specifications and high-level designs for applications, and developers to generate code from the high-level designs. Moreover, the ideal tool will also accurately represent changes to the design that result from inevitable low-level coding decisions made in the interest of efficiency and other factors. In this way, managers would always be able to review the current designs of an application. In CASE terminology, this is known as *traceability* between design and code.

MULTILINGUAL DEVELOPMENT. In addition to breaking down the wall between design and code, the ideal CASE environment will also remove the walls between the different languages used to build an application. For example, the environment would allow developers working with 4GLs to manipulate low-level data structures created by programmers working in C. The 4GL

Four General Requirements for Success

programmers would not have to leave the comfort of their language environment to work with these structures, even though they were created using a different language.

Reduce the Cost of Development

Productivity is the primary focus of CASE and other attempts to improve software development tools and techniques. Yet, most CASE users say that the primary benefit of this technology has been to improve the individual programmer's ability to generate lots of lines of code. Current CASE tools have been much less successful at promoting and supporting code modularity and reuse. To skeptics, code reuse is the Holy Grail of software development. They are wrong. Reuse is attainable, but it requires communications between developers and ironclad testing procedures that are not typical today.

Code reuse is the most promising method available today to reduce the cost of software while meeting new challenges of quality and functionality. Reuse promises to save both time and money—time in that developers won't have to reinvent existing pieces of functionality when they build an application, and, for the same reason, money. Reuse also saves money on testing and integration of software modules.

The key to reuse is not new technology (although some new tools would certainly help to automate reuse). Rather, reuse can be achieved by development organizations that institute disciplined development processes and quality-control policies and procedures, including unit and integration testing. These steps will help to create the atmosphere of accountability and trust required before developers will reuse the work of other developers.

Make Software That Can Be Changed to Reflect New Conditions

Markets change, and businesses that hope to operate profitably within those markets must also change if they hope to thrive. Unfortunately, the software that supports a business is often an impediment to healthy change. Most custom software is designed not to change, but to represent the state of the business at a point in time a year or two earlier than today.

To support change, software designs must be modular, and modules must have three essential characteristics. First, they must be self-defining. Second, they must be self-advertising so that they can tell other modules what they are and which functions they can perform. Third, modules must be intelligent enough to field general requests for services. Modules that require prior knowledge of their implementations will not support change without substantial modification of themselves and other modules.

Migrate from Current to New Technologies

The most promising general solution to the crisis in software development is object-oriented technology. Object-oriented software development offers corporations a foundation for creating intelligent software modules that can change over time without compromising their ability to interact with other modules. Current implementations of object-oriented concepts are not perfect, but it is important to distinguish the *conceptual foundation* from the available products that seek to deliver the foundation.

Object-oriented development techniques and tools require a shift away from current knowledge about software design and implementation to new ideas. The shift is inevitable for the reasons outlined above, but the change won't take place overnight. Rather, corporate development shops will take years to move from their current base of procedural development technology to object-oriented development technology. The need to extend the life of current investments in data and information technology is also a conservative force in this migration.

An important requirement of any comprehensive CASE framework/environment is to aid this migration. To the extent that they are specific to a single language, development methodology, or application model, software development frameworks and environments make the transition between procedural languages and object-oriented languages that much more difficult. The ideal situation is for the development environment to be open (based on vendor-neutral standards) and flexible enough to bridge today's requirements and tomorrow's. This

Four General Requirements for Success

means bridging procedural languages and object-oriented languages, for example. However, it also means bridging centralized applications architectures (based on monolithic relational databases, for example) and distributed architectures such as client-server and distributed peers.

Object-Oriented Programming Meets the Challenge

Reduces Complexity and Provides Reuse

Object-oriented software operating in a distributed computing environment is the future toward which we are moving. Object-oriented software development supports two important user goals. First, it can reduce the complexity of building software for distributed, heterogeneous environments by encapsulating intelligence about the logical components of those environments within objects. This makes it possible for developers to make more effective use of distributed environments than is possible today.

Second, objects provide a reasonable basis for reusable software components. Reusable components promise to speed development by eliminating the need to code all elements of every application from scratch. In addition, however, objects promise to raise the overall quality of software by placing pretested components into the hands of developers.

Corporate Pioneers at the Frontier

A modest number of corporate software development shops are making this transition today by shifting their software development to object-oriented development environments rooted primarily in either SmallTalk or C++. These shops are the leading edge in corporate software development, and they are demonstrating the benefits of object-oriented development despite a lack of a complete set of tools. Most object-oriented development environments today require hand-coding with rather low-level languages. Tools that *generate* these low-level languages from high-level specifications will accelerate the movement to the object-oriented approach to systems development.

The majority of corporations are not yet prepared to make the substantial change in their current tools and practices that object-oriented development requires. These shops will need interim solutions to help them move from procedural software development—principally with Cobol—to object-oriented design and implementation. The ideal interim solutions will help users change their practices using procedural tools in object-oriented ways, eventually paving the way for a full shift to object-oriented software development.

Unisys ASD Framework Offers an Alternative

Unisys Advanced Solution Development (ASD) Framework is one of the few alternatives available to corporate developers today. Fortunately, it is a good alternative. ASD Framework gives developers a mature set of procedural tools that support more productive approaches to software development by borrowing key concepts from object orientation. Make no mistake—ASD Framework is not an object-oriented development environment. It is 4GL based, but it delivers the modularity and separation of application specifications from implementation that are available in object-oriented environments. Over time, ASD Framework will incorporate object-oriented languages, paving the way for users to make the shift to completely object-oriented development, including inheritance of specifications to create new modules from existing ones.

A Practical Software Engineering Platform

ASD Framework is both Unisys's strategic CASE product line and its development architecture. ASD Framework is a platform that places an existing suite of development tools into a unifying context. (See Illustration 1.) Those tools are:

Unisys ASD Framework Offers an Alternative

- Business Planning Workbench (BPW), which is a tool for defining business requirements and mapping them to information systems at a high level. BPW comes in DOS and MS Windows versions.

- Designer Workbench, which is a tool that provides modern graphical user interfaces to existing applications built with Unisys's existing 3GL and 4GL tools. The environment works without requiring modifications to existing applications. Designer Workbench is an MS Windows-based tool.

- The Mapper System, which is a fourth-generation language for creating user front ends and environments in a network environment spanning mainframes, midrange processors and Unix servers, and PCs.

- LINC II and Ally, which are designed to create transaction-oriented applications. LINC II's unique value is its ability to generate the database and network structures required to support an application from a high-level application specification. LINC II runs on Unisys's A Series and 2200 mainframes, as well as its U Series Unix processors. Ally runs on the U Series and several other Unix boxes and is designed to create Unix-based, database-independent applications. Ally supports the X/Open Company Limited Distributed Transaction Processing (DTP) model.

A Set of Related Tools

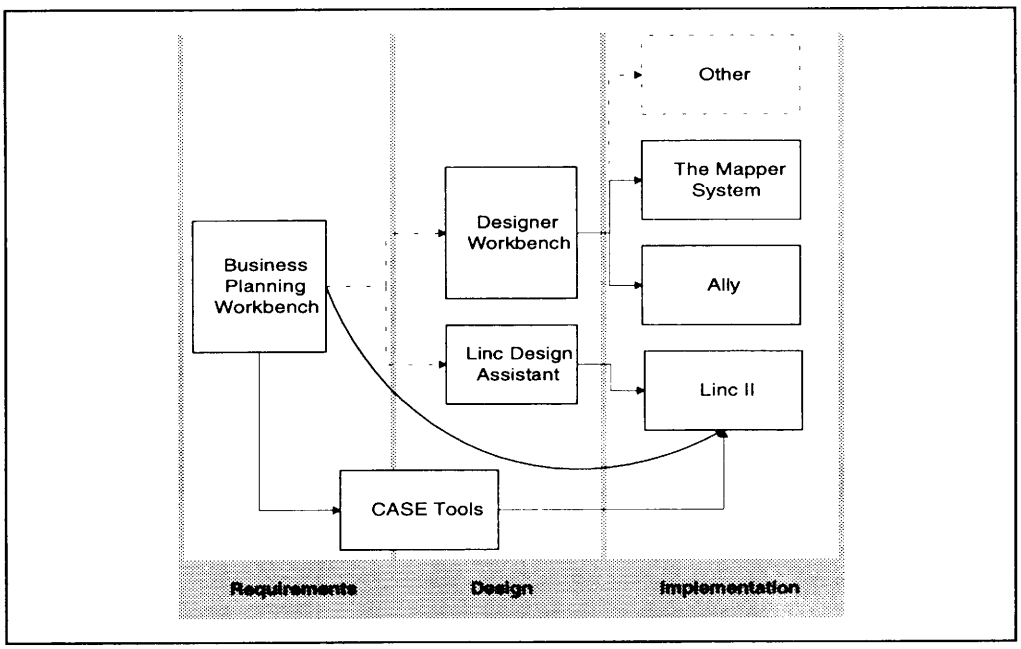

Illustration 1. ASD Framework is a platform that places a set of existing tools into a unified context. This illustration shows which phases of development each ASD tool is designed to address, along with the direction connections between tools that exist today. Dotted lines indicate near-term extensions of the ASD tools' capabilities.

How ASD Framework Is Different from Other CASE Frameworks

ASD Framework is different from most other CASE frameworks. Users can't go out and buy ASD Framework as such. Rather, it is a Unisys internal architecture designed to ensure the integration of the various Unisys tools and the ability of all of the tools to support new technologies over time. To satisfy these requirements, Unisys plans to incorporate support for a wide variety of standard interfaces and protocols within ASD Framework.

Unisys ASD Framework Offers an Alternative

A STRONG FOUNDATION. Thus, ASD Framework is really a set of tools with a strong foundation. Most other CASE frameworks also offer developers tools and APIs to repositories, data integration, and other useful services. Most CASE environments offered by major vendors are skeletons of these proprietary APIs waiting to be fleshed out with third-party development tools. Vendors espousing the API approach to CASE frameworks are seeking to establish a set of long-lived standards that lock customers into their technical direction for the foreseeable future.

A DIFFERENT APPROACH. Unisys's approach is different. So far, Unisys has been uninterested in creating its own proprietary APIs for such services as repository and data exchange between disparate tools. Rather, Unisys plans to support as many CASE APIs and messaging protocols as possible and to interoperate with all major competing systems. Unisys has a genuine commitment to neutral standards.

