Annals of Software Engineering 13, 71–96, 2002

A Framework for Evaluating Distributed Object Models and its Application to Web Engineering

HOSSEIN SAIEDIAN saiedian@eecs.ku.edu
Electrical Engineering and Computer Science, University of Kansas, Lawrence, Kansas 66045, USA

NABIL GHANEM
Sprint PCS, 10881 Lowell Street, Suite 200, Overland Park, Kansas 66210-1666, USA

JEYABARATHI NATARAJAN
Department of Computer Science, University of Nebraska at Omaha, Omaha, Nebraska 68182, USA

Abstract. The success of building distributed object systems depends on important factors such as architecture, the distributed object model (DOM) selected, and the process adapted in the selection of the DOM. There are a number of DOMs. Although the primary goals of these models are the same, each model has a unique underlying architecture, maturity, and features provided. A critical evaluation of DOMs is thus needed by those organizations that are considering migrating to distributed object computing. The evaluation process can be time-consuming and may drain organizational resources. Most of the current evaluation processes adopted by organizations are not generic enough, and they concentrate only on the problem on hand. Hence, they cannot be used by any other organization, sometimes not even a different project at the same organization. Therefore, a more generalized framework or template is required to evaluate DOMs. This paper proposes a framework to evaluate DOMs. A number of important managerial items such as cost, personnel, and technology resources, training, enterprise changes, and time constraints have been identified, explained, and justified as the evaluation criteria. An evaluation of the most widely used DOMs, CORBA, DCOM, and RMI, is provided using the above criteria. Finally, a case study of a production web-based system is presented to demonstrate the use of the framework.

1. Introduction

Distributed Object Computing (DOC) has become the key word for developing mission-critical client–server applications. With the development of World Wide Web (WWW), Intranet, Internet, and in order to cope with competition, large organizations are migrating to leading edge technologies like electronic commerce and DOC.

For many of the large industries like health care, railroad, and airline, the key factors for success are customer service, a high performance software system, and real-time processing. These industries store and process their base and critical data using mainframe systems due to their high capacity and performance. Many of these corporations reorganize their business by way of business process reengineering (BPR), relocation of business and operations units, and building partnerships with international organizations.

To enable the availability of distributed and secure data to their users through web and client–server applications, these corporations require a high performance, scalable, secure, and interoperable distributed object architecture. Such an architecture is provided by the distributed object models (DOMs), such as Common Object Request Broker (CORBA) by Object Management Group (OMG), Distributed Component Object Model (DCOM) by Microsoft, and Java Remote Method Invocation (RMI) by Sun Microsystems. Any organization wanting to transit to the DOC paradigm has to choose and adopt one or more of these models that best suit their business requirements. The advantages, disadvantages, and success of their transition depend on the DOMs adopted and the limitations they might bring in terms of cost, complexity, flexibility, etc.

The DOM selected could have substantial managerial and technical impacts on the transition process and the organization. The technical aspects affect the managerial aspects either directly or indirectly. Even though studies, benchmarks, and evaluations of DOMs are available, they are arbitrary and concentrate more on technical aspects. The goal of this paper is to introduce a common framework to evaluate DOMs in terms of managerial aspects. The framework provides a systematic approach to analyze and evaluate DOMs on their managerial aspects, which could be used by organizations to select best-suited DOMs for their organization and applications.

Once an organization decides to transition to DOC, it will have to select one or more DOMs. Since there are a number of models available in the market, an organization must select one that is most suited to its needs and also most cost-effective. The objective of this paper is to provide a framework which will help organizations choose the models best suited to their requirements.

The organization of the paper is as follows. In section 2, we will present a brief description of the most widely used DOMs: CORBA, DCOM, and RMI. The evaluation framework is discussed in detail in section 3. Section 4 includes the evaluation result of DOMs CORBA, DCOM, and RMI followed by a web-based system case study in section 5. Conclusions are given in section 6.

2. Distributed object models

There are numerous DOMs available in the market. Also, there are many references available for these DOMs. So, the following sections will provide a brief introduction to the evolution of three major DOMs: CORBA, DCOM, and RMI.

