Business-Object Architectures and Standards

Cory Casanave

Data Access Corporation
14000 S.W. 119 Avenue, Miami, Florida 33186 USA
cory_casanave@omg.org

ABSTRACT: *Business information systems have become an integrated part of the modern enterprise and as such are required to enable the enterprise to serve and adapt to complex and dynamic business needs. An application architecture based on "business objects" is proposed as a way to build information systems to better meet these needs. Business objects are defined as components of the information system that directly represent the business model.*

KEY WORDS: *business object, interoperability, OMG BOMSIG, CORBA, Business model.*

BIOGRAPHY: Cory Casanave is the founder and a co-president of Data Access Corporation, a developer of object-oriented application-development tools, and chairman of the OMG Business Object Domain Task Force (BODTF) as well as a member of the OMG board of directors.

1. Introduction

The quality of a company's information system has become recognized as a strategic corporate advantage. Information systems have become the backbone of the modern enterprise and as such are crucial to its functioning. An organization with the appropriate information tools can take advantage of business opportunities quickly and can adapt itself to changing business requirements.

Despite advances in hardware, software, client/server technology, right-sizing, distributed computing, and better methodologies, corporate information processing continues to fight the complexity, inflexibility, and poor performance of its current mix of solutions.

As the enterprise has become more dependent on its information-processing capability, this same growth of dependence has put stress on that very capability. Poor performance, software backlogs and inflexible systems are, unfortunately, the norm.

Many solutions have come (and some have gone) to help with this problem. Some of these solutions—the ones that hold the most hope—are difficult to integrate and move to from existing technologies. Client/server and distributed-object computing in particular are seen as hopeful solutions—and are hard to integrate.

Products, services, and techniques that help overcome these problems can be critical to the success of the enterprise. OMG Business Objects and the Business-Application

Architecture are intended to enable such products, services and techniques by creating a standard framework for business applications, using OMG's CORBA.

It is not the intent of this paper to define or specify a business-application architecture or a business-object protocol. Rather, it is the intent to identify the advantages of, need for, and practicality of such standards in order to foster further work in this area.

1.1 Terms used

Terms used in the Business-Object domain correspond to terms used for similar (but lower-level) concepts in other disciplines. The following table relates terms used in this model to the other domains.

Business Objects	Object-Oriented Software Engineering	SmallTalk
Business Object	Entity	Model
Presentation	Interface	Presentation
Business-Process Object	Controller	Control

2. OMG Business Objects

Object-oriented systems have existed for about twenty years, but have only gained widespread acceptance in the last five years. In particular, objects have come to dominate user interfaces and system programming. Objects are visible to users as icons, boxes, and windows on the screen that they manipulate directly. This style of user interface (originally developed by Xerox PARC) has spawned a huge advance in the ease of use, esthetics, and power of end-user software.

Objects have also been used extensively by advanced programmers in systems software and applications. Objects are now part of the implementation of almost every major piece of software. While not fully exploited, object-oriented programming is currently helping make software more reliable and reusable.

Paradoxically, objects have not been widely used to represent the business itself. A business can be "modeled" in terms of objects that make up and reflect it. Objects can represent inventory and invoices, customers, and salespeople. Objects can also represent events in a business, such as purchases, sales, and other types of transactions.

Modeling the world as objects and then implementing them in an object-oriented system is the basis of object-oriented technology. It is time that the power and ease of understanding inherent in objects be applied to the business itself. Anything that is related to the finances, products, or customers of an enterprise can be a business object and work as part of a cooperative business-object system.

Put another way, business objects represent things, processes or events that are meaningful to the conduct of the business. Business objects can be distinguished from programming objects such as arrays and I/O channels or from user-interface objects such as buttons and windows. Business objects can also be distinguished from system objects such as your word-processing program. Business objects make sense to business people.

2.1 Definition of a Business Object

> A business object is a representation of a thing active in the business domain, including at least its business name and definition, attributes, behavior, relationships, rules, policies and constraints. A business object may represent, for example, a person, place, event, business process, or concept. Typical examples of business objects are: employee, product, invoice and payment.
> The business-object abstraction, which models the real world, is represented by an object in the information system. Each such object in the information system is a component of that information system and must be supported by a technology infrastructure. [Burt 95]

2.2 Description of an OMG Business Object

The following *Description of an OMG Business Object* has been adopted by OMG BOMSIG, and is included here for reference [Burt 95].

OMG Business Objects are representations of the nature and behavior of real-world things or concepts in terms that are meaningful to the business. Customers, products, orders, employees, trades, financial instruments, shipping containers, and vehicles are all examples of real-world concepts or things that could be represented by Business Objects.

Business Objects add value over other representations by providing a way of managing complexity, giving a higher-level perspective, and packaging the essential characteristics of business concepts more completely. We can think of Business Objects as actors, role-players, or surrogates for the real world things or concepts that they represent.

Business Objects can act as participants in business processes, because as actors they can perform the required tasks or steps that make up business processes. These Business Objects can then be used to design and implement systems in such a way that these systems exhibit and continue to maintain a close resemblance to the business that they support. This alignment is maintained because object technology allows the development of objects in software that mirror their counterparts in the real world.

Business Objects allow an enterprise to communicate, model, design, implement, distribute, evolve and market the software technology that will enable them to run their business. The implications of Business Objects include:

- **Communication:** Business Objects provide common terms and ideas at a level of detail which can be shared among business and technical people to articulate and understand the business in business terms.
- **Modeling:** Business Objects have certain characteristics and behavior which enables them to be used naturally in modeling business processes, and the relationships and interactions between business concepts.
- **Design:** Business Objects represent real-world things and concepts which enable design effort to be concentrated in manageable chunks.
- **Implementation:** Business Objects have late and flexible binding and well-defined interfaces so that they can be implemented independently.
- **Distribution:** Business Objects are independent so that they can be distributed as self-contained units to platforms with suitable installed infrastructure.
- **Evolution:** Business Objects can be used in a variety of roles and evolve with the needs of the business. They provide a means for integrating, migrating and evolving existing applications.