PRACTICAL STRATEGY. Unisys's CASE strategy is practical. The company has quality software development tools in its product catalogue, but, historically, it has sold these tools only to its hardware customers. Now, Unisys wants to make its CASE tools product line independent of its hardware business. To do so, it must reach out to users that don't have Unisys hardware and don't know of Unisys's strengths as a vendor of software development tools. Unisys has to give software developers at large a reason to care about ASD Framework.

PRAGMATIC FOCUS. Unisys is tackling this challenge by focusing on a particular type of developer and by seeking to solve problems today rather than promising solutions tomorrow. The target customer for Unisys is the mainstream corporate developer working in Cobol. ASD Framework gives these developers a high-level environment that can carry them into development on Unix servers and distributed computing platforms without a wrenching change.

NOT TARGETING OBJECT-ORIENTED DEVELOPERS. Unisys is specifically not seeking to appeal with its current product line to leading edge developers who have already made the switch to C++ or SmallTalk. For now, Unisys will let other vendors service these developers. Unisys's goal is to provide developers with tools that permit them to specify applications at a high level and then generate C++ (among other languages) to implement the applications. This strategy will also enable Unisys to support transparent access to existing business data.

Unisys's focus on solving problems now is evident in its promotion within ASD Framework of its *tools,* rather than a set of APIs. Over time, Unisys is promising to build up the technical underpinnings of its tools to provide greater functionality, greater integration, and support for a greater number of tools, methodologies, and programming styles.

The Guiding Principles of ASD Framework

ASD Framework is both a conceptual framework to help users understand Unisys's direction and priorities with its development tools and an architecture for those tools. The framework is depicted in Illustration 2. The concepts supporting ASD Framework are expressed in a set of principles. The architecture is expressed in a set of common components for Unisys's current development tools and an architectural direction for future versions of those tools.

The difference in Unisys's strategy is evident in its four guiding principles:

- Support rapid application development. This means support for prototyping and for improved programmer productivity using high-level tools.
- Support transition to new technology by hiding implementation details beneath an extensible, high-level development environment. Among the new implementation tech-

nologies Unisys plans to support within ASD Framework are object-oriented programming languages and distributed computing facilities, such as remote procedure calls.

- Support as many interfaces as possible to enable integration and interoperability with competing environments, as well as integration of third-party tools with ASD Framework.
- Closely link business requirements analysis and planning with the software development process.

ASD Framework

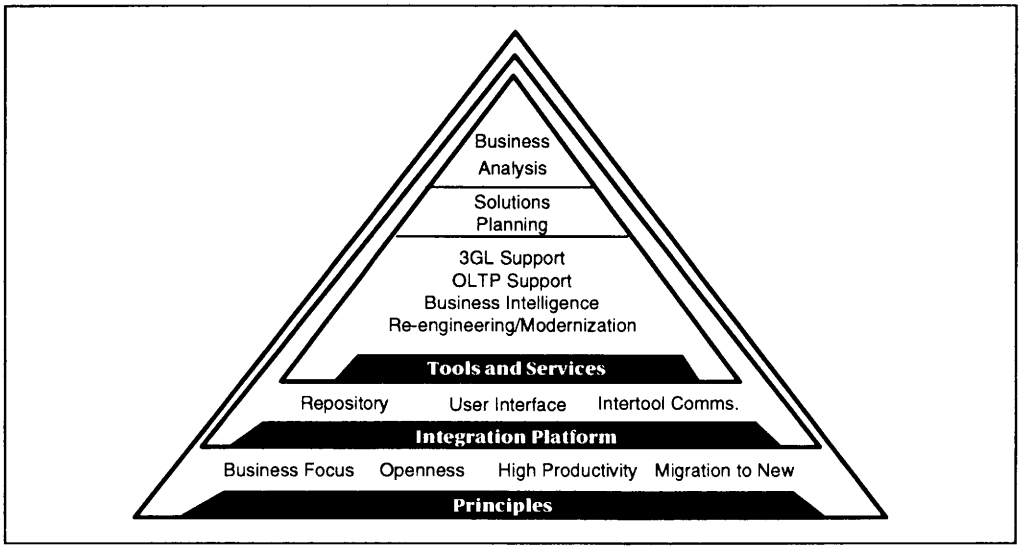

Illustration 2. ASD Framework has three conceptual layers and a variety of individual components. The keys to the future architecture of Unisys development tools, however, are the forthcoming Unisys Repository, a common user interface to multiple tools, and a future intertool messaging service.

Commitment to High-Level, Language-Neutral Development

The last of these principles is the key to Unisys's CASE strategy, which builds on ideas from fourth-generation languages that are still valid today (but often forgotten). The breakthrough of 4GLs was that developers could build systems faster and with greater accuracy by using high-level concepts about systems rather than low-level structures for programming.

Defining applications using high-level concepts is a hallmark of ASD Framework. But Unisys takes the concept an important step beyond 4GLs. Rather than linking high-level development to a particular language, as 4GLs do, Unisys supports the creation of a variety of language implementations from high-level specifications. The result is an important bridge between today's procedural languages and new languages, including object-oriented programming languages. As long as the system's high-level specifications are available, Unisys will be able to generate a variety of implementations from them.

This strategy has had an interesting result for Unisys. Whereas most other framework vendors—IBM, Hewlett-Packard, and CenterLine Software, for example—are focused on providing integrated programming environments for C or Cobol or C++, Unisys is not. Unisys has programming environments already in its LINC II, The Mapper System, and Ally products, but its focus is on allowing systems to be specified in a language-independent fashion, allowing developers to generate code in a variety of languages from them. Thus, Unisys isn't competing with language-environment vendors. Instead, it is building a framework that will surround and coexist.

The Guiding Principles of ASD Framework

Architectural Details, Today and in the Future

ASD Framework is more than a set of principles. It is also a strategy that dictates an architecture for software development tools—both in the short term and in the indefinite future.

The architecture seems unremarkable in Unisys's depiction of ASD Framework (Illustration 2). There is a repository, which is a database of information about the code that comprises applications; a unified user interface; and inter-tool communications services. Every major CASE vendor has these elements in its architecture for the future. However, each of the Unisys environments—LINC II, The Mapper System, and Ally—conform to this general architecture today.

SEPARATE TOOLS—COMMON FUTURE. Each of Unisys's tools has its own repository, user interface, and development integration facility. Unisys does not yet have a common framework of these services that spans its applications. The plan is to first build interconnections between the individual tools and then migrate to a common layer beneath all of them. The beauty of this approach is that it delivers functionality today, and it will deliver increased interoperability in the near term and a single powerful platform in the future *without forcing users through jarring technology transitions.*

MAKE OR BUY TOOL INTEGRATION? In addition, the ASD Framework architecture is different in an important way from most other CASE architectures. Of the three major components in the architecture, Unisys is furthest from implementing a development tool integration system (such as SunSoft's ToolTalk and HP's Broadcast Message Service). Unisys is currently studying how to implement this functionality, and may, in fact, buy it rather than build it.

FOCUS ON RICH REPOSITORY. Unisys's primary focus is on designing a repository that will support rich multitool data-sharing—a much more robust level of integration than that supported by inter-tool messaging. Unisys needs to define an information model for its repository that can envelop a wide variety of data models, not just a few chosen in order to achieve its goal. The company has an object-oriented model it believes will accomplish this goal and is currently testing it with partners.

COMMON USER INTERFACE. Unisys is close to completing a common user interface to its tools. This interface will be the Designer Workbench, a Windows-based tool for building application code using graphical aids. Designer Workbench has been available for a year, but not for every Unisys platform. Unisys plans to make it the primary development interface regardless of underlying language. So far, the company has delivered on this promise for The Mapper System and Ally. LINC will be next. This strategy will allow Unisys to slide new implementation technologies under a consistent interface. Unisys plans to enhance Designer Workbench over time to add new semantics to support this strategy.

Unisys's priorities in the evolution of ASD Framework promise that the company will have powerful integration facilities before its competitors. Getting the information model for the repository right is the hard problem in building a CASE framework. Building a mechanism for tools to exchange requests for service and responses is an easier problem to solve. But, without a common data model, tools can only send one another messages; they really can't effectively share information.

Does ASD Framework Meet the Four General Requirements?

We have found no development environment that meets all of the four requirements outlined in the previous section. Many object-oriented tools come close but falter on the issue of migration. Most procedural development environments fall short in every area. Unisys ASD Framework meets most of the requirements today and promises to keep providing improvements over time. Detailed discussions of each tool follow in other sections. The following is a summary:

Does ASD Framework Meet the Four General Requirements?

Linking Software Development to Business Requirements

ASD Framework is strong on linking software development to business requirements. Using Business Planning Workbench, Designer Workbench, and LINC II, it is possible to begin an application by defining high-level business requirements and then proceed right to code generation on LINC II-supported platforms.

Indeed, Unisys has a unique commitment to helping managers outline their strategic business imperatives in terms that software designers and developers can understand. Unisys has not yet built the "upstream" connection between code and design, but this is an obvious enhancement of future versions of its products.

Reduce the Cost of Development

With ASD Framework, developers have available to them all of the productivity advantages of high-level programming environments. Each of the ASD Framework tools offers developers greater power for each line of code than 3GLs do.

In addition, with LINC II, Unisys has begun to support reuse of software modules across applications. The support isn't complete, however. Unisys has no facility to catalog or exhaustively test modules. But the basis for reuse is there, and this is impressive.

Making Software That Can Be Changed

Unisys fares well on making software that can be changed. With LINC II, The Mapper System, and Ally, developers create software by specifying an application's functionality and then generating the code required to implement it. In the case of LINC II, the generated code is Cobol native to the target environment. In the case of Ally and The Mapper System, the code is the language of those environments. Because the application is defined at a high level, though, developers can change the application by changing the specifications and then regenerating the application.

Migrate from Current Environments to New Ones

ASD Framework is explicitly designed to support the migration from today's most commonly used development tools to new development technologies and techniques. Unisys plans to support this requirement by increasing the number of languages it can generate from its high-level specification languages. The Unisys Repository will be key to supporting this approach.

Is ASD Framework Really Open?

Is ASD Framework really open? To answer this question, Unisys must deal with two issues. First, will Unisys provide its own tools on non-Unisys platforms? Second, will Unisys allow integration of third-party development tools with ASD Framework, and, if so, how?

MULTIPLATFORM IMPLEMENTATIONS. Unisys has demonstrated its commitment to multiplatform ports of its development tools. The Mapper System is already available on Sun SPARCstations, OS/2, and Microsoft Windows. During November 1992, Unisys announced availability of The Mapper System for IBM's RS/6000 and Intel platforms under SCO Unix.