2.1. CORBA

Common Object Request Broker Architecture (CORBA) was defined and controlled as a standard architecture for distributed computing by the Object Management Group (OMG). OMG, the largest software consortium, was founded in 1989 by eleven companies, including 3Com Corporation, American Airlines, Canon, Inc., Data General, Hewlett-Packard, Philips Telecommunications N.V., Sun Microsystems, and Unisys Corporation. Now it has more than 800 companies as its members. The objectives of OMG are quoted by Haughey [1999] as:

> "... to create a component-based software marketplace by hastening the introduction of standard object software. The organization's charter includes the establishment of industry guidelines and detailed object management specifications to provide a common framework for application development."

OMG defines only specifications and does not implement or deliver products; OMG's member vendors adopt the standards and specifications and deliver CORBA products.

Vinoski [1998] describes CORBA as "*an application integration technology.*" The goals of CORBA are to facilitate the development of client–server applications, rapid integration of legacy systems, off-the-shelf applications, and new development. On the basis of object orientation, CORBA achieves the integration of heterogeneous applications, reusability, and portability [Emmerich 1997]. Guttman and Appelbaum [1998] have traced the evolution of CORBA in generations as follows: first generation running from 1990–1996, second generation, during 1996–1998, and third generation since 1998. Within a year from its establishment in 1989, OMG adopted CORBA 1.0. Over the next several years, few CORBA Services were designed, Interface Definition Language (IDL) interfaces for languages C, C++, and Smalltalk were developed. Some vendors adopted the specification and came up with products having minimal functionality.

The CORBA 2.0 specification was released in 1995. CORBA 2.0 made a big difference in the Object Request Broker (ORB) and commercial world. It was more stable than earlier versions, answered many questions, and provided the Internet Inter-ORB Protocol (IIOP). Many vendors supported CORBA 2.0 and delivered reasonably robust and good products. But, these products varied in the level of support for standards, performance, robustness, scalability, ease of use, etc. The names of few CORBA vendors include Iona, Visigenic, ICL, Sun, IBM, HP, and Digital [Guttman and Matthews 1998]. Between 1996 and 1998, OMG continued to refine the specification and provide some extension to the specification. It has added language binding to many other languages and improved the existing language bindings and core functionalities. The last formal version of CORBA specification released was CORBA 2.3. The CORBA specification and its products evolved and matured during this period. Many CORBA products are available in variety of platforms. Also, Guttman et al. [1998] have commented that:

> "We have seen a major consolidation within the CORBA vendor community. ...the CORBA vendor community consists of fewer but stronger vendors. This is good for the industry. As this consolidation occurs, we are seeing a maturation of compliant products from these vendors."

OMG's focus has expanded to provide business objects for many domains, including healthcare, finance, telecommunications, and manufacturing. Also, it is working on extending the specifications to include object-modeling techniques. OMG's CORBA 3 also referred as CORBA Component Model (CCM) has three major categories of specification, including Internet Integration, Quality of Service (QOS) Control, and the CORBA Component Architecture as mentioned by John Siegel in the release information docu-

ment of CORBA 3. CORBA standards are evolving, and providing additions and improvements to CORBA is a continuous process.

2.2. DCOM

Microsoft developed Distributed Component Object Model (DCOM) technology, which enables objects residing on different computers or on the same computer to talk to one another. Microsoft's initial objective in developing the object model was to make communication possible between different Windows applications. With this in mind, Microsoft created the Object Linking and Embedding (OLE) scheme, which facilitated linking compound documents in the Windows platform. OLE became a popular way of integrating and linking data between applications. Further enhancement of this technology resulted in Microsoft developing the Component Object Model (COM).

Since then, Microsoft has extended and added more functionality to COM on a frequent basis. COM was considered "*a complex, fragile collection of incremental solutions*" [Quoin 1998]. It became stable and earned its credits with its successor ActiveX. COM allowed the integration of binary components in the same machine in an object-oriented fashion, and has been standardized as a binary integration technology in the Windows world.