- **Marketability:** Business Objects have the potential to be commercially distributed and combined with Business Objects from other sources to facilitate a market in Business Objects.

More formally, a Business Object and its component parts are defined as:

- **Business object**: a representation of a thing active in the business domain, including at least its business name and definition, attributes, behavior, relationships and constraints. A business object may represent, for example, a person, place, or concept. The representation may be in a natural language, a modeling language, or a programming language.
- **Business name:** the term used by business experts to classify a business object.
- **Business definition:** a statement of the meaning and purpose assigned to a business object by business experts.
- **Attributes:** facts about the business object relevant to fulfilling its business purpose.
- **Behavior:** the actions a business object is capable of performing to fulfill its purpose, including: recognizing events in its environment, changing its attributes, and interacting with other business objects.
- **Relationship**: an association between business objects that reflects the interaction of their business purposes.
- **Business Rules**: constraints which govern the behavior, relationships, and attributes of a business object.

2.3 Business Objects Are Not DBMS Tables

Business Objects may, at first, seem much like tables in a relational DBMS, since tables also represent business information. In some simpler cases, there may be a direct correspondence between a business object and a DBMS table. But in most cases, the business objects will be implementing rules and processes beyond the capability of a DBMS. They may be combining multiple tables, managing distribution or managing information that is not even stored in a DBMS (like online stock price quotations). Business objects represent multiple tables, processes and rules at a higher level than the DBMS table.

2.4 Business Objects and Legacy Systems

Business objects can be built using any form of new development tool or they can be built on top of existing software.

For example, let's assume you have an application with 800 users running on a proprietary DBMS and there is just no way for you to flip a switch and have these users run on a newly designed system. However, you would like to add some new functions today and then transition to a new, more-powerful DBMS over time ~~how~~ **how?**

A business-object "wrapper" is written in the language of the existing DBMS (business objects do **not** have to be implemented in an object-oriented language) using a business-object framework. The relationships, rules, and procedures for using the object are implemented as part of the business object using the existing libraries and methods of the legacy application. This new business object can then be used as part of the new business-object architecture while still using the existing legacy application. Critical new functions can be added on top of the business objects. This is using object techniques without

changing the legacy programming environment. Conceptually, the user interface of the old program can be replaced by the Business-Object framework.

New presentations are designed with a business-object toolkit or another language to give users a consistent view of their applications through the business objects. These new presentations can be used at the same time as the original programs.

As time permits, the legacy application can be replaced—piece by piece, until it is gone. Once the legacy application is gone, you are free to re-implement the business object with more-current DBMS systems and tools *without changing the other business objects or applications (presentations) that depend on it.*

Legacy applications may be wrapped at the DBMS level (as in the above example) or at the application level. Business-object wrappers may communicate directly with the legacy programs, which may or may not store information in a DBMS.

It is a unique feature of the business-object architecture that it works so well for building new applications**and** providing a transition strategy for legacy applications and data.

Frameworks, adaptors, and re-engineering tools can be produced to assist with the transformation of legacy systems in any language, on any DBMS or transaction processor.

3. The Business-Application Architecture

The Business-Application Architecture (BAA) represents an application architecture and a protocol for "cooperative business objects" [Sims 95]. It is not the architecture of the business or of a specific application, but an architecture for how to represent and implement business concepts as business objects. The BAA is the "glue" that binds the business model with the technology. The BAA, together with an appropriate implementation, will provide a architecture in which business-object attributes, relationships, business rules, and application rules can be implemented. Objects implemented in this way will then be interoperable with other business objects that were implemented in this way.

All information systems have an architecture. That architecture may be formalized and structured, or it may be informal and implied. But for a system to operate, there must be agreed-to conventions, structures, and protocols - this is the architecture. Most "application-development systems" combine an application architecture with tools and sometimes a language to help implement that architecture. The architecture becomes part of the way you use the system or language.

The application architecture can be thought of as that layer between the high-level business objects being implemented and the low-level languages, operating systems, object-request brokers, and DBMS systems. As part of the architecture, a "protocol" exists for the components of that architecture to interact The protocol includes an object model, all interfaces, rules, constraints and ordering considerations.

The BAA is **not** a standard business model; it does not attempt to specify the standard or common components, object structures, or processes in a business. It is a standard way to represent any business model as a structure of executable distributed objects.

3.1 How Does the BAA Fit with Tools and Languages?

The business-application architecture does not attempt to specify the correct or best method for implementing business objects. Any combination of computer languages, 4GLs, design tools, frameworks, rule-based systems, and expert systems may be employed to implement a business object. Frameworks and other forms of tools and components are anticipated as products that assist developers or users in defining and implementing business objects that enable the business-object protocol. The BAA and underlying technologies provide a structural layer that allows differing implementation vehicles to work together in the same businesses.

It is expected that higher-level interfaces will be provided so as to hide the highly technical Interface-Description Language (IDL) interfaces from application developers. These higher-level tools and frameworks will provide standard BAA-to-IDL interfaces as a "framework" that application developers can use more easily. The high-level frameworks and tools will provide interfaces appropriate for directly defining business objects, attributes, relations, and business rules. In that these high-level interfaces may interoperate via the BAA protocol, we do not expect these interfaces to require standards of their own.

