Coordinating support for every phase of computing.

OMG OVERVIEW: CORBA and the OMA in Enterprise Computing

~Jon Siegel~

When the Object Management Group (OMG) was formed in 1989, interoperability was its founders' primary, and almost their sole, objective: A vision of software components working smoothly together, without regard to details of any component's location, platform, operating system, programming language, or network hardware and software.

When this interoperability goal was realized in 1996 with the issuance of the CORBA 2.0 (Common Object Request Broker Architecture 2.0) specification [2, 6], two notable things happened: First, the infrastructure-oriented OMG members who had been concentrating on CORBA and the basic services were suddenly free to expand their horizons to include extended features; as a result, the basic CORBA object model is being extended to include multiple interfaces per object, objects passable by value, a beans-like component model, and support for real-time, fault-tolerant, and embedded CORBA [3].

A second result was that companies and organizations working in vertical markets (or, as OMG refers to them, Domains) started to use the OMG Interface Definition Language (IDL) to specify standard objects that all could share. This led OMG to expand its scope through a major reorganization in early 1996, creating its Domain Technology Committee that has grown to include subgroups in Finance/Insurance, Electronic Commerce, Healthcare, Manufacturing, Telecommunications, Transportation, Life Science Research, and Business Objects. Nearly simultaneous with this expansion was the formation of OMG's Analysis and Design Task Force, which adopts specifications supporting software analysis and design.

The result, shown in Figure 1, is an extensive suite of OMG specifications with enormous appeal to the enterprise that is now able to field the large-scale applications they need within a single coherent architecture that extends from design to run time. Even though the upper layers of this architecture are only starting to emerge, the basic infrastructure is well-tested and has already produced many stories of CORBA success in mission-critical enterprise applications [4].

An general overview of the OMG architecture is provided in the sidebar "Architectural Overview," which gives an idea of its scope. We survey and introduce each of the areas listed in the sidebar here and provide references to additional information. Other articles in this section will build on this material in many of the areas. All OMG specifications are

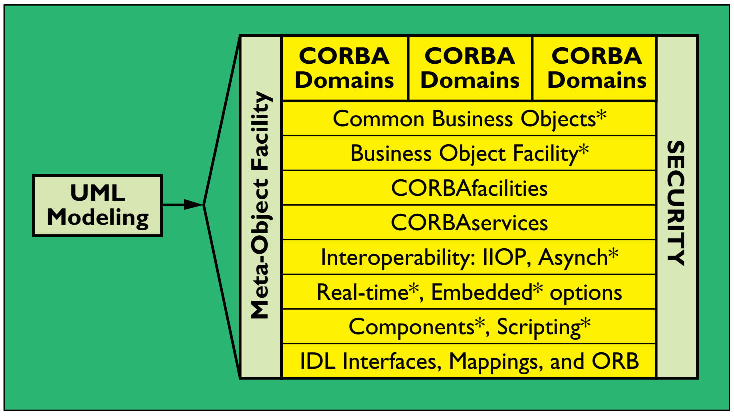

Figure 1. OMG produces more than just CORBA, as this diagram of the full suite of OMG specifications shows. Parts that were still in progress when this article was written in mid-1998 are denoted by asterisks (*).

available without charge from the OMG Web site www.omg.org/library/specindx.htm.

UML and the MOF: Supporting Analysis and Design

Modeling is a key first step in the building of industrial-strength enterprise software systems. Prior to the development of OMG's Unified Modeling Language (UML) specification [4], there was no way to transfer models from one modeling tool to another because tools on the market supported similar but not-quite-identical metamodels and output artifacts. UML is a visual language for the development and exchange of well-defined models for software development. Standard UML language components provide much of the functionality that users need, but mechanisms for extension and specialization allow use in areas in which standard components fall short. It's designed to support tools and collaborations, using frameworks, patterns, and components.

UML defines a graphical notation with these standard diagram types: use-case diagrams; class diagrams; behavior diagrams including state charts, activity diagrams, and interaction diagrams; and finally implementation diagrams, including component diagrams and deployment diagrams. There is no data-flow diagram; the activity diagram provides this functionality and more. This graphical notation *is* the language; there is (at least presently) no corresponding text-based equivalent. It is formally defined in the *UML Notation Guide* [4], available from OMG; in addition it is described in the documentation of various UML-based tools, and is already the subject of a number of books [1].