Microsoft wanted communication between objects to be extended to remote computers, across networks and even on the Internet. This led to the birth of a very significant and popular object model called DCOM, which provided solutions for distributed applications. The next-generation of COM and DCOM technology is COM+, which has enhanced runtime and object services.

2.3. RMI

Sun's [1998] definition of RMI is as follows:

> "Java™ Remote Method Invocation (RMI) is distributed object model for the Java language that retains semantics of the Java object model, making distributed objects easy to implement and to use. The system combines aspects of the Modula3 Network Objects system and Spring's subcontract and includes some novel features made possible by Java."

The RMI system goals include the Java language goals for supporting distributed objects with some extensions supported by the RMI system [Sun 1998]. Remote Procedure Calling (RPC) is a common way for processes to communicate with each other in a distributed system [Wollrath *et al.* 1997]. In RPC, control can flow from a point in one program to a point in another program on another machine. But RPC is not good enough for distributed object systems because in distributed object systems communication between program-level objects residing in different address spaces is needed [Sun 1998]. Also, there is a need for a mechanism to match the semantics of object invocation.

Sun Microsystems designed Java's RMI to support pure-Java distributed objects in a seamless manner by combining qualities of Java with a language-centric design, which

significantly simplifies the traditional RPC systems. Since the distributed computing takes place entirely in Java, its systems can make use of the features of the Java type system and garbage collection. RMI supports distributed polymorphism and pass-by-value objects in remote calls.

Sun developed interim RMI as part of Java Development Kit (JDK) release 1.0.2, which came after the first JDK release 1.0 in January 1996. RMI's goal is to let an object running in a Java Virtual Machine (JVM) to communicate to an object running in another machine across a network [Waldo 1998]. The JVM is a machine simulated by software on the computer [Curtis 1997]. RMI also requires that the objects involved in the communication are Java classes.

There are many releases of RMI. The core RMI was released with JDK 1.1 in February 1997 even though JDK 1.0.2 had all the RMI APIs as JDK 1.1. The new enhancements to RMI including activation and custom socket protocol capability were released with Java 2 SDK v1.2, and Java 2 SDK v1.3 evolved with more enhancements for RMI and serialization used for remote communication. The RMI Security Specification draft has been revised for the third time as of April 2000. Sun and IBM have worked together and developed Remote Method Invocation over Internet Inter-ORB Protocol (RMI over IIOP) 1.0.1 based on OMG's specifications "The Java-to-IDL-mapping" and "Objects-by-Value", which was shipped in June 1999. The objective of RMI over IIOP was to add CORBA compliant distributed computing capabilities to the Java 2 platform. It provides the best features of RMI and CORBA. As of February 2001, RMI over IIOP specification is an integral part of the Java 2 Platform, and it will continue to be part of future versions of Java.

3. Evaluation framework

The DOM adopted by an organization plays an important role in its transition process towards DOC. Organizations willing to move to DOC will be looking at one or more DOMs depending on their present environment and the business requirements. As discussed earlier, there are a number of models available in the market.

There are many approaches an organization can adopt to choose a DOM. An organization may pick a DOM which is very successful in the market; it may use the benchmarks and evaluation done by research organizations and other enterprises. However, these approaches may not work for organizations having unique and specific requirements. In this case, the organization has to deal with the DOMs evaluation and selection process. But understanding all these models and selecting the most suited model for an organization is a difficult and time-consuming process.

This section provides a framework which will help organizations choose the best-suited models based on their requirements. The basic criteria to choose DOMs are categorized as part of the framework. The framework research identified managerial and technical categories of criteria to be evaluated. According to Nakamura [1998] the goals of enterprise middleware are to support productivity improvement, cost reduction, im-

proved customer service, and enhanced profitability. The DOMs, which are a type of middleware, have to be evaluated for how well they support these business goals.