Unisys has also announced that it is porting LINC II to Unix System V Release 4, making it available on a wide variety of Unix platforms (not just the Unisys U Series).

THIRD-PARTY TOOLS INTEGRATION. Unisys believes the ultimate purpose of a repository-based CASE framework is to support tools from many vendors. A repository allows each tool to present an appropriate view of the same underlying information about software. As Unisys implements de facto and de jure API standards in ASD Framework, it will provide third-party vendors with the basis for integration with ASD Framework's underlying repository and inter-tool communications facilities.

However, integration via APIs is a future direction, not a current reality. For the present, Unisys itself plans to integrate popular tools with ASD Framework. It will rely on its own gateways between third-party tools and ASD Framework tools to accomplish this integration.

Unisys will select third-party tools for integration based on customer demand. For example, Unisys is interested in integrating PowerSoft Corporation's PowerBuilder with ASD Framework because of PowerBuilder's popularity with Unisys customers.

Business Planning Workbench

Strategic Planning Meets IT Planning

The first step in application development is an analysis of user requirements and business imperatives. These factors inevitably determine whether or not an application development project adds value to the business. Unlike most CASE frameworks, Unisys's ASD Framework includes a tool to aid in the process of requirements gathering and analysis.

Business Planning Workbench (BPW) is a tool to help ensure that a corporation's information technology investment plans are in tune with its strategic business goals. BPW is a specialized database for gathering and organizing business requirements information and then using this information to plot an effective and efficient information technology strategy.

In most corporations, business, or strategic, planning is segregated from technology planning. The result is predictable: Many information technology solutions do not support very well the attainment of strategic goals. The ideal approach is for strategic business planning and the design of information systems to proceed in lock step.

But how? A number of strategic consulting firms have bridged the gap by building information technology planning into their methodologies. However, users of these methodologies generate documents describing the corporate goals and the organizational and technology steps needed to achieve them. Planning documents do not meet the need to analyze and update strategic and information technology plans on an ongoing basis. This is the need BPW seeks to meet.

BPW has been available in a DOS version in Europe only. Beginning in 1993, Unisys is making available a new Windows version of the tool in addition to the older DOS version in Europe, along with associated consulting services. We expect BPW to be made available in the United States soon as well.

What Is Business Planning Workbench?

Unisys created BPW with Coopers & Lybrand's U.K. partnership. Coopers & Lybrand/U.K. needed a PC tool to automate its Summit-S strategic planning methodology. Unisys needed a tool within its ASD Framework that addressed business requirements analysis and planning—the first steps in successful software design.

What emerged from the partnership is a database tool that automates Summit-S out of the box but can also be easily tailored to implement any other strategic planning methodology. (Unisys is not wedded to Summit-S.) The primary function of BPW is to collect information from high-level executives and front-line managers about the strategic goals of the business, the distinct functions (procedures and operations), and the opportunities and barriers to success in a database.

This information typically is collected during interviews with executives and managers. The resulting database contains a model of the business. BPW can capture all information about all aspects of the business, bringing to light assumptions and priorities that might otherwise never have been made explicit. The model can be updated as needed to reflect new business conditions, or to make corrections. Managers can generate views of the model to run simulations without corrupting the integrity of the underlying database. BPW supports both table views and diagrammatic views of its information. (See Illustration 3.)

BPW Basic Mode of Operation

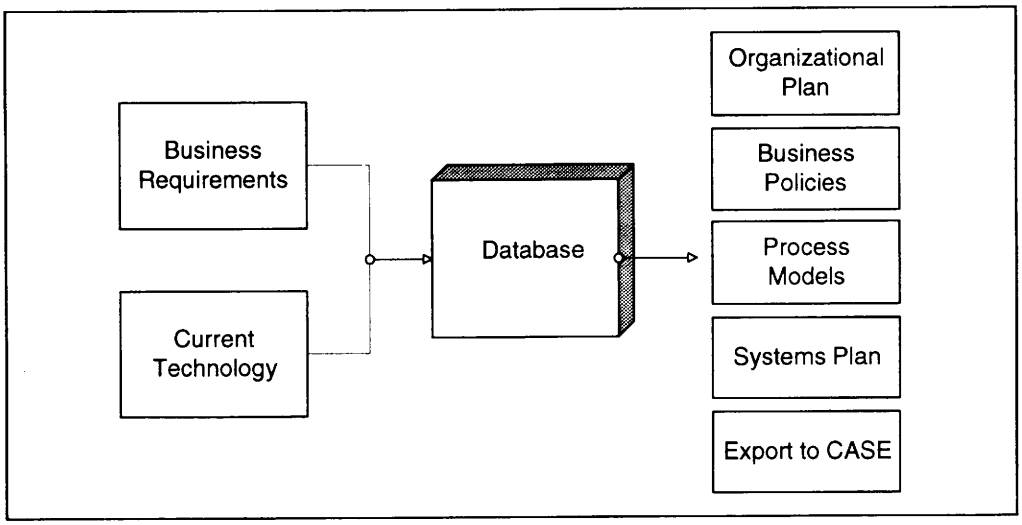

Illustration 3. The basic mode of operation for Business Planning Workbench is to input information about business goals and existing systems into a database, which can then be used to generate analytical reports. In addition, BPW can export a basic requirements specification to either Unisys' LINC II environment or the Systems Engineer design package from LBMS Incorporated.

BPW's Categories of Planning Information

Summit-S segments strategic and information systems planning into major categories:

- Business Strategy, which defines the high-level goals of the business
- Business Objectives, which define the basic goals of the business
- Opportunities, which define unaddressed customer needs or voids in markets
- Deliverables, which define products and other "output" to customers
- Functions, which define actions
- Organization, which defines the structure of the corporation
- Business Architecture, which defines the relationships among strategy goals, functions, deliverables, opportunities, and barriers
- Critical Success Factors, which define management's view of the most important factors in a successful business strategy
- Programs and Projects, which group Functions into applications-development units
- Goal Information Technology Architectures, which present technology architectures in a map-like format, with full definitions of every element

At each stage, managers define the names and nature of elements, from business goals to departments and divisions, along with their benefits and relationships to other elements. For example, if a business strategy is to automate the corporation's warehouses, the definition of that strategy would include expected benefits.

Analyzing the BPW Model

The BPW database stores information about each goal, subgoal, operation, procedure, and so forth—including the relationship of one element to another. For example, a strategic goal

might be to increase profitability. A subgoal for this goal might be to reduce overhead costs. A function might be the billing of customers. Users define the relationships among these items.

Relationships Support Analysis of Plans

The BPW database supports analysis of business strategies and plans in two major ways. The first is to explore the relationships between elements in the model of the business. For example, if the model identifies a Deliverable called Get Product Out On Time as being crucial to success, a user can easily see which Functions are required to achieve that Deliverable. This might lead to a decision to accelerate an application-development project designed to deliver those crucial functions.

The second way BPW supports analysis is by correlating information in very flexible ways. For example, managers might correlate organizational units by function to determine if there is any functional overlap between departments. Having collected this information, BPW allows managers to analyze the relationships between goals, strategies, and even systems. Managers can use BPW to create any number of scenarios to test their assumptions or try out different plans.

BPW users can generate a number of scenarios and reports from the underlying database. By focusing on the Functions defined in the database, managers can begin to create a rational approach to information systems planning. In addition, over time, BPW gives managers a way to continually track the effectiveness of their technology investments in meeting business goals.

Integration of BPW and Other ASD Framework Components

At the current stage of its development, BPW is available to users of Unisys's tools (and the CASE environments of other vendors as well) as the first step in describing a strategic plan that is connected to information technology plans. By focusing on Functions defined within BPW, managers can target their investments in information technology on high-payoff applications.

The ideal situation would be to push a button in BPW to generate a high-level functional specification for an application. However, BPW does not yet do this. What Unisys does support is importing a BPW functional description into either Unisys's LINC II environment or to Systems Engineer from LBMS Incorporated, which, in turn, generates a requirements specification for the application.

Users also cannot use LINC II specifications to populate a BPW database. To do so would support re-engineering projects nicely, and also ensure that, when inevitable changes are made to an application during coding, the impact of those changes on functional requirements would be immediately accessible to managers.

An obvious future direction for Unisys is to support greater integration with LINC II and its other tools.

Designer Workbench

Common ASD GUI: Tool for Development of Front Ends

Unisys's tool for building user environments is Designer Workbench, a Windows-based tool for creating application front ends. The product operates in a way that is consistent with other graphical tools on the market. This mode of operation is often called *spatial programming* because it allows developers to work by arranging graphical elements on a screen, rather than by writing lines of code. Designer Workbench's Forms Designer, the main component of the product, allows developers to design an on-screen form by painting the form on the screen and defining its fields in place.

In addition, Designer Workbench outputs fully functional Windows applications that will work alongside other Windows applications. In other words, Designer Workbench is not a closed environment.

Designer Workbench is Unisys's common high-end graphical environment for all of the ASD Framework tools. Designer Workbench is currently available for The Mapper System. Versions of Designer Workbench for other ASD Framework tools can be expected in the future.

Beyond Generic Graphical Front End Features

Although Designer Workbench in many ways typifies graphical front-end development tools, it also has some unique aspects. For example, it has features that permit development of touch-screen interfaces. Touch screens are a helpful interface for applications that allow executives to browse through data online (executive information systems). Since many executives don't type, touch screens are a vital alternative interface.

Designer Workbench also supports the full range of image-presentation, graphics, and font display features available under Microsoft Windows.

In addition, Designer Workbench also gives developers an important application-maintenance feature. Developers can store a master copy of the specification for their user screens on a host or server. When developers need to change screens to incorporate new features, satisfy user requests for changes, and so forth, they can update the master copy of the specification. The next time each user seeks to use the applications that use the changed screens, Designer Workbench will download the new specification and update the user's local copy of the screen to conform to it.

Development Priorities for Designer Workbench

Unisys has four key development priorities for Designer Workbench: support for more platforms, expansion of its functionality in host integration, the addition of a multiuser repository for the tool, and support for development of full client logic at the workstation.

First, Unisys wants to support Designer Workbench on a variety of platforms. Designer Workbench is currently a Windows tool, and Unisys is working on ports to other client platforms as well.

Second, Unisys sees a great opportunity in the use of Designer Workbench to modernize legacy applications with graphical front ends without requiring modifications to the underlying code. Unisys plans to add to Designer Workbench a "screen-scraping" facility that allows developers to easily map new graphical displays to old, character-based screens. Screen-scraping is a technique available today with the use of discrete tools, but Unisys plans to fold this function into its graphical development environment.