Supplementing the UML, the Meta-Object Facility (MOF) [4] provides a standard repository for metadata within the CORBA architecture. Defined in terms of a metamodel and set of IDL interfaces, the MOF supports component-based computing from modeling and design through implementation to run time. The common metamodel can support introspection, enabling sharing of components across heterogeneous distributed environments, and through revisions as components' life cycles evolve.

The CORBA Computing Model

Figure 2 shows a request passing from a client to an object implementation in the CORBA architecture. Two aspects of this architecture stand out:

- Both client and object implementation are isolated from the Object Request Broker (ORB) by an OMG/ISO IDL interface. CORBA requires that every object's interface be expressed in OMG IDL. Clients see only the object's interface; never any implementation detail. This guarantees substitutability of the implementation behind the interface—our plug-and-play component software environment.
- The request does not pass directly from client to object implementation. Requests are always managed by an ORB. Every invocation of a CORBA object is passed to the ORB; the form of the invocation is the same whether the target object is local or remote. (If remote, the invocation passes from the ORB of the client to the ORB of the object implementation). Distribution details reside only in the ORB where they are handled by software the user bought, not built. Application code, freed of this adminstrative burden, concentrates on the problem at hand. Sophisticated implementation design ensures that the ORB layer is lightweight and fast, with extended features called upon only when needed, for example, distribution or fault recovery. (The ORB is not a separate process—in virtually all current implementations, ORB functionality is provided through library routines that

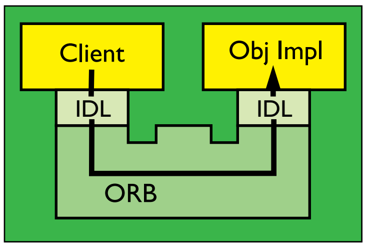

Figure 2. A request passing from a client to an object implementation in the CORBA architecture. In this case, Client, Object, IDL stub and skeleton, and ORB all reside within the same process (shaded region).

are linked into an executable module along with clients and object implementations. The presence of the ORB enables both in-process and remote invocations, by both clients and objects in the executable module. In-process invocations have little ORB involvement and execute rapidly, while remote invocations typically execute with performance comparable to RPC mechanisms that provide a similar level of service.)

In CORBA, an object's interface is defined in the OMG IDL. The interface definition specifies the operations the object is prepared to perform, the input and output parameters each requires, and any exceptions that may be generated along the way. This interface constitutes a contract with clients of the object, who use the same interface definition to build and dispatch invocations as the object implementation uses to receive and respond. This design provides a great amount of flexibility, and many benefits. It enforces encapsulation, and allows clients to access object implementations independent of each other's programming language.

To the client or user, the OMG IDL interface represents a *promise*: when the client sends a proper invocation to an object through its interface, the expected response will come back. To the object implementor, the interface represents an *obligation*: the implementor must implement, in some programming language, all of the operations specified in the interface. Writing the contract (in OMG IDL), and fulfilling it (in a programming language such as C++, C, or Smalltalk), are usually two separate steps in the writing of a CORBA application, although some vendors' CORBA products generate OMG IDL automatically from either source code or application design information.

For every major programming language, an OMG standard language mapping [2] specifies how OMG IDL types and method invocations convert into language types and functions. This is how the OMG IDL skeleton and the object implementation come together: The OMG IDL compiler uses the mapping specifications to generate a set of function or method calls from the OMG IDL operations. Programmers, usually assisted by an automated or semiautomated tool, refer to the OMG IDL file and use the language mappings to generate the corresponding set of function or method declarations. After compilation and linking, these resolve so that the skeleton makes the right calls to invoke operations on the object implementation. Currently, CORBA specifies OMG IDL language mappings for C, C++, Java, Cobol, Smalltalk, and Ada. Mappings don't have to be standardized by OMG in order to be useful; implementations of not-yet-standard mappings are available now for Objective C, Eiffel, and other languages. Mappings for Visual Basic, while not standardized, are usually based on OMG's COM/CORBA interworking specification [2].

