FEATURE ARTICLE

Why I Love the OMG: Emergence of a Business Object Component Architecture

Jeff Sutherland
IDX SYSTEMS CORPORATION,
BOSTON, MASSACHUSETTS

■ Object technology, a necessary but insufficient condition for software reuse, requires an infrastructure that supports plug-compatible business object components for fast and flexible delivery of new or enhanced products to the marketplace. This article is a retrospective view of key conceptual issues driving the standardization of a business object component architecture (BOCA) within the Object Management Group (OMG). The seamless integration of BOCA with the Unified Modeling Language (UML), a standardized meta-object facility (MOF), and an emerging CORBA component specification is essential for making a design-driven generation of runtime components into heterogeneous distributed object frameworks. BOCA standardization can enhance software productivity with plug-compatible reusable components, the holy grail of object computing.

The OMG Business Object Domain Task Force (BODTF) has been the focal point for standardization of a business object component architecture (BOCA).[1] The emergence of this standard could have far-reaching effects on worldwide software development. Priming this effort required joint work of the OMG BODTF, the Accredited Standards Committee X3H7 Object Information Management, and their joint sponsorship of the OOPSLA Workshop on Business Object Design and Implementation for the years 1995–1998. Completing BOCA standardization required the united efforts of over 800 of the leading software development companies and user organizations worldwide—the members of the OMG.

This article serves as a retrospective of some key concepts driving BOCA standardization and the global effort to build a unified set of standards for component-based systems throughout the software development lifecycle. In mid-1998, BOCA was moving through the final OMG vote for adoption. Initial tools are available that will generate BOCA applications running in the IBM San Francisco Java Framework from an annotated UML design document. The same tools will generate Enterprise JavaBean and CORBA component applications, as soon as frameworks for these emerging technologies become available.

Background

X3H7 OBJECT INFORMATION MANAGEMENT [NCITS 1998]

The International Standards Organization (ISO) has approved a new work item to refine and extend the current international standard Reference Model for Open Distributed Processing (RM-ODP) [ODP1998]. X3H7 (now NCITS Technical Committee X7: Object

1 Data Access Technologies, Inc., Electronic Data Systems (EDS), National Industrial Information Infrastructure Protocols (NIIIP), SEMATECH, Inc., Genesis Development Corporation, Prism Technologies, IONA Technologies, OMG [1998a].

Information Management), the US technical committee for this international work item, is tasked with the following:

— Refine the RM-ODP Enterprise Language, expanding the relationship of an enterprise specification of a system to other RM-ODP viewpoint specifications of that system, so as to enable use of the RM-ODP for specification of object-based application architectures.
— Ensure that the enterprise language together with the other viewpoint languages is suitable for the specification of a concrete application architecture to fill a specific business need.
— Measure success with a demonstration of the use of RM-ODP viewpoint languages to specify a concrete application architecture.

OMG [1998b] BUSINESS OBJECT DOMAIN TASK FORCE (BODTF)

With a membership of over 800 software vendors, software developers, and end users, OMG's goal is to establish CORBA as standard middleware through its worldwide standards specifications: CORBA/IIOP, object services, internet facilities, and domain interface specifications. Established in 1989, OMG's mission is to promote the theory and practice of object technology for the development of distributed computing systems, and to provide a common architectural framework for object-oriented applications based on widely available interface specifications.

The Object Management Group has chartered the BODTF to facilitate and promote

— use of OMG distributed object technology for business systems;
— commonality among vertical domain task force standards;
— simplicity in building, using and deploying business objects for application developers;
— interoperability between independently developed business objects;
— adoption and use of common business object and application component standards; and
— to issue requests, evaluate responses, and propose for adoption by the OMG specifications for objects, frameworks, services, and architectures applicable to a wide range of businesses.

OOPSLA WORKSHOP FOR BUSINESS OBJECT DESIGN AND IMPLEMENTATION

OOPSLA (Object-oriented programming, systems, languages, and applications) has been the leading object technology conference for more than a decade. There is a wide variety of participant-driven workshops, tutorials, invited speakers, panels, debates, and technical papers capturing the latest both in research and development.

The OOPSLA Workshop on Business Object Design and Implementation [1998] is jointly sponsored by X3H7 and the OMG BODTF to solicit technical position papers relevant to the design and implementation of business object systems.

The goals of the OOPSLA workshop are to

— enhance the pattern literature on the specification, design, and implementation of interoperable, plug-and-play, distributed business object components;
— clarify the design and implementation of object-oriented systems, particularly systems in which workflow patterns and the resource-event-agent (REA) accounting model [Geerts 1997] are basic building blocks for production business systems;
— contribute to emerging architectures for Intranet/Internet/Extranet applications, particularly those that integrate business objects, web servers, object and relational databases, and new approaches to client delivery of content;
— pursue issues developed in previous workshops on heterogeneous distributed workflow systems; specify business object solutions to mobile agents, process engines, and systems that exhibit emergent behavior; cross-fertilize business object design concepts with experience from the field of complex adaptive systems;
— provide explicit experience reports on business object systems developed and in production.

