Organizational System Integration through Emerging Technologies

Benjamin Khoo
Computer Information Systems Department
California State Polytechnic University
Pomona, CA 91768, USA

Abstract

Modern organizations face a highly dynamic environment that requires management to integrate the distributed and disparate systems for organizational efficiency. Such integration can promote knowledge growth within an organization. Integrated systems can be developed to overcome the distributed nature of existing information systems or models through the use of advanced emerging technologies. Common Object Request Broker Architecture (CORBA) is one such emerging technology. This paper describes how this emerging technology can integrate organizational systems by illustrating with a simple example and suggests how CORBA can be delivered in the IS curriculum. When students learn how to use an emerging technology to resolve a current organizational problem, they learn the relevant skills for industries' felt needs.

Keywords: design and implementation, emerging technologies, organizational systems

1. INTRODUCTION

The technological explosion of the Internet in the late 1980s and early 1990s has resulted in a paradigm shift that has affected all aspects of our lives. This new information infrastructure has resulted in a proliferation of new technologies; arguably, the most prevalent of which are the data processing capabilities of information systems. The rapid adoption by businesses of the Internet, Intranet and Extranet has pushed the fringes of information systems towards a new frontier. The advent of new programming languages such as Java ("write once, run anywhere") and JavaScript, combined with the client-server architecture adopted by many organizations, has opened new opportunities for information systems researchers to develop distributed, network centric systems. Students in Information Systems (IS) need to be exposed to these emerging technologies. Technologies are considered emerging if it still is being developed or it has not been fully adopted by industry. The Common Object Request Broker Architecture (CORBA) is one such technology. CORBA has come into prominence currently as a solution to integrate distributed systems.

Background

Since the last century, organizations have been motivated to develop the rules of work design based on a model of decentralization (specialization of labor) and economies of scale derived from the Industrial Revolution. This motivation resulted in the propagation of departments within organizations, each specializing in a specific work process. Over time, each of these departments develops some form of computerized models that help to enhance its own productivity. These systems are known as "islands of automation" due to the distributed isolated nature of these systems. In a typical enterprise, there are "islands of automation" distributed throughout the enterprise. In the mid-twentieth century, departments in an organization were able to function as close-knit units. But with the paradigm shift,

organizations became more high tech, efficient, and more complex as technology hid most day-to-day operations from human users. Late in the twentieth century, organizations were so large and complex that often one department did not know what the other related departments were doing. Currently, organizations operate in a highly dynamic environment, requiring management to respond speedily and flexibly to external changes (Morton, 1991). Organizations now must compensate for the loss of visibility of overall organizational functions due to the increased complexity and advancement in technology if they are to remain competitive (Eirich, 1993). Industrial researchers, from the mid-1990s to 2001 (Clarke, Stikeleather & Fingar, 1996; Firestone, 1997; Firestone, 1999; Dabke, 1999; Hummingbird White paper, 2001), had consistently been expounding the need to integrate these distributed and disparate knowledge systems as a possible solution to support a quick response mechanism in an organization. An approach to integrate these systems is by using the services available through CORBA.