Clients may construct their invocations in one of two modes: *static* or *dynamic*. Static invocations are routed to the client's ORB through a stub that is compiled in the target programming language from the IDL interface definition. Dynamic invocations are assembled at run time by the client using a set of ORB functions defined for this purpose. The Dynamic Invocation Interface (DII) provides an extremely high degree of flexibility, although it requires more coding than static (compiled-IDL) invocations. For purely historical reasons, the DII provides a *deferred-synchronous* distributed invocation mode unavailable for static invocations. A new OMG asynchronous/messaging specification remedies this shortcoming, details of which are provided later in this article.

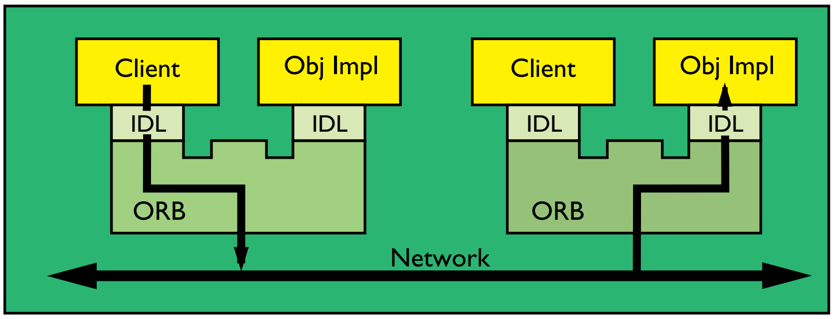

Figure 3. A request passing from a client to an object implementation in the CORBA architecture. In this case, the client and target object reside on different machines, and communicate over a network. The ORB handles network communications for both Client and target Object.

The CORBA Distribution Model

Figure 3 shows the basis of CORBA distribution: ORB-to-ORB communication. OMG IDL supports distribution in a number of ways: it enforces encapsulation, and unambiguously defines operations and types. Named, identifiable, sharable Interface Repositories (IR) [2] ensure that all ORBs on the network have access to IDL interface definitions.

Building on this, OMG's standard General Inter-ORB Protocol (GIOP) specifies all aspects of interoperability up to (but not including) network transport: GIOP specifies a small set of standard messages that ORBs send to each other, so that client and object see object-oriented invocation and response; a Common Data Representation (CDR) for IDL datatypes (with byte-order corrected by the receiver for efficiency); and a set of transport assumptions. The GIOP content, layered upon TCP/IP transport, forms IIOP—the Internet Inter-ORB Protocol, OMG's mandatory standard for CORBA-compliant distribution [2].

The scope of CORBA is so vast that no single protocol could ever meet service and efficiency goals for all target platforms. Thus the specification supports additional protocols that may be provided in two fundamental ways: first, GIOP may be layered on reliable protocols other than TCP/IP. Examples include OSI, IPX, ATM, and the Telecommunications protocol SS7. Secondly, alternative protocols may be based on content formats other than GIOP. Gateways may be built to bridge between alternative protocols and IIOP; ORBs that communicate using IIOP in addition to other protocols, or that utilize only non-IIOP protocols but provide bridges to IIOP are considered CORBA-compliant.

To the client, the object instance is represented by its Interoperable Object Reference (IOR). The exact format of the IOR, while an OMG standard for interoperability purposes, is opaque to the client. This enforces encapsulation, and allows ORB implementors some freedom to optimize for efficiency and reliability. Internally, the IOR format is an extendable, multiple-component structure that includes a slot for each protocol that may be used by a client ORB to access an object; for some protocols, the IOR also includes the network address of the object's ORB (or, at least, the network address of the last known location of the object's ORB, since objects may move; when this happens, the original ORB will refer the client ORB to the new location with a *location_forward* IIOP message).

CORBA messaging semantics have just been expanded. As explained previously, OMG specifications until recently supported only synchronous invocations for static invocations, although deferred synchronous invocations are allowed with the DII. A new asynchronous/messaging specification standardizes a range of asynchronous, deferred-synchronous, and time-independent (store-and-forward) invocation modes. Under the new specification [6], quality of service may be specified in various ways including automatic priority-raising for requests whose time-to-live has almost expired. And CORBA clients on portable computers, for example, may make a CORBA invocation over a dial-up connection, disconnect from the network, connect again later, and retrieve the response.