Why Business Object Component-Based Development?

For many years members of X3H7 and the OMG BODTF have been intensely aware that the global market has become a highly competitive environment moving at an accelerating rate of change. Gradual improvement in productivity and enhancements in quality are no longer enough to maintain market leadership. Time-to-market of new products and rapid evolution of old products and applications are key to success. This awareness led these groups to join forces in 1995 to initiate a radical change in software development environments, a change that took years to specify and will take decades to implement.

Accelerating product evolution requires reinventing the processes that bring products to market and eliminating processes that do not add value. Since modern corporations have embedded many rules and procedures for product delivery in computer systems, software applications that run business must undergo significant change. To gain the strategic advantages of speed and flexibility, corporations must remodel their business processes, then rapidly translate that model into software implementations. The rapid adoption of the Internet since 1995 has accelerated the pace of software evolution dramatically and pushed it in the direction of global, distributed object computing, the designated environment for BOCA.

Business process reengineering (BPR) sets the stage for continuous evolution of business processes to meet rapidly evolving business requirements. Implementation of software systems that support BPR requires business objects that can both simulate corporate procedures and translate smoothly into software objects. Well-designed business object implementations can be easily modified as a business changes. In particular, if software implementation can be automated from design, change becomes easy, rather than difficult or impossible.

Reorganization of business processes is most effective when there is a well-understood model of the existing business—an evaluation of alternative future models against the current business—and when a model-driven approach is used to realign the business strategy, processes, and technology. A multilayered, object-oriented blueprint of the enterprise can drive the refocusing, realignment, and reorganization of the business [Jacobson et al. 1995]. Current attempts to implement this process under the rubric of BPR have been largely ineffective due to difficulties in changing monolithic organizations, processes, and information systems.

Figure One [Digre 1997] demonstrates that enhancing the productivity and performance of integrated circuits (IC) has led to exponential growth in computing power over the past thirty years. This has been driven by

> the observation made in 1965 by Gordon Moore, co-founder of Intel, that the number of transistors per square inch on integrated circuits had doubled every year since the integrated circuit was invented. Moore predicted that this trend would continue for the foreseeable future. In subsequent years, the pace slowed down a bit, but data density has doubled approximately every 18 months, and this is the current definition of Moore's Law, which Moore himself has blessed. Most experts, including Moore himself, expect Moore's Law to hold for at least another two decades.[2]

More recently, Moravec [1998] observed that information-handling capacity in computers has been growing about ten million times faster than it did in our nervous systems during our evolution. The power doubled every two years in the 1950s, 1960s, and 1970s, doubled every 18 months in the 1980s (Moore's law), and is now doubling each year.

Custom chip development, which is largely software-based, has followed Moore's law due to the heavy capital investment in tools and technology common in the IC chip industry. However, this has not led to comparable gains in business application software development, largely due to lack of automated software construction from design artifacts and failure to achieve large-scale reuse of software components in business.

Understanding the reasons for rapid gains in hardware productivity clarifies the direction that application software development must take to achieve comparable results. Software productivity is a core issue for X3H7 and the OMG BODTF as they assess how to maximize the impact of software standards development on worldwide business.

X3H7 Contributions

Document X3H7-93-23 [Kent 1993], Objectives and Operations, provided the following guidelines for the work of the X3H7 during the period 1993–1996.

— Develop liaisons with groups working on object-oriented standards and know what they are doing.

— Complete the object model features matrix document that defines in some detail the characteristics of object models proposed by different groups.

— Develop an X3H7 reference document based on the features matrix for designated groups working on object model standards.

— Based on the importance of each liaison group and the timing of each group in the standards-development process, present formal proposals to these groups to facilitate harmonization of object model standards and enhance interoperability of distributed object systems.

— Develop problem scenarios arising in the interaction of object systems to illustrate clearly the technical issues in distributed object interoperability.

The majority of members of X3H7 are also members of the OMG and committed to seeing relevant standards implemented by industry bodies. Under the editorship of Frank Manola [1997], the object model features matrix document developed an analysis of issues in harmonizing object models, showing that competing object models provide not only different structures, but often different semantics underlying the concepts that support these structures.

Object model interoperability requires understanding the structure and semantics of commonly used object-oriented frameworks and the interfaces among these development frameworks. Object models must interoperate within widely adopted frameworks and the number of frameworks should be few. An X3H7 consensus was reached in 1994 that 80% of new object-oriented development would be done in three application languages (Smalltalk, OO COBOL, and C++) and that these applications would communicate through a business object request broker to four external environments—X3H2 SQL standard databases, ODMG standard object databases, Microsoft's COM environment, and the OMG CORBA environment. Figure Two [Sutherland 1994] illustrates the views of X3H7 at that time.

The widespread adoption of the Internet since 1995 has only accentuated the need for interoperable, distributed object standards and added Java to the list of widely used development languages. One of Java's primary benefits is enhancing interoperability of distributed systems, a primary object of X3H7.

Even before the rapid growth of the Internet, there was consensus that application developers should be

2 pcwebopedia.com. *Moore's Law.* http://pcwebopedia.com/Moores_Law.htm.