2. APPROACH FOR INTEGRATION

The distributed and disparate nature of organizational systems needs to be coherently integrated to "promote knowledge growth, promote knowledge communication and in general preserve knowledge within an organization" (Steels, 1993). The integration approach should: (a) be able to integrate software on diverse platforms, (b) be proven and use mature technology (or has strong industrial support), (c) provide intelligent and dynamic model access, (d) enable the models to have location transparency, (e) enable the user interface to adapt to both knowledgeable and novice users (by providing guidance), and (f) enable the knowledge gained over time to be leveraged to solve similar problems. An emerging technology that meets these criteria is the Common Object Request Broker Architecture (CORBA). CORBA also uses web technologies. CORBA offers an attractive mechanism for the desired integration. CORBA can be used to deliver the integrated system by integrating the various components. When multiple entities can collaborate to perform the reasoning process, the process is usually more efficient and the solution usually is more optimal. The CORBA specification describes how software components can inter-operate across networks, languages and platforms. CORBA provides the services to enable enterprise-wide systems integration and in so doing, provide the framework to make relevant knowledge available on demand to employees (Garone and Buck, 2000). The prevalence of the Internet has also propagated the explosive growth of web-based environments in organizations. CORBA has distinct advantages in such an environment. It is easy to use -- design and code without explicit knowledge of the communication mechanism; it is object-oriented -- since distributed systems have a large number of states that need to be monitored, encapsulation and the use of exceptions can significantly increase the ease of managing the states; it optimizes communication traffic -- its objects are passed by reference rather than by value; it has a set of common services -- including object initiation and a naming service; and it is platform independent (Kashima, 1999). In addition, CORBA provides a mechanism for reuse. If problem-solving methods and knowledge bases are reusable components, then this should lead to a number of benefits for the developer. Since knowledge-base construction is expensive and time-consuming, the reuse of an existing knowledge base, even if it requires adaptation or augmentation, will lead to significant savings. A unified point of access that connects the intranet to the corporate business by integrating all of an organization's applications and back-end systems provide access to employees, which enhances productivity.

A system design can be developed for such an integrated system. A system design refers to a technical blueprint for evolving a corporate infrastructure resource that can be shared by many users and services. To be effective, a system design must satisfy three, often conflicting, requirements (Porter, 1990): (a) it should provide as much vendor-independence as practical; (b) it should be capable of rationalizing the frequent multi-vendor, multi-technology chaos of incompatible elements; and (c) it should incorporate the standards being adopted or likely to be adopted by leading vendors and electronic trading partners. The backbone of the system design is provided by the CORBA services (Orfali, R. and

Harkey, D, 1998; Zahavi, R., 1999; Brose et al., 2001). An example of a CORBA application will be discussed next.

3. A CORBA EXAMPLE

The notion of distributed information systems each with its own capability collaborating to solve a problem was unheard of a decade ago. Today, with the advancement of technology and software, it is possible to develop such a system. There are some mechanisms that are required for such a scenario to work:

1. Discovery mechanism: where a component can know the existence of other components and their capabilities (location transparency can be inherent in the system);

2. Communication mechanism: where components can communicate with one another;

3. Interfacing mechanism: where components can interface with each other or with database systems; and (optional) to add intelligence,

4. Reasoning mechanism: where an inference broker can reason about the capabilities of other components so that the correct component can be invoked to solve the problem at hand.

The integrated system uses the CORBA services to meet the above requirements with a central naming service for the discovery mechanism. The naming service allows clients to find objects within a distributed environment based on abstract/natural names.

A simplified view of the system is illustrated in Figure 1.

