Patricia Seybold Group

Editor-in-Chief
Michael A. Goulde

INSIDE

OPEN INFORMATION SYSTEMS

Guide to Unix and Other Open Systems

Vol. 8, No. 7 • ISSN: 1058-4161 • July 1993

SAP's R/3 Software Suite

Architecturing Open, Integrated Software for Distributed Enterprises

By Joshua M. Greenbaum

IN BRIEF: SAP is an application software company based in Germany that has made the migration from mainframe to open systems. A dominant supplier in its native country, SAP hopes to achieve a strong position worldwide with R/3, its client/server suite of applications. Based on a flexible, tiered architecture, R/3 illustrates many of the challenges, as well as some solutions, facing companies migrating from host-based architectures to distributed architectures based on standards. The complexity of R/3 also illustrates both the value of and the pitfalls in buying an integrated suite of applications from a single vendor.

Report begins on page 3.

Information Tools

A Paradox-Solving Paradigm

THE STANDARD PARADIGM for the use of information technology (IT) in business is a department of technology experts and managers who translate business information requirements into software code to be run on company-owned computers. These experts investigate and make decisions about which hardware, software, and networking products the company should purchase or lease. They have knowledge that is broad and deep regarding the technology that comprises products under consideration.

A significant portion of IT budgets goes into developing and maintaining internal information technology expertise, even though it is the information, not the technology, that is relevant to the business. Many IT departments function more like internal systems integrators than they do like departments of a pharmaceutical company, an automobile manufacturer, or a brokerage.

This paradigm grew out of the historic complexity of the technology and a lack of standards that made it difficult to assemble complete solutions from components supplied by multiple vendors. Single-vendor solutions would be simpler, perhaps, but no single vendor can consistently provide all the solutions a customer requires.

User demand for open systems has been partly a plea to be freed of the requirement of being in the computer technology business. Open systems promise to make technology less of a concern for users. Open systems—which focus on products that implement commonly agreed upon specifications—enable users to shop by brand name, not by ingredient.

But open systems don't go far enough in distancing users from the complexity and expense of technology and its integration. They do not provide a sufficiently abstract view of information technology, and they still require extensive programming in arcane languages and knowledge of underlying formats and protocols in order to build large distributed systems. Open systems are "right" in that they provide a choice of products based on standards, but the products that deliver open systems aren't the right ones yet.

What users need is a new set of tools—we'll call them "information tools"—that allow them to build customized business applications without having to know about the underlying technology. Information tools aren't compilers or 4GLs, although they might be produced by them. 3GLs and 4GLs might be considered to be analogous to machine tools—tools used to build the tools that users actually employ in their work. The key to information tools is not new technology, but a new way of thinking about how vendors should be packaging their products. Some products should be sold as strategic tools and some as a commodity infrastructure, invisible to users.

Some visual programming tools are headed in the right direction, but they are limited in scope, remain dependent on knowledge of underlying protocols, and still require "Wizards" to manage the systems on which the applications they develop run. Even more important, the developers of these tools haven't changed the way they think about the role the tools should play. Most see their visual programming tools as extensions of traditional programming methodologies.

Information tools have to allow companies to essentially get out of the computer business, not by outsourcing, but by focusing on information, not technology.

The objective is to push the responsibility on the vendors to hide the technological complexity of their products from the users of those products. Things like system and network administration and management have to be made push-button simple. The industry is headed in the right direction, with GUIs and visual programming, but the basic paradigm shift has not yet been made. Vendors have to understand that they are selling low-maintenance information appliances, not high-maintenance works of technological elegance.

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
STANLEY H. DOLBERG
MITCHELL I. KRAMER
DAVID S. MARSHAK
RONNI T. MARSHAK
JOHN R. RYMER
ANDREW D. WOLFE, JR.

Copy Editors
ALBERT C. D'AMATO
MIRIAM F. D'AMATO

Art Director
LAURINDA P. O'CONNOR

Sales Director
PHYLLIS GUILIANO

Circulation Manager
DEBORAH A. HAY

Customer Relations 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.

SAP's R/3 Software Suite

Architecturing Open, Integrated Software for Distributed Enterprises

The great dynamic of the information technology market in recent years has been the switch from proprietary to open systems. Many vendors have stumbled in the attempt and few software companies have made the switch as thoroughly or convincingly as SAP AG, a highly successful mainframe software company based in Waldorf, Germany, that turned in over 800 million deutsche marks, some $532 million, in revenues last year.

Those revenues were earned primarily from sales of SAP's R/2 product, an integrated software suite that runs in traditional mainframe environments. But, starting in 1992 in Germany and at the beginning of 1993 in the United States, SAP has begun marketing the open systems version of R/2, dubbed R/3.

Since that time, SAP has sold over 250 R/3 systems, not a bad start for a product that has fewer than half its software components currently available. But, as SAP rolls out the remaining portion of its giant open systems product through the end of 1994, the expectations are that R/3 will prove to be quite successful in both the European and U.S. markets.

To achieve that success, SAP still has a long row to hoe: The company must release its remaining software modules in a timely fashion and replace its unwieldy, dumb-terminal interface with one worthy of the graphical user interfaces R/3 hopes to serve. Also, SAP must rely for its future success on a large number of third-party services and product providers, managing both the relationships and the product support that such a widespread offering entails. It won't be easy, but then, building the complex and thorough architecture described here hasn't been easy either.

The Essential R/3

When R/3 is available in its entirety sometime next year, SAP will have one of the most extensive collections of off-the-shelf Unix and open systems software on the market. There are over seven million lines of code in the R/3 system, excluding the database, user interface, and other elements external to the R/3 code base. And within these lines of code lie literally dozens of individual applications, development tools, and connectivity products.

Applications Cover a Spectrum of Operations

The applications are designed to automate virtually every part of a medium- to large-sized business, with an emphasis on the manufacturing and industrial sectors. Accounting and finance, production planning, materials and plant management, human resources and office automation are just some of the functions covered by R/3. Support for third-party PC productivity software is also built into the R/3 environment.

Behind these applications lies a large body of software development code, including SAP's proprietary ABAP/4 fourth-generation language (4GL), as well as systems management software that allow for a high degree of customization and tuning for particular business requirements.

The Essential R/3

Highly Integrated, Yet Open Environment

But R/3 is more than just a collection of software applications, systems software, and development tools. It is also a highly integrated environment that fully exploits the philosophy of distributed, client/server computing. R/3 makes use of an impressively orthodox open systems approach, all the more impressive because of SAP's mainframe roots. SAP has learned to make extensive use of de facto and de jure standards for interface, database, and communications functions within R/3. The result is that the company has produced a scalable, portable system that allows for considerable sharing of data between software modules as well as with the outside world using a well-documented series of internal and external communications protocols.

SAP has left little reason to doubt R/3's openness: To date, the software supports six Unix operating systems, six proprietary environments, three relational databases, and three major user interfaces, as well as a host of de facto and de jure communications and interface standards.

Repository-Based

It is important for the developer or integrator that R/3 also makes extensive use of a data repository that allows for consistency and control across this gigantic environment. All data elements, with their descriptions and functionality, are contained in the Data Dictionary, as well as help texts, screen descriptions, and mappings to the relational database that acts as a physical and logical storage mechanism for R/3.

Partnerships Are Key

Backing up this plethora of technology are R/3's partners. These partnerships—with systems integrators, management consultants, third-party software developers, and hardware vendors—exist in order to service R/3 users who wish to push the use of R/3 to its limits. The complete product suite is extremely complex and highly functional, and, in its full configuration, R/3 can require significant resources in addition to the software, consulting, and training services that SAP provides.

SAP likes to point out that R/3 has also been designed to be installed and configured with relatively little effort, and in SAP's current R/3 customer list are a number of users who are effectively using a "turnkey" version of R/3 that has required little or no custom development or integration.

But a full-blown R/3 installation can also become an extremely complex project, involving a complete re-engineering of business procedures and the installation of dozens of applications supported by dozens of individual software tools. New products and skill sets must also be brought into these user organizations. In its full manifestation, R/3 runs in highly distributed, heterogeneous environments and can require the resources of third-party relational databases and their associated tools as well.

This complexity brings with it the need for extraordinary consulting and development resources, and, therefore, R/3 stands at the top of a large food chain of third-party partners. The success of R/3 will, in turn, have an extraordinary impact on the ancillary products and services that surround R/3. And those products and services in their own right will have an impact on how R/3 is diffused into the larger open systems market.