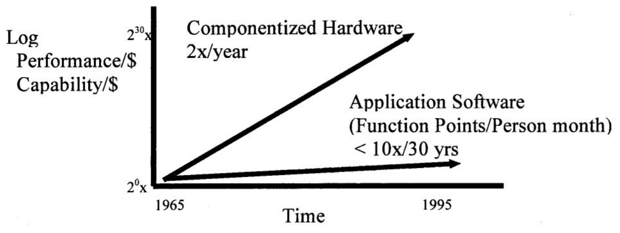

FIGURE *ONE*
Hardware price/performance versus software price performance.

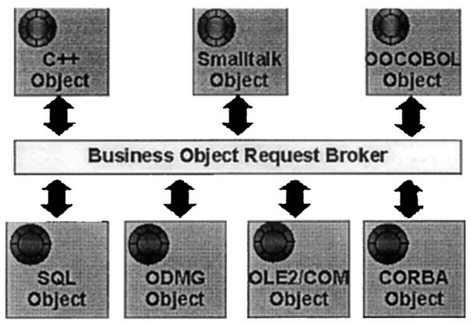

FIGURE *TWO*
ANSI X3H7 standardization targets. September 24, 1994.

shielded from the details of these implementation environments. They should be able to use object-oriented analysis and design (OOAD) tools to build an application in a standard notation. OOAD tools should be able to import legacy models from CASE tools. The application model and all of its artifacts should be stored and versioned in an object repository and the runtime application binary objects should be generated from the repository to conform to standard component interface specifications. Request broker technologies should provide automated mapping among development frameworks. Figure Three shows the X3H7 conceptual view of this problem.

X3H7 members participating in OMG and other standards bodies began driving the agenda of object model harmonization in multiple organizations. They were key technical contributors to the ISO standard RM-ODP, the distributed processing reference model that OMG technologies must conform to, in accordance with agreements between ISO and OMG. They also agreed to cosponsor, with the OMG BODTF, a business object design and implementation workshop at the OOPSLA'95 Conference on object-oriented programming, systems, languages, and applications, in order to encourage research in the drive for common business object component standards.

OMG BODTF Contributions

In 1994, Sutherland [1995] began discussing his findings on key issues in building lifecycle object-oriented development environments for business objects within standards organizations, including the OMG Business Object Management Special Interest Group (BOMSIG, now BODTF). Simultaneously, Casanave [1995], now chair of the OMG BODTF, edited the BOMSIG business application architecture white paper and later OMG Common Facilities RFP4: Business Object Facility and Common Business Objects.

Business Objects as Reusable Components

Objects are not enough to gain the benefits possible with object technology. Only plug-compatible, larger-grained components can achieve a productivity breakthrough. Early adopters of object technology asserted that packaging software in object classes would allow software to obtain the benefits of Moore's law seen in IC chip fabrication [Cox 1986] and some projects have achieved major productivity benefits. For example, a maintenance management system at General Motors originally written in PL/I was rewritten under an EDS contract in Smalltalk and achieved a 14:1 increase in productivity in design, coding, and testing [Taylor 1992]. Detailed analysis showed 92% fewer lines of code, 93% fewer staff-months, 82% less development time, 92% less memory needed to run, and no performance degradation.

Although there are many isolated projects that used object technology to achieve dramatic productivity gains during the past decade, this success did not translate into broad improvements across the software industry. In 1995, META Group [1995] reported that, "despite the promise of reusable objects, most IT organizations have realized a scant 10–30% productivity improvement from object technology (OT)." Failure to achieve larger productivity gains was attributed to data-centric, task-oriented applications development; methodologies and cultures that do not promote reusability, and few linkages between BPR-defined business processes and IT support initiatives.

Business objects are designed to support a clearly defined relationship between BPR-defined business processes and their software implementation. Using an object-oriented development methodology yields quick time-to-market and object-oriented design allows rapid evolution of business objects in response to market conditions. The bottom line is that object technology is a necessary, but not sufficient condition for large returns on investment. It must be combined with focus on delivering business object components that enable fast and flexible delivery of new or enhanced products to the marketplace.

A Business Object Component Architecture

As business models are renewed, software architectures must be transformed. A BOCA is an effective solution for dynamic automation of a rapidly evolving business environment.

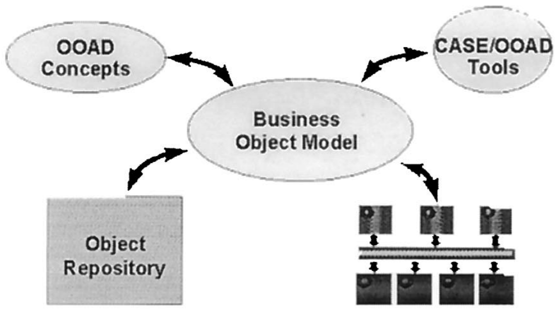

FIGURE *THREE*

ANSI X3H7 standardization targets. September 24, 1994.