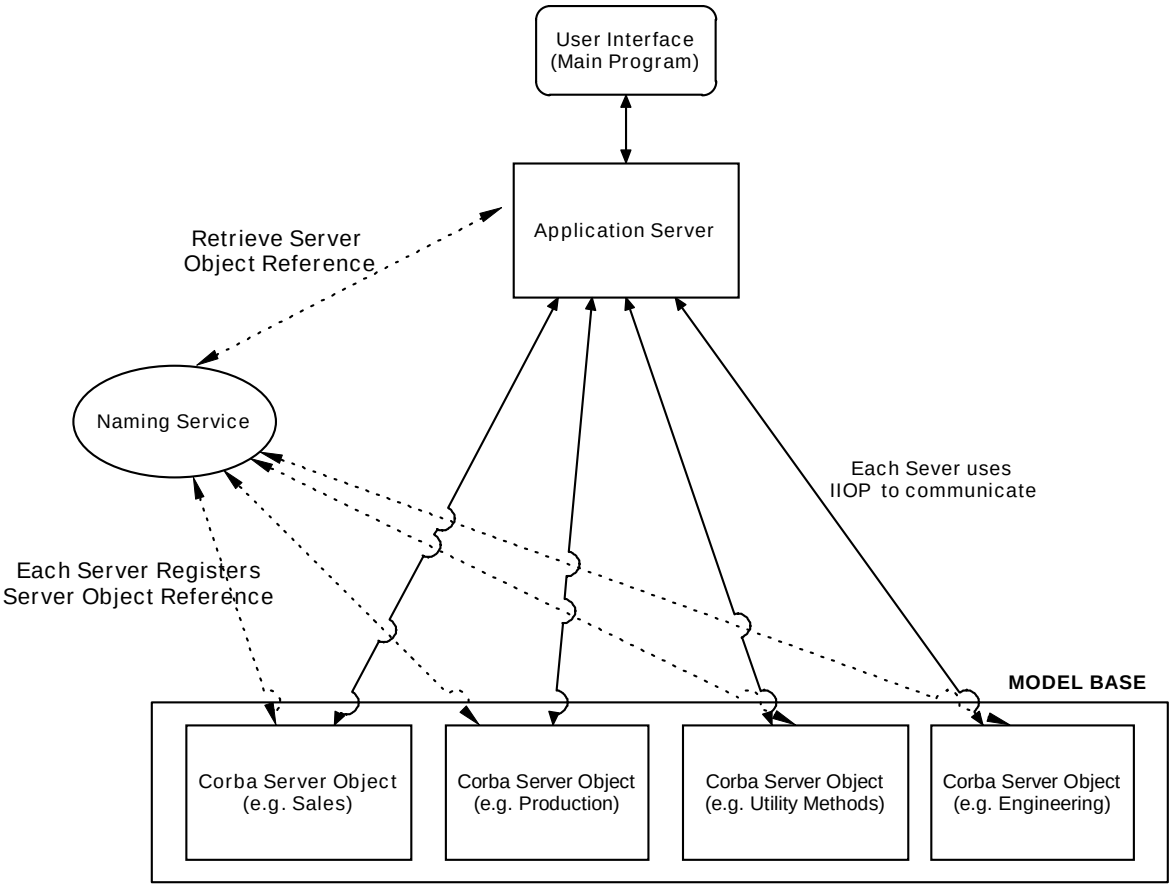

Figure 1. Simplified view of the integrated system.

As Figure 1 shows, a server object registers with the naming service by invoking a *bind* to associate a logical name with an object reference. The naming server adds this object reference/name binding to its namespace database. A client application looks for a server object by invoking *resolve* to obtain an object reference with this name, and the client uses the object reference to call or invoke methods on the target object. Clients (user programs) use the Interoperable Object References (IOR) to communicate with server objects (component models and sub-programs).

The Model Base is a collection of different component models. A CORBA Interface Definition Language (IDL) defines the public interface between the application and each component model. This IDL is independent of the programming language in which it is implemented. It can also be used as a design tool for partitioning systems into components. The application communicates with the CORBA Naming Service to obtain the object reference for the component model and then binds to the object reference for the component model. The Internet Inter-Object Request Broker Protocol (IIOP) request to the component model contains the method call of the component model and the required parameters. Upon receipt of the parameters, the component model will process the request and return the results in the data type as defined in the IDL. The results are returned as an IIOP response to the application.

An ontology of the component models can be set up for the different component models of the organization-integrated system to inter-operate effectively. An ontology is a conceptualization of the different component models and their relationship. This ontology is implemented through the naming service. The role of the naming service is to provide an association between abstract names and CORBA objects. Structured tree type unique names must be assigned to various application components within the distributed environment. This assignment will also allow for flexibility in terms of navigation and configuration. The ontology and naming convention structure is illustrated in Figure 2.