Third, Designer Workbench is a single-user tool in its current versions. It does not support coordinated work by teams of developers using the same repository information. In the corporate environments that Unisys is targeting, support for multiple developers is important. Unisys intends to provide multideveloper support by adding a multiuser repository to Designer Workbench in a future release.

Fourth, Unisys plans to gradually expand the extent to which of an application can be built using Designer Workbench. Today, Designer Workbench addresses mainly the construction of user interfaces—no mean feat. Over time, Unisys plans to add features that permit the construction of application logic through the high-level tool as well.

The Mapper System

The Mapper System: A 4GL Grows Up

The Mapper System is one of the original fourth-generation languages. During its long history, it has changed dramatically to meet new needs. This ability to change is a strength of the underlying language and architecture of The Mapper System.

The primary purpose of The Mapper System is to assist developers in creating rich end-user computing environments and applications. The Mapper System is a tool, then, for defining the "front-end" applications that users employ to access data stored in "back ends," such as LINC II production systems and relational databases. The Mapper System applications access and integrate a variety of data from disparate sources without requiring knowledge of the underlying access semantics. Having obtained data, Mapper applications can then also provide users with the means to model and manipulate those data.

The secondary purpose of The Mapper System is to provide sophisticated end users with an easy-to-use programming tool that allows them to build their own database query and reporting applications.

Developers (or sophisticated end users) can work with The Mapper System via two routes. First, they can write Mapper code. Second, they can use Designer Workbench for The Mapper System (based on Microsoft Windows) or The Mapper System's own Forms Designer. Both Designer Workbench and Forms Designer allow the developer to define the user interface for the application using high-level drawing spatial programming tools. Under the covers, both Designer Workbench and Forms Designer generate the Mapper code needed to implement the application.

Critical Success Factors for Front-End Development Tools

Not all development environments designed to create the end user's view of information systems are created equal. Many are special-purpose tools designed to build only one type of application. Others are restricted only to one platform. The best tools meet four criteria, which we call the critical success factors of a front-end tool. The Mapper System meets all of these criteria today. The four criteria are:

- Ability to create any and all types of front-end applications
- Ability to access to multiple databases and back-end systems
- Support for multimedia
- Support for multiple platforms

General Purpose Application-Creation Environment

The Mapper System is a general purpose development environment, not a tool aimed at creating only one type of user front end. Thus, developers can use The Mapper System to build decision-support applications in which the most important functions are modeling and simulation routines. However, they can just as easily use The Mapper System to create the online graphical "briefing books" associated with Executive Information Systems (EISs). In addition, the most recent versions of The Mapper System support creation of multimedia applications, including real-time video displays within more conventional forms-based applications.

Multidatabase Access

The first versions of The Mapper System were dependent on an internal database (like most 4GLs). However, Unisys has long since made The Mapper System a tool that is independent of its sources of data. The Mapper System retains it own database, but it can also access a number of other sources.

The key to this ability is The Mapper System's data-access features. The Mapper System has an interface to access LINC II data. It also has a generic interface for accessing relational databases, called The Mapper System Relational Interface (MRI). Using MRI, developers can define the data sets required by a user within The Mapper System's internal database. The

data required to fill these data sets may be acquired from a number of back-end databases. MRI, then, generates the query syntax required to obtain the data from each back-end data source.

Over time, Unisys will have to modify The Mapper System to add a "standard" interface for multidatabase access. This shouldn't be a problem for Unisys. The SQL Access Group (SAG), Microsoft, Borland International, and IBM are all defining different interfaces to multiple back-end data sources, each hoping to establish an industry standard. With MRI, Unisys has a solution to the problem of multidatabase access today and an architecture that will allow it to migrate to one or more "standard" interfaces in the future.

Multimedia Facilities

The original 4GLs were designed to make it easier for developers to code character-based user interfaces. Not all of these products have made the transition to graphical user interfaces, such as Microsoft Windows. The Mapper System has. The latest version of The Mapper System supports all of the media types available under Microsoft Windows and Unix, including voice and video. Developers face no dead ends when it comes to media support in The Mapper Systems.

Support for Multiple Platforms

The Mapper System has been available on Unisys's four major platforms for a long time. During late 1992, Unisys announced its availability on the IBM RS/6000 and Intel platforms under SCO Unix. Other ports will follow during 1993.

LINC II

LINC II Advanced Procedural Environment

LINC II is a procedural development environment that uses some object-oriented concepts to increase developer productivity and flexibility as well as to improve the maintainability of applications code. The object-oriented concepts LINC II uses include: modularity of code, creation of self-defining modules, separation of an application's specification from its implementation, and the close coupling of data with business rules.

Building Blocks: Ispecs and Events

Using LINC II, developers build applications by defining *information specifications* (ispecs) and the business rules of the application. Ispecs are the basic elements of a LINC II application. An ispec is not an object; it is an independent module. LINC II supports two basic ispecs, *standard ispecs* and *events*, and two variations of standard ispecs.

A LINC II application is really a matrix of ispecs and events, or transactions. The Designer Workbench permits a developer to easily view the ispecs and events in an application, including their interrelationships.

ISPECS. A standard ispec is a data structure with a single key. Standard ispecs define the fixed concepts in an application—concepts such as plant locations, vendors, and customers. The actual locations and vendors the user deals with may change, but the definition of what a vendor is and what role it plays in an application are not likely to change. The two variations on the standard ispec are *memo*, which has multiple keys, and *table*, which is a standard ispec that is automatically loaded into memory by the application to improve performance.

EVENTS. An event defines a transaction between standard ispecs. Thus, an event will define what a customer does in a business transaction but not the information that defines what a customer is.

BUSINESS RULES. An application's logic is expressed in *business rules*. Developers associate business rules with a particular ispec. For example, an ispec representing a data entry screen layout defines the data associated with that screen *and* the code blocks that define rules governing those data (such as validation routines). The result of this structure is to give mainte-

nance programmers easy access to all relevant information about the general structure of an application as well as the structure of each element and transaction within an application.

ISPEC DEFINITION. Developers can define ispecs either by using the Link Definition Language (LDL), a 4GL based on Cobol syntax, or by using Designer Workbench and its graphical tools. Developers define the application logic (the business rules) using LDL only. LINC II generates ispecs based on the developer's Designer Workbench specifications. (In addition, ispecs are modifiable using LDL).

All ispecs and events are stored in LINC II's repository. Access to them is governed subject to security permissions. For example, the software development manager can make certain ispecs read-only to the majority of developers, limiting full access to selected members of the development staff.

Why Ispecs Are Important

LINC II builds an application from an abstract specification. Thus, the design for an application is separate from that application's implementation in Cobol. This is an important design principle. Typically, in the creation of an application, developers begin with a design, usually sketched out on paper. When developers begin to implement their design for the application by writing code in a programming language, they usually make design changes. However, the original design specification is rarely, if ever, modified to reflect these changes. Thus, most organizations can only understand the design of their applications by reading low-level code to discern its inherent design. The result is code that is difficult to maintain and change.

Major Benefits of LINC II's Modular Approach

LINC II's approach to software development yields two important benefits that are not commonly available from other procedural software development environments. The first of these benefits is a reduced maintenance burden. The specification of each ispec *and its relationship to other ispecs* are always available to developers. This means that developers who are assigned to update an existing application can very quickly analyze that application's structure and purpose. The result is less time spent understanding the application code and more time spent on the maintenance job.

In addition, by working at the level of design specifications to build and modify applications, developers are assured of always maintaining a current design specification for an application. By exposing the design of an application, developers can more easily verify for business managers and users that their applications are accomplishing the business's objectives.

The second major benefit is the ability to change existing software without breaking it. By defining an application's elements and actions in abstract specifications, LINC II gives developers the opportunity to change and update their code with great flexibility. LINC II allows a developer to change any of the ispecs within an application (subject to access controls) to add new features or modify existing ones. For example, a developer can easily add a field to a data entry form ispec with the confidence that the change will not have unintended consequences for other applications that use the ispec. LINC II tracks the usage of ispecs in applications, and so it can propagate a change in one ispec to all uses of that ispec in all applications. The developer then must regenerate the application on the target platform to implement the change.

LINC Design Assistant also gives developers the ability to test a changed ispec before committing it to the data dictionary.

"Living" Design Specifications

LINC II is unique among CASE environments in its close integration of a tool for specifying applications and the low-level structures that implement those applications. Most CASE environments allow users to use design specification tools to outline an application's architecture. Very few tools, however, generate implementations from high-level specifications. LINC II

does. It generates the database schema, data structures, and programs required to implement the database, network, and the application functionality *on multiple native platforms.*

This integration yields a vital benefit for corporate developers. It allows developers to specify information systems using a high-level design tool—the LINC Design Assistant—and then maintain the resulting applications using the same high-level tool. The result is that design specifications in LINC II are living elements, not just throwaway documentation for the early stages of an application's development.

Native Environments, Not Runtimes

LINC II generates native implementations, not a runtime environment that is hosted on a target environment. The environments supported are Unisys A Series mainframes (Cobol, DMS II databases), Unisys Series 2200 mainframes (Cobol, the 2200's relational DBMS), and Unisys U Series (MicroFocus Cobol, Oracle Version 6 relational DBMSs). LINC II generates the required network code for only the A Series and Series 2200 environments, however.

Unisys also provides a facility for integrating new LINC II systems with preexisting Cobol programs.

LINC Design Assistant: Upper CASE Tool

A crucial step in an application development project is the creation of a high-level design for the application. It is at this stage that software architects lay out the functional modules, data structures, database schema, and user interface for the application. Unisys's LINC Design Assistant is a tool for aiding this process.

Developers define the elements and events within their applications by manipulating shapes and arrows on the screen and associating business rules (procedures) with those elements.

The LINC Design Assistant Process

Developers begin a Design Assistant session by defining the functional area in which they are working. Typical functional areas include accounting, human resources, and marketing. The next step is to define an activity within the functional area. For example, an activity within accounting would be budgeting. The next step is to define the data and events within that activity.

The next step is to define the business rules for the object. The rules are not created using LINC Design Assistant, but, rather, with the language of the underlying implementation environment—LINC II. The last step is to paint the screens associated with each object. This set of functions accelerates the creation of user interfaces.