Dynamic change requires reuse of chunks of business functionality. A BOCA must support reusable, plug-compatible business components. The two primary strategies in use now for implementing client/server systems to support reengineering business processes are visual Fourth generation languages and classical object technology. Both are better than COBOL, but neither can effectively implement plug-and-play business object components.

Building Business Object Components

A group of objects is the ideal unit of reuse, which should behave as a higher-level business process and have a clearly specified business language interface. Business object components are encapsulated with a protocol that allows efficient communication with other objects on the network. Work on the concept of Ensembles [Love 1993] shows that there is a minimal design specification for a plug-compatible component.

Consider a typical client/server application such as an order entry system. This system takes a purchase order as input and produces a validated order as output. The internals of this component should be a black box to the external world. The resulting order is input for another subsystem or, alternatively, an exception condition is raised if the purchase order is not valid for processing (see Figure Four).

To support plug-compatible reuse, a business component needs encapsulation in the following ways. The external world must not know anything about component internals, and the internals must not know anything about external components, other than allowing interested objects to register for notification of specific events or exception conditions.

The internals of a business component are made of other encapsulated business components. For example, when a purchase order passes through the membrane of the order entry business object, an internal component must see it, validate it, look up customer information, inventory availability and catalogue pricing, and build an order consistent with business rules and procedures. Each of these tasks is accomplished by embedded components, many of them communicating with external data sources.

External databases must be encapsulated as business objects components or reuse will not be easily

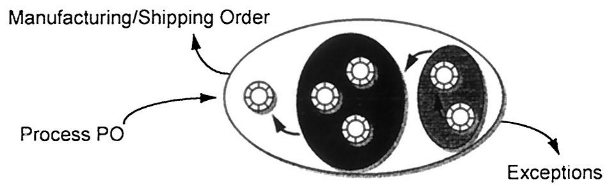

FIGURE *FOUR*

An order entry business object.

achieved. There must be a database access component that causes values from any kind of database to materialize as objects inside the business component. Whether object-oriented, relational, or other database access is required, a set of class libraries designed to automate this interface will result in a major savings in development resources [Sutherland et al. 1993].

An order entry business object will typically have multiple user interfaces. A clerk may be taking the order over the phone, entering purchase information, validating customer records and credit data, and reviewing an order for consistency and customer acceptance. Other users may require different presentation screens. User interfaces are difficult and time-consuming to build at the code level and should be encapsulated as separate objects that communicate by message passing to the order entry component.

A simple order entry client/server component has at least three large-grained components, one or more presentation objects, a business component that models the business process, and a database access component that shields the application developer from database access languages, database internals, and network communications (see Figure Five). Business object programmers focus their efforts on building business components, or large-grained business objects, which can be distributed easily on the network.

Distributing Business Object Components

System evolution will invariably distribute business object components to maximize network performance and processor utilization, and to ensure proper control, integrity, and security. With the widespread adoption of standards-based Internet technologies, distributed object systems have become the norm. Business reengineering implies implementing a distributed environment where components encapsulating business functionality can be migrated to nodes on the network that allow maximum flexibility, scalability, and maintainability of a business object component system.

Business objects made up of nested components allow distribution of these components across a network. Figure Six shows the logical application as a coherent set of nested client/server components; its deployment may include distributing subcomponents across multiple heterogeneous computing resources in dispersed locations. Thus, an application designed on one processor is scattered across a network at runtime.

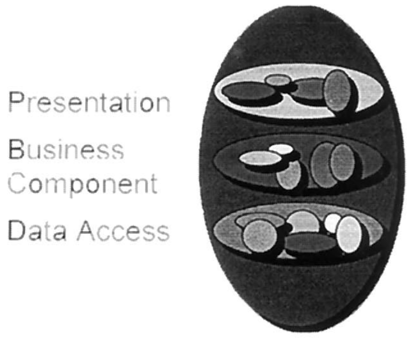

FIGURE *FIVE*
Client/server component.

	Before	After	Improvement Ratio
Process Steps	52	3	17.3
Staff	7	1	7
Time	9 weeks	9 min	2400

TABLE *ONE.* **Reengineering a Purchase Order Component**

Developers of business information systems have taken advantage of building applications with OLE components. At Object World in San Francisco, Allied Signal won the Computerworld Award for best object-oriented application of 1995 [VMARK 1995]. It reengineered the supply management business process (which took 52 steps to purchase a single part), so it now requires only three steps to complete the same transaction. The old process required seven people and took nine weeks to produce an approved purchase order. The new supply management specialist tool (SMST) allows one person to complete the same process in nine minutes for established suppliers with long-term agreements in place. In the case of new suppliers, where a request for quote (RFQ) is required, the process takes nine days. Table One summarizes these benefits.

In this example, process cycle time is reduced 2400:1 for established suppliers, and 5:1 for new suppliers. Cost reduction in operational staff is 7:1. Improvement in business efficiency leading to greater customer satisfaction and resulting market share is far greater than any reduced costs in operations overhead or development time. It is the main reason for using business object component design tools to assure success of business process reengineering practice.

