The Emergence of a Business Object Component Architecture

Jeff Sutherland, IDX Systems Corporation

Abstract

Object technology, a necessary but not sufficient condition for software reuse, requires an infrastructure that supports plug compatible Business Object Components for fast and flexible delivery of products to the marketplace. The Object Management Group (OMG) Business Object Domain Task Force (BODTF) was the initial focal point for standardization of a Business Object Component Architecture (BOCA).[1] Priming this effort required joint work with the Accredited Standards Committee X3H7 Object Information Management[2], and their joint sponsorship of the OOPSLA Business Object Component Design and Implementation Workshop for the years 1995-99. Emergence of W3C XML standards will further enhance BOCA and enable a distributed business object system that provides interoperability between disparate enterprise applications on the Web.[3]

1 Background

Standardization of a framework for a Business Object Component Architecture requires coordination between ANSI/ISO standards bodies and the Object Management Group. The effort is informed by academic, government, and industry research papers presented annually at the OOPSLA Workshop on Business Object Component Design and Implementation.

1.1 X3H7 Object Information Management

The International Standards Organization (ISO) is extending the international standard Reference Model for Open Distributed Processing (RM-ODP[4]) to incorporate enterprise modeling. RM-ODP compliance is a requirement for OMG standards submissions, and is the primary linkage between the ISO and the OMG.

X3H7 (now part of NCITS Technical Committee T3: Open Distributed Processing) is the U.S. technical committee for this international work item and is tasked with the following:

- Refine the RM-ODP enterprise language, explicating the relationship of an enterprise specification of a system to other RM-ODP viewpoint specifications of that system to enable the RM-ODP to be used for specification of object-based application architectures.
- Ensure that the enterprise language together with other RM-ODP 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 the RM-ODP viewpoint languages to specify a concrete application architecture.

1.2 OMG Business Object Domain Task Force (BODTF)

With a membership of over 800 software vendors, software developers, and end users, the OMG goal is to establish CORBA as standard middleware through its worldwide standards specifications: CORBA/IIOP, Object Services, Internet Facilities and Domain[5] 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. The goal is 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:

- the 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,
- the 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.

1.3 OOPSLA Workshop on Business Object Component Design and Implementation[6]

OOPSLA (Object-Oriented Programming, Systems, Languages, and Applications) has been the leading object technology conference for more than a decade. There are a wide variety of participant-driven workshops, tutorials, invited speakers, panels, debates, and technical papers

capturing the latest in both research and development experiences.

The OOPSLA Workshop on Business Object Component Design and Implementation is jointly sponsored by X3H7 and the OMG BODTF for the purpose of soliciting technical position papers relevant to the design and implementation of business object component systems.

The goals of the OOPSLA Workshop on Business Object Component Design and Implementation 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 component systems, particularly systems in which workflow patterns and the REA accounting model[7] are basic building blocks for business applications.
- Contribute to emerging architectures for net-based applications, particularly those that integrate business object components such as web servers; object and relational databases; and XML technologies.
- Pursue issues raised by papers on heterogeneous distributed workflow systems presented in previous workshops; 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.
- Share experience reports on business object component systems both in development and in production.

2 Why Business Object Component-Based Development?

Gradual improvements in productivity and enhancements in quality are no longer enough to maintain market leadership in a global environment. Time to market of new products and rapid evolution of old products and applications are key success factors. In 1995, X3H7 and the OMG BODTF joined forces to initiate a radical change in software development environments, a change that would take years to specify and 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, the software applications that run the 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 and pushed it in the direction of global, distributed object computing, the target 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 Object Components that can both simulate corporate procedures and translate smoothly into software objects. Well-designed Business Object Component implementations can be easily modified as the business changes. In particular, if software implementation can be generated 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 is performed, and
- a model-driven approach is used to realign the business strategy, processes, and technology.

A multi-layered, object-oriented blueprint of the enterprise can drive the refocusing, realignment, and reorganization of the business.[8] Current attempts to implement this process under the rubric of Business Process Reengineering (BPR) have been largely ineffective due to difficulties in changing monolithic organizations, processes, and information systems.

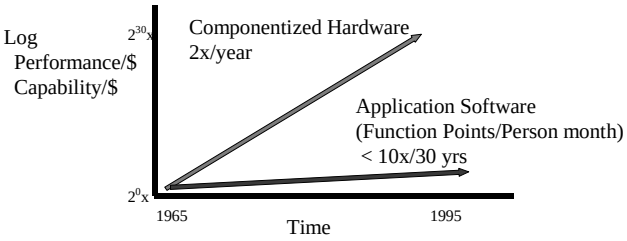

Figure 1: Hardware Price/Performance vs. Software Price Performance[9]

Figure 1 demonstrates that enhancing the productivity and performance of integrated circuits (IC) has led to exponential growth in computing power over the past thirty years (Moore's Law), while software development productivity has increased only one order of magnitude. Most experts, including Moore himself, expect this trend to hold for at least another two decades."[10]

Moravec[11] has more recently observed that information handling capacity in computers has been growing about ten million times faster than it did in nervous systems during human evolution. Computing power doubled every two years in the 1950s, 1960s and 1970s, doubled every 18 months in the 1980s, and is doubling each year in the 1990s.