LINC Design Assistant is an interpreted environment based on a single-user repository of information. Developers generate application specifications for code generation from the repository. However, developers can work within the LINC-interpreted environment *without* committing changes to the repository. This allows for prototyping and testing of new structures before they are committed to the repository for use. (Further, the LINC Interactive Test Environment—LITE—allows developers to test their modules interactively without generating code to implement them.)

Upper CASE Migration

LINC II allows users to import design specifications created using third-party tools. Unisys supports, through the LINC CASE Interface (LCI), importation of specification files from structured design tools from LBMS, Intersolv, IEW/ADW, and Gamma. LINC II automatically translates these specifications into its own format. Users can then use Unisys's Design Assistant to optimize these translations or to modify them to reflect changed business requirements.

Development Priorities for LINC II

Unisys's top priorities for LINC II are: support for third-party Unix platforms, multidatabase applications, compliance with the Distributed Transaction Processing interfaces defined by X/Open Company Limited, and an improved SQL implementation.

Ally—Unix-Oriented Development Environment

Multidatabase, Distributed OLTP Development

Like LINC II, Ally's purpose is to create transaction-oriented production applications. Ally, however, is designed as an environment for application creation and execution strictly in Unix-oriented distributed environments supporting one or more varieties of relational database products.

Developers can use Ally to create client/server transaction processing applications involving, for example, Oracle and Informix databases. The Ally runtime, when used in conjunction with Unix Systems Laboratories' Tuxedo transaction manager, supports the integration of data from both of those databases within a single application.

Ally is a high-level tool, not a third-generation programming language. Developers work by selecting commands and functions from menus in a character-based environment. Ally includes, however, a Pascal-like language to give developers total control over an application's logic and flow of control.

Complete Distributed Transaction Applications

Developers can use Ally to define complete transaction-processing applications in OLTP environments. Developers use the tool to define clients and servers, parameters for service requests, database schema (including data fields), and primary and foreign keys, as well as views, restrictions on fields, and so forth.

In addition, Ally generates calls to Tuxedo from specifications, shielding developers from Tuxedo's native C language interface. This feature is supported for both client and server requests. Having created an application specification, a developer then generates the applications code required for the application directly from Ally. The Ally runtime environment ensures applications portability across multiple versions of Unix.

Development Priorities for Ally

Unisys has two major development priorities for Ally: adding GUI features and providing Windows and Motif presentation drivers.

Ally has a character-based menuing user interface. For many users, this interface is outdated. Unisys will provide a Motif-based interface for Ally during 1993 to satisfy this user need. Additionally, in 1993, Ally will add support for XA-compliant databases with new Ally servers based on open SQL access.

Future Directions for ASD Framework

ASD Framework is based on proven technology and tools, not new concepts. A test of the product line will be its ability to evolve during the future to accommodate the latest trends in software development. Unisys has a plan to support this evolution. The major components of this plan are an enterprise-scale repository, expanded support for distributed computing, and support for object-oriented development within ASD Framework.

Cross-Tool, Enterprise-Scale Repository

At the heart of each of the ASD Framework tools today is a repository that stores information about the specifications for software, including relationships between components. Today, these repositories are discrete entities specific to each tool. During the next year, Unisys will begin to roll out new repository technology that can span all of the ASD Framework tools.

The first step in this roll-out will be support for information exchanges among the discrete repositories using proprietary import/export facilities. Next, Unisys will introduce a new repository based on its object-oriented model for all of its tools on the PC and Unix platforms. The switch to the new repository will be invisible to the user. The last step of the roll-out will be to offer the new repository on Unisys's mainframe platforms and to support a single logi-

cal view of data in repositories on multiple platforms. The schedule calls for the entire program to be completed by 1997. The distinguishing characteristics of the Unisys Repository are an object-oriented information model and distributed database technology.

The purposes of the Unisys Repository are to support data-sharing among tools and to support integration between tools. This approach would allow a high-level graphical tool to access information (objects) created using a different tool—a low-level language, for instance. The result is support for collaboration on software creation by users of different tools *and* support for reuse of existing components in new applications.

Object-Oriented Information Model

Object orientation is important when it comes to repositories because an object-oriented model is inherently flexible and extensible. Unisys has not taken the approach of defining the attributes of what it believes to be relevant tools and then calling that its repository information model. The model is general; it does not reflect the capabilities (and limitations) of existing tools. Rather, it represents a general model of all tools as they are today and might be tomorrow. Because it is object-oriented, the model is extensible to new capabilities. Unisys uses this base model to create views that are tailored to the particular storage models of various development tools. Individual tools vendors do not, then, have to modify their storage models to work with the Unisys Repository. They can just change the view of the model to suit their purposes. Unisys plans to publish its repository information model sometime during 1993.

Distributed, Multiplatform Database Foundation

Unisys is going with a distributed database underpinning for its repository as a way of supporting large-scale use of the technology. Monolithic repositories won't scale well as demands increase, but distributed structures will. Unisys plans to support distribution across mainframes, midrange Unix servers, and PCs, giving users flexibility in their choices of repository platforms. Unisys's distribution architecture utilizes both transaction control and replication to coordinate information stored in distributed databases.

Repository API Support

Unisys does not plan to publish its own APIs for repository services. It will use the IRDS 2 APIs coming out of the National Institutes of Standards and Technology (NIST) for the Unisys Repository. During 1993, the company plans to implement a version of IRDS 2 in advance of the formal adoption of that standard sometime in 1994 or 1995, and then make modifications as needed to comply with the final standard. Unisys also plans to implement the PCTE APIs for repository services being developed in Europe. Unisys will publish its repository model, which is the information third-party developers need to build views of the repository to fit their tools.

Finally, Unisys plans to implement support for IBM's AD/Cycle APIs to allow interconnection with IBM's repository. Unisys will also support import/export, using the CDIF format to interchange data with other repositories.

Object-Oriented Development and Other Advanced Technologies

The new repository is the key to Unisys's ability to gracefully support new technologies within ASD Framework. The repository is the place where all sorts of information models can be represented in a way that makes them accessible to multiple tools. Thus, the new repository technology will allow Unisys customers to continue to use high-level tools to specify applications, but *generate* the code to implement those applications in languages besides Cobol or C.

Unisys plans to support advanced languages, such as C++ and SmallTalk, only when it can be sure that, in doing so, it will be able to preserve the high-level view of development it has achieved with Designer Workbench. The same is true of any other new coding technologies. Unisys will treat them as low-level technology to be generated, not as replacements for its high-level specification tools.

Future Directions for ASD Framework

Distributed Applications Development

Unisys does not promote ASD Framework as a solution for building distributed applications, although it could. The ASD Framework tools currently allow distribution of function, as shown in Illustration 4.

Current Support for Distributed Applications

	Presentation	Database	Application Logic
Designer Workbench	•		
The Mapper System	•	•	•
LINC II	•	•	•
Ally	•	•	•

Illustration 4. The current versions of ASD Framework tools support a fair amount of distribution in applications implementation. This chart segments applications into three parts: presentation, database, and logic. It then identifies which of these elements each of the ASD Framework tools can distribute.

Unisys does support distributed applications through a variety of mechanisms, although none of them is very standard. The reason Unisys doesn't promote ASD Framework as an environment for creating distributed applications is that it hasn't yet implemented distributed computing standards into its base software. However, the company has already demonstrated its commitment to standards in its existing networking products. The company has very complete support for both the SNA and OSI protocol suites. Distributed computing standards are a natural evolution for these products.

The standards that Unisys expects to implement in future versions of its tools are:

- The ONC+ Transport Independent (TI) RPC, OSF RPC, and ISO ROSE RPC
- OSI's Remote Data Access (RDA) protocol
- X/Open's Distributed Transaction Processing (DTP) interfaces, including XA, TM-TM, and AP-TM
- The Object Management Group's Common Object Request Broker Architecture (CORBA) interface

The result of these efforts will be to add standard interfaces to tools that already support distributed applications.

Conclusions

Does ASD Framework Meet the Four General CASE Requirements?

We haven't found any development environments that meet all four of the requirements outlined in the introduction to this report, though many object-oriented tools come close. Object-oriented development provides a means to establish a common language to foster communications between business managers and software developers. Object-oriented tools also are explicitly designed to support reuse of code to improve productivity.

However, object-oriented tools don't yet support change in the software very well. Most object-oriented environments are built on source-level technology. When software has to be changed, these environments require recompilations to update all affected applications. Some vendors allow multiple applications to share the use of single objects, eliminating the need to recompile code. But, particularly in C++ products, recompilation is the general requirement.

Conclusions

In addition, few vendors have figured out how to allow users to refactor the underlying class structures for object-oriented applications.

Object-oriented development environments also falter on the issue of how today's corporate developers will migrate to new ways of designing and building applications. In general, object-oriented development environments require developers to learn new (and strange) languages. The vendors of object-oriented environments are just beginning to offer high-level graphical tools—analogous to 4GLs—that shield developers from the underlying languages.

Most of the CASE environments that use procedural technology fall short of meeting every criterion outlined above. They don't solve the problem of management-developer communication, they don't offer support for reuse, and they don't address change very well. Unisys ASD Framework is rooted in procedural technology, but ASD Framework is different in its approach from other CASE products. The Patricia Seybold Group sees ASD Framework as being much more in tune with meeting users' real needs in software development. Unisys has done a better job than the other procedural CASE vendors in meeting the four requirements that we believe are important in solving the corporate software crisis. Unisys ASD Framework doesn't completely satisfy all of them, but it does a good job on most. Moreover, Unisys's strategy promises to provide advances during 1993 and beyond.

Next month's *Open Information Systems* will address
PC and Unix Integration

For reprint information on articles appearing in this issue,
please contact Donald Baillargeon at (617) 742-5200, extension 117.

Open Systems: Analysis, Issues, & Opinions

FOCUS: STANDARDS

Common Open Software Environment: Not a Monopoly—Not a Consortium

Six Unix vendors have banded together to agree on a specification for a common desktop environment, with the hope of establishing a precedent for future efforts designed to accelerate the inclusion of de facto standards in open systems standards.

The three leading vendors of RISC-based Unix systems and workstations—Sun Microsystems, Hewlett-Packard, and IBM—have teamed with three suppliers of Unix software technology—USL, Univel, and Santa Cruz Operation—to define APIs for a common desktop environment. They have also agreed to adopt (sell) common networking products and to endorse standards in the areas of graphics, multimedia, object management, and systems management. If nothing else, these agreements signal an end to the Motif versus OpenLook war and the end to Sun's resistance to directly providing the Open Software Foundation's Distributed Computing Environment (OSF's DCE).