Despite isolated success stories, Brodie [1997] reported, after a survey of 201 distributed object computing (DOC) applications worldwide, that this technology is not and will not be ready for prime time until vendors can deliver standards-based business object component frameworks.

> For the moment, DOC is in its infancy and does not meet industrial-strength requirements or the claims of its proponents . . . There are even very recent claims that a major breakthrough has occurred and that a DOC renaissance is upon us [Microsoft 1996]. Based on our experience, GTE has decided to halt the design, development, and deployment of DOC technology and applications. In part this relates to our recognition of the problems described . . . In part, it also relates to our pursuit of commercial off the shelf (COTS) applications for which the vendors are largely responsible for the issues raised . . . Following a significant study of and investment in DOC technologies and methodologies, we have concluded that the benefits do not currently warrant the costs to overcome the challenges described . . . The claims for increased productivity, re-use, and lowered costs cannot be achieved with other than very highly skilled staff who must work with immature technology and methods. We will continue to investigate the area and observe its progress and will be prepared to take full advantage of the technology when DOC is more mature. I look forward to a highly competitive market for the DOC infrastructure and highly competitive products.

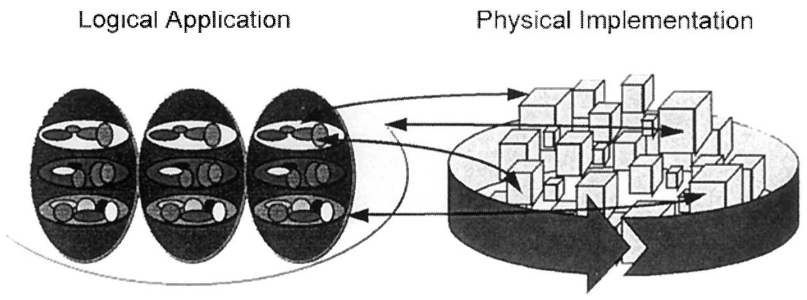

FIGURE *SIX*
Application business object with nested client/server components.

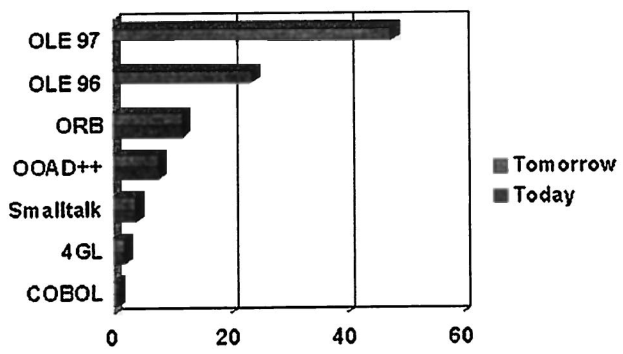

FIGURE *SEVEN*
Moore's Law for software as projected in 1994.

Challenge of Achieving Moore's Law for Software

Working with Capers Jones at Software Productivity Research, Sutherland [1995] in an analysis performed in 1993, using a database of thousands of projects on language environment productivity [Jones 1996], showed that 4GL environments are twice as productive in the real world as COBOL environments in a full lifecycle analysis.

Smalltalk doubled the productivity of a 4GL environment, but only if there was 80% reuse. Since the average amount of reuse by Smalltalkers was only 20% (not much better than C programmers at 15%) special tools had to be used to achieve this level of productivity.

In Figure Seven, OOAD+ is an example of a tool that guarantees 80% reuse, largely through automation, that enables roundtrip engineering from design to code and back, and is tightly integrated with user interface tools that allow nonprogrammers to develop user interfaces, and generates runtime components from design. Achieving these objectives, consistent with the X3H7 design targets noted previously in Figure 3, doubles the productivity of a Smalltalk environment.

The ORB bar in Figure Seven points to an OOAD+ environment that automates mapping between application objects and their storage in a relational database. Sutherland observes that in multiple projects in heterogeneous business environments, hand-coding object/relational mapping absorbed more than 30% of development resources.

Sutherland estimated that by 1996 it would be possible to buy 50% of an application as off-the-shelf components, effectively doubling productivity. By 1997, early adopters would be buying 50% of the application as external components and reusing internally generated components for another 25% of the application, effectively doubling productivity on an annual basis, and beginning to achieve Moore's law for software. Cox's vision of software as IC chips could be realized in such a component environment.

Success in achieving the projections above has occurred only in isolated instances. At OOPSLA '98, Zincke [1997] gave a report showing a production system that was developed at the rate of 7.52 function points per person-day, an order of magnitude faster than the industry average. Widespread success has been limited by redeployment of software tools for Internet applications, effectively forcing the industry to repeat the lessons of the last decade of Smalltalk development, and the lack of standard component environments in which to build domain-based object-oriented frameworks.

OMG BOMSIG

By mid-1995, BOMSIG [1995] completed its second revision of a business application architecture, noting that

> with a system comprised of a set of cooperative business objects, the outmoded concept of monolithic applications becomes unnecessary. Instead, your information system is comprised of semi-autonomous but cooperative business objects which can be more easily adapted and changed. This type of *component assembly and reuse* has been recognized as a better way to build information systems.