This paper presents a framework based on the managerial aspects. There are many reasons for looking at the managerial aspects vs. the technical aspects. First of all, the technical criteria affect the managerial criteria directly or indirectly and it becomes an inseparable part of the managerial aspects. Secondly, as Sneed [1995] has argued, the technical details are irrelevant if we cannot make a business case for solving a problem. Much research has shown that the ultimate and real factors for introducing tools and techniques will be organizational and managerial, not technological [Guttman and Matthews 1998]. And finally, the discussions with corporate Information Technology (IT) managers also emphasized that the ultimate goal of the organization was business and managerial oriented rather than technology oriented. The resultant framework is as shown in table 1. The table shows the framework criteria, factors to be evaluated as part of the framework criteria, and the technical criteria affecting the framework criteria.

The chosen framework criteria for evaluation are given in the following subsections. Section 4 provides the evaluation result of the DOMs: CORBA, DCOM, and RMI as well as a case study illustrating use of the framework.

Table 1
Framework for evaluation of DOMs.

Framework criteria	Factors	Technical criteria
Cost	• Software cost including initial acquisition cost, licensing and licensing costs, bundled software cost • Resources cost including human resources, training and support services cost • Development cost including cost for design, development, deployment, maintenance, time cost	• Reusability • Leading edge technology support (e.g., Internet development) • Complexity/ease of use • Evolution and maturity • Security
Training	• Management training • Developers training • Installation and administration training • Additional tools and technology training	• Complexity/ease of use • Evolution and maturity
Resources	• Software and hardware • Documentation • Support and service • Training resources • Skilled personnel	• Evolution and maturity
Enterprise changes	• Business process reengineering • Mergers, acquisitions, partnership, and globalization	• Reusability (integration and interoperability)
Time constraints	• Design and development time • Maintenance time	• Complexity/ease of use • Reusability

3.1. Framework criteria

The following subsections describe in detail the framework criteria, the importance of each criterion as part of framework, and reasons for the evaluation of each criterion.

3.1.1. Cost

In software industry the first question asked about a project or product at stage is: how much does it cost? The top management and the marketing experts will be looking at the immediate, instant benefits and increase in revenue. Building quality software takes lots of time, effort, and cost. The same is true with building software for reuse.

Even though it might seem that quality software and reusable components costs a lot to develop, in the long run they save time, effort, and cost. In some projects the actual development may not cost much but the tools, maintenance or the time span of the system might be costly. Migration projects like legacy system migration to DOC may add some expenses, such as training the existing experts on the new technologies and tools, training the skilled technicians for domain expertise, and upgrading existing software and hardware. There are many legacy migration projects that failed because they exceeded their budgets. There are many projects abandoned after spending so much money and time [Schneidewind and Ebert 1998]. The US Internal Revenue Service project to replace 27 aging systems started in 1989, was scrapped around 1997 after spending $4 billion [Bacon 2000].

DOMs are different in their nature, characteristics and level of maturity. Some characteristics of DOMs affect the cost involved in the development of distributed object applications. The maturity level of a DOM would complement this point. The cost involved in training software engineers might be less expensive for a mature model compared with an immature DOM. Though management would want to achieve their goals, the ultimate motivation behind those goals is cost-driven. Cost is an important criterion for choosing a DOM. There are many categories of cost to be considered to calculate the total cost involved in using a particular DOM. These categories are as follows:

Software cost – includes cost such as buying the runtime software, licensing or leasing of software, bundled software, etc. The brief description of these costs are as given below.

- *Initial acquisition cost* – the cost for purchasing DOMs, required hardware, networking tools and equipment, and the cost in installing and configuring the software and hardware.
- *Licensing costs* – the cost involved in purchasing additional licenses for developers, and any additional licensing cost the organization might have to be responsible for when it deploys the product to their customers.
- *Supporting hardware and software costs* – since some of the DOMs are complex in nature, the organization might have to purchase some additional tools to expedite the development work. For example, distributed applications are very difficult to

test and debug, so organizations may have to purchase some test tools to expedite the development work.

- *Leasing of software to reduce upfront cost* – sometimes DOMs and other required software could be leased instead of purchased to reduce the upfront cost, and could be paid on a term basis.
- *Increased bundling of software with hardware* – some vendors reduce the price of software or may provide more features for the same price if the hardware is also purchased from them.

Resources cost – would include the cost to bring human resources, training resources, etc.