Basic Architecture and Components

The basic architecture of R/3 is contained in the BASIS system (no relationship to the Batelle/IDI product), which comprises the system software and services that underlie the R/3 applications. BASIS can best be understood by looking at its different software layers, from the applications on the outside to the data dictionary at the center.

The most important outer layer of the R/3 world for users is the applications themselves, and, for the sake of clarity here, we will describe R/3 as though all the applications due to

be part of the R/3 family were currently available. Their actual availability will be staggered over the next 18 months, as noted in Table 1.

Availability of R/3 Modules

Version 1.0 Currently Available	Version 2.0 Available August 1993	Release 2.1 Available January 1994
BASIS Component, Financial Accounting, Product Planning, Sales and Distribution, Office and Communication, Controlling, Fixed Assets Management, Materials Management, Human Resources.	New modules include: Purchasing, Inventory Management, Warehouse Management, Material Requirements Planning, Sales, Billing, Computer-Aided Selling, General Ledger, Cost Center Accounting, Order and Project Accounting, Profitability/Market Segment Analysis, Assets Accounting.	New modules include: Legal Consolidation, Funds Management, Activity Types and Processes, Order and Project Accounting, Product Cost Accounting, Profitability/Market Segment Settlement, Project Center Accounting, Investment Controlling, Assets Accounting, Technical Asset Management, Costing, Work Order Cycle, Preventive Maintenance, Quality Management, Basic Project System Data, Planning and Forecast, Project Integration, Information System Projects, Purchasing, Inventory Management, Sales, Basic Data in Production Planning, Master Production Schedule, Capacity Requirements Planning, Production Activity Control.

Table 1. The roll-out of the full suite of R/3 applications will occur over three releases. Full availability will be achieved by the beginning of 1994. Customers select those applications they require and add additional applications at will.

In addition to the R/3 applications, SAP supplies a host of system administration tools, data query tools, and installation and development environments that also occupy this outer layer of technology.

ABAP/4 Modules

The layer below R/3's applications and services is made up of collections of software routines or modules written in ABAP/4, SAP's fourth-generation language. The ABAP/4 modules control the underlying functions of the applications, from database and data dictionary access to the processing of screens and reports. ABAP/4 is an interpreted language very much like a structured 4GL, and its programs are interpreted at runtime.

The vast majority of the modules used in a given installation will have come with R/3 itself, though users can develop their own using ABAP/4 and associated tools such as the Enterprise Data Model discussed below.

Modules are grouped in module pools according to their relative functionality, with individual applications and their processes linked to a given module pool or pools. Internally, running an application then becomes a matter of sequentially stepping through a series of modules contained in one or more pools. An application may not use every module in a given pool, and it may access modules in several pools. In addition, a number of

different transactions can access the same module pool. Regardless, applications derive their functionality by effectively traversing a sequence of individual modules according to the specific functionality required.

The collections of modules that function as screens and report writers (as opposed to those that supply the underlying logic to the application) fill the role of shells or templates for the data underlying the applications. These data enter the modules and, hence, the applications through one of six mechanisms:

- User keyboard input
- CPI-C protocol for program-to-program communications
- SAP SQL, a set of SQL-like extensions to ABAP/4, for most general database needs
- Native SQL when dealing with a specific relational database and its unique functions or capabilities
- Remote Function Calls—a form of RPC—for data interchange between heterogeneous systems
- Electronic Data Interchange (EDI)

These mechanisms will all be discussed in greater detail below.

Relational Databases and R/3

The relational database component is the most important of these data sources, and it constitutes the next key layer of the R/3 environment. It is in the database that the applications data—for accounting, personnel, parts, factory management, and the like—are stored. And, in turn, an application's functionality is dependent largely on the speed and accuracy of the database, particularly in the distributed, client/server environments in which R/3 excels.

SAP has taken an open approach to the relational database component—the only major software element other than operating systems and communications that SAP does not supply itself. R/3 plans to support three different commercial databases: Oracle, Informix, and Software AG's (Reston, Virginia) Entire SQL/DB, although, at present, Oracle is the only database used with R/3.

Software AG's database, which is due out for R/3 later this year, is expected to have a most favored database status, reflective not only of the product's low cost and basic functionality but also of the close working relationship between SAP and fellow German software vendor Software AG. For the user who simply requires a runtime database to go with R/3, that database will most likely be Entire SQL/DB. If, however, the user has a compelling need for a particular database feature—such as the heterogeneity of the Oracle environment—that choice can be made without prejudice. SAP has pledged to support as open a database strategy as possible.

R/3 Data Dictionary

Sitting at the center of this layered software universe is the data dictionary, a repository that stores the essential information about the data and structures within R/3. The information stored in the data dictionary is accessible by a number of components of R/3, including ABAP/4 programs, application modules, the Enterprise Data Model, the database interface, and the help system.

The data dictionary's fundamental purpose is to hold the descriptions of the fields contained in tables, screens, and ABAP/4 programs, including formatting, data typing, and specific functions relative to the field and its data. The main goal of this approach is to maintain

consistency within the R/3 environment, and its presence makes development and customization a much easier and more consistent task.

The data dictionary—itself an SAP proprietary product—has a peculiar relationship to the relational database component. While there is a theoretical one-to-one mapping of R/3 data between the data dictionary and the database, in reality, the relational database exists more as a storage and low-level data management tool for R/3. Genuine database logic exists primarily in the data dictionary and the modules themselves.

For example, R/3 often works with several logical tables from the data dictionary at once, effectively grouping them as a single object. This kind of mega-table can be difficult to map in a relational database environment. R/3 solves the problem by maintaining these groups or clusters of tables as discrete elements within the data dictionary and storing them as a single large table in the relational database. Likewise, when a relational database is restricted to using a lesser number of tables than R/3 requires, the system can group individual tables into a large pool that is stored as a single table in the database.

SAP plans to offer some modifications to this database/data dictionary relationship in its next version of R/3. If the user doesn't need the logical efficiency of clustered tables or if the database environment provides sufficient speed of execution to obviate the need for clusters, then the data dictionary tables can be set up in perfect one-to-one correspondence to the relational tables. This will allow greater consistency between the data dictionary environment and the relational model of the user's data.

The data dictionary can also be made to define its own data tables independently of the database structure itself and then utilize these tables in application modules. This can be done using what R/3 calls *views* and *matchcodes*. A view can be considered as an ad hoc, one-time subset of a database table or tables; a matchcode provides wildcard options that can specify a range of values and return an appropriately filtered table. Both are used by applications modules to work with or display specific elements of the database.

Contents of the Dictionary

Underneath the data dictionary layer lie the elements or objects contained within. The different types of information about R/3 contained in the data dictionary speak volumes about the central role this particular software element plays in the R/3 world.

- Tables, their fields and relationships, are the larger elements that make up the Data Dictionary. The tables in R/3's dictionary are conceptually identical to relational tables. They are two-dimensional matrices made up of individual fields, each of which, in turn, is defined by its domain and data element.

- Domains are the technical mechanism by which the Data Dictionary supplies the rules for the data stored in tables. Domains define the attributes of the data according to format, length, data type, and other features.

- Data elements are where the Data Dictionary stores the business rules that define a field. These rules essentially take a general purpose domain and define how it is to be modified for a specific purpose within a given program. The effect is not unlike inheritance in the object-oriented world: The data element of a given field gives specific attributes to a generalized "class" defined by the domain to which it belongs.

- Clusters and pooled tables are defined by the information stored in the Data Dictionary.

- Matchcode objects and Views are also stored actively in the Data Dictionary.

- Help information is also stored in the Data Dictionary. Context-sensitive help information can be accessed from the applications.

Having this active Data Dictionary gives the R/3 programs and programmers a centralized source of the meta-data contained in the R/3 environment. The contents of the Data Dictionary effectively define the state of the installation at a given user site, and, from the information in this structure, R/3 is able to protect the consistency of its data and business structures across every program in use.

Role of the Information Model

The richness of the Data Dictionary and the openness of the R/3 environment are derived from the use of an Information Model to describe the overall environment on which the general R/3 system and any individual system are built. This model—which lends a computer-aided software engineering approach to development under R/3—represents, in conceptual as well as concrete terms, the totality of R/3 functionality, of which any individual user's installation becomes a subset.

The Information Model represents the cornerstone of R/3's openness, both between its many modules as well as in relation to the outside world. The Information Model's blueprint of R/3 allows the developer or integrator to see and understand the interrelations among the different elements of R/3 in a multidimensional way. By understanding these relationships and programming accordingly, the user can modify, add to, or simply replace any module or software element in R/3 without losing its inherent data integrity and software integration.