In Anticipation of Windows NT

The parties involved did not directly position their announcement as a defensive move against Microsoft and Windows NT. However, there is no doubt that the threat to Unix posed by Microsoft's new operating system, both on the desktop and on servers, helped overcome the host of past obstacles that have stood in the way of reaching agreement on a single consistent Unix desktop environment and programming interface. While this move is unlikely to immediately blunt the impact of Windows NT on the desktop, it lays the foundation for Unix to be in a better position to respond should Windows NT fall short of its pre-release hype.

The agreement also signals a recognition on the part of the Unix industry that it must present a consistent, unified environment for users and ISVs. It was refreshing to see this group finally overcome the inertia that has for so long characterized Unix vendors when they consider agreement on specifications and standards. If anything, the Common Open Software Environment (COSE) announcement signals a willingness to compete on the basis of who has the better implementation, not on who has the better specification.

Components of the Agreement

The six vendors agreed to four objectives:

- Develop a specification for a common desktop environment which will contain a consistent set of APIs that will eventually be supported on all the vendors' Unix systems.
- Adopt a common set of networking products, either developed internally, licensed from one another, or developed outside, and place them on their price lists. Included are ONC+ and DCE.
- Endorse specifications for standards and technologies in the areas of graphics, multimedia, and object technology.
- Establish a working group in systems management and administration to work toward a goal of defining a framework and associated tools to support interoperability and management of distributed systems. The vendors will also address the need for systems management standards.

In order to accomplish these objectives, the six vendors will have to work closely together over the next few years. Opening up their process to additional companies before their work is submitted to standards bodies is not a high priority. We expect, however, that Unix International members will be given the opportunity to have input, out of courtesy if nothing else.

Specification for the Desktop Environment

The COSE specification will provide for a common desktop environment that will have a consistent look and feel across all compliant implementations. There will be a single API for software developers to use in writing applications that can then be ported across all of the sup-

ported platforms. The goals are to have applications and data that are portable across the network, to support interoperability across platforms, and to have a user interface that is recognizable as Unix. The group plans to complete the specification for this environment by June 1993. There will be a developer's conference sponsored by all of the participants in the fall of 1993. The integration will be performed by the four developing partners, IBM, Hewlett-Packard, USL, and Sun, in the first half of 1994. A reference implementation which will be openly licensable will be produced by at least one of the partners.

Desktop Components. The desktop environment will be a specification that codifies the integration of elements from HP's Visual User Environment (VUE), IBM's Common User Access (CUA) model and Workplace Shell, features from Sun's OpenLook and DeskSet productivity tools, USL's SVR4.2 Desktop Manager, and OSF's Motif toolkit and window manager. SunSoft's ToolTalk interapplication communication mechanism will be included with extensions from HP for encapsulation. There is no telling at this point what this will look like. (It's too bad Apple didn't bite at the opportunity it had last year to have Macintosh be the common look and feel for Unix.)

Desktop Functionality. When the integration work is completed, the result will be a desktop environment that will support applications at the source-code level, support interoperability across platforms, and have a common look and feel across platforms. The idea is for users to be able to look at any of the vendors' desktops and say, "Oh, that's Unix." The common desktop will support:

- Electronic mail, group calendaring, text editing, audio, and other productivity tools
- Task and window management along with online help
- Procedural and object-oriented application integration with drag-and-drop, linking, embedding, and data interchange capabilities
- Dialogue and forms-building with icon editing
- Graphical object and file management
- Security features including start-up, login, locking, and authentication
- Installation automation and configuration at runtime

In developing the specification, the group will have to determine whether every platform should look identical, including whether each should have identical icons, and whether icons should have uniform location at startup, common default colors, and common styles.

Making It through X/Open. The completed specification will be submitted to X/Open to be considered for inclusion in X/Open's XPG. (See *Open Information Systems*, Vol. 8, No. 2, for more information on the XPG.) X/Open has a process, known as Fast Track, that allows completed specifications to be submitted to it for review. After technical review, the specification can be approved by a three-quarters majority vote of the Board. Or, as was the case of the OSF DCE Fast Track submission, some part of the specification may be returned for revision.

In addition to the COSE desktop environment specification, the Motif specification will be submitted by OSF for X/Open consideration, and Novell/Univel will submit the NetWare Unix client specification as well.

The Fast Track process could result in integration of these specifications into XPG in as little as three months after submission. However, there is no guarantee that X/Open will accept the COSE specification in its entirety or without modification. On the other hand, the COSE vendors could go ahead and implement their specification, ignoring any modifications coming out of X/Open. That is extremely unlikely, however.

One thing is certain: Even though the six vendors involved in the COSE specification represent over 70 percent of the Unix market, X/Open will not be railroaded into approving it. X/Open is bound by ISO 9000 quality assurance requirements to follow its specified procedures, and those procedures will be applied to COSE just as they would to any other specification.

Product Roll-Out. Specifications are meaningless without products. Most likely it will be mid-1994 by the time a specification has been approved by X/Open, compliance tests developed, and a branding program put into place. The reference technology will be released by all four development partners in 1994, which means that compliant products will most likely not be on the market until early 1995. Although vendors will claim compliance much earlier than that, true compliance cannot be claimed until a compliance test is passed, and that cannot happen until after a reference technology is completed. Microsoft will be preparing to release the object-oriented Cairo version of Windows NT in 1995, which is shaping up to be a very interesting year.

The Free Market Approach to Networking

No attempt was made by the six vendors to settle the issue of different approaches to distributed computing the way they resolved the Motif versus OpenLook battle. The COSE vendors will support efforts to reference, sell, and deliver ONC+, NetWare client, and DCE technology. In other words, none will put up roadblocks to keep customers from having supported products in these areas. Although these technologies are not part of the desktop environment specification, they do have published API specifications, making implementation straightforward. All six vendors made specific commitments to work to assure interoperability of each of their versions of these technologies.

The various vendors can be expected to make announcements on availability within the next few months. It will be important to see just how each prices and supports the various technologies. If customers are truly going to feel free to chose ONC+ or DCE for their distributed computing framework, then they have to feel that all the vendors will support either approach without prejudice and will price each technology comparably.

Commitment to Graphics Libraries

The commitment was made by each vendor to provide support for standard graphics libraries/specifications in its products, including Xlib/X for basic 2D pixel graphics, PEXlib/PEX for 2D and 3D vector graphics, and XIElib/XIE for advanced imaging. The companies will share validation and test suites for compliance with these libraries and ensure that their programming documentation is consistent as well.

This is another area where the software developers' task of ensuring that their applications run on the various vendors' platforms will be made easier and more cost effective.

Support for Multimedia Standards

The six companies agreed to work to define an infrastructure for multimedia called Distributed Multimedia Services (DMS) and multimedia access and collaboration tools for the user called the Desktop Integrated Media Environment (DIME). The specifications that are developed will be combined and submitted in response to the Interactive Multimedia Association's (IMA) Request for Technology (RFT) for Multimedia System Services. That RFT calls for a robust set of requirements to establish a foundation upon which multimedia application and title developers can rely for predictable and consistent results across a wide variety of platforms and networked environments.

The IMA expects to receive responses to its RFT by May 1, 1993, and to publish "Recommended Practices" by the third quarter of 1993.

More Than Meets the Eye in Object Management

Although the six vendors pledged to work together to accelerate the development and delivery of interoperable object-based technology, this area bears close watching because of the differences in approaches taken by the companies. They all expressed support for the Object Management Group (OMG) and committed to provide Common Object Request Broker Architecture-(CORBA) compliant implementations in future products. However, current CORBA-compliant products do not interoperate particularly well, so much work is needed in this area.

In addition, the companies pledged to establish common style guidelines for developers, to specify a common set of core object-related capabilities for object construction and development (and hopefully storage), and to work through standards organizations to develop testing and certification methodologies.

If the group is to meet the challenge of Microsoft's Cairo for object-oriented systems, it will have to step quickly. Cairo developer training is likely to begin in 1994, with product rolling out in the form of a second major release of Windows NT in late 1995. The gauntlet is thrown down.

Systems Management

The best the companies could do in the area of systems management is to agree to create a working group to define a framework and associated tools. The working group is expected to present a road map and detailed plans by the third quarter of 1993. The areas the working group will consider include:

- User and group management, including security
- Software installation and distributed management
- Software license management
- Storage management (backup/restore)
- Print spooling and management
- Distributed file system management

Considering the difficulty the OSF has had in moving ahead with Distributed Management Environment (DME) because of vendor bickering and the Not In-

vented Here (NIH) syndrome, this work group has no easy task. Among the vendors, there are at least three viable alternative approaches, any one of which could become a reasonable framework for systems management. Which will win out is too uncertain to warrant a prediction. What we can predict are some very lively working group meetings.

Where Does COSE Fit In?

The discussions that took place around this agreement have been going on for over a year. Some discussions were between different pairs of the six parties. Others involved more participants. It is safe to assume that Hewlett-Packard and USL were prime movers, since Unix is so key to both their strategies. The last entrant, believed by some to have agreed less than a week before the announcement, was Sun. After all, Scott McNealy, Sun's CEO, had been steadfast in his rejection of Motif and adherance to OpenLook as Sun's permanent direction for GUI.

It is unclear exactly how many of the various pieces of technology that the agreement purports to bring together will fit. For example, HP's VUE and IBM's Workplace Shell seem to be redundant, rather than complementary. It is also unclear which elements from each will be brought into the final specification. What is clear, however, is that something will come out of the process and whatever that is will be supported by the six vendors.

Specification Versus Technology. The work being done to develop a specification that will be unencumbered by licensing means that any technology can be used to implement the specification. It is not limited to SVR4.2, Solaris, or any of the other vendor's products. This means that vendors who are not direct participants in the effort are free to support the specification with their products. And, since the specification is not limited to Unix, non-Unix technologies could support the specification. Although we do not expect Microsoft to rush out and make Windows NT-compliant, there is nothing to prevent it from licensing another company to take the Windows NT technology and develop a compliant variant.

Neither is Digital Equipment excluded from supplying a compliant product, particularly since its current Unix offering supports many of the components included in the specification. However, Digital may want X/Open to consider the specifications of certain of Digital's technologies, like Application Control Services (ACS), for inclusion as part of the specification. Why Digital wasn't included is anybody's guess. Speculation and excuses range from excluding Digital because of its relationship with Microsoft to the idea that nobody at Digital could make a decision quickly enough to be included in the announcement. Of course, that doesn't account for why Digital wasn't included months ago. But then, we understand that Sun didn't buy into the deal until a week before the announcement. Of course, if Sun didn't buy in, there probably wouldn't even have been an announcement.