- *Training and support services cost* – cost required to train the software engineers, any other support that may be required with the DOM and the ongoing development service and support required by the organization from inception through deployment phases of the application.
- *Cost for additional and better-trained staff* – the cost for the additional and specialized staff and software tools.

Development cost – cost involved in design, development, maintenance, and deployment.

- *Design and development cost* – cost involved in design, development, integration, and the time required. The complexity of the DOM architecture contributes to this cost.
- *Long-term maintenance cost* – this would involve the cost for the upgrade and migration of the DOMs as well as the applications.

The DOM's support for technical features such as Internet development, security, reusability, expendability, complexity, and maturity needs to be looked into along with the cost evaluation. Distributed Internet computing promotes workflow automation, business-to-business, and business to consumer transactions. Those features in turn reduce the organizations business administrative or maintenance cost, sales cost, and inventory cost [Nakamura 1998]. Research has shown that reuse and extensibility of components reduce the software development cost [Oberndorf *et al.* 1997]. With respect to the criteria shown above, the DOMs impact the revenue of an organization. An organization should evaluate DOMs for any cost impacts as the success of an organization is measured by its revenue.

3.1.2. Training

DOC is well known for its complexity. This complexity requires the organizations involved in DOC development to train their software and domain engineers constantly. The DOMs are immature and, still evolving, and new standards and processes are emerging continuously. This leads to constant releases of standardization documents, new

product releases, upgrades, and enhancements. Organizations need to validate and evaluate the new releases, upgrade tools, and train their software engineers with the new releases.

This causes additional work to the organization to manage any issues with the maturing process and to train the technical and domain experts. Apart from training the experts on the technical side, an organization has to train the domain experts, software engineers with the domain technology, and new business process requirements. It is required to train the software engineers for a good mixture of technical and domain expertise so that they can understand the business requirements and develop an architecture that will reflect the business goals.

Training is a time-consuming and costly process. Proprietary and immature DOMs may drive up the complexity to the training process and increase the overall training cost. The DOMs differ in their implementation details and support for features. An example would be DCOM, where security can be achieved by both configuration and by programmatic control depending on component-wide and method-wise security needs. For these reasons DOMs have to be evaluated for variety of training requirements, including the feasibility of getting internal and external training services, and any cost and management issues the DOMs might bring. DOMs have to be evaluated for the potential areas of training an organization might have to plan for including:

- Educating both the upper and middle management;
- Training for programmers;
- Training the installation and administration engineers; and
- Training for any other additional tools and technologies. Any other special training might be required in areas such load balancing, performance tuning, security, clustering, etc.

Some of the other criteria that impact the training are the complexity of the DOM, its ease of use level, availability of the training firms, and the organization's training process. A DOM designed with the goal of simplicity and ease of use may not require in-depth training.

3.1.3. Resources

The DOMs are unique in their characteristics and differ with their vendor support, platform support, maturity, hardware, and software requirements. These differences may require additional categories of resources as part of the migration process. If the model is not matured enough, finding skilled software engineers will be a challenge for management. There are a few options to handle this issue.

The organization could try to bring in contractors and consultants from the vendors who were involved in the development of DOMs. This may not be feasible because most likely the vendors cannot provide support throughout the life cycle of the migration process except for providing some specialized consultation in the inception phase of the

project. Another option is to bring in consultants from firms providing specialized services in object-oriented and distributed computing. But this is not a cost efficient option. Alternatively, the organization could give the migration project for outsourcing, which might bring the drawback of maintenance issues and other issues related to outsourcing. It would be easier to find documentation, services, and support for matured models compared with immature models.

Many of organizations data are distributed, dispersed, and maintained on different platforms. So it is required that the DOMs are available in many platforms and interoperability and integration are necessary. In some organizations components and business processes may have been developed in different programming languages. The DOM's specification may not be able to unify those components and processes. For example with RMI the programming language is restricted to Java, which may not fit the organization's business process.