The Information Model thus plays the role of both map and mapper: It defines R/3 as it exists "off the shelf" as well as showing how to change R/3 without losing its essential qualities of openness and integrity between different elements.

The Information Model maps the R/3 data world and its functionality using an entity relationship approach. It is itself divided into an Enterprise Data Model, an Enterprise Function Model, and an Organization Model.

Roles of the Enterprise Data, Function and Organization Models

The Enterprise Data Model defines the data, files, and other elements in an R/3 application, while the Enterprise Function Model defines the business processes that relate to the Data Model. The Organization Model contains the organizational descriptions and dynamics contained within R/3. Descriptions listed in the Information Model of data, relationships, and entities are stored in the Data Dictionary and are mapped directly to the Data Dictionary's domains, tables, and keys.

Two Ways to Use the Information Model

The Information Model is used in two ways. As a static tool, the model can be used by developers to understand the relationships that underlie the Data Dictionary and to maintain consistency with the overall R/3 Information Model when building a specific instantiation of R/3. In essence, the Information Model serves as a larger road map of the R/3 system.

SAP also provides an Online Information Model tool that allows the Data Dictionary and Information Model to be modified according to the needs of the individual user. Again, this modification is undertaken with the express condition that the Data Dictionary and Information Model maintain a close degree of consistency. The result is that R/3, when installed, has a fundamental architecture underpinning its functionality that can be studied and modified as the user needs.

Configuring R/3: Three-Level Client/Server Architecture

The Information Model and the Data Dictionary are essential elements in defining and managing both the internal elements of the R/3 system and the means by which R/3 is distributed across a wide variety of platforms and networks. The ability to be configured as a distributed system and its support of open systems hardware and software are the most important design features of R/3.

The physical layout of an R/3 system is distributed within a networked environment according to what SAP calls its "three-layer" architecture. The essence of this approach is to segment the three major functional components of R/3: the presentation element, the applications services, and the database services. By adroit use of communications software, the Data Dictionary, and the Dispatcher, this segmentation allows each component to be physically located on a different system.

Flexible Configurations

The beauty of this approach is precisely that it lends itself handily to a highly distributed, heterogeneous environment. Users are theoretically able to configure R/3 in a variety of forms depending on their needs: Presentation code can be on a local PC, workstation or terminal, with applications code (the modules) on a departmental server and the database on a much larger server down the hall or in the next state.

The three components can also be combined in different permutations: presentation and applications on a workstation and the database on a server, or applications and database on a single server, with presentation code running on the workstation or PC. A third alternative is that all three can be run on a single machine. In every instance, the hardware can be mixed and matched as the user sees fit within the large range of client and server systems and their operating environments. (See Table 2.)

Platform Support: Currently or Announced

Hardware	Operating Systems	User Interface	DBMS
HP 3000 and 9000, SNI, DEC VAX and Alpha, SUN, IBM RS/6000	HP MPE/ix, HP UX, SINIX (SNI), BS2000 (SNI), OpenVMS, Solaris, AIX, BOS (Bull), Windows NT	Motif, Windows 3.1, Windows NT, OS/2 Presentation Manager	Oracle, Informix, HP Allbase, DB2, Software AG Entire SQL

Table 2. R/3 can run on a broad range of server and client configurations, which, combined with its standards support, qualifies it as an open systems software product.

That's the theoretical version. In reality, the distribution of resources is limited in part by the underlying network services. In particular, SAP recognizes that current network bandwidth requires the database and applications servers to be located on the same machine or in a close, clustered configuration.

Three-Tier Architecture

In order to distribute its functionality across three potentially discrete sites, R/3 isolates three groups of code: the user interface, application logic, and database management. Each of these elements can exist in a single, nondistributed system as well as in a variety of combinations on separate systems. (See Illustration 1.)

Protocol Support

Creating the distributed environment requires having a number of means for communicating data elements, tables, and programs between applications and across platforms. To do this, R/3 defines a large number of communications protocols that support distributed processing within R/3. These elements are, in turn, controlled by the Dispatcher.

Configuring R/3: Three-Level Client/Server Architecture

R/3's Three-Tier Architecture

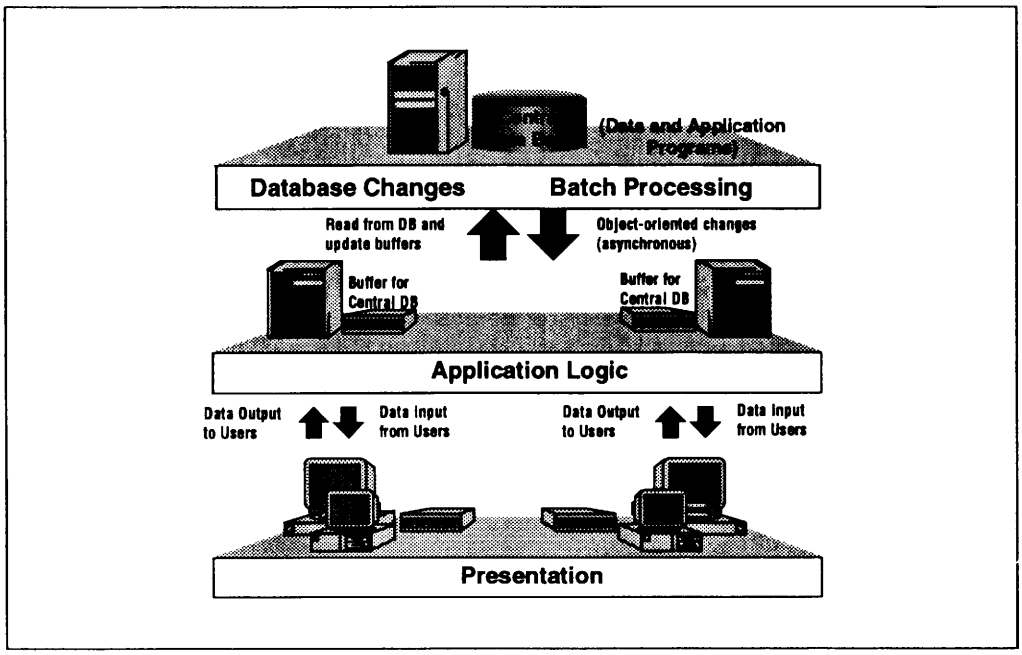

Illustration 1. The architecture of R/3 has three tiers. Each can be distributed or located on the same hardware platform.

- CPI-C: Part of IBM's SAA as well as an X/Open specification, CPI-C is used when applications and their modules require synchronized data communications. The CPI-C protocol is built into ABAP/4 and is the basic form of interprogram communications.
- Remote Function Call: Essentially a superset of CPI-C, the Remote Function Call (RFC) mechanism in R/3 acts like an RPC, allowing for communications between heterogeneous elements in the R/3 architecture.
- Electronic Data Interchange: Electronic Data Interchange (EDI) is an important form of communications for R/3, despite the fact that EDI standards are hardly uniform across the industry. But, as befits an application suite targeted at large manufacturing concerns, EDI interfaces are made available and are used in order to communicate orders and other information between R/3 users and their external customers and partners.

Interfaces, Databases, and Operating Systems

R/3 also accomplishes this physical distribution and heterogeneous support by defining two sets of interfaces that mediate between the R/3 environment and the larger operating environment that surrounds it. These are the presentation interface and the database and operating system interface.

PRESENTATION INTERFACE. The presentation interface exists so that the use of Motif, Windows 3.1 and Windows NT, or X-terminals remains independent of the rest of the R/3 environment. Guaranteeing this cross-platform functionality is the R/3 terminal process, which isolates the specific functionality of a given windowing system from the rest of R/3. This terminal process can reside on the workstation itself, although, in the case of an X-terminal, it must exist as part of the application or database server.

DATABASE INTERFACE. Conceptually, the database interface isolates the mechanisms for transporting and managing data between applications and the database. It also has the key

task of mediating between the logical structures of the data dictionary and the physical structures of the relational database itself.

The database interface handles data access through the use of SAP SQL, which is essentially a library of standard SQL calls available within the ABAP/4 environment. The database interface is responsible for translating SAP SQL into the native SQL of the database installed in the system.

R/3 provides a second mechanism for database access that allows for native SQL to be embedded within an ABAP/4 program. ABAP/4 then passes the SQL directly to the database, bypassing the database interface completely.

SAP tends to promote the use of SAP SQL for obvious reasons: As part of the ABAP/4 environment, SAP SQL is independent of the underlying relational database and thus allows database independence for the R/3 applications. And bypassing the database interface also allows the application to bypass much of the data dictionary's role, interfering with the integration and consistency of R/3. But SAP also recognizes that users may want to access specific functions within their relational databases that are not generally available within SAP SQL. Thus, it is possible to call and execute native SQL routines within the ABAP/4 language.