Figure 2 depicts a possible enterprise naming context graph. This graph allows for enterprise level services as well as subsystems and services that are specific to a particular department. This contextual graph will assist navigation to a CORBA object within a multi-departmental enterprise facility. Leaf nodes associate a name with a reference to a CORBA object. This naming context graph includes a departmental context below the enterprise root context. This layering is one approach for handling multiple departments within the enterprise. When all the component models within the enterprise is associated and named on this ontology (or naming context graph), any application object within the enterprise network can be invoked. This referencing leverages one of the advantages of CORBA -- location transparency.

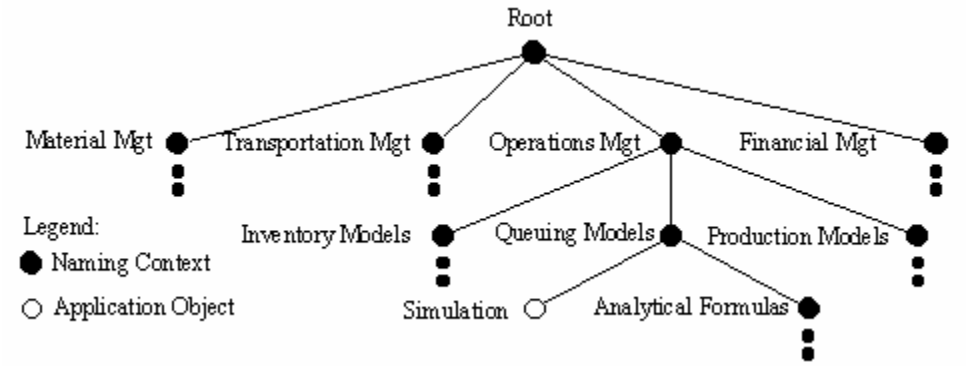

Figure 2. A Component Model Aware Naming Context Graph (Ontology).

CORBA provides the Naming Service as the discovery mechanism. Before an object or components can be accessed, a client must obtain a remote object reference. There are a number of standard specifications that have been developed in different contexts, which address the object discovery. The CORBA Naming Service provides a mapping between a name and an object reference. Storing such a mapping in the Naming Service is known as binding an object. Obtaining an object reference, which is bound to a name, is known as resolving the name. Names can be hierarchically structured by using contexts. Contexts are similar to directories in file systems and they can contain name bindings as well as sub-contexts. Object references are opaque data types, and their string form is a long sequence of numbers. When a service is restarted, its objects typically have new object references. However, in most cases clients want to use the service repeatedly without needing to be aware that the service has been restarted. The Naming Service solves these problems by providing an extra layer of abstraction for the identification of objects. It provides readable object identifiers for the human user. Users can assign names that look like structured file names, giving a persistent identification mechanism.

Objects can bind themselves under the same name regardless of their object reference. The typical use of the Naming Service involves object implementations binding to the Naming Service when they come into existence and unbinding before they terminate. Clients resolve names to objects, on which they subsequently invoke operations. Use of the naming service provides the following benefits: (a) it provides a central repository for object references accessible from applications, thus avoiding having to distribute them to clients via an alternative method, and (b) it allows clients to locate objects through standard names that are independent of the corresponding object references.

The implementation of this integrated system can be done using any CORBA software package with a programming language such as Java or C++. Free online CORBA software can be found at http://www.omg.org/technology/corba/corbadownloads.htm and also at http://adams.patriot.net/~tvalesky/freecorba.html while other materials, like tutorials, can be found at http://www.intelinfo.com/newly_researched_free_training/CORBA.html. The next section will discuss the delivery of CORBA in the IS curriculum.