Note: It is possible for developers to create business objects that directly implement the BAA protocol; however, this protocol must expose some of the complexities inherent in a distributed-object system and for this purpose, implementation frameworks and intermediate components are useful in simplifying the job of the application developer. However, developers are free to use (or extend) the BAA protocol directly for special needs.

For support of legacy systems, business-object frameworks may be built for COBOL, RPG-II, IMS, and CICS. While the "source code" for these systems would appear completely different, the resultant application architecture would be the same and the objects would be interoperable.

3.2 How Does the BAA Fit with Other Application Architectures?

Many application architectures exist for both business and non-business applications. In that such architectures must be able to co-exist with each other and the BAA, the BAA must be sufficiently general to facilitate the interaction of BAA applications with applications of other architectures. The "wrapping" technique previously discussed in connection with legacy applications provides the capability to implement the BAA protocol in conjunction with other architectures—it is **not** an exclusive option. Thus the BAA is intended to provide the interaction protocol for application components in a variety of architectures.

Application architectures outside the business domain generally become part of the implementation of business concepts represented as business objects. For example, while a software-development company may monitor a project with a configuration-management system that uses its own application architecture (such as PTCE), the company's business system may refer to a single entity, which is the development project for that application. The implementation of the "project" business object may use the configuration-management system to provide business information (such as project status) to the business system.

It is unclear at this time whether the BAA can be sufficiently general to represent **all** business applications. It is our hope that other architectures can be built as **extensions** to the BAA rather than **alternatives** to the BAA. Such a determination can only be made after further work is done in this area.

4. Advantages of OMG Business Objects and the BAA

4.1 Flexibility

Maintaining a simple, standard interface to objects relevant to your business makes the information facility much more flexible. Changes in business policies or structure can be reflected directly by the business objects, and applications based on these will frequently adapt automatically to the changes. New business objects and business structures can be developed and deployed while still maintaining the old interfaces for a cross-over period.

Since the implementations of business objects directly reflect the structure of your business, business objects and applications are easier to produce and maintain, providing a more-responsive information-processing facility.

4.2 A single place to put business rules

The rules, policies and procedures of an enterprise can become quite complex and interrelated. By having a single, known place to put each rule (and express it only once!) the management and evolution of your rules, procedures and policies become much less complex.

4.3 High-level

The business objects operate at a “high” level, one that is understood by business people. Entire organizations—and in particular top management, can participate in the design of its information model and business rules without having to be burdened by implementation details. Business Objects use business names and terms.

4.4 Works with legacy systems

Legacy systems and data can be “wrapped” as business objects to become part of the new generation of applications without discarding the value of the legacy applications.

4.5 Insulation from insufficient or transient standards

Standards which were intended to prevent the business from becoming dependent on a particular vendor tend to be frustrated in real-world situations. Information-systems departments seem inevitably to depend on proprietary extensions and features sooner or later that again cause “lock in”. With business objects, the enterprise’s own information model becomes the standard, insulated from the DBMS or “tool *du jour*”. Advances in technology and new standards can be more easily integrated with working systems.

4.6 Open architecture

The business-object architecture is open and extensible. Interfaces and capabilities can be added as required for the business's need. Even the business architecture itself can be implemented on top of any distributed-object standard. As standards come into place, business objects become interoperable and tools can be provided to create and maintain them.

Any type of tool can be used to implement business objects or exploit their existence. Advanced Business Process Re-engineering tools, Workflow systems, CASE tools, 4GLs or 3GLs can all be employed to create or use business objects. The high-level nature of business objects makes them ideal for advanced decision-support systems and report writers.

4.7 Scaleable

Since business objects can employ advanced distribution mechanisms "behind the scenes" and the same or a related business object can be distributed across multiple systems, the architecture is infinitely scaleable. The applications are insulated from changes made to scale the system.

4.8 Reusable components

Business objects represent well-defined reusable components for application development. Reusable components leverage design and development efforts, increasing responsiveness and reducing costs. Business objects may be purchased from third-party vendors and integrated into an existing system.

Since business objects directly represent the business model, reuse becomes natural. The business model and objects (which have a natural order) become the library of reusable components.

4.9 Opens system to "power users"

Since business objects are visible to the "desktop," any program or user can access *and safely manipulate* the objects of the business. Power users and end users get unprecedented accessibility to enterprise resources.

Business objects are safe to manipulate because data integrity and business rules are enforced by the business objects.

4.10 Ideal for business-process re-engineering

Business-process re-engineering (BPR) is heavily dependent on a strong and flexible information system. Business objects are an ideal way to implement an information system that supports BPR. The type of analysis done to "re-engineer" a company can produce the of business model that business objects can implement.

Ivar Jacobson, in his excellent book *Object Advantage* [Jacobson 94], shows how BPR and object-oriented analysis can be combined and are complementary.

4.11 Ease of use

Providing a pre-built application framework places the user in a better position to concentrate on the application problems. Users who are forced to build an application framework "from the ground up" can face a huge effort in design and implementation that has nothing to do with their business problems. A well-thought-out, proven and standard framework can save massive amounts of work. Combine this with the possibility of purchasing pre-built objects and pre-built tools and the user's work is really leveraged!

Business objects use business terms in ways that business people understand. Keeping the terminology in line with the business makes the entire system more understandable.

4.12 Business objects are "happening"

Business objects are a hot topic. The press is talking about them, standards bodies, like the Object Management Group (OMG), are talking about them. IS professionals are asking for the functionality. Vendors are implementing them. Users who currently are trying to use "two-level" client/server systems *know* they need them.

4.13 Standards and the OMG

While products based around this architecture are attractive, standards will make it an industry. By standardizing on the Business-Object Architecture, objects created in diverse systems can interoperate and companies can provide specialized tools to create and maintain the business objects.