Use of native SQL does entail some limitations. The notion of clustered and pooled tables is dependent on the intervention of the Data Dictionary, where the information defining these mega-tables is stored. The use of native SQL bypasses the Data Dictionary and thus, bypasses these mega-tables. This fact is of little practical consequence from the programming side, as long as the user or developer is willing to maintain database tables outside the Data Dictionary environment. But the key role that the Data Dictionary plays in maintaining both the consistency and portability of R/3 would make using native SQL less than advantageous unless there were a compelling reason to do so.

Operating System Interface

Finally, though no less importantly, R/3 also isolates the operating system from the internals of the software, allowing for a high degree of operating system independence. With support for a number of flavors of Unix, as well as OpenVMS, HP's MPE/ix, and Windows NT, it is important that the functionality of elements like the Dispatcher be kept as operating system-neutral as possible.

Using R/3 Applications

Running R/3 is similar to running most menu-driven, graphical applications suites. Users are assigned logon IDs and passwords, as well as a client number that identifies the specific company for which the suite of applications they will use are assigned. The client number allows R/3 to be installed on one or more physical systems and to be used among several companies or functional entities simultaneously.

Logging on brings up the main menu, which allows the user to select an application or enter the System Menu to manage specific sessions, create user profiles, or otherwise run non-application-specific services.

A Not Very Graphical UI

R/3 screens are graphical up to a point. Although the outside frames look like any Windows or Motif-based application, the text in the interior of the screen is displayed in nonproportional, typewriter fonts. The contrast is almost shocking. The surrounding windows support most of the standard widgets expected in graphical environments, while the text within the window looks like a throwback to a dumb terminal, which indeed it is. R/3's nonproportional fonts, which cannot be sized, were taken wholesale from R/2. The lack of true graphical user interface (GUI)-based fonts is a glaring omission in the marketability of R/3. Except

that they cannot be sized, the R/2 fonts don't lack functionality, but they definitely give the impression either that R/3 is incomplete or that SAP lacks understanding of what the client-side requirements are for open, client/server systems.

SAP has recognized this omission and has promised to upgrade the screens in Version 2.1, which will be available next year.

Nonetheless, the screens perform their tasks well, if not elegantly. Functions within individual windows or screens are selected by pointing and clicking on a series of menus or by entering a transaction code that identifies the desired function on the command line at the bottom of the menu. This command line is very much the refuge of the power user, and its inclusion frees the user from the slow process of scrolling through series of menus to arrive at a desired transaction or function.

The Dispatcher

From the operating system perspective, the functions performed in a given screen are treated as a collection of parallel processes controlled and executed by the SAP Dispatcher.

The Dispatcher acts much like a transaction monitor, allocating resources as needed to perform specific transactions. The basic functioning of the Dispatcher mirrors processing in the Unix world. Activating a function by one of the methods above signals the Dispatcher that an internal transaction has been initiated that requires a work process—essentially a Unix process—to be allocated to it. The Dispatcher's role is to manage the allocation by using resources available according to the distribution of the R/3 system. Once the specific transaction has been accomplished, that Dispatcher frees the work process for a subsequent task.

Supporting Software for the R/3 Environment

Surrounding R/3 are a large number of software tools and other elements that allow for customization, administration, tuning, and other functions above and beyond the actual running of the R/3 applications. These include system administration tools, software development tools, software for third-party applications integration, and installation tools.

System Administration and Installation

R/3 supports three levels of system administration tools: Unix-based tools, database-specific tools, and R/3-specific tools. Unix tools need little elaboration here, and database tools, such as those provided with Oracle, are included as part of the database system when it is purchased by the user, though they are not regularly included in the runtime database that is installed with R/3.

The R/3 tools include SAP DBA, which assists in the update and maintenance of all databases supported by R/3. There is also a Monitoring and Administration tool, which includes a performance monitor that can display information on the status of work processes within R/3. This tool also supports the administration of multiple users and services, allowing for the allocation and distribution of resources.

The Collector is another administration tool that collects statistics on SAP's use of Unix, ABAP/4, and the database. The statistics garnered by the Collector include network use, memory and paging, response time, database requests, and other performance and tuning data.

SAP also provides software to assist in the installation of this complex environment, from a knowledge-engineering program that helps develop the system configuration to a database installation program that configures the database.

Another important tool for installation is SAPinst, a menu-driven program that creates the R/3 environment and sets up the applications servers.

Development under R/3: ABAP/4

The principal development tool for R/3 is ABAP/4, which, as stated in "ABAP/4 Modules" above, is the same language used in the creation of modules and other elements of the R/3 environment.

ABAP/4 is well-suited to the needs of R/3 in a number of ways. Its interpreted nature supports prototyping and facilitates the development of multilingual applications. Its data types and libraries are designed to support the data and text needs of R/3, and it provides a rich set of operations within the framework of a fourth-generation programming environment. Data input and output from most external devices can also be handled directly from the ABAP/4 language. (See Illustration 2.)

As noted above, ABAP/4 also contains a rich SQL environment and a set of communications interfaces built around the CPI-C protocol. The combination of these two elements allows R/3 modules to contain the necessary programming functionality and logic for distributed computing.

Also assisting in ABAP/4's distributed programming logic are two forms of subroutine: functions modules and dialogue units. While functionally identical to a classic subroutine, these have the added benefit of being callable from a remote system. Thus, function modules and dialogue units, like the program modules of which they are subsets, can be distributed across the R/3 three-level architecture as needed.

Miscellaneous Tools

SAP also supplies a list of tools for developing applications using ABAP/4. In addition to the function libraries that come with ABAP/4, R/3 also provides debuggers, editors, and utilities that support the programming environment.

In addition, R/3 also comes with Screen Painter and Menu Painter software, which allow developers to design application screens using a fully graphical environment. These two components also aid in the development of custom applications or modifications of the R/3 suite.

Ad hoc data extraction and analysis are supported by ABAP/4 Query, an end-user-targeted application that generates reports based on R/3 data.

The Data Dictionary also has a number of software tools that support its development and maintenance, such as the Online Information Model tool. In addition, R/3 has a Data Dictionary Information System that develops a series of reports on the status of the dictionary, including lists of current applications, tables, fields, and data elements.

Integrating Third-Party Software

One of SAP's major goals is to give third-party applications a role in R/3. This recently became more important when SAP and Microsoft Corporation (Redmond, Washington) entered into an alliance that will see R/3 ported to Windows NT and will include the support of Microsoft applications such as Excel and Access in the R/3 environment.

To this end, SAP has developed the Alice interface, which allows Windows-based applications to connect to the R/3 environment and share data with R/3 applications. This product is due out by the third quarter of 1993, and it will be extended to support the Macintosh environment at a later date.

ABAP/4 Language Syntax

```
REPORT R310LO63
TABLES; TABNA
DATA:   BEGIN OF TAB OCCURS 100,
                    COUNTRY LIKE TABNA-COUNTRY,
                    ID          LIKE TABNA-ID
                    NAME1   LIKE TABNA-NAME1,
                    SALES   LIKE TABNA-SALES,
                    END OF TAB
                    COUNTRY_SUM LIKE TAB-SALES
                    PERCENT(S) TYPE P DECIMALS 2
SELECT * FROM TABNA
        MOVE - CORRESPONDING TO TABNA TO TAB.
        APPEND TAB
ENDSELECT
IF SY-SUBRC NE 0.
*D: TEXT-001: 'No entry available
*D:                  in table TABNA'
        WRITE: TEXT-001
        EXIT.
ENDIF
SORT TAB BY COUNTRY SALES DESCENDING
LOOP AT TAB
        AT NEW COUNTRY
        SUM.
        COUNTRY_SUM - TAB-SALES
        NEW-PAGE
        WRITE:   TEXT - 002
                            TAB - COUNTRY.
                  31 TEXT - 003
                            COUNTRY_SUM.
*D: TEXT - 002: 'Country :'
*D: TEXT - 003: 'Total Revenues :'
        ULINE
        ENDAT
        IF COUNTRY_SUM EQ 0.
                    PERCENT = 0.
        ELSE.
                    PERCENT = TAB - SALES * 100/ COUNTRY_SUM
        ENDIF.
        WRITE: / TAB-COUNTRY,
                            TAB-ID,
                            TAB-NAME1,
                            TAB-SALES,
                  60  PERCENT, '%'
END LOOP
```

Illustration 2. Sample ABAP/4 code used to create a table, called Report R310LO63, that displays country of origin, company ID, company name, sales to that company, and percentage of overall country sales represented by each customer within a given country. At the top of the table, the total revenues for all customers within the reporting country are displayed.