The consensus notion of a business application architecture evolved to what is now the standard three-tier architecture with business objects in the middle tier. A distinction began to be made between business objects as entities and business objects as processes (see Figure Eight).

Towards the end of 1995, the business application architecture concepts had evolved into the release of OMG Common Facilities RFP-4: Common Business Object and Component Interoperability Facility (later known as the Business Object Facility (BOF)). The thrust of the RFP was to begin to build a layer on top of the OMG CORBA infrastructure to enable a plug-and-play environment.

CORBA infrastructure provides an environment for communication among distributed objects, but 100% of a business application has to be hand-coded in this environment. With a component architecture, it should be possible to buy 80% of the application components and have to write only 20% of the code. A component interoperability facility could provide generic superclasses for business objects. Common business objects crossing domains could be standardized, and domain frameworks developed to use both the common business objects and the component interoperability facility (later the BOF).

It's Never as Easy as It Looks

OMG Domain Task Forces were created at the end of 1995 to emphasize the importance of user organizations and vertical domain software to the future of OMG. BOMSIG metamorphosized into the OMG BODTF with the authority to issue its own RFPs, and Common Facilities RFP-4 evolved into BODTF RFP-1.

The dominant response to the business object facility portion of BODTF RFP-1 matured, after several collaborative efforts, into the BOCA. At the time of

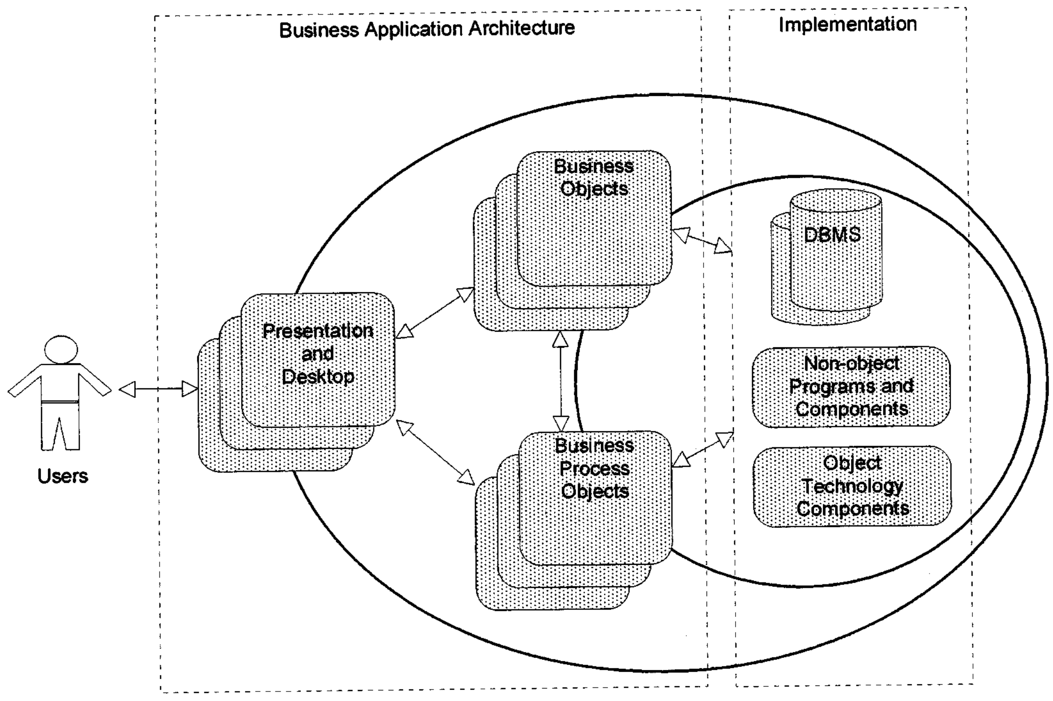
Business Application Architecture
Implementation
Business Objects
DBMS
Presentation and Desktop
Non-object Programs and Components
Users
Business Process Objects
Object Technology Components

FIGURE *EIGHT*
Business application architecture revision 2, OMG 95-04-01.

this writing, BOCA was approved by the OMG Architecture Board and is in the voting process to become an OMG adopted technology.

In order to bring BOCA to the voting process, several phases of integration with and definition of other OMG standards had to evolve.

It was necessary to harmonize BOCA with parallel work in multiple areas:

UML. The Unified Modeling Language for object-oriented analysis and design became an OMG adopted technology in 1997, through the united efforts of Rational Software, Microsoft, Hewlett-Packard, Oracle, Sterling Software, MCI Systemhouse, Unisys, ICON Computing, IntelliCorp, i-Logix, IBM, ObjecTime, Platinum Technology, Ptech, Taskon, Reich Technologies, and Softeam corporations [OMG 1997a].