Market Impact

The desktop specification, when it is finalized and included in the XPG, will be a great boon to developers and, ultimately, to customers. Some estimate that ISVs will save a total of $1 billion in redundant development costs incurred in porting to/from OpenLook and Motif. Companies like Lotus, WordPerfect, and even Borland, now will have a much larger market to support Unix application development. However, this environment will not succeed in stemming the Windows NT tide unless it works better, offers more functionality, and is noticeably better than the Microsoft offering. Being consistent could also mean being consistently inferior.

—M. Goulde

UNIPLEX INTEGRATION SYSTEMS

Bringing Open Office Systems into the '90s

Switching from Office Focus to Mail-Enabled Focus

Uniplex is very clear in its mission: It plans to continue its market leadership in open systems with the introduction of modular software components that deliver mail-enabled applications for enterprise-wide computing. And this could be an achievable goal for Uniplex Integration Systems, the leading, and practically the only remaining viable, vendor of office systems for the Unix market. Uniplex has continued to make money while competitors like Quadratron and Data General (office applications—not hardware) disappeared, and companies like Applix refocused their product lines away from integrated suites of office applications.

But, though Uniplex had an impressive suite of products for the 1980s, the handwriting has been on the wall for some time now. The company has finally announced its new approach and products, first revealed as the Nouveau strategy in the September 1991 issue of *Unix in the Office* (September, 1991, Vol. 6, No. 9). It has taken all this time to announce the onGO product line, which is now shipping.

Before we discuss the onGO products, let's look at the existing Uniplex product line.

EXISTING SUITE OF OFFICE APPLICATIONS. Uniplex has been selling a suite of integrated office applications for almost a decade. The current products on the market include Uniplex Mail and Uniplex Business Software. The latter product consists of:

- Uniplex II+ (word processing, spreadsheet, relational database, business graphics, and a screen- and menu-builder)
- Advanced Office System (includes II+, electronic mail, time management—calendaring and scheduling, a personal organizer, card index, and report writer)
- Advanced Graphics System (advanced presentation graphics package and a presentation editor)
- Uniplex Datalink (menued SQL front end to the most popular relational databases on the market)
- Uniplex Windows (X Window-based shell)
- Uniplex DOS (support for DOS clients for word processing and spreadsheet)

SEPARATING OUT THE MAIL. The key difference in the bundling of the existing product line is that Uniplex Mail is now available separately from the office product. The mail application can work with any editor of choice, and it includes the calendar/scheduler. Uniplex Mail (sans office) has proven to be very popular on the SCO, RS/6000, and AViiON platforms. The current product is what might be called quasi-GUI—it has a GUI face, but reverts to character mode at all but the main-screen level. Uniplex plans to continue selling Uniplex Mail as its character-based offering to those installations that haven't yet migrated to GUI environments or for those still in transition.

ENTERING THE GUI WORLD WITH onGO. onGO is Uniplex's true GUI product. The onGo suite demonstrates a modular software component approach. onGO Mail does not feature the same format as Uniplex Mail, but the company is shipping a transparent mail gateway with both products to facilitate transparent mail routing in a mixed installation of the two products.

However, by next summer, onGO will offer a character-based client, allowing a single mail system throughout an organization. The advantage to this will be a single enterprise-wide directory service. Currently, Uniplex Mail requires its own directory, which is more cumbersome to manage than is onGO's directory service.

MODULARITY IS THE GOAL. Uniplex's goal is to provide a group of modular products that can be customized by each site to the way the people really work, supporting group and/or enterprise preferences (though not a tremendous amount of individual customization is supported). The company calls this ability to customize *workflow*.

MAIL-ENABLED WORKFLOW. It is interesting that Uniplex is touting its mail-enabled focus as the basis for workflow, rather than promoting a database-enabled workflow model. Uniplex was one of the first integrated office systems that recognized from the beginning the strategic value of including relational database management system (RDBMS) capabilities as part of a basic office suite, including its own RDBMS (an Informix look-alike), and eventually offered a menu-based interface into any popular RDBMS. Sure enough, all "workflow"-related information in onGO, such as information from the calendar, alarms, triggers, and the actual data being sent around, is stored in the RDBMS of choice at the customer site. The mail-enablement is how the information gets routed to users. Uniplex considers this a vital requirement for ad hoc workflows. And we agree. The most robust workflow products will take advantage of both the underlying mail systems and corporate databases.

The onGO Product Line

STRATEGIC CATEGORIES OF SOFTWARE. This brings us to onGO, a modular suite of software components that provide an open infrastructure for scalable enterprise groupware applications. Whew, what a mouthful!

Basically, the strategic directions—major categories—of onGO were articulated in our report on Nouveau in *Unix in the Office*, September 1991. Let me present them here again:

- Business Communications: X.400 mail with routing and directory management
- Document Preparation: a WYSIWYG compound document editor and architecture
- Document Management: management of compound documents and multilevel security
- Information Access: database links, DDE support, and intelligent filtering
- Software Integration Tools (formerly active integration management): an end-user-oriented macro programming language

A LAYERED PHILOSOPHY. Uniplex offers a layered diagram of the office framework for the '90s that demonstrates the philosophy of the onGO product line. (See Illustration A.)

Underlying Platforms. The bottom layer in this framework is the platform. Right now, only Unix is supported. The most likely candidate for the next platform release will be NT. Uniplex includes OS/2 and Taligent on the chart, but the company, like everyone else, is waiting until these become market requirements.

Enterprise-Wide Messaging. The next layer up is an enterprise-wide messaging system, which Uniplex believes is a strategic, fundamental layer for any enterprise automation solution. In onGO, this is a standards-based mail layer with a native X.400 architecture. The mail system can be executed over any other transport, such as TCP/IP, not just over native X.400. This is a boon for internetwork communications because it doesn't require the overhead of X.400 mail.

Mail-Enabled Applications. Mail-enabled applications are next. Currently, the Uniplex-provided applications include some filtering, routing, calendaring, events-triggering, and alarms. These applications can be considered services to the mail system and to personal productivity and/or custom software. Uniplex will provide published APIs to these applications for VARs, systems integrators, and customers. Uniplex is planning to provide a higher-level graphical toolkit for business users building workflow applications at this layer.

Personal Productivity Tools. Productivity tools should be able to take advantage of mail-enabled applications (or services). In the Uniplex framework, users are free to choose whichever word processor, spreadsheet, etc., that they want to use—though onGO does include a compound document editor and graphics tools.

Desktop User Interface. The top level of the office framework reflects the concept of allowing users a choice of desktop interface. Currently, X-Window, native Motif, and MS Windows are supported. Uniplex is looking at supporting OpenLook, character Unix, Presentation Manager, and, possibly, the Mac.

TECHNOLOGY FEATURES. The following represent the primary technology features of the onGO product:

- *Support for Standards.* Currently, onGO supports X.400, X/Open XPG, POSIX, APP, ANSI C, Motif, and Windows. Future support and connectivity is planned for X.500, XAPIA, VIM, and MAPI. (Uniplex will adopt the XAPIA version of MAPI. If that turns out to be insufficient for customer demands, the company plans to provide a facility that will allow MAPI clients to hook into onGO mail services. The company does point out that, out of the box, onGO will have X.400 connectivity and can communicate with MAPI via the protocol.)

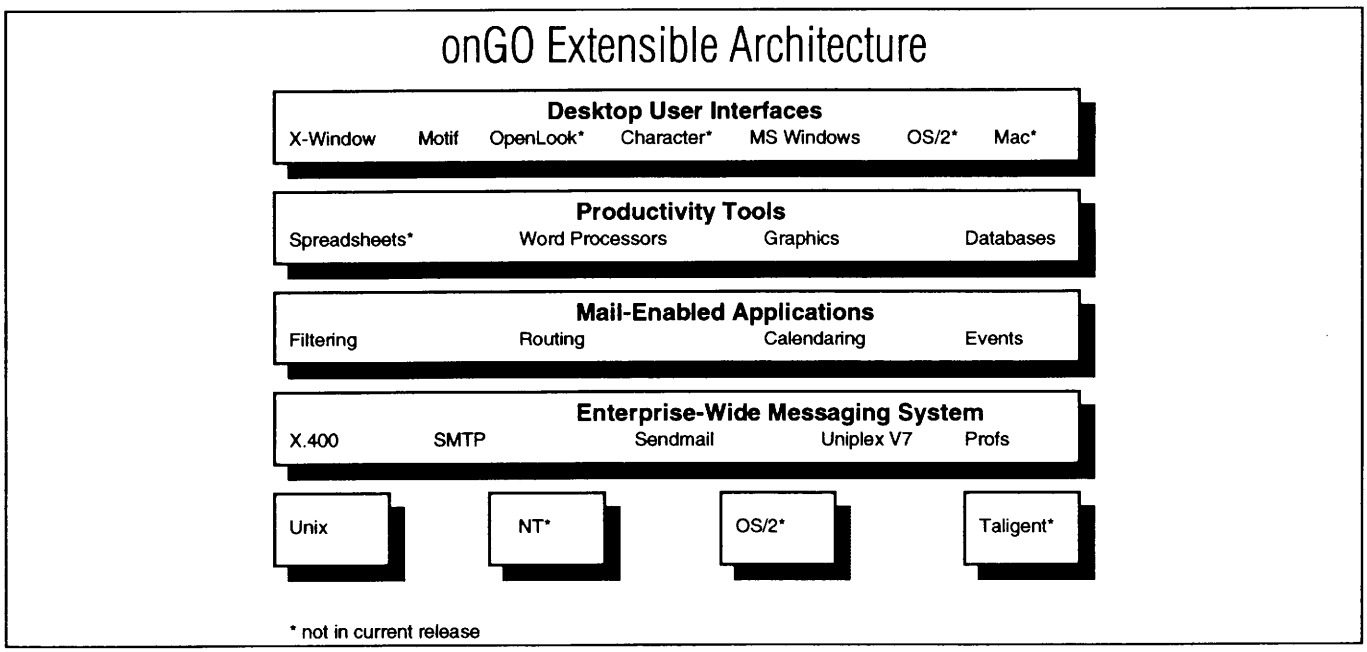

Illustration A. This diagram demonstrates the future directions of onGO. The current offering just begins to address the open environment that Uniplex envisions.

- *Inherent Scalability.* The design of the onGO mail engine will accommodate upward and downward scalability. You can, for example, run SMTP with the mail engine; you don't have to use the X.400 MTA to move mail around, which isn't practical for small groups.