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 the lack of automated software construction and failure to achieve large scale reuse of software components in business applications.

The software productivity problem is a core issue for the X3H7 and the OMG BODTF as they assess how to maximize the impact of software standards development on the worldwide business community.

2.1 X3H7 Contributions

Document X3H7-93-23, Objectives and Operations[12], provided guidelines for work of the X3H7 during the period 1993-96, and included:

- Developing liaisons with groups working on object oriented standards and review their progress.
- Completing the X3H7 Object Model Features Matrix document[13] that defines in some detail the characteristics of object models being proposed by different groups.
- Developing an X3H7 reference document based on the Features Matrix to present to targeted groups working on object model standards.
- Based on importance of each liaison group and the timing of each group in the standards development process, presenting formal proposals to these groups to facilitate harmonization of object model standards and enhance interoperability of distributed object systems.
- Developing scenarios of problems arising in the interaction of object systems to clearly illustrate the technical issues involved 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, the X3H7 Object Model Features Matrix developed an analysis of issues involved in harmonizing object models. This showed that competing object models provided not only different structures, but often different semantics underlying the concepts that supported these structures.

Interoperability of object models requires understanding the structure and semantics of commonly used object-oriented frameworks and the interfaces between these development frameworks. Object models must interoperate within widely used 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++). 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 2 illustrates the views of X3H7 at that time, updated to reflect the impact of the Web.

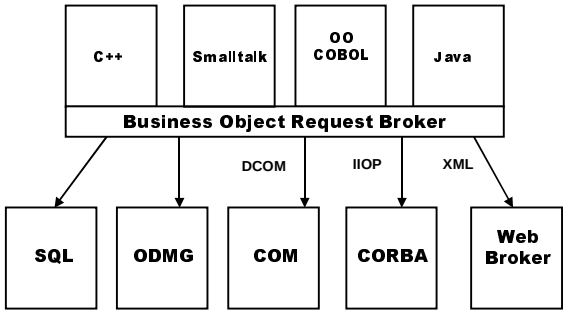

Figure 2. ANSI X3H7 Standardization Targets. 24 Sep 1994[14], updated 16 Feb 1999

The widespread adoption of the Internet since 1995 has 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 objective of X3H7. In 1998, a World Wide Web Consortium (W3C) proposal for a WebBroker combined with an evolving set of standards for an XML distributed object computing infrastructure, added a fifth environment requiring interoperability.[15]

Even before the rapid growth of the Internet, there was a consensus that application developers should be 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 between development frameworks. Figure 3 shows the X3H7 conceptual view of this problem.

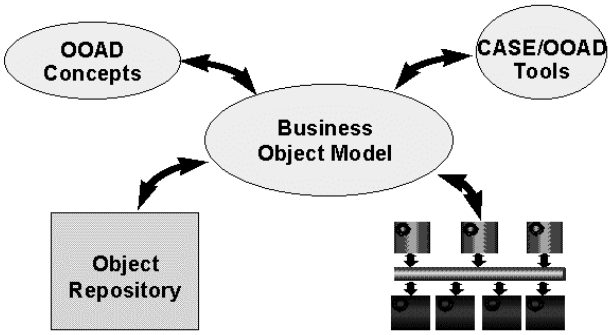

Figure 3. ANSI X3H7 Standardization Targets. 24 Sep 1994, Injecting a Business Model into a Runtime Environment

X3H7 members participating in OMG and other standards bodies began driving the agenda of object model harmonization in multiple organizations. They were key contributors to the ISO standard RM-ODP, the distributed processing reference model with which OMG technologies must conform. They also agreed to co-sponsor, with the OMG BODTF, a Business Object Component Design and Implementation Workshop at the 1995 Conference on Object-oriented Programming, Systems, Languages, and Applications (OOPSLA), in order to draw research contributions into the drive for common Business Object Component standards.

2.2 OMG BODTF Contributions

In 1994, at the OMG BODTF, Sutherland[16] presented his findings on key issues in building life cycle object-oriented development environments for business objects to standards organizations, including the OMG Business Object Management Special Interest Group (BOMSIG, now BODTF). Simultaneously, Cory Casanave, 1995 Chair of the OMG BODTF, edited a BOMSIG Business Application Architecture White Paper[17] and later OMG Common Facilities RFP4: Business Object Facility and Common Business Objects.[18] These became the reference documents for the BOCA standardization efforts to follow.