In the current software-computing environment, the developers need tools and technology, which are fully integrated as well as 7×24 support service [Murphy 1995]. Today's software systems require strengths from both domain and computer experts. The following are the categories of resources required by organizations those are willing to transition to DOC:

- *Software and hardware* – availability of required software and hardware required for the DOM with feasible cost.
- *Documentation* – documents pertaining to the DOM's specification, standards, history, past experiences and issues, product vendors, and DOM users.
- *Support and service* – support from the DOM's vendors.
- *Training resources* – organizations offering training on DOMs and related tools, online training resources, and computer-aided training resources.
- *Skilled personnel* – availability of skilled personnel whom can be hired by organizations, sources for specialized help, and resources offering outsourcing services.

These resources impact the success of an organization's transition to DOC. All the categories of resources may not be available for a DOM. DOMs have to be evaluated for the availability and feasibility of the required resources, because some of the resources may be easily available and accessible but may not be cost effective. The criteria such as evolution and maturity of the DOM would greatly impact the resource criterion.

3.1.4. Enterprise changes

To meet the customer's needs and to be on the competitive edge, organizations undergo changes such as business process re-engineering (BPR), mergers, and acquisitions. In addition, to reach international customers and their needs, organizations build partnership with international enterprises, relocate their operations and facilities, and diversify their solutions and services.

The above organization changes bring complexities to the Information Technology/Systems (IT/S) and their maintenance. BPR may cause system migration or adoption of new software engineering models and methods. For example enterprises have moved from mainframe to client–server object-oriented software systems to store and process their business data and are now adopting electronic commerce and Internet systems as the means to do business with their customers. The information systems have to cope with the changes by aligning the technology with the new business process.

Acquisition, merger, and partnership changes may bring about the need for legacy system integration and migration of systems. They may produce their own integration, interoperability, and heterogeneity issues.

Sometimes these changes may not only require extending the applications and components but also making changes to the DOM itself. Not all the DOMs are extensible by nature. The DOMs CORBA and DCOM support extensibility [Devarakonda 1998]. The CORBA design allows users to add services and replace object adapters. There could be components that can be reused from the merged organization. The DOMs should be evaluated for reusability with out degrading performance.

The chosen DOM should handle these changes and the challenges they may bring in. Hence it is necessary for an organization to evaluate DOMs from the perspective of enterprise changes to see how well they can handle issues and requirements caused by the changes. The enterprise changes require DOMs to support reuse, which is impacted by the integration and interoperability capabilities of the DOMs. According to McArthur [1999], the factors affecting interoperability in component-based architectures, which applies to the DOMs, also include the following.

- *Interface* – support for standard and consistent and multiple interfaces.
- *Location transparency* – location and connection transparency.
- *Synchronization* – support for asynchronous and synchronous communication.
- *Threads* – support for multi-threaded control.
- *External dependencies* – there should not be any dependency constraints to elements external to the architecture.
- *Performance* – the level of performance with in the architecture.
- *Availability* – level of availability of components with in the architecture.
- *Reliability* – level of reliability.

There are two types of integration required with DOMs. One would be the integration support with the application development environments and the other would be the legacy system integration, which need to happen at the back-end of the application. The factors impacting integration include:

- *Portability* – environmental independence, support for plug and pull in multiple environments without changes;
- *Adaptability* – ability to modify, modularity and granularity support;
- *Fault tolerance* – how well failures are handled;

- *Extendibility* – ability to add new services;
- *Scalability* – expansion capability on the requirements of a specific functionality;
- *Environment constraints* – free from platform, programming language and service constraints; and
- *Robustness* – fewer environmental or context dependencies.

Since the above-mentioned factors of DOMs play a viable part in the enterprise changes the DOMs have to be evaluated for the above factors in regard to the enterprise changes criteria of the framework. The DOMs relationship to the organization's future technology plans also need to be looked into too.

3.1.5. Time constraints

The delivery dates and budgets for software projects are hard to estimate as the evaluation criteria are arbitrary and they mainly depend on the problem on hand. For many service-oriented enterprises slippage in deadlines is not acceptable because of the nature of their business. Time constraints are an important factor for mission-critical and customer service-based projects. So DOMs have to be evaluated considering time constraints to see how well they can accelerate the development work to finish the project on time.