As noted above, third-party software can also be tightly integrated using the Enterprise Data Model's architecture.

The Challenge Faced by SAP and R/3

When installed in all its complex, distributed glory, R/3 presents a huge challenge and opportunity for the user as well as the systems integrator or consultant charged with designing and installing the system. Installing R/3 can only have a limited impact on a user organization, but, then, R/3 is not necessarily intended for the user seeking a minor adjustment to his or her IT environment—unless that individual is already using R/2. Some users are implementing small parts of R/3 in limited environments, but the efforts of SAP's huge engineering staff has been directed toward making R/3 a highly adaptive and adaptable product.

Indeed, R/3 is at times so feature-rich as to be cloying, and, although the basic system can be installed with little modification, optimal use can only come through a rather extensive consulting and systems integration approach. Making full use of a comprehensive suite of software like R/3 in a complex, distributed client/server environment can be a daunting task, and one that a user can undertake only with the understanding that the installation of the system will effect a radical change in the company's business practices, even if they were previously fully automated. If done well and planned for carefully, that adoption of R/3 can make the difference between mere data processing and strategic information technology.

But it is important to note this "business re-engineering" issue when considering R/3. Installing the full R/3 package without a major reworking of business procedures will only result in a better automation of an inherently outmoded or even problematic system. It's not for the pleasure of their company that SAP has linked up with major management consultants and systems integrators both in the United States and Europe. SAP recognizes this issue and has sought the partners required to deal with it. And users should recognize that the cost of hardware and software could quickly become a minor element in the installation of R/3.

R/3 and the Role of the Integrator

As implied by this complex environment, R/3 requires an extraordinary amount of consulting and systems integration work when doing a custom installation requiring more than just the installation of a "turnkey" R/3 system. For the most part, this is accomplished by one of SAP's many partners, ranging from Deloitte & Touche and Price Waterhouse in the United States to CAP Gemini Sogeti in France. SAP has vowed to farm out some 80 percent of the integration and consulting work to its partners. (See Table 3.)

R/3 Consulting and Systems Integration Partners

Andersen Consulting	Price Waterhouse	Deloitte & Touche
Coopers & Lybrand	Ernst & Young	KPMG
CAP Gemini Sogeti		

Table 3. Ultimately, consulting partners will be key to the success of R/3 by engaging in the re-engineering of customers' business processes.

These partners are extremely active in the R/3 market. Deloitte & Touche, Price Waterhouse, and CAP Gemini have set up R/3 expertise centers in order to promote the development of this market. Their success is important to the success of R/3. Because SAP is unwilling or unable to take up the bulk of the re-engineering and systems integration effort, and end users will probably be unable or unwilling to handle it themselves, R/3 will become very dependent on the service side of the market for its success.

Is R/3 What the Market Wants?

SAP obviously did its homework in designing R/3, though its use of nonproportional, dumb-terminal fonts is disappointing and suggests a technological pragmatism that defies common marketing wisdom. Indeed, the company has been criticized by users expecting a more graphical user interface, and SAP, in response, is working hard to upgrade its screens in a future release.

Competitive Architecture

Architecturally, R/3 is as close to what the market is looking for as any of its rivals' products. Its support for standards and open systems is without reproach, and its three-level architecture makes sense in this increasingly client/server world. Support for multiple databases, interfaces, and hardware platforms is exactly what customers are expecting in their software products today, and R/3 supports most of the major market leaders in each of these categories.

Complex Service and Support

The multivendor support matrix has its liabilities, however. Maintaining a complex software product like R/3 that is dependent on so many non-R/3 products and standards will force SAP to devote considerable resources to the support side of the business. As the product matures and its customer base spreads, SAP will be forced to keep tabs on aging versions of databases, operating systems, and interface software that its customers may not want to upgrade. As companies like Oracle already know, that support matrix can present a daunting task.

SAP's reliance on third-party service partners is also a good strategy, even if it leaves SAP out of a growing services revenue base. Consulting and systems integration can be distracting enterprises for a software company; many companies certainly have had their problems in growing that side of the business. But SAP may want to change its mind about services at some point. The 80 percent of the consulting that SAP doesn't want to do today could eventually represent a significantly larger revenue source than the R/3 software alone.

Getting It to Market

SAP's next tasks are to move its many applications modules into the market in a timely fashion and to continue the practice of integrating third-party software, like Microsoft's Access database and Excel spreadsheet, into R/3. As the mid-1990s approach, R/3 may become one of the premier packaged software products in the open systems market. From a technological and marketing standpoint, there seems to be little to stop that from happening.

Next month's *Open Information Systems* will address
The Future of Unix: Microkernels and More

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: DEVELOPMENT TOOLS

OSF DCE: In Search of a Few Good Tools

Four years and many megabucks into the cycle, the Open Software Foundation (OSF, Cambridge, Massachusetts) Distributed Computing Environment (DCE) is visibly flirting with adoption in the marketplace. DCE has survived industry politics, uninspired marketing, overblown expectations, and diffusion of responsibility among the main sponsors. Yet, it persists as the closest thing to a standard for developers who are committed to building distributed applications for heterogeneous environments. X/Open has incorporated DCE into its Distributed Computing Profile (See *Open Information Systems,* Vol. 8, No. 3, February 1993) and is trying to address the issue of DCE cross-vendor, transport-level compatibility. IBM (Armonk, New York) has come out of the blocks with a DCE-based enterprise strategy and has backed it by shipping DCE Core Services on both its AIX and OS/2 platforms. Digital Equipment Corporation (Maynard, Massachustts) and Hewlett-Parkard (HP, Palo Alto, California) have announced DCE Core Services products, and even Microsoft Corporation (Redmond, Washington) and Novell Corporation (Provo, Utah) have made the "if and when" pledge to respond to demand.

The DCE Tools Deficiency

Despite these indications of imminent acceptance, the nagging lack of development tools plagues DCE. Relatively few programmers can design and build a distributed application working at the DCE application programming interface (API) and remote procedure call (RPC) Interface Definition Language (IDL) programming level. And the tools vendors who could solve that problem have been waiting on the sidelines for a solid indication of a commercial opportunity, watching to see where and when the investment should be made. This uncertainty has made the timing of DCE adoption hard to pin down.

Leaping into the Void: Gradient Technologies

While the timing remains uncertain, the likelihood of acceptance in the marketplace just got a booster shot from the same small company that already saved DCE from PC-client oblivion—Gradient Technologies (Marlboro, Massachusetts). Gradient beat the long odds once by successfully delivering a secure DCE client based on Windows 3.1 with a product called PC-DCE. Gradient might be playing a key role again by rushing into the vacuum of DCE development tools with a product called Visual DCE, a DCE development tool based on Microsoft's Visual Basic and Visual C++ products. Gradient rolled out a prototype of Visual DCE at the May OSF Challenge '93 event in Boston and demonstrated some examples of how Visual DCE has been designed to dramatically reduce the complexity and the coding involved in building DCE applications.

Targeting the Cobol Developer

The high-level goals for Visual DCE are to decrease application development time and shift the requirement for programmer skills toward skills that are mainstream in the market. Visual DCE essentially provides a Visual Basic and Visual C++ interface to DCE core services. With Visual DCE, DCE bindings, interfaces, names, and access control lists (ACLs) translate into Visual Basic objects (called *controls*) with configurable property sheets; IDL data types automatically translate into Visual Basic data types. Gradient has extended Visual Basic in order to take the complexity out of working with the DCE and Windows APIs.

Visual DCE: An Object-Based View

The C language interface to DCE presents an awkward and tedious development environment, including the need to repetitively write procedure calls to perform basic operations such as binding, name lookup, and error-checking. The Gradient design for Visual DCE characterizes the typical ways these procedures are performed, groups them into classes with default properties, and provides a relatively simple mechanism to change the object properties at design time or runtime. Visual DCE represents Windows controls as classes with standard default properties and supports the

development of custom controls. The architecture of Visual DCE is shown in Illustration A.