The lower "technology" layer is already available and standard as CORBA 2.0. The next layer of standardization can provide the higher-level business-object protocol.

We expect the infrastructure and interface to become standardized by the OMG sometime in 1996. Once this happens, the now-uncoordinated efforts being put into business objects can become cooperative technologies supporting a common application architecture.

Post-OOPSLA update: An RFP for business objects was issued by the OMG on January 11th, 1996 [Casanave 96].

5. How Business Objects Fit into a Business

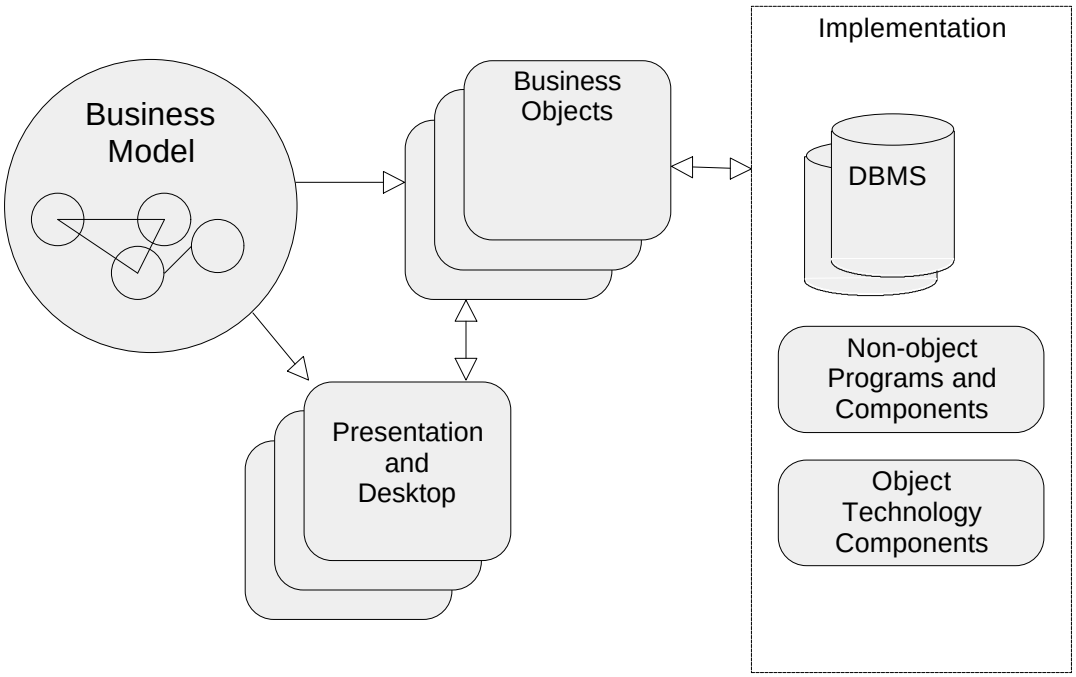


Diagram BAA-1

5.1 The business model

The basis for any business-object system is the "model" of the actual business. This model is built using abstract business objects and processes and/or more-specialized versions of these abstract objects.

This model should include every person, place, thing, event, or transaction that needs to be captured in the information system.

The business processes are likewise identified and modeled as business-process objects.

Once complete, this business model becomes a valuable reference to how your business is organized and operates.

5.2 Business objects and implementation components

Each object in the business model is used to create an executable representation of that object in your computer system. This executable object will contain and encapsulate the information and rules associated with that object and its relationships to other objects.

Some business objects may be implemented on top of existing applications as "wrappers", exposing the legacy application as business objects. Other objects may be implemented using Workflow tools, computer languages or 4GLs. Provided all of the tools and wrappers can "speak" the BAA protocol, consistency of implementation environment is not required.

When used with a traditional DBMS, the executable objects sit between the DBMS and the user interface providing an object-oriented, multi-tier client/server system.

The direct representation of the business model as executable and user-accessible objects is the essence of the business-object concept!

5.3 Presentation and system interfaces

Given the executable business objects, user interfaces are generated to allow users and other applications to view and manipulate the business objects. The business-object user interface becomes the new "look and feel" for your applications. Desktop applications may also interface with the business objects through interfaces such as OpenDoc and OLE.

5.4 The outdated concept of "application"

With a system composed of a set of cooperative business objects, the outmoded concept of monolithic applications becomes irrelevant. Instead, your information system is composed 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.

An application, in terms of business objects, becomes a set of cooperative business objects combined to facilitate business processes.

6. The Requirement for OMG Standards

6.1 Options for an application architecture and framework

Given that an organization wishes to implement a business application, there must be an application architecture. That architecture may be custom, proprietary, or standard. Each approach has its advantages and disadvantages.

6.1.1 Custom

A custom architecture provides maximum internal flexibility to the enterprise. The applications can be designed and tuned to the organization's needs. Since the organization has developed much of its own infrastructure, it is not dependent on as many external suppliers (unless such dependencies are built into the custom framework).

Creating a custom architecture is not a small job. Experience has shown that a highly capable and specialized development team requires one to two years to field a stable infrastructure for applications development in a distributed environment.

The application infrastructure, like all software, will also require costly maintenance and future development. Of course, the application created in a custom environment will not interoperate with external software—considerable effort must be expended to integrate other software and data.

6.1.2 Proprietary

A proprietary application framework may be purchased from a vendor, frequently with some standard business applications. This solves the problems of producing a custom

framework, but it does not solve the problems inherent in integrating the system with software that uses another framework.

Many organizations are also concerned about being locked in to a proprietary-framework vendor, since the organization may become very dependent on the provider. However, with a good provider relationship, a proprietary framework may be very productive.