MOF. The meta-object facility defines a simple meta-metamodel with sufficient semantics to describe metamodels in various domains starting with object analysis and design. Integration of metamodels across domains is required for integrating tools and applications across the lifecycle using common semantics. OMG adopted technology [OMG 1997b] represents the integration of efforts currently underway by the Cooperative Research Centre for Distributed Systems Technology (DSTC), IBM, International Computers Limited, Objectivity, Oracle, System Software Associates, and Unisys in the areas of object repositories, object modeling tools, and meta data management in distributed object environments.

CORBA Components. The current OMG RFP for CORBA components begins [OMG 1996],

While abstract interfaces are at the heart of object-oriented technology, they are only one dimension of the complex space within which distributed object applications are designed and built. In recent years, the concept of component technology has emerged as a more complete mechanism for expressing object-oriented software entities and assembling them into applications. Two prominent examples of component models are JavaBeans, and ActiveX Controls.

The Gang of Four (IBM, Netscape, Oracle, and Sunsoft) initiated this effort in 1997 stating that [OMG 1997c]

a component framework must provide a standard way to ask questions at *design time* as well as run time about the external interfaces, presented as *methods, properties* and *events.* The CORBA component model must support *interface composition,* so that components and the applications that use them are decoupled, and can evolve independently while maintaining compatibility. It must be possible to pass *component state and methods by value* so that native language interfaces can be mapped naturally into CORBA distributed operations. The CORBA component infrastructure must interoperate with existing non-proprietary component standards, such as *JavaBeans.* The component framework must support the Internet deployment of multi-tier applications, with *URL naming* of CORBA objects, and easy access to CORBA objects and services from *Java.*

The way components are composed currently and "snap together" is left to the implementation (Java-

Beans is one such implementation). Without this piece there are no true "plug-and-play" business components. When CORBA components come on-line, BOCA IDL mappings can be extended to utilize CORBA components, achieving true "plug-and-play.") Without the BOCA, CORBA components provide a way to "snap together" implementations, but no business application architecture into which to snap them.

CDL. BOCA proposes a component definition language (CDL), a way to write down, in a textual form, business object specifications that use the metamodel. With a way to express the metamodel textually, BOCA provides a way for writing publishable business object standards.

IDL. BOCA IDL provides the mapping from the metamodel to OMG IDL interfaces. IDL interfaces necessarily contain technology details that need to be shielded from the business developer but are necessary for interoperability. Given a particular business object model, interfaces must be expressed in a consistent way that supports the underlying framework and interoperability. IDL mapping specifies the form and content of business object interfaces based on the metamodel.

BOF Interoperability Specification specifies a mapping between CDL and CORBA IDL interfaces that business objects have to support and use to achieve technical interoperability. Standardization of BOF interoperability specification depends on clarifying interfaces to CORBA services and its relation to future adoption of a CORBA component specification standard.

CORBA + IDL. Distributed business objects would not be possible without the underlying distributed object infrastructure. The CORBA metamodel, ORB, and IDL are the bases on which the BOCA framework is built.

CORBA Services. The framework is supported by the library of CORBA services uses by business objects in well-defined ways.

Frankel [1998] notes that the following key points outline the relationship of BOCA and UML to CORBA.

(1) UML allows the semantics of a type to be described, not merely the syntax. A type can be specified to be abstract, invariant rules can be specified for the type, constraints can be specified for attributes of the type, pre- and post-conditions can be specified for the type's operations, and so on.

(2) Although it is an OMG specification, UML is independent of CORBA, COM, Enterprises Java Beans (EJB), or any middleware. The intention of its creators is to provide mappings to the various middleware technologies.

(3) CORBA IDL and its (implied) underlying metamodel do not support the expression of the semantics of a type. This results in massive loss of semantic information when mapping a UML-based business domain model to CORBA.

(4) BOCA is a metamodel derived from (i.e., is a superset of) the CORBA metamodel, and CDL therefore is a proper superset of CORBA IDL. BOCA and CDL support the expression of a type's semantics. Thus, BOCA allows a UML-based business model to be mapped to CORBA without loss of semantic information.

BOCA Current Implementation Status

Currently, software is available from Data Access Technologies [BOCA CDL 1998] that generates business object components into the IBM San Francisco Project Java Framework. An annotated UML design autogenerates CDL and MOF metadata, and can be used to generate Java classes in IBM's proprietary framework. Enterprise JavaBean frameworks will be available soon and BOCA will generate Enterprise JavaBeans code.

OMG ORBOS is receiving responses to the CORBA component facility RFP. Initially, IBM, Netscape, Oracle, and Sunsoft proposed that this be JavaBeans-based. Multiple competing proposals are now being reconciled. When the component facility is available, BOCA will generate CORBA components.

Why I Love the OMG

The harmonization of multiple OMG task force standardization efforts in widely disparate technologies to provide a standard infrastructure for generation of business objects from design specifications is a monumental task. The OMG is the only organization that can mobilize the best technical resources of over 800 leading software vendors and user organizations worldwide to bring such an effort to completion. When this effort is complete, we will have a standard analysis and design language, a standard business specification language, a standard plug-and-play component environment, a standard meta-object facility for designs, applications, and repositories, and standard interfaces for distributed object environments.