2.3 Business Objects as Reusable Components

Objects alone 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[19] 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 of design, coding, and testing. Detailed analysis of this project showed 92% fewer lines of code, 93% fewer staff months of effort, 82% less development time, 92% less memory needed to run, and no performance degradation.[20]

While there are many isolated projects that used object technology to achieve dramatic productivity gains during the past decade, this success has not translated into broad improvements across the software industry. In 1995, META Group reported that, "despite the promise of reusable objects, most IT organizations have realized a scant 10%-30% productivity improvement from object technology (OT)."[21] Failure to achieve larger productivity gains was attributed to:

- data-centric, task-oriented application 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 software implementation of these components. Using an object-oriented development methodology yields quick time to market and object-oriented design allows for 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 in the marketplace.

3 The Need for a Business Object Component Architecture

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

Dynamic change requires the reuse of chunks of business functionality. A BOCA must support reusable, plug-compatible business components. Historically, the two primary strategies used for implementing client/server systems to support reengineering of business processes where visual 4th Generation Languages and classical object technology. While both of these approaches are better than COBOL, neither of them can effectively implement plug and play Business Object Components, nor do they allow systems to rapidly evolve with changing business requirements.

What is needed is a component standard that can be implemented with object technology. These components need to plug and play into standard frameworks. The concept of a BOCA incorporates objects, components, and frameworks along with the essential tools and infrastructure to allow autogeneration of full blown distributed object systems from design artifacts. Furthermore, these systems can be dynamically updated by changing the design and regenerating code, allowing synchronization of software systems with the constantly changing business processes seen in most enterprises.

3.1 Building Business Object Components

A group of objects is the ideal unit of reuse. These groups of objects 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[22] (a rigorous definition of a software module) has shown that there is a minimal design specification for a plug compatible component.[23]

Consider a typical client/server application like 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 to another subsystem or, alternatively, an exception condition is raised if the Purchase Order is not valid for processing (see Figure 4).

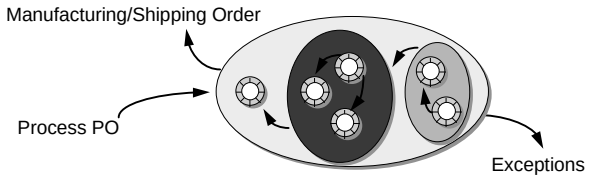

Figure 4. An Order Entry Business Object

To support plug-compatible reuse, a business component needs encapsulation in the following ways. The external world should not know anything about component internals, and the internals should 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 that is 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 should be encapsulated as business objects components or reuse cannot be easily achieved. There must be a resource tier with 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.[24]

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. Today, much of this process can be automated. They 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 four large-grained components, one or more presentation objects, a workspace component that manages context during the creation of a transaction, an enterprise component that models transaction state, and a database access component that shields the application developer from database access languages, database internals, and network communications (see Figure 5).

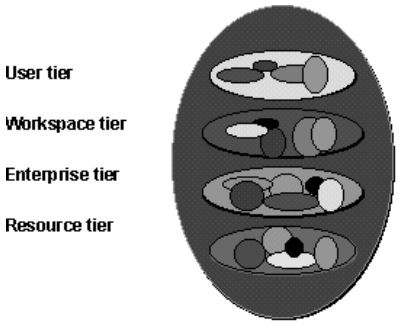

Figure 5. Business Object Component

Herzum[25] defines a business object component as follows:

A business component represents the software implementation of an autonomous business concept or business process. It consists of all the software artifacts necessary to express, implement and deploy a given business concept as an autonomous, reusable element of a larger information system.

Business component programmers focus their efforts on the software implementation of a concept which is a composition of software artifacts, including distributed components.

Szyperski[26] takes a complementary approach:

A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component is deployed independently and is subject to composition by third parties.

The first definition focuses on the concept of a component within an enterprise system. The second surfaces the key issues for components used across enterprise systems.

3.2 Distributing Business Object Components

System evolution will invariably distribute these Business Object Components to maximize network performance and processor utilization, and to ensure proper control, integrity, and security of information. 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.

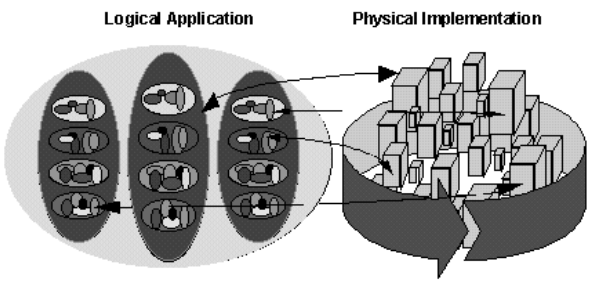

Figure 6. Application Business Object with Nested, Tiered Components

Business objects made up of nested components allow distribution of these components across a network. Figure 6 shows the logical application as a coherent set of nested client/server components. Deployment of this large-grained object 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 run time.