CORBA Component Model (CORBAbeans)

A specification still in process when this article was written in July 1998, CORBA components [6] extend OMG's object model to include a number of features important for distributed systems. The Request for Proposal, OMG's requirements document that precedes a new specification, calls for a component model for CORBA systems that is structured as a natural extension of the existing CORBA object model. Current specification efforts for Multiple Interfaces per Object, Objects Passable by Value, and the emerging Messaging Service will be taken into account in the new specification. The notion of "component" may not correspond one-to-one to a CORBA interface, nor to a CORBA object; the RFP asks submitters to propose the exact relationship between a component and a set of interfaces. Components have instance identities, as well as properties, that are an externally accessible view of a component's abstract state that can be used for design-time component customizing, and that support mechanisms for notification (event generation) and validation when a property's value changes. Components will support an introspection mechanism and the additional functionality this implies. CORBA components will map not only to currently supported programming languages, but also to commercially available component models including Java beans (although the CORBA component model will be distributed while some current models are not).

A separate specification effort will define a CORBA scripting facility to enable user-level assembly of applications from CORBA components. When this article was written in mid-1998, it was not possible to say in more detail how this would be done. Up-to-date information on both components and scripting is available on OMG's Web site www.omg.org/library/schedule.htm.

Real-Time, Fault-Tolerant, and Minimal/Embedded CORBA

Real-time distribution support extends the scope of CORBA to many environments that could not otherwise consider it. Even without formal standards for real-time CORBA, enough vendors provided real-time ORBs to support a vigorous market. In mid-1998, OMG adopted its first standard in this area [4, 6], so recently that details weren't ready when this article was written. Real-time CORBA is an optional extension for all ORBs; that is, no ORB is required to support real-time execution, but if an ORB does claim real-time support, it must provide it in the specified way, accessable via the specified interfaces, in order to be considered compliant. OMG expects to release its real-time specification over a number of years; the first specification will encompass fixed-priority scheduling, control over ORB resources for end-to-end predictability, and flexible communications.

The Portable Object Adapter that couples an object implementation to its ORB is flexible and robust enough to support redundant, fault-tolerant CORBA installations but there is still a need to standardize the process by which this is accomplished. Thus the OMG is presently adopting a standard for fault-tolerant CORBA with a specification encompassing both active and passive redundancy modes.

The mandatory parts of the CORBA 2.0 specification a product must support for certification include many that, although useful in a programming environment, become useless appendages when an executable is burned into silicon. For example, an ORB burned into a chip will never have to support any additional clients; if all of its current clients use the static invocation interface, this ORB could shed its DII support without consequence. Also, if none of the original clients use the *any* IDL type, the ORB could similarly shed the code used to support this complex type. The minimal CORBA RFP will establish an OMG specification that enables such ORBs to bear a CORBA brand without the "excess baggage" that would otherwise be required. Embedded applications tend to be reproduced in great quantities, thus support here has important consequences for the CORBA marketplace.

The CORBAservices: Basic Services for Distributed OO Applications

The Object Management Architecture (Figure 4) builds upon the CORBA architecture and interoperability foundation to realize OMG's vision of plug-and-play component software. A foundation of standard services invoked using standard interfaces, the OMA defines an environment where interoperability penetrates upward from the system level into application components.

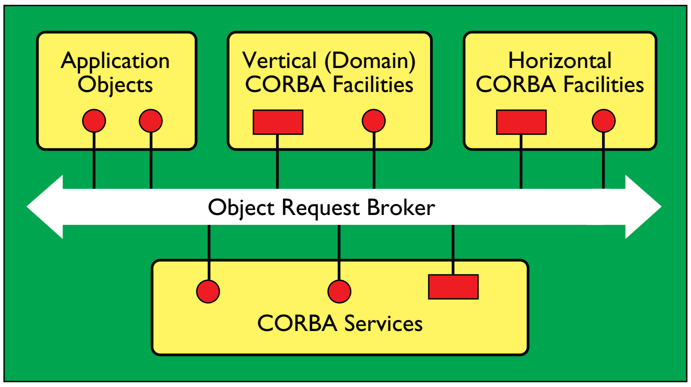
Application Objects
Vertical (Domain) CORBA Facilities
Horizontal CORBA Facilities
Object Request Broker
CORBA Services

Figure 4. The Object Management Architecture (OMA) defines OMG's foundation of standard services and facilities for distributed object architectures

The goal of the OMA is simple: when applications provide basic functionality, let them provide it via a standard interface. This enables a component software market both above and below the level of the interface: below it, multiple *interchangeable* implementations of the basic functionality (compound document management, for instance) may still provide differences in performance, price, or adaptation to run on specialized platforms, while above it, specialized components (a sophisticated editor object, for example) come to market that can operate on any compound document managed by a component that conforms to the standard interface.