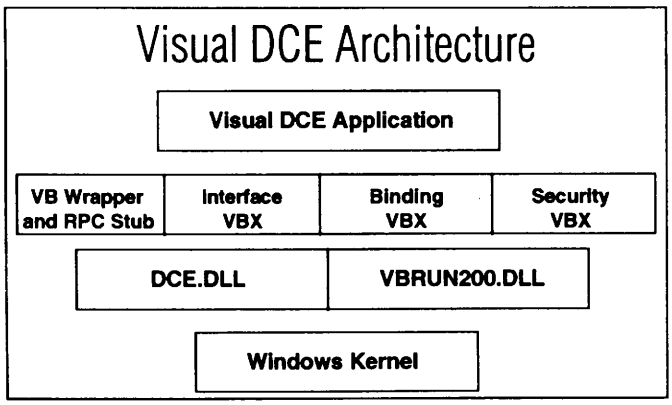

Illustration A. The Visual DCE design encapsulates the DCE layer and generates the calls to the DCE API from a Dynamic Link Library. This modular product design isolates Visual DCE from Windows 3.1 and supports future use of the Windows NT RPC and possibly other architectures.

Visual DCE enables the developer to sidestep the complicated DCE API calls, such as the DCE RPC binding routines. Instead, the developer can create and then manipulate high-level "Binding" objects, "Interface" objects, "Directory" objects, or "Security" objects. Interface objects are used to link Binding objects with RPCs and can be defined through the selection of RPC interface and operation from a listing of available options. The Binding objects can be defined based on the selection of time-out, authentication, server name, string binding, or other attributes. The Directory and Security objects can be similarly defined, modified, and manipulated within Visual DCE. Double-clicking on a Visual DCE icon displayed on the Visual Basic tool bar pops up the object properties window, which supports pull-down menus for point-and-click selection of object attributes.

Developing with Visual DCE

With Visual DCE, as few as three simple assignment statements are required to bind a client and a server, for example:

Statement	Action
Interface1.Action = RPC_Interface_Open	Open the interface
Binding1.InterfaceID = Interface1	Link the interface to the binding
Binding1.Action = RPC_Binding_Bind	Activate the binding

This code sample compares with 25 to 50 lines of DCE API calls to accomplish the same actions, leveraged further by the reusability of the Visual DCE objects compared with the standard C language approach. An entire page of RPC code to create an RPC binding is replaced by just one selection from the Binding object properties window. Directory and Security objects can similarly be linked to a Binding object with Visual DCE, again multiplying the savings in effort and time to the DCE developer. Visual Basic initializes and breaks down context automatically, saving the developer from having to code special operations such as starting or terminating task-level operations. Also, Visual DCE objects are automatically saved when the DCE program is saved.

The developer does not have to be aware of threads nor provide for error-handling with Visual DCE. Exceptions that result from an RPC operation are handled by the Visual Basic exception handler, mapped to Visual Basic error codes and with a default to the display of a message box.

Visual DCE IDL Compiler

As a key part of Visual DCE, Gradient offers an Interface Compiler back end for the PC-DCE IDL compiler that generates a Visual Basic interface module from a standard IDL file containing the interface calling parameters and the attributes of the client and the server. The Visual DCE Interface Compiler handles differences in pointers, strings, and structures, and generates a Visual Basic version of the IDL constants, type definitions, call declarations, and arguments for use in the Visual Basic program. The Visual Basic interface module is compiled and linked with the stub module to create a Windows Dynamic Link Library (DLL) that includes both the client and server stubs generated by the IDL compiler and the additional code necessary to marshal the Visual Basic data types for remote procedure call execution. The RPC in the DLL can be called directly from Visual Basic and can be pasted as a template into an application.

Not Just for DCE Client Development

Contrary to expectation, Visual DCE is not aimed exclusively at developing DCE client applications. Although it is a Windows-based tool and the current prototype does operate under PC-DCE, Visual DCE was conceived for developing both the client and server pieces of a DCE application. In fact, part of the broader Gradient mission is to change the rules about how Windows PCs fit into the DCE picture. Gradient believes that developers are turning to Windows not only for less expensive hardware and software, but also

because the range of choices and the support for application development, system management, and client/server functionality have become more than acceptable. In a narrow context, Visual DCE offers hope to today's DCE developers that they can get there from here without completely retooling their skill sets. In the broader context, Visual DCE could foreshadow a new dimension of the role of Windows in the DCE framework.

Product Status and Availability

Visual DCE was in prototype form in June, and Gradient is hoping to stave off the intense interest in the product to allow for a full beta test before shipment by the end of 1993. The version demonstrable in early June runs on PC-DCE 1.0.1 and Visual Basic 2.0, and it can be used to develop client-side applications. The product reliably binds and communicates with multiple server applications, such as the DCE "Market Minder" and "greet" applications, which have been used for the past couple of years to demonstrate the heterogeneous distributed capability of DCE. The product runs with very good performance on a garden-variety 486 clone and offers quite reasonable ease of use for the non-C programmer. — *S. Dolberg*

CORBA WATCH

SunSoft's Project DOE: The Shape of Solaris to Come

Toward Ubiquitous Distributed Objects

Sun Microsystems Incorporated (Mountain View, California) has finally weighed in with a CORBA-compliant distributed object computing product. The Project Distributed Objects Everywhere (DOE) Developer's Release is SunSoft's first product to implement the Object Management Group's (Framingham, Massachusetts) Common Object Request Broker Architecture (CORBA) interfaces for distributed object management services. The Project DOE Developer's Release is a toolkit for defining distributed applications as objects that can interact in flexible, powerful ways across LANs and WANs.

When the Object Management Group (OMG) defined CORBA during 1991, SunSoft was a prime mover in the effort. SunSoft, the operating systems software subsidiary of Sun Microsystems, and its partner, Hewlett-Packard Company (Palo Alto, California), were responsible for the Interface Definition Language (IDL) adopted as the static client interface within CORBA.

Almost two years later, SunSoft announced its first CORBA product. In the meantime, HyperDesk Corporation (Westboro, Massachusetts) and Digital Equipment Corporation (Maynard, Massachusetts) released products that implemented some of CORBA with promises to fully abide by the OMG standard later. Why did SunSoft require so much time to get to market? (The same question is relevant also for HP, which has yet to announce a CORBA product. But that's another story for another issue.)

The answer to this question can be found in SunSoft's evolving strategy for its CORBA-related technology. SunSoft has now stabilized its ideas about how to put its next generation of distributed object computing technology to work for customers. This was not true two years ago.

SUNSOFT'S DISTRIBUTED OBJECT STRATEGY. Project DOE is the heart of SunSoft's next-generation computing platform. SunSoft has formulated a strategy for the software that balances an ambitious vision for future Sun platforms with the need to give current users value today.

SunSoft's distributed object computing strategy has three goals:

- Reduce the cost of building and maintaining complex software. Project DOE is SunSoft's foundation for reusable, component-based software for both existing and new applications. Project DOE can be used to "objectify" existing code as well as to create new object-oriented code. In both cases, objects allow developers to reuse existing code in applications, to distribute computing to take advantage of network resources, and to integrate different pieces of code with much greater ease than before. The result: Developers can create complex software for a lower cost than they can using conventional methods.

- Build enabling technology that makes distributed applications easier and less expensive to build. Project DOE will eventually become the next generation of SunSoft's Solaris operating system environment. It will support a variety of underlying operating system and distributed computing services, as well as applications and application development tools. SunSoft's measure of success will be whether or not these tools reduce the effort and cost required to build distributed applications.

- Retain for SunSoft's Solaris operating system a place of prominence among computing environments. Project DOE must succeed for SunSoft to turn back Microsoft's assault with Windows NT on the Solaris customer base. SunSoft hopes to beat Windows NT in the market by offering support for robust distributed applications and support for distributed object computing before Microsoft does and by conforming to the Object Management Group's standards before Microsoft does. At the same time, SunSoft hopes to hold onto its ability to ask premium prices for Solaris.

These are ambitious goals, and the technology required for SunSoft to attain them must be powerful and sophisticated. Interestingly enough, the DOE Developer's Release *does not* contain all the technology SunSoft will need to achieve these goals, a fact that SunSoft acknowledges. Rather, it contains enough for SunSoft and key early customers to begin filling in the outlines of Sun's next-generation platform by drawing on practical experience building applications.

What Is Project DOE?

Project DOE is both a strategy and a set of products. The first product shipped under the Project DOE program was ToolTalk. ToolTalk allows two independent applications to interact using simple messages, as opposed to using calls to complex network APIs.

SunSoft's inclusion of ToolTalk in Project DOE has been the source of some confusion. ToolTalk is not a CORBA implementation now and isn't ever likely to be. Rather, ToolTalk gives developers the rudimentary features of an object request broker, if only because it makes communications between independently created pieces of software object-like in their simplicity.