4. DELIVERY

The distributed objects (through CORBA using Java) course can be taught at any university whose IS curriculum includes the Object Oriented Analysis and Design (OOAD) and Object Oriented Programming (Java or C++) courses. If there were an additional Telecommunications (Networking) course, it would be even better. Most campuses have networked personal computer laboratories; computers in such client-server environment can be used for the hands-on programming exercises. A Java2-capable PC or workstation per student with connection to the local area network via TCP/IP is recommended. A sample syllabus can be as follows:

Course title: Distributed Objects with CORBA using Java

Catalog Description: This course covers the fundamentals of distributed object computing using CORBA. Students learn the CORBA architecture, the Interface Definition Language (IDL) including mapping the IDL to Java and connecting CORBA objects. Students also get hands-on experience building client and server applications in Java using Static Invocation and Dynamic Invocation Interface.

Pre-requisites: Working knowledge of OOAD and Java programming.

Learning Objectives (http://www.objectinnovations.com/CourseOutlines/107.html):

- Understand the Object Management Architecture, the role of an Object Request Broker, and the assistance of the Object Adapter.

- Understand the role of Interface Definition Language in achieving interoperability between various components and design distributed systems using IDL.

- Understand the significance of CORBA Services and the implications of using or integrating various services into a distributed component design.

- Understand the relationship between the Java environment and CORBA, including the details of the IDL-to-Java language mapping.

- Build Java/CORBA server and client applications

- Address practical distributed design issues

Recommended Textbooks:

1. "Client/Server Programming with Java and CORBA." By Robert Orfali, and Dan Harkey. John Wiley & Sons.

2. "Java Programming with CORBA." By Andreas Vogel and Keith Duddy. John Wiley & Sons.

3. "Core Java2, Volume II Advanced Features." By Cay Horstmann and Gary Cornell, Prentice Hall.

Course Outline:
(partially adapted from http://www.objectinnovations.com/CourseOutlines/107.html)

1. Introduction to CORBA

1.1 CORBA Overview (and its advantages)
1.2 The CORBA Standard
1.3 The Object Management Architecture
1.4 The Object Management Group
1.5 CORBA and the Goal of Interoperability
1.6 Object Request Brokers
1.7 Object Adapters

2. Interface Definition Language (IDL)

2.1 Introduction to IDL
2.2 IDL Constructs
2.3 Modules, Interfaces, Operations, Attributes, Inheritance
2.4 IDL Design Issues
2.5 IDL Compiler (Code Generator)

3. CORBA Services

3.1 Naming
3.2 Events and Notifications
3.3 Transactions and Concurrency
3.4 Trader
3.5 LifeCycle
3.6 Persistent State
3.7 Collections
3.8 Security
3.9 Externalization

4. CORBA Basics

4.1 A Simple Illustrative Example
4.2 Creating the IDL
4.3 Creating the Server Application
4.4 Creating the Client Application

5. Java and CORBA

5.1 Relationship Between Java and CORBA
5.2 Portability
5.3 RMI and CORBA
5.4 Other Java APIs and CORBA
5.5 JavaIDL
5.6 CORBA Applications and Applets
5.7 Practical Motivations to Use CORBA

6. The CORBA Runtime Environment

6.1 Static versus Dynamic Invocation Models
6.2 Interface Repository
6.3 Using the Interface Repository
6.4 Dynamic Invocation Interface (DII)
6.5 Using the Dynamic Invocation Interface

7. The Java ORB

7.1 The JavaIDL ORB
7.2 Use of Alternate ORB Implementations
7.3 Initializing the ORB
7.4 Creating Requests
7.5 Object References

8. The Java IDL Mapping

8.1 JavaIDL Compiler
8.2 Mapping for Basic Types
8.3 Mapping for Modules
8.4 Mapping for Interfaces
8.5 Implementation Base versus Tie
8.6 Helpers and Holders
8.7 Mapping for Structs
8.8 Mapping for Exceptions
8.9 Mapping for Sequences
8.10 Mapping for Arrays

9. CORBA Object Implementation

9.1 The Object Request Broker
9.2 What's Done For You
9.3 CORBA Objects and Servers
9.4 Basic Object Adapter and Portable Object Adapter
9.5 Writing the Servant
9.6 Writing the Server
9.7 Publishing the Object Implementation
9.8 Simple Object Persistence