6.1.3 Standard

A standard framework solves the problems of creating a custom framework and becoming locked in to a single vendor. The organization may deal with multiple vendors to supply and support the standard framework.

The standard framework will have a much-larger support base and as such will probably be worked-out and debugged to a greater degree.

The most-important factor in a standard framework is commercial support. Given a standard framework, it is practical to purchase pre-built business objects in an open market. Pre-built objects can be used as-is or enhanced using standard object-oriented techniques, vastly leveraging development. On the tool side, the organization can purchase design and implementation tools, data-analysis tools, languages, libraries and utilities to help use and build applications in a standard framework. Standard desktop applications can interface with the architecture components.

A standard framework also leverages training. A development organization will be better able to find employees and consultants who already understand how the business system operates.

A standard framework can also be expected to have a longer lifetime. While standards take longer to produce, they also last longer. Business applications have an average lifetime of 10-15 years, while some proprietary architectures have a lifetime of one-to-two years. Standards have a lifetime more in keeping with business needs.

The only downside to a standard framework may be flexibility. The framework may not do just what is required in very special conditions. But, the object-oriented paradigm helps here as well, since the standard framework can be extended, as can all object systems.

In short, a standard framework can foster an *industry of business objects*.

6.2 Goals of standardization

The reasons to standardize components of the BAA are directly reflected in the purpose of the OMG...

(a) to promote a single object-oriented applications-integration environment based on appropriate industry standards;

(b) to promote a framework for compatible and independent development of applications;

(c) to enable coordination among applications across heterogeneous networked systems in a multinational, multilingual environment;

(d) to adopt a core of commercially available implementations of this framework and to promote international market acceptance and use;

(e) to actively influence the future direction and development of these core products and technologies; and

(f) to foster the development of tools and applications that conform to and extend this framework and to provide a mechanism for certifying compliance with the core technologies.

(Article I of the OMG by-laws [OMG 95])

Such a purpose for OMG and the BAA will have a range of advantages...

6.2.1 Synergy

To synergize the work being done in creating business applications and distributed object components into a cooperative industry effort.

6.2.2 Interoperability

To make independently developed business objects interoperable with a minimum of effort.

6.2.3 Federation of systems

To allow diverse business systems to be integrated.

6.2.4 Ease of use

To make the information understandable in business terms and easily meet business needs.

6.2.5 Open market

To foster an open market in business-object-related components, both in pre-built business objects and in tools for using and building business objects.

6.3 What needs to be standard?

With all the advantages of a standard, there is a dark side also. Restrictive standards can stifle innovation, and poor standards can do more harm than good. To minimize the inherent problems of standardization, standards should be *minimal*. That is, they should provide a sufficient level of standardization to meet the goals but no more. Simple, minimal standards are also easier to adopt to future innovation.

Another question of a standard is its scope. We are targeting business applications because of the extreme importance of business data processing and because of the high degree of commonality among business applications. Business applications represent billions of dollars of expenditure worldwide and directly impact the productivity of society—they deserve special attention. Trying to design a framework for all applications may not sufficiently benefit business applications; it may not even be possible. Applications outside the business domain may still use the BAA where appropriate, but it is not the design

intent of the BAA. The term “business application” is intended in its more-general sense. The data processing of governments and organizations fall within the domain of the BAA.

6.4 Existing OMG standards

The existing OMG CORBA standards are required to implement a distributed-object business system. They provide the basic mechanisms for creating and using objects in a distributed network.

The existing and proposed OMG standards provide the necessary interfaces for transactions, User interface, events, object lifecycle and object query are all required for a business system. The proposed application architecture must build on and work with the existing standards. For example, the IDL interface to the user interface should conform or work with the user-interface component adopted by OMG common facilities.

The application architecture should build on this existing foundation.

Are the existing standards sufficient? If the existing standards were given to two development teams with the charter of producing the same application, it is unlikely that the above goals would be achieved. Both teams would have to come up with their own answers to fundamental questions like:

- What is the appropriate structure of an application built with these tools?
- How are changes and dependencies propagated?
- Should the user-interface and business rules be together?
- Should the data and business rules be together?
- How does the user interface interact with the data in the business object?
- Where are the business rules put?
- How does an object locate another cooperative object?
- What are the common events that drive the system?
- What happens when a business rule is broken?
- How are errors handled?
- What happens when rules or data change?
- Will the structure scale-up to a running system?

Answering these questions and building the infrastructure to support them is the process of designing the application architecture and framework. Given that no two teams are going to come up with the same rules, the requirement for interoperability will not be achieved, and considerable effort will have been duplicated.

6.5 Required new standards

Two elements are essential to an application architecture and protocol. The architecture represents the components that are used to “model” the business problem and build the system, while the protocol is the set of rules that govern how these components behave and communicate with each other.

For example, in the reference model (Diagram BAA-2), we have presentations and business objects. If users change data in the presentation, how is that change communicated to the business object? If that change violates a business rule, how is that

violation communicated to the presentation? Which object is responsible for side effects of that change and how and when are the side effects made visible to the presentation?

Business application are very "state-" (or data-) oriented. That is, business systems are driven by actions changing data and properly propagating the effects of that change. The protocol must provide very clear rules for dealing with that state and propagated effects.

6.5.1 Basic architectural framework

The basic framework outlined in the reference model (Diagram BAA-2) has three components: business objects, business-process objects, and presentations. These are the building blocks of the applications. The same building blocks are used to model the business and to build the application. Each component of the BAA application becomes a subclass of one of these components.

As part of the architectural framework, each of the following must be addressed:

- What the appropriate structure of an application built with these tools is.
- Whether the user-interface and business rules should be together.
- Whether the data and business rules should be together.
- Where the business rules are put.
- What happens when a business rule is broken.
- What happens when rules or data change.
- How the structure will scale up to a running system.