Building a complex distributed system takes time, and the complexity or ease-of-use criterion of DOMs impact the design and development of distributed applications. Because of the tremendous development of Internet computing and experience on the World Wide Web, users' expectations for distributed systems are very high. Users want the information to be available worldwide instantly. So organizations are expected to deliver the information instantly to be competitive in the market.

The objectives and goals of DOMs and the features they offer vary. For example, CORBA and DCOM offer Interface Definition Languages (IDLs) that significantly reduce complexity, which in turn promotes the productivity of building distributed systems. Hence, the DOMs have to be evaluated for how well they can simplify the complexities involved in building distributed systems in order to increase productivity so that organizations meet their time deadlines and expectations. Because, it is on-time delivery that keeps an organization on the competitive edge, growing, and achieving its goal.

The DOMs have to be evaluated for the features they provide, which would facilitate the organization in meeting its time constraints. These features should be targeting the life cycle of the development process including the requirement gathering, design and development, deployment, administration, and maintenance phases of the application.

4. Evaluation of distributed object models

In section 3 we reviewed the framework for evaluating DOMs. This section presents the evaluation results of CORBA, DCOM and RMI in terms of technical and managerial aspects and a web-based case study to demonstrate the use of the framework.

Table 2
Summary of technical features of DOMs.

Feature	CORBA	DCOM	RMI
Vendor support	CORBA is a specification defined and controlled by OMG and not an actual reference implementation	DCOM is a Microsoft specification and reference implementation	RMI is a Sun Microsystems specification and reference implementation
Platforms availability	Available in many platforms	Available in Microsoft and few flavors of UNIX platforms	Any platform with JVM
Language support	Depends on the CORBA product	Supports main languages	Mainly Java and others are via JNI and RMI over IIOP
Portability	Provides portability on the basis of object-orientation	Supports portability in Windows platforms	Supports near-perfect portability with the expense of performance
Fault tolerance	As per the standard	NT clustering	Server side Java
Load balancing	Depends on the CORBA product	By connections and thread	By connections and thread
Synchronization	Asynchronous	Synchronous	Asynchronous
Protocol	IIOP	DCE/ORPC/HTTP	JRMP/IIOP/HTTP
Objects Support	Fully distributed heterogeneous objects	COM objects and distributed behavior	Distributed RMI, RMO over IIOP and HTTP objects
Interfaces	OMG IDL, IIOP, Interface Repository	MIDL	OMG IDL
Garbage collection and memory management	Not supported automatically. Applications have to deal with the problems	Taken care by DCOM	Provides automatic garbage collection and memory management
Security	Specified in Security Service	Provided by DCOM, and also can be controlled in application	Java built-in Security Manager and also can be controlled in application
Legacy integration	IDL, Wrapper	MIDL, COM objects, Wrapper	Wrapper, OMG IDL, JNI, and RMI over IIOP
Database integration	Specified in Query Service	OLE DB/ODBC	JDBC
Deployment	ORB needs to be deployed on each server and client	Runtime is included with Windows platforms	Runtime is included with Java environment

4.1. Evaluation of distributed object models

In this section we will be looking at the ready-to-apply evaluation results of the DOMs. These results will help an organization in understanding the unique characteristics of each model and how they differ from one another. Such an understanding of the models

Table 3
DOMs evaluation on cost.

Cost factor	CORBA	DCOM	RMI
Software cost	• Costs more depending on the product and platform support	• Costs less as the runtime software is shipped with the operating system • Development tools do cost but comparatively less	• RMI comes as part of the JDK and can be freely downloaded from Sun's website • But IDE tools cost more than DCOM tools
Resource cost	• Costs more, as skilled and sophisticated resources are required	• Costs less and plenty of resources are available	• Costs more, as it is still emerging and not easy to find experienced resources
Development cost	• Costs more compared to DCOM	• Costs the least in Windows platform	• Costs more than DCOM

is required by organizations willing to transit to DOC.

A summary of the DOMs' evaluation based on technical features is given in table 2.