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.[27] They reengineered the Supply Management Business Process that 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 1 summarizes these benefits.

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

Table 1: Reengineering a Purchase Order Component

In this example, cycle time of the process is reduced 2400:1 for established suppliers, and 5:1 for new suppliers. Cost reduction is operational staff is 7:1. The impact of improvement in business efficiency leading to greater customer satisfaction and resulting market share is far larger than reduced costs in operations overhead or development time. It is the prime objective for use of Business Object Component design tools to assure success of Business Process Reengineering practice.

Despite isolated success stories, Brodie[28] reported in 1995, after a survey of 201 distributed object computing (DOC) applications worldwide, that this technology was not and would not be ready for prime time until vendors could 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.[29] 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."

Progress has been made towards addressing Brodies' concerns since 1995. After the turn of the millenium, BOCA standards and Web technologies will resolve these problems and make distributed object computing easily accessible to the average developer. Significant events leading up to this capability are outlined below.

4 Achieving "Moore's Law for Software"

Working with Capers Jones at Software Productivity Research, Sutherland did an analysis in 1993 using a database of thousands of projects on productivity of language environments.[30,31] This study showed that 4GL environments were twice as productive in the real world as COBOL environments in a full life-cycle analysis.

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

In Figure 7 below, OOAD+ is an example of a tool that guarantees 80% reuse largely through automation, enables roundtrip engineering from design to code and back, 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 7 refers to an OOAD+ environment that automates the mapping between application objects and relation database storage of these objects. Sutherland observed that in multiple projects in heterogeneous business environments, hand coding object/relational mapping absorbed more than 30% of development resources.

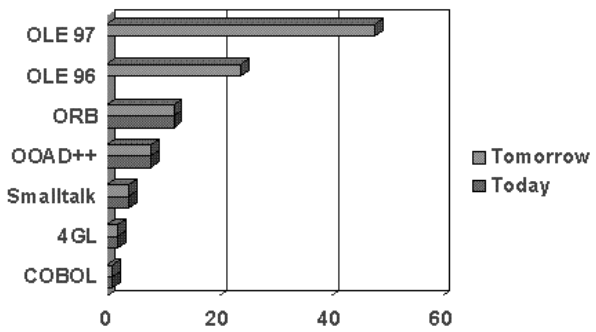

Figure 7: Moore's Law for Software

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. Brad Cox's vision of software as IC chips could be realized in such a component environment.

Successes in achieving these goals have occurred on a isolated basis. At OOPSLA'98, Zincke[32] gave an experience 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 industry average. Widespread achievement of these results has been limited by redeployment of software tools for Internet applications, effectively forcing the industry to repeat the lessons of the last two decades of Smalltalk innovation, and the lack of standard component environments in which to build domain-based object-oriented frameworks.

4.1 OMG BOMSIG Business Application Architecture and Common Facilities RFP-4

By mid-1995, BOMSIG completed its second revision of a Business Application Architecture,[33] 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 that 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 had evolved to what is now the standard three-tier architecture with Business Objects in the middle tier. A distinction began to be drawn between Business Objects as entities and Business Objects as processes (see Figure 8).

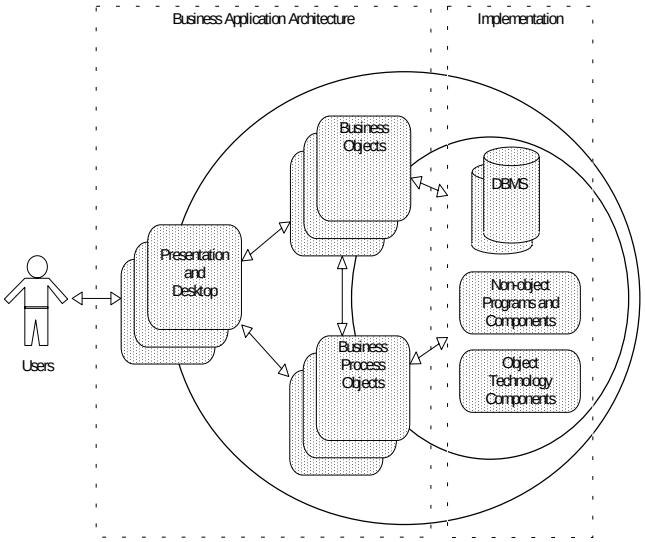

Figure 8. Business Application Architecture Revision 2. OMG 95-04-01

Towards the end of 1995, the Business Application Architecture concepts had evolved into the issuance 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. Figure 9 became the central view of the problem:

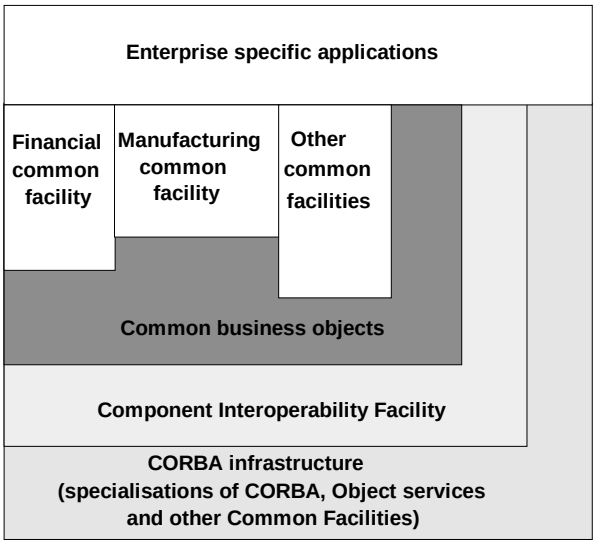

Figure 9: Business Application Architecture. CFRFP-4, OMG 95-12-13

The CORBA infrastructure provides an environment for communication between distributed objects. However, 100% of a business application needs to be hand coded in this environment. It should be possible with a component architecture to buy 80% of the application components and only have to write 20% of the code. A Component

Interoperability Facility would provide generic superclasses for business objects. Common business objects crossing domains would be standardized, and domain frameworks would be developed to use both the Common Business Objects and the Component Interoperability Facility (later the Business Object Facility (BOF).

4.2 It's Never as Easy as it Looks

At the end of 1995, OMG Domain Task Forces where created to emphasis the importance of user organizations and vertical domain software to the future of OMG. BOMSIG metamorphized into the OMG Business Object Domain Task Force (BODTF) with the authority to issue its own RFPs and Common Facilities RFP-4 evolved into BODTF RFP-1.

The leading response to the Business Object Facility portion of BODTF RFP-1 matured, after several collaborative efforts, into the Business Object Component Architecture (BOCA). BOCA was approved by the OMG Architecture Board but failed to get the required vote of the entire OMG membership required for adoption as an OMG technology.

The problems with adopting a standard in 1998 revolved around several phases of integration with and definition of other OMG standards. It was necessary to harmonize BOCA with parallel work in multiple areas:

- UML - The Unified Modeling Language (UML) 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.[34]
- MOF - The Meta-Object Facility[35] defines a simple meta-metamodel with sufficient semantics to describe metamodels in various domains starting with the domain of object analysis and design. Integration of metamodels across domains is required for integrating tools and applications across the life cycle using common semantics. This OMG Adopted Technology 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 corporations 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, "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." [36]

4.3 The CORBA Component Conundrum

While BOCA achieved a reasonable level of integration with UML and the MOF in 1998, the lack of clear definition of a CORBA component model caused some OMG members to question whether a standard should be approved when it depended on an underlying component model that was still in the process of initial specification.

The Gang of Four (IBM, Netscape, Oracle, and Sunsoft) initiated the CORBA component effort in 1997 stating that "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."[37]

Currently, the way components are composed and "snap together" is left up to the implementation (Java beans being one such implementation). Without a standard component model 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 to snap them into.

In early 1999, the leading proposal for a CORBA Component Model is similar to Enterprise Java Beans with some significant differences. It is likely that the industry will be faced with three component models to deal with (ActiveX, JavaBeans, and a CORBA Component).

4.4 BOCA Current Implementation Status

The BOCA consists of several key concepts:[38]

CDL: BOCA proposes a Common Specification Language (CDL), a way to write down, in a textual form, business object specifications that use the meta-model. By specifying the meta-model in textual form, BOCA facilitates the creation of publishable business object standards.

IDL Mapping: The BOCA IDL mapping capability provides the mapping from the meta-model 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. The IDL mapping specifies the form and content of business object interfaces based on the meta-model.

Interoperability Framework: These are the CORBA interfaces that business objects have to support and use to achieve technical interoperability. The framework provides the technical underpinnings for BOCA objects using the meta-model. Standardization of the Interoperability Framework depends on the future adoption of a CORBA Component Specification standard.

Distributed business objects would not be possible without the underlying distributed object infrastructure. The CORBA meta-model, ORB and IDL are the basis on which the BOCA framework is built. The framework is supported by the library of CORBA services used by business objects in well defined ways.

Software is available from Data Access Technologies[39] that will generate business object components into the IBM San Francisco Project Java Framework. An annotated UML design autogenerates CDL and MOF metadata. These products can be used to generate Java classes in IBM's proprietary framework. Enterprise Java Bean frameworks will be available soon and BOCA will generate Enterprise Java Bean code. When the CORBA Component Facility is available BOCA will generate CORBA components.