The *real* Project DOE, however, is rooted in CORBA and a larger environment based on it. Sun's DOE platform has much in common with its current Solaris operating system. (Indeed, we can expect Sun to retain the Solaris brand name for its distributed object technology in later releases.) Both platforms support distributed computing. Both are also based on 32-bit, multitasking architectures. But Project DOE goes much further than today's Solaris to make distributed computing accessible to a wider range of developers.

The major components of the DOE platform are (see Illustration A):

- *SunOS 5.x and ONC+.* This is a separable "support" layer. SunOS 5.x is a multithreaded version of Unix System V Release 4 that runs on SPARC RISC chips and on Intel 486 and higher. ONC+ provides networking support, including TCP/IP, and associated services, including the Network Information System Plus (NIS+) naming service, the transport-independent remote procedure call (TI-RPC), and security.

- *The DOE Object System.* The heart of the DOE Object System is SunSoft's CORBA-compliant object request broker (ORB), the Distributed Object Management Facility (DOMF). SunSoft's Project DOE Developer's Release supports both of CORBA's client-side interfaces (the Static Invocation Interface and the Dynamic Invocation Interface) as well as its server-side interface, the Basic Object Adapter. This layer of the DOE platform also contains basic services to support DOMF.

- *DOE Application Environment.* The DOE Application Environment contains higher-level services required to build distributed applications. The services included in the Project DOE Developer's Release are: naming, which governs how object names are mapped to implementations; associations, which support the linking of objects and embedding of objects within other objects; event notification, which allows one object to notify other objects of the occurrence of an event; and properties, which allow developers to attach descriptions to objects. Other services, including transactions, will be shipped in later releases.

- *DOE Applications.* The DOE Applications are indistinct at this juncture, and probably will remain so for a year or more as SunSoft delivers production versions of the base platform. SunSoft's initial ideas for applications that it might ship as options with DOE include systems management, multimedia, and groupware. Third parties will deliver others.

The key differences between today's Solaris and DOE can be found in the ORB and the application services. To build distributed applications using Solaris, developers have three choices, each of which simplifies distributed applications development in its own way. They can program to Sockets, they can use RPCs to link applications, or they can use ToolTalk.

DOE promises to simplify development still further. One of the two keys to Sun's DOE architecture is IDL, which SunSoft will promote to developers as the main facility to define DOE objects. Actually, developers use IDL to define *interfaces* to *object implementations*. An object implementation can be a binary executable, a script, a database—whatever. All object implementations are accessible only through their interfaces. This makes it possible to change the way an object is implemented without breaking applications that use it. It also

makes it possible for objects to be moved on a network without breaking existing applications.

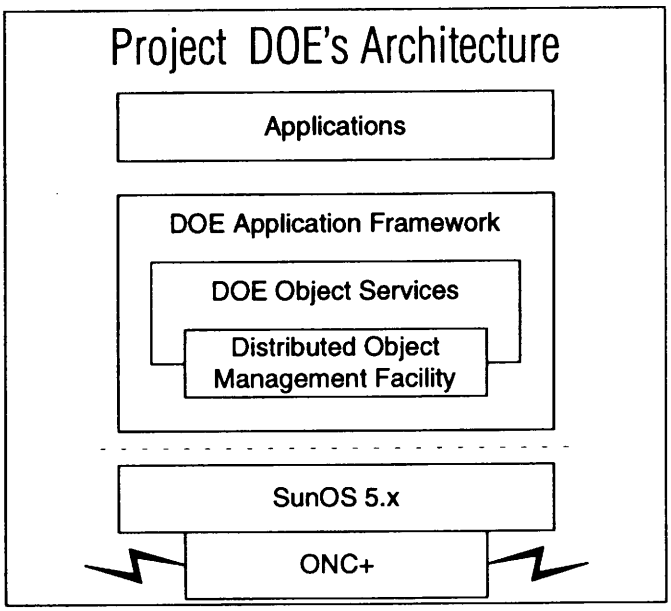

Illustration A. Project DOE outlines four major layers of system software, beginning with a base of SunOS 5.x and ONC+, and proceeding up through an object management layer, a service layer, and, finally, a tools layer.

IDL may be DOE's center of gravity, but SunSoft also has implemented CORBA's Dynamic Invocation Interface in the Project DOE Developer's Release. SunSoft just doesn't seem to believe that it and its customers will gain as much benefit from use of the dynamic interface as they will from using IDL. SunSoft believes the immediate use of the dynamic interface is to allow developers to browse the objects that exist in an environment.

The other keys to the DOE architecture are SunOS 5.x and ONC+. SunOS gives SunSoft a threads service to support interobject messaging. Every request from one object to another invokes a new thread, not a full process. ONC+ gives SunSoft an RPC mechanism to use as the underlying transport mechanism for interobject messages.

Architecture of DOE Developer's Release

SunSoft's Project DOE Developer's Release is a first step toward the realization of the full DOE environment. The toolkit includes the following:

- The Distributed Object Management Facility for SunOS, which contains the CORBA Static Invocation Interface (as realized in IDL), Dynamic Invocation Interface (DII), and Basic Object Adapter
- A set of Object Services to support DOMF. The services are object-naming, associations, properties, and events
- C and C++ language bindings
- Documentation, including the source code for four applications and a videotape
- A four-day training class and support

The DOMF is the heart of the toolkit, and SunSoft has also made available its C and C++ language bindings as part of the offering.

The Object Services are perhaps the most interesting part of the release. The services include some of those that SunSoft and 20 other vendors have submitted to the OMG in response to its Object Services Request for Technology. The Joint Object Services Submission (JOSS) appears headed for approval by acclamation, since it has no real competitors.

The Project DOE Developer's Release, then, appears to have the first implementation of JOSS. SunSoft's implementation may need to be fine-tuned to accommodate revisions called for during the OMG's review process, but the basic services should remain the same.

JOSS introduces important new interfaces to CORBA, the most important being object-naming. The JOSS interface is based on federated naming as its underlying model. That is, the interface does not force all CORBA implementations to conform to any particular naming scheme, but it manages naming across schemes. This is vital to support interoperability of objects across object request brokers.

JOSS also standardizes the interface to object storage services, using an interface based on the work of the Object Database Management Group, an ad hoc group of database vendors in which SunSoft played a key facilitation role. SunSoft plans to make this interface available in the next version of the Project DOE technology it releases.

DOE Applications

The Project DOE Developer's Release may be at an early stage of release, but it does work. SunSoft demonstrated the software on the show floor at the Object World trade show in San Francisco during mid-June.

The demonstration application was DOE Cards, a system that allowed users to trade a variety of cards with one another to accumulate points. DOE Cards ran on the show floor over wireless Ethernet for the three-day conference, supporting thousands of trades and hundreds of users.

DOE Cards was a whimsical demonstration of SunSoft's capabilities with DOE. The company believes that initial users of its distributed object technology will fall into three applications categories:

- *Interconnecting Binary Applications.* SunSoft has a demonstration application that links Lotus 1-2-3 spreadsheets and a visualization package using DOE technology. Any of these three packages can update the other in real time. The spreadsheets and visualization package are each equipped with IDL interfaces, which define the objects each contains and the messages other objects can use to invoke those objects. The result is like a Microsoft OLE application that works over a network—LAN *and/or* WAN.

- *Extending Existing Applications.* In another demonstration project, SunSoft used DOE to add video-message and speech-message objects to the Solaris Mailtool. In this case, SunSoft added an IDL interface to its Mailtool that allowed it to invoke the video or speech objects. The result is added functionality without rewriting of the Mailtool.

- *Distributing Processing on a Network.* In a third demonstration, SunSoft used DOE technology to define calculation engines in a network as compute objects to support a wind-tunnel simulation. As the user increases the velocity of the wind in the simulation, the object driving the simulation invokes more of the compute objects to support it.

Pricing, Packaging, and Availability

The Project DOE Developer's Release is essentially a pre-beta release of the DOE Object System. SunSoft plans to distribute the toolkit to a selected set of customers, produce a new revision based on their feedback, and be ready for a general beta program during the spring of 1994.

To participate in the pre-beta program, an organization must have a project in hand. The training class will focus during the first day or so on defining the first-cut IDL interfaces required to implement the participant's project, and then, during the final days, on refining those interfaces.

SunSoft has not revealed the costs of participating in this early release program.

Conclusions about Project DOE

VALUABLE ADDITION. SunSoft's Project DOE technology is a valuable addition to the emerging market in distributed object computing tools. SunSoft's approach offers users additional choices among CORBA products in three primary areas.