The CORBAservices specify basic services that almost every object needs; this part of the OMA started first and the CORBA facilities take advantage of much of it. The CORBA facilities provide intermediate-level services to applications. Application Objects, at the uppermost level, will not be standardized by OMG; this is where vendors will compete in innovative ways to provide the best combination of features and value for the customers.

For each component of the OMA, the OMG provides a formal specification—a published document prescribing the syntax (how to invoke each operation on each object) in OMG IDL, and semantics (what each operation does) in English text. Vendors of OMA-compliant systems then implement and sell the services (some bundled with an ORB package, others not), which are accessed via the specified OMG OMG IDL interfaces. Vendors do not have to provide every service, but every service they provide must conform to the OMG specifications in order to be considered compliant.

The CORBAservices provide fundamental, nearly system-level services to OO applications and their components. Out of the approximately 15 defined services, four functions are key:

- Access to object references across the network, supported by the Naming Service and the Trader Service;
- Notification of significant events or objects' change of state, supported by the Event Service and the Notification Service;
- Support for transactional semantics (two-phase commit and rollback), supported by the Object Transaction Service (OTS); and
- Support for secure interoperability, supported by the Object Security Service.

Other CORBAservices support Object life cycle, relationships, and additional functions.

In summary, CORBA and OMG IDL provide the interoperability infrastructure that objects will use to link together. Then the OMA standardizes a set of common foundation objects, including the key "matchmaking" services Naming and Trader that get clients and object implementations together as necessary, along with other basic services. Whenever a client needs to use a service or an object, it can find it, communicate with it, and invoke it.

The Horizontal CORBAfacilities

As shown in Figure 4, the CORBAfacilities (encompassing both horizontal and vertical/domain portions) fill in the architecture between the basic CORBAservices and the marketplace-provided Application Objects. When the full architecture is realized in off-the-shelf products, companies will be able to share application-level data and functionality to integrate their IS. Since top-level Application Objects will not be standardized, the mid-level CORBAfacilities will be accessed either by innovative clients purchased in a competitive software marketplace, or by targeted modules specifically tailored to each company's needs.

OMG's original plan for the horizontal CORBAfacilities was ambitious, mapping out four major categories: User Interface, Information Management, Systems Management, and Task Management. While these are still useful to conceptualize the scope of this part of the OMA, the Common Facilities Task Force that does this work has been absorbed into the ORBOS (ORB and Object Services) Platform Task Force and no longer exists as a separate group. Thus new facilities are being added slowly, usually in response to specific industry demands. Currently defined CORBA facilities include the XCMF Systems Management Specification, shared with The Open Group, and a Print Spooling facility [1, 2].

The Vertical (Domain) CORBAfacilities

There are many advantages to IDL as a language for specification of standard, shareable objects: It was designed for this purpose, and an interface specified and maintained in IDL is also formally defined in every programming language that has an OMG-specified mapping. And, since late 1997, OMG IDL has also been an ISO standard (number 14750), enabling even formal standards organizations to build and adopt IDL interfaces.

As might be expected, even before the reorganization that created the Domain Technology Committee (DTC) and started OMG's formal entry into domain specification setting, organizations were defining their own specifications in IDL. Because of active OMG support since the committee reorganization in early 1996, these efforts have increased manyfold. At their center is OMG's Domain Technology Committee, which is empowered through the organization's procedures to charter Task Forces in the various domains. These Domain Task Forces (DTF) write requirements documents (RFPs, in OMG-speak) for new specifications, and evaluate and recommend candidate specifications. Based on a DTF recommendation, the DTC conducts a formal vote of adoption, ensuring that every domain specification has the endorsement of all domains and not just the one in which it originated. Thus, general specifications (such as the one for a Currency facility), that benefit from the expertise of members of the Finance DTF that undertook its specification effort, must meet the needs of the other domains in order to pass their DTC vote. In a final step, recommended documents need approval by OMG's Board of Directors to become official specifications.