4.5 Work in Progress

In 1999, the OMG BOCA effort refocused into a Business Object Initiative (BOI) addressing the characteristics of object-oriented analysis and design for enterprise computing systems. The OMG BOI Roadmap document noted that enterprise systems are closely tied to the business processes of an institution. As a result, the design of these systems will closely reflect the business domain of the enterprise. Several important characteristics need to be addressed in the design process:

- "The object types defined represent artifacts of the entities, processes, rules and events that occur in the enterprise business environment.
- "The number of object types is very large, sometimes numbering in the hundreds or thousands.
- "The associations among the myriad object types are numerous and the semantics of these associations are particularly crucial to ensuring the ability of different components to interoperate. The greater incidence in enterprise-scale distributed object systems of interactions among objects implemented by different development teams further magnifies the need for precision in this regard."[40]

Since the OMG had approved the Unified Modeling Language (UML) as a standard in 1998, BOI proponents argued that the important question was what, if anything, needed to be added to UML to properly specify a business object component system. What needed to be added to specify systems that needed to respond to high level business rules and events? What was important for building loosely coupled components that would be created by multiple development teams? How would UML be used to specify plug and play component models? These questions led to creation of four Request for Proposals (RFPs):

RFP 1: A UML Profile for Enterprise Distributed Object Computing (EDOC). This UML profile[41] must support specification at the design level of information required by all of the emerging OMG industry component models. It must distinguish between business process objects, business entity objects, and business rule objects. And it must support modeling business events.

RFP 2: A UML Profile for CORBA. This RFP calls for a UML profile supporting analysis and design semantics that are unique to CORBA.

RFP 3: A Human-Readable Textual Notation for the UML Profile for EDOC. This RFP calls for a Human Usable Textual Notation (HUTN) that would allow object models expressed in terms of the UML Profile for Enterprise Distributed Object Computing to be expressed textually.

RFP 4: A Mapping to CORBA of the UML Profile for EDOC. This RFP mandates that the mapping specify how an object model expressed in terms of the UML Profile for EDOC would be expressed in CORBA terms, including how the CORBA objects would use the CORBA services.

These RFPs allow for competing proposals for UML specification of business object component systems and human readable text manifestation of those specifications, essential requirements for automating the creation of components from design. Presumable, portions of the original BOCA specification will be resubmitted for standardization in an environment where there is broader understanding within the OMG of BOCA issues and the opportunity to be fully consistent with the UML standard and the emerging OMG component standard.

5 Future Directions – XML and the WebBroker

In January 1998, Manola presented "Towards a Web Object Model" to the OMG Internet Special Interest Group.[42] This paper was developed under a DARPA contract to determine how the Web would become a distributed object computing environment. As the editor of the ANSI X3H7 Object Features Matrix, an extensive comparative analysis of object models developed over the last decade, Manola was ideally suited to the task.[43]

Conclusions were that the standardization of the Extensible Markup Language (XML), associated XML infrastructure standards, and the the World Wide Web Consortium's (W3C) Document Object Model (DOM), would provide a native Web environment for distributed XML objects. The important point is that XML is not just a document model. It is the basis for a distributed object model which includes messages, state, methods, and object-oriented interfaces.

In April, 1998, a proposal was submitted to the W3C for a WebBroker as a lightweight alternative to COM and CORBA distribution models. The abstract stated:

A "necessary technological foundation exists to create a unified distributed computing model for the Web encompassing both document publishing and distributed software object communication. For lack of a better term, this model is referred to here as "WebComputing." Applications designed for the WebComputing environment exhibit a mix of features from both the Web publishing and the traditional distributed objects paradigms, blended into a unified model. The goal of this model is to extend the current Web application model such that the benefits of distributed object computing systems such as the OMG's CORBA and Microsoft's COM+ can be realized in a Web native fashion. The objective is to have a system which is less complicated than the above mentioned distributed computing systems and which is more powerful than HTML forms and CGI."

A Resource Description Framework (RDF) Model and Syntax Specification became a W3C Proposed Recommendation in January 1999. Together with Uniform Resource Identifiers (URI) which provide object identity and Namespaces, RDF provides the capability to implement an object model of an interoperable, distributed Web objects. It enables interoperability between applications that exchange machine-understandable information on the Web.

"RDF can be used in a variety of application areas; for example: in resource discovery to provide better search engine capabilities, in cataloging for describing the content and content relationships available at a particular Web site, page, or digital library, by intelligent software agents to facilitate knowledge sharing and exchange, in content rating, in describing collections of pages that represent a single logical "document", for describing intellectual property rights of Web pages, and for expressing the privacy preferences of a user as well as the privacy policies of a Web site. RDF with digital signatures will be key to building the "Web of Trust" for electronic commerce, collaboration, and other applications."[44]

The Web is driving many companies towards XML as a messaging paridigm within and between systems. Since it is a tagged data format with variable length records, allowing semantics to be defined separately from the data, one can envision systems which dynamically recognize and unravel XML packets of information and transform them into useful internal messages within a business component.