- *Heterogeneous Computing Environment.* Right now, heterogeneity means X-Window, Motif, and MS Windows desktops and multiple Unix servers. At first ship this spring, onGO will run on the IBM RS/6000 and on the DG AViiON platform. This summer, the company will add support for the HP 9000, SCO Open Desktop/Unix, and Sun Solaris. Fall will see the addition of support for the DEC Alpha/OSF/1 and UnixWare (Univel). Supported clients at first ship are native X-Window and Motif. MS Windows support will not come until this summer, as will character Unix support. Support for NT, OS/2, and Mac desktops has not yet been scheduled.

- *Native Client/Server Architecture.*

- *Incorporated Object Technologies.* Besides writing a goodly portion of onGO in C++, Uniplex is promoting the object paradigm in its user interface and within its applications. The company is working on a relationship with Hyperdesk to leverage DCE. Though no specific time frame was given, the company states that this is definitely on the agenda for a subsequent release.

Product Suites

Uniplex is marketing two different onGO product suites, both of which were scheduled to ship at the end of March 1993:

- onGO office
- onGO Write/Paint/Draw

onGO OFFICE. The following are the fundamental components of onGO Office:

Mail Manager:

- Native X.400 universal mail
- Filtering
- Routing
- Directory management

Directory Manager:

- Administrative tools and enterprise-wide address book
- Public and local directories and distribution lists
- Soundex search
- Database resident

Calendar Manager:

- Schedules over LAN and WAN environments
- Resource scheduling
- Comprehensive scheduling
- Alarms and events management

Universal Office Server:

- Management and administration of office communications

- Mail administration, routing messages, monitoring traffic—can handle upwards of 100,000 people on an enterprise-wide network

- Directory synchronization

- Calendar data database resident

onGO Office is priced at $70 per user for 100 users.

ONGO WRITE/PAINT/DRAW. The second onGO suite includes a comprehensive document publishing system that combines WYSIWYG word processing with page layout facilities, a full-featured paint package, and a full-featured draw package. Write/Paint/Draw is available separately or with onGO. This product will initially be available only on the Unix platforms under OSF/Motif.

One of the advantages of this product is its multieditor architecture, which allows you to add additional multimedia editors (image, voice, math, etc.).

onGO Write/Paint/Draw is priced at $298 per user for 100 users.

Future Directions for onGO

Though there are currently only the two onGO modules, more are forthcoming. Uniplex plans to include the following components in future releases of the products:

- *Document Management.* Uniplex is working in partnership with an established industry player (whose name has not yet been revealed) on integrating the document management facility transparently into Uniplex's core technology.

- *Information Access.* Uniplex plans to add additional database links, OLE support, and active data filtering—for example, the ability to connect to external data sources and create rules and triggers based on data events. This is how the company plans to inte-

grate workflow capabilities into onGO. Uniplex's philosophy of workflow is to provide people with tools that mirror the way they work. The company has an established history of building tightly integrated applications; in the new model, the applications aren't originally built to be tightly integrated, but Uniplex will provide the glue and data flow within and throughout its own and third-party applications to ensure that applications can share data, rules, filters, etc.

- *Toolkits.* As mentioned earlier, the company plans to provide graphically-oriented programming tools, APIs, and a cross-application macro facility for the onGO environment. Many APIs are already available, but these are low-level, programming APIs. Uniplex wants to offer higher-level tools that are easier to use, aimed at the business user who wants to develop applications.

Conclusions

ADVANTAGE IN THE ENTERPRISE MARKET. The difference between the onGO strategy and the strategies of the traditional desktop office applications providers (Microsoft, Lotus, etc.) is full commitment to Unix clients and servers and to character clients as equal citizens to MS Windows clients. Uniplex's experience in the Unix market gives the company an advantage: It doesn't have to learn how to scale up from the desktop because it already knows. However, at issue is how well does Uniplex know the way to provide the services at the desktop level? The company is used to dealing with workgroups of up to thousands of users—something the desktop competitors are still ramping up to do. But Uniplex has no history in supporting desktop-level users, who require a different style of service and support than do traditional Unix system administrators.

A MODULAR APPROACH TO OFFICE. Uniplex is actually still addressing the same areas that it addressed with the office products: providing database access, mail, scheduling, word processing, spreadsheet, etc. But now it is providing these pieces in a modular fashion—you don't need to use it all or buy it all from Uniplex. There is also a change in focus from providing productivity tools to providing an environment and enabling services. onGO is newly written in a combination of C and C++ to take advantage of object technologies. It also assumes using personal applications of choice and multiple clients of choice with complete transparency from one to the other. In the Unix marketplace, there really aren't any other competitors that are still offering this breadth of service for workgroup computing. WordPerfect is the closest to providing this level of workgroup support, but that company has the opposite problem from Uniplex. WordPerfect must learn to scale up with service and support while Uniplex learns to scale down to the desktop. — *R. Marshak*

FOCUS: VENDOR STRATEGIES

Siemens-Nixdorf: Is Opening up Enough?

Germany's Siemens-Nixdorf Informationssysteme AG (SNI) has launched a spring marketing push that highlights new products and new strategies designed to strengthen the company's position in the open systems market. From "opening" its proprietary operating system to pairing up with Pyramid Computer for database and transaction processing solutions, SNI's new offerings will allow its product package to look stronger and more coherent to open systems customers. But it's a two-edged sword, and, for SNI, as with all vendors moving toward a standards-based, fully interoperable world, the imperative is to go beyond "me-too" products and come up with unique qualifications that allow customers to distinguish one commodity-based offering from another. In that regard, SNI still has some work to do.

While SNI does have strong offerings for vertical markets like banking and government, its most promising value-added offering has been nipped in the bud. Plans for a full-service, soup-to-nuts systems integration offshoot (See *Open Information Systems,* Vol. 7, No. 11) have been greatly scaled back, with SNI preferring not to alienate existing and prospective integration partners. It's an unfortunate retreat from a promising market that is becoming synonymous with open systems.

Opening Proprietary Systems

On the product front, SNI is at least putting in place a convincing open systems strategy. One of the highlights of the spring drive is the plan to open up SNI's proprietary BS2000 operating environment to the open systems world. But all in good time; SNI is phasing compatibility in slowly, with POSIX and XPG4 compliance not scheduled until sometime in 1995.

The first step is to make BS2000 look more open on the front end and stretch lower-level interoperability and file-sharing capabilities on the back end. Much of this capability is now in place—support for NFS has been around for a while—and SNI announced recently a two-product package, called FHS Doors and Dialog Builder, that puts a widget-based graphical interface on top of

existing BS2000 applications. The redesigned front end can then be run on Windows, Motif, and generic Unix boxes.

Step two—sometime in 1994—is to add support for OSF's Distributed Computing Environment in BS2000, allowing a mixed Sinix/BS2000 environment to distribute data and processing between the two systems. And step three, due in 1995, is to bring BS2000 into compliance with POSIX and XPG4.

The Pyramid Connection

While SNI is extending BS2000 toward being an open system, it's also putting its efforts into that slice of the Unix market that has made the deepest inroads into the proprietary mainframe market: symmetrical multiprocessing Unix systems. And, for high-end SMP, SNI has recently signaled a much closer relationship with Pyramid, whose technology already figures prominently in SNI's RM600 database machine.

The relationship—based on a 7 percent holding and a board seat brought over from the Nixdorf acquisition—will get tighter as SNI takes over sales, marketing and support for Pyramid systems in Germany this spring. And the two have promised closer technological cooperation on SMP systems as well. All this in spite of—or perhaps because of—the fact that ICL has a worldwide reseller agreement with Pyramid, and Olivetti touts the Pyramid systems it sells as the Olivetti database system of choice.

Why is SNI paying all this attention to "open" BS2000 and Pyramid? The simple answer is that the company is reading the market well. Mainframes still have a role to play among SNI's customers, many of whom might be convinced to use an existing environment for the move to open systems if the technology were right. SNI claims it has been very successful in hanging on to proprietary customers—as well as customers needing to upgrade to a Unix System V.4 environment—when they move to open systems and/or Unix. "Open" BS2000 will help the company keep these mainframes installed and in use at existing customer sites.

Including Mainframes in Open Systems Direction

SNI also has serious plans to keep its mainframes within the scope of its Open Systems Direction distributed computing architecture, playing a new role as the "superserver" for its client/server Unix and PC systems. Using the mainframe in this capacity is not new—IBM has similar plans for MVS as a superserver for RS/6000 systems— but it is significant. Despite the promise of mainframe-like Unix systems such as Pyramid, the ability of BS2000 to handle large numbers of users, very large data files, and very high-speed transactions still gives it a technological leg-up, though at a cost.

And this new role for the BS2000 does little to diminish SNI's emphasis on the Pyramid line. Pyramid, along with the Tuxedo transaction processing monitor and databases like Oracle7, is stealing a lot of mind share when it comes to transaction processing and Unix. While still fresh, this TP market is heating up in Europe, as the close embrace of fellow Europeans ICL Ltd. and C. Ing. Olivetti & Co. SpA with Pyramid testifies. SNI does well to try to leverage its unique relationship with Pyramid against its rivals.

Overall, the key to success in the market lies not in doing what everyone else is doing, but in doing much more. For now, SNI seems to be concentrating its efforts on reaching existing customers with migration and compatibility strategies and touting value-add in terms of vertical market strategies and office automation solutions that are similar to those sold by its competitors. More and more aspects of SNI's hardware market—PCs, Sinix, and even Pyramid servers—are becoming commodities, and its Open Systems Direction, while exemplary, is hardly unique in itself.

Abandoning Systems Integration to the Competition

On the systems integration front, SNI may be making a major mistake, despite vice chairman Horst Nasko's assertion that if the electric power generator division of parent company Siemens AG doesn't sell power, he shouldn't sell systems integration. The reply to the former is clearly government regulation, not market opportunity. As to the latter, it's a question of shortsightedness. Over in France, Groupe Bull, already the fourth largest integrator in Europe, is adding a facilities management group to its impressive service offerings. IBM Europe is also struggling to get its integration service revenues up, and ICL is making moves in this regard as well. SNI seems to stand alone in restricting its once broad integration ambitions to a pre-sales consulting activity.

SNI's present course should stand the company well in its bid to become profitable after cutting its losses in half last year to 500 million DM, or more than $300 million. But the answer to the question of what vehicles SNI will use to move to a forward position in the market is still up in the air. *—J. Greenbaum*