There are currently eight Domain Task Forces: Business Objects, Finance/Insurance, Electronic Commerce, Manufacturing, Healthcare (that task force has adopted the name CORBAmed), Telecommunications, Transportation, and Life Science Research. Also meeting at OMG but not yet chartered as DTFs are Special Interest Groups in Utilities (primarily electric power, that is currently undergoing deregulation in the U.S.) and Statistics. By March 1998, just over two years after the domain effort started, six domain specifications were either adopted or being considered by the DTC vote and new specifications were moving through the pipeline at a rate of 12 or more per year.

The six completed specifications cover a wide range of items:

1. A Currency Facility, from Finance DTF.
2. A set of Product Data Management Enablers, from the Manufacturing DTF.
3. The Person Identifier Service (PIDS), from CORBAmed.
4. The Lexicon Query Service, also from CORBAmed.
5. An Audio/Visual Stream Control Object, from Telecomm DTF.
6. The Notification Service, also from Telecomm.

Market acceptance is excellent; in spite of their newness, several of these specifications are implemented in products at beta or GA level: Audio/Visual Stream Control Object, PIDS, PDM, and others. PDM is being implemented by a consortium of manufacturers and software vendors. Interoperability of the PIDS specification was demonstrated at the medical software conference HIMSS last year, in a multivendor configuration using the show's network.

Summary

With an architecture based on the Interface Definition Language OMG/ISO IDL, the Object Request Broker, and the Internet Inter-ORB Protocol (IIOP), CORBA components interoperate regardless of location, platform, programming language, system vendor, or network. Desirable features such as asynchronous invocation support, real-time, embedded, and fault-tolerance support are being added, as is a component model. Building on this architecture, the OMA adds the CORBAservices, a set of standardized definitions of components providing services such as object naming, life cycle, security, transactions, and more. Domain-provided additions specify frameworks for specialized but industry-standard components in Finance, Electronic Commerce, Manufacturing, Healthcare, Transportation, Telecommunications, Life Sciences, and general Business. At the beginning of the software process, the Unified Modeling Language and Meta-Object Facility support modeling and design.

This environment has tremendous appeal for virtually any enterprise, but especially for those for which heterogeneity is an issue. With coordinated support for every phase of computing from design through implementation to run time, CORBA integrates legacy functionality with today's sophisticated hardware and software, allowing businesses to shop for the best products and integrate them into a coherent, maintainable architecture. The OMG Web site lists a large number of success stories, large mission-critical CORBA applications in use at various enterprises today (specifically at www.corba.org). The broadly based, growing support for CORBA attests that the number of success stories will continue to expand rapidly. C

References

1. Eriksson, H.E. and Penker, M. *UML Toolkit*. Wiley, New York, 1998.
2. Object Management Group. CORBA Specifications; www.omg.org/library/specindx.htm.
3. Object Management Group. OMG TC Work in Progress; www.omg.org/library/schedule.htm.
4. Object Management Group. CORBA Success Stories; www.corba.org.
5. Object Management Group. OMG TC Work in Progress: Technology Adoptions; www.omg.org/library/schedule/Technology_Adoptions.htm
6. Siegel, J., Ed. *CORBA Fundamentals and Programming*. Wiley, 1996.

Jon Siegel (siegel@omg.org) is the director of Domain Technology at the Object Management Group in Framingham, Mass.; www.omg.org

Architectural Overview

The OMG architecture offers:

- Support for Analysis and Design: UML [5] and the MOF [4];
- Basic object-oriented computing model: The ORB (Object Request Broker); OMG/ISO IDL (Interface Definition Language) and its mappings to C, C++, Java, Smalltalk, Cobol, and Ada [2];
- Distribution: The protocol content specification GIOP and its mapping to TCP/IP, IIOP [2]; alternative mappings and protocols; extensions to messaging and asynchronous invocation semantics [6];
- Component Model (now in process): CORBA Components and scripting; multiple interfaces; objects passable by value [6];
- Specialized modes: Support for Real-time, Fault-tolerant, and Embedded CORBA [6];
- CORBAservices, basic services for distributed object-oriented applications: The most important are the naming and trader services, the event and notification services, the Object Transaction Service (OTS), and the security service [2];
- Horizontal CORBAfacilities: Systems management, print spooling, and similar services [6]; and finally
- Vertical market (Domain) CORBAfacilities: Support for the enterprise, built upon this firm and wide foundation, including standard objects for standard functions, shareable within and across domains [2, 6].