Consider an example from a health content provider on the Web. A request for an article from the New England Journal of Medicine causes the content server to serve up an XML copy of the article. This XML package is then sent to the pricing server which looks at the XML and determines the pricing for the article. It appends pricing information to the XML package and passes the expanded information set to the advertising server. This server, in turn, appends the appropriate ads and sends the XML to the personalization server which appends the appropriate personalization information. Finally, the entire packet of XML information arrives at a Web server which transforms the XML into the appropriate HTML to ship to the client browser.

There are major benefits to this approach that radically reduce programming and maintenance for such applications:

- XML software is rapidly appearing on the Web to generate, process, and interpret XML, significantly reducing programming requirements.
- Complex XML interactions can execute very fast while human readable format significantly reduces debugging costs.
- Data ordering and data field lengths can be interpreted dynamically, drastically reducing transaction errors in operations.
- An organization or standards body can centrally define semantics and post them on the Web. XML tools can dynamically access and interprete this information.
- XML allows separation of presentation from semantics and data formatting. Multiple presentations of the same data can be generated through XSL style sheets.
- A single XML object can be a globally distributed complex concept via URL pointers, or an autonomous agent roaming the Web, interpretable and runnable on any Web server, or both simultaneously.

The natural direction for business object component systems in the Web environment will be to use XML distributed objects as the basis of the Workspace Tier of a business object component shown in Figure 5. As a tagged data model suitable for serializing objects, an XML Workspace Tier of a business object component is an ideal package, or kit of information, to pass around the enterprise as a workflow object. Various components in an enterprise system could "process" this XML kit, provide added value as appended XML information, and ship it on to the next service specified in a workflow graph.

6 Conclusion

When the development of a standard Business Object Component Architecture 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 objects. Tools will be provided to generate business systems from design into heterogeneous distributed runtime environments. Code changes may be reengineered into design supporting round trip engineering. A lightweight WebBroker may render distributed object computing as easily deliverable and as ubiquitous as HTML pages. This will position the software industry for the twenty-first century and launch a global effort to break down the barriers to implementing Moore's Law for Software.

7 References

[1] Data Access Technologies, Inc., Electronic Data Systems (EDS), National Industrial Information Infrastructure Protocols (NIIIP), SEMATECH, Inc., Genesis Development Corporation, Prism Technologies, IONA Technologies. *Business Object Component Architecture (BOCA), Revision 1.1*. OMG Document: bom/98-01-07.

[2] X3H7 is now part of NCITS T3: Open Distributed Processing Technical Committee. See the NCITS X7 Technical Committee Home Page at http://enterprise.systemhouse.mci.com/X3H7/default.html.

[3] Portions of this paper were previously published as Sutherland, Jeff. Why I Love the OMG: The Emergence of a Business Object Component Architecture. *ACM StandardView 6:1:4-13, March 1998*

[4] *Open Distributed Processing Home Page*. http://enterprise.systemhouse.mci.com/WG7/default.html

[5] The term "domain" refers to a realm of business interest such as transportation, manufacturing, or finance. It is also used in this document to refer to more specific areas of business interest such as marketing, sales, shipping, etc.

[6] OOPLSA Workshop on Business Object Component Design and Implementation. http://www.jeffsutherland.org/oopsla99/index.html

[7] Geerts, Guido. *The Timeless Way of Building Accounting Information Systems: The 'Activity' Pattern*. OOPSLA Workshop on Business Object Design and Implementation, 1997.

[8] Jacobson, Ivar, Maria Ericsson, Agneta Jacobson. *The Object Advantage : Business Process Reengineering With Object Technology*. Addison-Wesley, 1995.

[9] Digre, Tom. Business Application Components. In Sutherland J., D. Patel, C. Casanave, G. Hollowell and J. Miller (Eds). *Business Object Design and Implementation: OOPSLA'95 Workshop Proceedings*. Springer, 1997

[10] pcwebopedia.com. *Moore's Law*. http://pcwebopedia.com/Moores_Law.htm

[11] Moravec, Hans. *Robot, Being: mere machine to transcendent mind*. 1998

[12] Kent, William. *X3H7 Objectives and Operations*. X3H7-93-023, 18 January 1993.

[13] Manola, Frank (Ed.) *Object Model Features Matrix*. X3H7-93-007v12b, 25 May 1997.

[14] Sutherland, Jeff. *ANSI X3H7 Standardization Targets*. X3H7-94-35, 24 Sep 94.

[15] W3C. Distributed Object Communication on the Web. W3C Note 11-May-1998. http://www.w3.org/TR/1998/NOTE-webbroker

[16] Sutherland, Jeff. Business objects in corporate information systems. ACM Comput. Surv. 27, 2 (Jun. 1995), pp. 274 - 276.