6.5.2 Inter-object protocol

The protocol is the standard IDL interfaces between presentations, business objects, and business-process objects. Anything done to these objects is done through these standard interfaces. The primary purpose of the interface to business objects will be to make and respond to changes in the objects' states. As part of the protocol, business objects should present their *metadata*. Metadata is information about the business object (as distinguished from the data the object is dealing **with**). By having the object present its own metadata, applications can change their behavior based on changes in the metadata, making the entire system more friendly, flexible and dynamic.

As part of the protocol, each of the following issues must be dealt with:

- How are changes and dependencies propagated?
- How does the user interface interact with the business object?
- How does an object locate another cooperative object?
- What are the common events that drive the system?
- How are errors handled?

6.6 What does not need to be standard

Anything that has to do with the e*xpression or implementation* of business objects or presentations should not be standard. The best and most-proper way to express business objects and business rules is still growing and changing; we do *not* need to lock that down in a standard. As long as the objects can implement the desired protocol, our goals are achieved. The following are some of the elements that do not require standards.

- High-level interfaces
- Computer language
- Operating system
- Source code
- Design tools
- Design methods
- Business-rule representations
- Implementation frameworks
- Presentation style
- Custom interfaces

6.7 Domain (application) object interfaces

Once the application architecture has a sufficient level of definition, the question of commonality of specific objects arises. Can we identify common objects like customers, accounts, products and orders and derive common names, attributes and relationships for those objects? Standards for common business objects are not required for the BAA to work, but they would enhance the ability for the objects to interoperate. Standards for business-domain objects is a separate issue from the BAA and is not covered in this paper.

6.8 The RFP

The OMG BOMSIG drafted an RFP (Request For Proposal) [Casanave 96] for common business objects and a Business Object Facility. This RFP was issued by the OMG January 11th, 1996. The RFP items are described as follows:

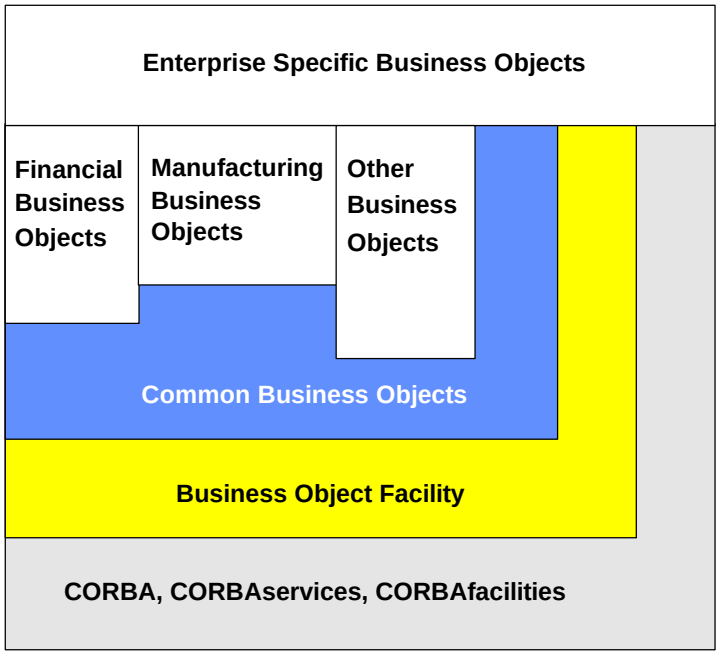
6.8.1 Common Business Objects

Objects representing those business semantics that can be shown to be common across most businesses.

6.8.2 Business-Object Facility

The infrastructure (application architecture, services, etc...) required to support business objects operating as cooperative application components in a distributed object environment.

The following diagram shows how these facilities fit in the current OMG architecture.



These facilities are seen as the "missing middle layer" between the CORBA facilities as the low-level infrastructure and the needs of standard and custom vertical applications as the high-level.

7. Business Application Architecture Reference Model

The reference model is a general model for business objects intended to encompass multiple interpretations and implementations of this concept. Diagram BAA-2 shows the abstract components of a business-object system and their interrelationships. Specific business-object systems may implement a superset or a subset of this model.