Tools will be provided to generate business systems from design into heterogeneous distributed runtime environments, positioning the software industry for the twenty-first century and launching the first global effort to break down the barriers to implementing Moore's law for software. **SV**

References

BOCA CDL, 1998. Development kit for OMG IDL. Prerelease revision 0.51, Data Access Technologies, Inc.

BRODIE, M. 1997. The emperor's clothes are object oriented and distributed. GTE Laboratories.

CASANAVE, C. 1995. OMG business application architecture white paper. OMG bomsig/95-4-1.

COX, B. 1986. *Object-Oriented Programming: An Evolutionary Approach.* Addison-Wesley, Reading, MA.

DIGRE, T. 1997. Business application components. In *Business Object Design and Implementation: OOPSLA '95 Workshop Proceedings,* J. Sutherland, D. Patel, C. Casanave, G. Hollowell, and J. Miller, Eds.

FRANKEL, D. 1998. UML, BOCA, and MOF: Presentation to the lifesciences DTF at the Orlando meeting. OMG Document lifesci/98-06-02.

GEERTS, G. 1997. The timeless way of building accounting information systems: The 'activity' pattern. In *OOPSLA Workshop on Business Object Design and Implementation.*

JACOBSON, I., ERICSSON, M. and JACOBSON, A. 1995 The Object Advantage: Business Process Reengineering With Object Technology. Addison-Wesley, Reading, MA.

JONES, C., 1996. Programming languages table, release 8.2. Software Productivity Research, March.

KENT, W. 1993. X3H7 objectives and operations. X3H7-93-023, 18 January.

LOVE, T. 1993. Object Lessons: Lessons in Object-Oriented Development Projects. SIGS Publications, Denville, NJ.

MANOLA, F. Ed. 1997. Object model features matrix. X3H7-93-007v12b, 25 May.

META Group, Inc. 1995. Making the case for use case. *Advanced Information Management,* File 324, 13 February.

MICROSOFT 1996. The Renaissance of distributed computing. White Paper, Nov. (www.microsoft.com/pdc/html/p&s.htm).

MORAVEC, H. 1998. *Robot, Being: Mere Machine to Transcendent Mind.* (In press.)

NCITS TECHNICAL COMMITTEE H7 1998. Object information management home page. http://enterprise.systemhouse.mci.com/X3H7/default.html.

ODP 1998. Open distributed processing home page. http://enterprise.systemhouse.mci.com/WG7/default.html.

OMG 1996. CORBA component model request for proposal. OMG Document: orbos/96-06-12.

OMG 1997a. UML proposal to the object management group in response to the OA&D task force's RFP-1, Version 1.1, Rational Software, Microsoft, Hewlett-Packard, Oracle, Sterling Software, MCI Systemhouse, Unisys, ICON Computing, IntelliCorp, i-Logix, IBM, ObjecTime, Platinum Technology, and Ptech, OMG Document: 97-08-11.

OMG 1997b. Meta object facility (MOF) specification joint revised submission. Cooperative Research Centre for Distributed Systems Technology (DSTC), IBM, International Computers Limited, Objectivity, Oracle, System Software Associates, and Unisys, OMG Document: 97-10-2.

OMG 1997c. CORBA Component Imperatives. IBM Corporation, Netscape Communications Corporation, Oracle Corporation, Sunsoft, Inc., OMG Document: orbos/97-05-25.

OMG 1998a. Business object component architecture (BOCA), revision 1.1, document bom/98-01-07.

OMG 1998b. Object management group business object domain task force home page. http://www.dataaccess.com/bodtf/.

OMG BOMSIG 1995. OMG business application architecture, revision 2. OMG 95-04-01.

OOPSLA 1998. OOPSLA workshop for business object designed and implementation home page. http://www.jeffsutherland.org/oopsla98/index.html.

SUTHERLAND, J. 1994. ANSI X3H7 standardization targets. X3H7-94-35, 24 September.

SUTHERLAND, J. 1995a. Business objects in corporate information systems. *ACM Comput. Surv. 27,* 2 (Jun.) 274–276.

SUTHERLAND, J. 1995b. An executive overview to object technology (tutorial). *In Object World, Boston, Sydney, San Francisco, Frankfurt and Executive Symposium on Object Technology* (Toronto).

SUTHERLAND, J.V., Pope, M. and Rugg, K. 1993. The hybrid object-relational architecture (HORA): An integration of object-oriented and relational technology. In *Proceedings of the 1993 ACM/SIGAPP Symposium on Applied Computing* (Indianapolis, Feb. 14–16, E. Deaton et al., Eds. ACM Press, New York 326–333.

TAYLOR, D. 1992. *Object-Oriented Information Systems: Planning and Implementation.* Wiley, New York, 320–322.

VMARK SOFTWARE 1995. Allied Signal Company wins the Computerworld Object Application Award at Object World. Press Release, August 21.

ZINCKE, G. 1997. How to achieve 7.52 function-points per person-day with object technology. Addendum to conference proceedings. In *OOPSLA Twelfth Annual Conference* Oct. 5–9, ACM Press, New York.