[17] Casanave, Cory. OMG Business Application Architecture White Paper. OMG bomsig/95-4-1

[18] OMG Business Object Domain Task Force. Common Facilities RFP-4: Common Business Objects and Component Interoperability Facility. OMG TC Document 95-12-13.

[19] Cox, Brad. Object-Oriented Programming: An Evolutionary Approach. Addison-Wesley, 1986.

[20] Taylor, David. Object-Oriented Information Systems: Planning and Implementation. John Wiley & Sons, 1992, pp. 320-322.

[21] META Group, Inc. Making the Case for Use Case. *Advanced Information Management*, File 324, 13 February 1995.

[22] Love, Tom. *Object Lessons : Lessons in Object-Oriented Development Projects*. SIGS Publications, 1993.

[23] Sutherland, Jeff and McKenna Jeff. *Ensembles*. Easel Corporation, 1993. http://www.jeffsutherland.com/papers/ensembles93.html

[24] Sutherland JV, Pope M, Rugg K. The Hybrid Object-Relational Architecture (HORA): An Integration of Object-Oriented and Relational Technology. *Proceedings of the 1993 ACM/SIGAPP Symposium on Applied Computing*, Indianapolis, 14-16 Feb 1993. Deaton E et al (Eds) ACM Press, pp 326-333.

[25] Herzum, Peter and Sims, Oliver. *The Business Component Factory*. John Wiley & Sons, 1999 (in press).

[26] Szyperski, Clemens. Component Software: Beyond Object-Oriented Programming. ACM Press, 1998.

[27] VMARK Software. Allied Signal Company wins the Computerworld Object Application Award at Object World. Press Release, 21 August 1995.

[28] Brodie, Michael. The Emporer's Clothes are Object Oriented and Distributed. GTE Laboratories, 1997.

[29] Microsoft. *The Renaissance of Distributed Computing*. White Paper, November 1996 (www.microsoft.com/pdc/html/p&s.htm).

[30] Sutherland, Jeff. *An Executive Overview to Object Technology (tutorial)*. Object World, Boston, Sydney, San Francisco, Frankfurt and Executive Symposium on Object Technology, Toronto, 1995.

[31] Jones, Capers, *Programming Languages Table, Release 8.2*. Software Productivity Research, March 1996.

[32] Zincke, Gerald. How to Achieve 7.52 Function-Points per Person-Day with Object Technology. Addendum to Conference Proceedings, OOPSLA 12thAnnual Conference, 5-9 October, 1997. ACM Press SIGPLAN.

[33] OMG BOMSIG. OMG Business Application Architecture, Revision 2. OMG 95-04-01.

[34] Rational Software, Microsoft, Hewlett-Packard, Oracle, Sterling Software, MCI Systemhouse, Unisys, ICON Computing, IntelliCorp, i-Logix , IBM, ObjecTime, Platinum Technology, and Ptech. *UML Proposal to the Object Management Group in response to the OA&D Task Force's RFP-1, Version 1.1*. OMG Document: 97-08-11.

[35] Cooperative Research Centre for Distributed Systems Technology (DSTC), IBM, International Computers Limited, Objectivity, Oracle, System Software Associates, and Unisys. *Meta Object Facility (MOF) Specification Joint Revised Submission*. OMG Document: 97-10-2.

[36] OMG. CORBA Component Model Request For Proposal. OMG Document: orbos/96-06-12.

[37] IBM Corporation, Netscape Communications Corporation, Oracle Corporation, Sunsoft, Inc. *CORBA Component Imperatives*. OMG Document: orbos/97-05-25.

[38] Frankel, David. *UML, BOCA, and MOF: Presentation to the Lifesciences DTF at the Orlando Meeting*. OMG Document lifesci/98-06-02.

[39] BOCA CDL Development Kit for OMG IDL, Pre release revision 0.51. Data Access Technologies, Inc., 1998.

[40] Frankel, David S.; Seidewitz, Ed; Rutt, Tom (Eds). *Roadmap for the Business Object Initiative: Supporting Enterprise Distributed Object Computing*. OMG Document bom/98-12-04, Version 3.0, 21 December 1998.

[41] UML Profile is defined in the Roadmap for the Business Object Initiative. It is essentially a subset, not necessarily a proper subset, of UML.

[42] Manola, Frank. *Towards a Web Object Model*. Position Paper for the OMG-DARPA-MCC Workshop on Compositional Software Architectures. Object Services and Consulting, Inc., 1998.

[43] Manola, Frank (Ed.) *X3H7 Object Features Matrix*. NCITS X3H7 Technical Committee Document X3H7-93-007v12b, 25 May 1997.

[44] W3C. *Resource Description Framework (RDF) Model and Syntax Specification*.