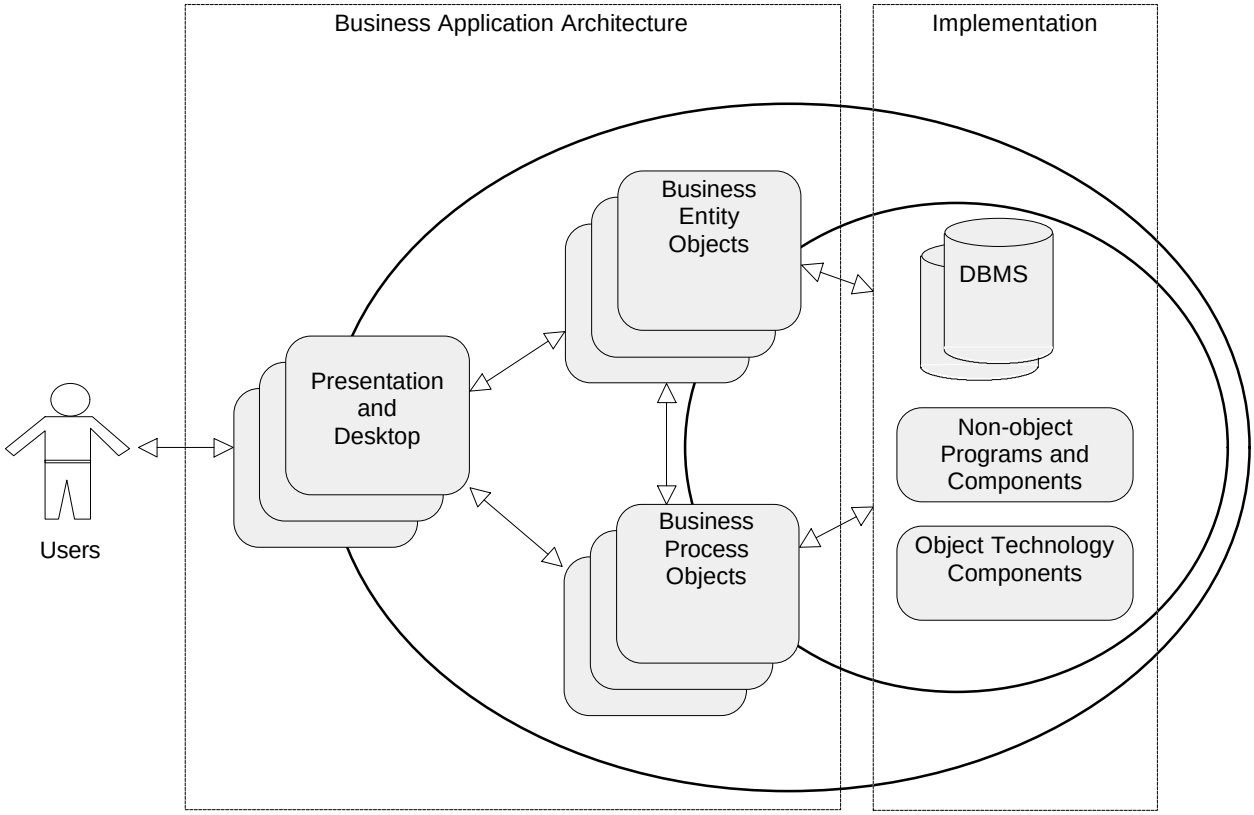


Diagram BAA-2

In this diagram, we can see that the tools used to build "traditional" programs, DBMS systems, technology components, and non-object programs, are encapsulated (shown by the inner circle). Only Business Objects will interface with this layer. Business Objects are encapsulated and made accessible to users by visual presentations and desktop programs. The Business Objects and their presentations comprise the Business Application Architecture.

There are several object-oriented meta-models to draw on for this purpose. A primary candidate is the model contained in the OORAM [Reenskaug 96] methodology. OORAM has the following features:

- the enterprise is modeled in terms of roles and collaborations between roles
- role collaborations and the information model are integrated
- models (frameworks) can be synthesized together

7.1 Components

7.1.1 Applications

Applications in this context are programs that are composed of a set of cooperative business objects. A program may implement one or many presentations and processes that work with business objects.

Any number of applications may be expected to share and reuse a common class of business objects. It is implementation-specific as to whether multiple applications share an instance of a business object.

Note: Not all applications are business-application-architecture applications. Other types of applications may exist for other purposes and architectures.

7.1.2 Business objects

Business objects encapsulate the storage, metadata, concurrency, and business rules associated with a thing, process, or event in a business. Multiple independent but related business objects may cooperate to service one application. Implementations may require different "flavors" of business objects for differing roles, such as: client-local objects and server objects. Business objects are responsible for all aspects of implementation including enforcement of business rules, application rules, data validity, concurrency, and storage. Business Objects are a representation of a thing active in the business domain including, but not limited to, its name and definition, attributes, behavior, relationships, and constraints.

7.1.2.1 Business-entity objects

Entity objects represent the actual things and concepts that make up the business. These are the nouns of the model: the people, places, things, and business events (such as a sale) that model the static state of the enterprise. Entity objects are an object-oriented extension to the concepts found in "ER" modeling and semantic modeling.

7.1.2.2 Business-process objects

Processes represent the flow of work and information throughout the business. These processes act on the business entities to cause the business to function. Business processes may be long-lived (such as an order life cycle) or may be short-lived (such as an end-of-year report). Long-life-cycle business processes are typically part of Business Process Re-Engineering (BPR) analysis.

Business-process objects may be implemented with Workflow systems, business-process managers, object-oriented languages, procedural languages, or interactive process-definition systems. The only requirement on the process implementation/definition environment is that the resulting business process supports the standard BAA interfaces or can be "wrapped" to provide such interfaces.

The executable business-process objects which represent the processes in the information system should not be confused with a Workflow definition that may take a part in implementing a business-process object. A Workflow definition, like any other business rule, is part of defining and implementing the object, not using it.

7.1.3 Presentations

Business objects have a companion—the Business Object Presentation, or "Presentation" for short. Each business object can have multiple presentations for multiple purposes. The presentation is the user's view of the business object for a given purpose. The presentations communicate with the business object in two ways: 1) To transfer information between the presentation and the business object on behalf of the user. 2) To learn how to display and manipulate the information (called "metadata" or, data about the data).

Having the presentations learn about the data from the business object makes them very simple and flexible. If anything about the business object is changed, that change is immediately reflected in the presentations.

Presentations are one type of application that can make use of business objects. Custom applications and automated processes (like Agents) can be part of a business-object system.

Presentations are always run on client machines but, thanks to the distributed-application architecture, the business objects and DBMS systems can run on the client machines, the server, or both. In large systems, the implementation of a single business object can be split into multiple pieces to better optimize performance across large networks. Since the mechanisms of implementing the business object and storing the data are encapsulated "behind the scenes", advanced DBMS distribution, object-oriented DBMSes, concurrency, and replication systems can be added to change the scale of operation without changing the interface to, or use of, the business object. Business objects can "scale" to the capacity of the underlying systems.