10. The CORBA Client

10.1 Building a Client
10.2 Object Types and Narrowing
10.3 Making Requests
10.4 Peer-to-Peer Systems
10.5 Controlling Location

11. Distributed Design Strategies

11.1 Factories
11.2 Naming
11.3 Lifecycle
11.4 Managing Location
11.5 Persistent Object Strategies and the PSS

Resources

- An Online Introduction to CORBA http://developer.java.sun.com/developer/onlineTraining/corba/corba.html
- The Object Management Group provides comprehensive information on CORBA and IDL http://www.omg.org/
- Visigenic VisiBroker for Java http://www.visigenic.com/
- IONA Technologies OrbixWeb http://www.iona.com/
- Sun Microsystems Neo http://www.sun.com/
- The CORBA website http://www.corba.org
- Software Technology – CORBA Tutorials and Introductions http://www.swtech.com/corba/tutorials/
- Professor Douglas C. Schmidt's CORBA webpage http://www.cs.wustl.edu/~schmidt/corba.html
- Cetus Links on Objects and Components/CORBA http://www.cetus-links.org/oo_corba.html

5. CONTRIBUTIONS

By integrating the CORBA course in the IS curriculum, students are exposed to the system design and implementation using the emerging technology. Students are also exposed to real organizational issues and are provided a "hands on" approach to resolve these issues. The skills that students learn will make them relevant to industry's needs.

6. REFERENCES

Brose, G., A. Vogel, and K. Duddy, 2001, "Java Programming with CORBA," Third Edition, Wiley, John & Sons, Incorp., 2001.

Clarke, J., J. Stikeleather, and P. Fingar, 1996, "Distributed Object Computing For Business," The Next Generation Computing Series, The Technical Resource Connection, Inc.

Dabke, Padmanabh, 1999, "Enterprise Integration Via CORBA-based Information Agents." IEEE Internet Computing, September-October 1999, pp. 2-10.

Eirich, Peter, 1993. SIG Chair and Report Editor. "Enterprise Modeling: Issues, Problems and Approaches." International Conference on Enterprise Integration Modeling Technology (ICEIMT), June 6-10, 1992 Hilton Head, South Carolina.

Firestone, J.M., 1997, "Distributed Knowledge Management Systems(DKMS): The Next Wave in DSS." White paper No. 6, Executive Information systems, Inc.

Firestone, J.M., 1999, "Enterprise Knowledge Management Modeling and Distributed Knowledge Management Systems." White paper No. 12, Executive Information systems, Inc.

Garone, Steve and Neena Buck, 2000,

"Capturing, Reusing and Applying Knowledge for Competitive Advantage." White paper, International Data Corporation, Farmingham, MA 01701.

Hummingbird, 2001, "Enterprise Information Portals: Enabling Knowledge Management in Today's knowledge Economy." White paper, Hummingbird Ltd.

Kashima, Hiroshi, 1999, "An Approach for Constructing Web Enterprise Systems on Distributed Objects." Technical report, IBM Japan Systems Engineering Co., Ltd.

Morton, S.E., 1991, "The Corporation of the 1990's: Information Technology and Organization Transformation." New York: Oxford University Press.

Orfali, R. and Dan Harkey, 1998, "Client/Server Programming with Java and CORBA." Second Edition, Wiley, John & Sons, 1998.

Porter, M. E., 1990, "The competitive advantage of nations." New York : Free Press, 1990.

Steels, L., 1993, "Corporate Knowledge Management." In: Barthès, J.-P. (ed) Proceedings of the 1st International Symposium on the Management of Industrial and Corporate Knowledge, ISMICK'93, pp. 9-30

Zahavi, R., 1999, "Enterprise Application Integration with CORBA Component and Web-Based Solutions." John Wiley & Sons, 1999.