- First, SunSoft stresses CORBA's static interface as the primary API for developers. In earlier CORBA products, CORBA's dynamic interface was stressed as the preferred API. Each API is useful in its own right. Some developers will prefer the strong type-checking of the static interface over the flexibility of the dynamic interface. In addition, SunSoft, in its training, will help developers understand how they can mimic the behavior of the dynamic interface using IDL.

- Second, SunSoft's architecture for Project DOE may enable greater distribution of distributed object management functionality than some of the earlier CORBA implementations. SunSoft believes that as much information as possible about each object in the environment should be packed into its IDL interface. Other approaches place information in central repositories residing on a single server. In some cases, the centralized repository may not be the best option, particularly in large applications.

- Third, SunSoft's approach is closely aligned with the OMG's standards. SunSoft's delivery of its Project DOE technology was delayed in part by completion of the three major CORBA interfaces. In its approach to object services, SunSoft used the OMG's Object Services RFT process to effectively ensure that its work would be standard. In the future, we expect SunSoft's fealty to the OMG to continue.

The downside of SunSoft's Project DOE technology is its strong link with Sun's existing hardware/software platform. SunSoft is forcing users to choose Solaris and ONC+ if they want to obtain the Project DOE technology. Many users will object. Indeed, Project DOE may force SunSoft to broaden its platform coverage to include more than SPARC and Intel and more than Solaris.

CAN DOE MEET MICROSOFT'S CHALLENGE? Jim Green is the director of Project DOE at SunSoft. At Object World, he looked like the most relieved man west of the Mississippi. His team had delivered a product—at last. But there's still much work for Green and his team.

The toolkit SunSoft is delivering is a preliminary product. It is a vehicle for the company to learn enough to understand how to package its advanced technology as part of its mainstream Solaris operating environment. SunSoft's deliberate approach is prudent. Few users or vendors claim to understand exactly how to put distributed object computing technology to widespread use today.

However, SunSoft's approach can't be too deliberate. The schedule for the completion of just the basic platform takes SunSoft through the end of 1994. During this time, SunSoft must figure out how to turn its reliance on Solaris's threads service and nested RPCs into benefits for developers at large. For most business developers, these facilities will be far too complex to be of immediate benefit.

It will take SunSoft even longer to deliver an environment conforming to its complete vision. Unfortunately, the complete vision is what many corporate developers and all business users will want.

This is a crucial point in SunSoft's ability to compete with Microsoft, its stated strategic goal. Microsoft's genius is to deliver to the market just enough technology to satisfy a set of *end-user* needs and then to build toward a complete vision from that point. Microsoft's Object Linking and Embedding 2.0 (OLE 2.0) is an example. OLE 2.0 provides the foundation for an eventual distributed object computing system that Microsoft calls "Cairo." Today, OLE 2.0 supports only document linking and embedding within a local environment. We expect Microsoft to add distribution and support for other types of applications (in addition to documents) during the next three years.

Therefore, SunSoft hopes to compete with Microsoft, it is going to have to demonstrate fairly soon some amazing *end user* capabilities. Focusing on supporting developers only with toolkits and like products won't be enough. *—J. Rymer*

Patricia Seybold's Computer Industry Reports Order Form

Please start my subscription to:

		U.S.A.	Canada	Foreign
☐ *Workgroup Computing Report*	12 issues per year	$385	$397	$409
☐ *Open Information Systems*	12 issues per year	$495	$507	$519
☐ *Distributed Computing Monitor*	12 issues per year	$495	$507	$519

Please send me a sample of: ☐ *Distributed Computing Monitor* ☐ *Open Information Systems* ☐ *Workgroup Computing Report* ☐ *Paradigm Shift—Case Studies in Distributed Computing*

Please send me information on: ☐ Consulting ☐ In-Depth Research Reports ☐ Conferences

Total $_____ ☐ **My check for $_____ is enclosed.** ☐ **Please bill me.** ☐ **Please charge my subscription to: Mastercard/Visa/American Express** (circle one)

Name: _____ Title: _____

Company Name: _____ Dept.: _____

Address: _____

City, State, Zip code, Country: _____

Fax No.: _____ Bus. Tel. No.: _____

Card #: _____

Exp. Date: _____

Signature: _____

Checks from Canada and elsewhere outside the United States should be made payable in U.S. dollars. You may transfer funds directly to our bank: Shawmut Bank of Boston, State Street Branch, Boston, MA 02109, into the account of Patricia Seybold Group, account number 20-093-118-6. Please be sure to identify the name of the subscriber and nature of the order if funds are transferred bank-to-bank. **IO-793**

Send to: Patricia Seybold Group: 148 State Street, 7th Floor, Boston MA 02109; FAX: (617) 742-1028; Phone: (617) 742-5200; MCI Mail: PSOCG

1992 and 1993 Feature Reports from Patricia Seybold Group
To Order these Back Issues, Fax (617) 742-1028, or Call (617) 742-5200.

Office Computing Report

Volume 15 issues—$40

- ❒ 10/92 Microsoft's Workgroup Strategy—Moving Group Functionality into Windows
- ❒ 11/92 Visual Programming—Application Design for End Users and Professional Developers
- ❒ 12/92 The Notes Phenomenon—The Industry Reacts to Lotus Notes

Volume 16 issues—$50

- ❒ 1/93 Microsoft Access—"Cirrus" Database Project Gets down to Earth

Workgroup Computing Report

- ❒ 2/93 WordPerfect Information Systems Environment—WordPerfect Reveals Its Blueprint for Workgroup Support
- ❒ 3/93 Lotus Notes Release 3—Extending the Notes Paradigm
- ❒ 4/93 Can Windows NT Meet the Challenge?—Microsoft's Next Generation Operating System Stirs the Industry
- ❒ 5/93 Action Technologies' Workflow Products—Coordinating the Activities of People as They Work Together
- ❒ 6/93 Implementing Groupware—Groupware May Be Hazardous to Your Organization's Status Quo!
- ❒ 7/93 Issues in Remote Computing—Notebook PC Boom Raises Question: How to Handle Detachable Computing

Back Issue Total $_____

Unix in the Office

Volume 7 issues—$50

- ❒ 6/92 Digital's DECworld Gems—Alpha and Accessworks Shine

Open Information Systems

- ❒ 7/92 Integrating Applications in the Real World—Evolution, Not Revolution
- ❒ 8/92 Windows NT 3.1—Microsoft's Bid for Desktop Dominance
- ❒ 9/92 Oracle's Version 7—Can It Leapfrog the Competition?
- ❒ 10/92 Galaxy from Visix—Application Portability Breakthrough?
- ❒ 11/92 European Open Systems Architectures—Europe's Vendors Strike out for Open Distributed Systems
- ❒ 12/92 The Unix Data Center—Fact or Fiction?

Volume 8 issues—$50

- ❒ 1/93 X/Open in the 1990s—Making Open Systems Safe for Users
- ❒ 2/93 Highly Available Open Systems—Expanding Today's Definition
- ❒ 3/93 Sybase System 10—Can It Manage Enterprise Data?
- ❒ 4/93 Unisys ASD Framework—Meeting the Challenges of Software Development in the 1990s
- ❒ 5/93 Unix and PC Interoperability—Toward the Utility Era of Computing
- ❒ 6/93 Open Electronic Mail—Interoperability through Standards
- ❒ 7/93 SAP's R/3 Software Suite—Architecturing Open, Integrated Software for Distributed Enterprises

Back Issue Total $_____

Distributed Computing Monitor

Volume 7 issues—$50

- ❒ 10/92 Database Interoperability—A Comprehensive Approach to Database Access for the 1990s
- ❒ 11/92 Transarc Encina—Will Distributed OLTP Systems Overrun the Mainframe's Last Stronghold?
- ❒ 12/92 UI-Atlas Distributed Management—Object-Oriented, Distributed Management for the Unix System V World

Volume 8 issues—$50

- ❒ 1/93 Component Software—A Market Perspective on the Coming Revolution on Solutions Development
- ❒ 2/93 Switched Internets—The Coming Gigabit Revolution in Enterprise Networking
- ❒ 3/93 IBM's System Object Model—Cornerstone of an Open Distributed Object Computing Environment
- ❒ 4/93 Encapsulating Databases—Practical Uses of Object Technology to Inprove the Value of Relational Data
- ❒ 5/93 OMG's CORBA 2.0—Industrial Grade Standard for Distributed Object Computing?
- ❒ 6/93 Enterprise System Management—The Quest for Industrial Strength Management for Distributed Systems
- ❒ 7/93 Detachable Computing—Vendors and Users Scramble to Support Occasionally Connected Computers

Back Issue Total $_____