7.1.4 Implementation

The implementation components are encapsulated by the business objects. They are not accessed directly by users, processes, or presentations. The business objects use and manipulate DBMS systems, technology components, and non-object programs to implement their functionality.

7.1.4.1 DBMS

The DBMS (or similar repository) is expected to store the representations of business objects and aid in their retrieval and concurrency. Many but not all business objects will use a DBMS to store their states.

7.1.4.2 Non-object programs and components

Business objects can encapsulate non-object or legacy programs so as to provide these older applications with the business-object interface. Existing non-object programs can also be modified to replace their user interface with a business-object interface.

7.1.4.3 Object technology components

Object technology components are the other pieces of technology required to implement the business objects. In the OMG model, these include CORBA, CORBAservices and CORBAfacilities. They also include other applications used to support the business objects.

7.2 Requirements

7.2.1 Encapsulation

The architecture of a business-object system is one in which the data, data storage, business rules and operations relating to each business entity are "encapsulated" (contained in and hidden by) a business object. These business objects have a simple,

standard interface that allows them to communicate with other business objects and with business-object presentations (presentations are what users see on terminals and reports). This represents the standard notion of object-oriented encapsulation applied to the business domain.

7.2.2 Responsibilities

Each business object is responsible for managing its own storage (usually in a DBMS), security, maintaining its relationship with other objects, and *implementing and enforcing the policies, procedures and rules of the business as they relate to that business object.* Business objects are information-centric in that they expose and manipulate business information. Business objects are encouraged, but not required, to utilize OMG object services and common facilities for implementation.

7.2.3 Distribution

Business Objects are implemented on top of a standard distributed-object broker such as CORBA (OMG), DSOM (IBM) or COM-OLE2 (Microsoft). These distributed-object systems have only recently become available as industrial-strength products and this technology is key to the business application architecture. The object broker allows **any** program (even your word processor or spreadsheet) to access and manipulate the business objects. Since rules are maintained by the business objects, complete control is exercised over the integrity and validity of the enterprise data. The object broker also allows any object to exist on any computer system and still integrate with the total information-processing infrastructure.

7.2.4 Ease of use

Business objects are intended for use by the developers and users of business applications. As such the design and implementation of business objects must support the requirements of these users. These requirements extend across the entire lifecycle of development from design to maintenance. The interfaces and services provided must make sense to these persons and allow them to define business applications without undue knowledge of, or restriction by, the technology.

7.2.5 Loosely coupled

Business objects exist within the dynamic environment of business. Business changes, merges, separates, and re-engineers. Business objects must cooperate within an environment that supports such dynamic change. As such, these objects must cooperate in ways that preserve the semantics but allow each object to change and grow independently.

7.3 Specialization of business objects

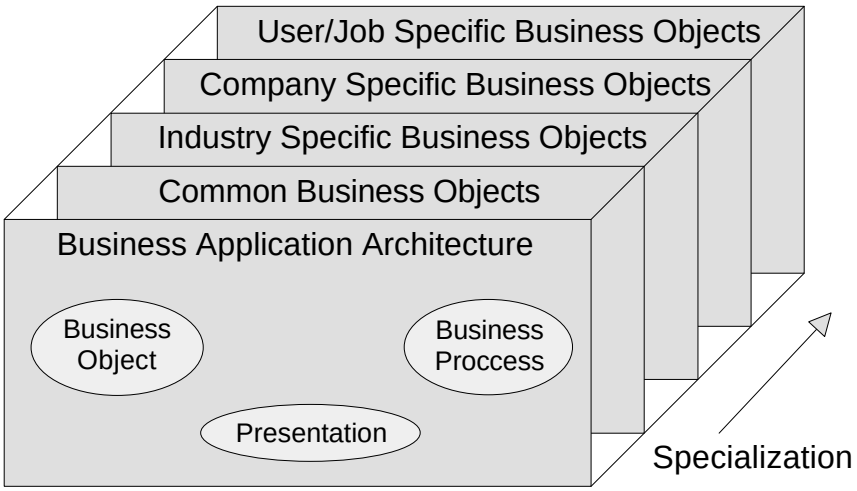


Diagram BAA-3

The generic Business Objects, Business Processes, and Presentations defined in the Business Application Architecture are specialized through common, industry, company, and user business objects.

For example, A Business Object might be specialized to create an "order" object in a general business suite. This order object may then be further specialized in a consulting company to be an "order for consulting services" object. A particular consulting company may add rules and attributes to that consulting-order object to enforce company policy. Finally, a particular department might further specialize the company's consulting-order object for a particular type of service.

The facility for specialization is inherent in the use of objects to represent the business in the information system. The degree of specialization required is driven by the business requirements of the users and the degree to which specialization will enhance business practices.

8. References

[Burt 95] Carol Burt [ed.]: OMG BOMSIG survey with published definition of a business object. OMG document 95-02-04. Www.omg.org

[Casanave 96] Cory Casanave [ed.]: OMG Common Business Objects and Business Object Facility RFP. OMG Document CF/96-01-04. Www.omg.org

[Jacobson 94] Ivar Jacobson, Maria Erricsson, Agneta Jacobson: *The Object Advantage, Business process reengineering with object technology* Addison Wesley 1994. ISBN 0-201-42289-1

[OMG 95] Object Management Group: Bylaws (not published).

[Reenskaug 96] Trygve Reenskaug with Per Wold and Odd Arild Lehne: *Working with Objects, the OORAM Software Engineering Method* Manning 1996. ISBN 1-884777-10-4

[Sims 94] Oliver Sims: Business Objects, *Delivering Cooperative Objects for Client-Server*. McGraw-Hill. ISBN 0-07-707957-4