4.1.1. DOMs evaluation summary for managerial criteria

We have seen the technical evaluation summary in the previous section. In this section the DOMs are evaluated on managerial aspects for each framework criterion and factor. The evaluation on the basis of cost, training, resources, enterprise changes, and time constraints are shown in tables 3–7.

5. Web engineering case study

Web engineering is an attempt to add systematic and sound scientific, engineering and management principles to the architecture, design, development, deployment and maintenance of web-based applications.

There is a need to bring and share experiences of building large web-based systems to develop more systematic, repeatable, reusable architectural styles and design patterns instead of repeating the same costly mistakes. Having discussed the framework for evaluation and selection of DOMs, it would be more appropriate to demonstrate the use of the framework by considering a practical example. This section applies the suggested framework to a practical web engineering application. This case study looks back on a decision to implement a DOC framework using Java and RMI in an IT department in a major telecommunications company. We will refer to this IT department as Service Delivery Platform or SDP. The following discussion evaluates the decision made in light of the suggested framework. SDP's architecture team had to persuade both management and the customers to make all new applications web-based and to move away from desktop applications. The main selling point was deployment and code portability.

The Routing Table Maintenance (RTM) system is a new web-based system which allows users to maintain a selected set of tables in the DMS 250/300 switches. This web-

Table 4
DOMs evaluation on training.

Training factors	CORBA	DCOM	RMI
Management training	• Need to train the upper management on OMG and CORBA • Middle management has to be trained	• No need to train the upper management; an orientation would be good as there are much changes going on • Need to train the middle management	• No need to train the upper management; an orientation would be good for the upper management considering the recent enhancement • Need to train the middle management
Developers training	• Extensive and sophisticated training required	• No need for an extensive training compared to CORBA • Products in non-Microsoft platforms require training	• No need for an extensive training other than the Java and IDE tools training
Installation and administration training	• Required, as the ORB has to be installed in both the client and server	• Need to train on tool such as MTS	• No need. The recent enhancements with the Java 2 platform need some installation and admin training
Additional tools and technology training	• Depends on the CORBA product	• Products in non-Microsoft platforms may require some training	• Training on security is required when integrated with existing systems • Training on tools such as RMI over IIOP

based system is thread-safe and graphically very rich. These tables control customers' voice and data services and features. These switch tables are dynamic, i.e., based on a value selected in one field. The following fields may vary in number and type. The business logic, which describes the rules of populating the different tables, is database-driven. This means that we could change the rules dynamically without the need to change code. The client graphical user interface is very rich graphically. The response time of the system should be less than two minutes.

The following architectures were our options:

- **Java-based architecture**.
  This architecture is a four-tiered architecture. The first tier was the client tier. This tier was strictly based on the Java Applet technology. The second tier was the Java Servers tier that contained the print server, database server, and other services on an NT server platform. The Java Servers in the second tier are reusable by future projects. This helps in focusing the development efforts on the business logic. The

Table 5
DOMs evaluation on resource criteria.

Resource factor	CORBA	DCOM	RMI
Software and hardware	• Many CORBA products available • Supports many languages • Support software depends on the CORBA product • Many platforms support • Supported by many hardware vendors	• Only one product by Microsoft • Supports all major languages • Integrated application development suits available • Supported mainly in Windows and few UNIX platforms • Supported in many hardware	• Many products available even though the specification is by Sun • RMI is restricted to Java • Available in all platforms that support JVM • Integrated application development suits available • Supported by many hardware
Documentation	• Online documentation available at `http://www.omg.org`	• Online documentation available at `http://www.microsoft.com` and MSDN `http://msdn.microsoft.com`	• Online documentation available at `http://java.sun.com`
Support and service	• Available in plenty and depends on the product	• Available in plenty	• Available in plenty
Training resources	• Available in plenty and matured	• Available in plenty and matured	• Available in plenty but not matured
Skilled personnel	• Need sophisticated skills • Available in plenty, matured, and costs more compared to DCOM	• Sophisticated skills not required • Available in plenty, matured and cost effective	• Need moderate skills • Available in plenty but not matured and costs more

third tier was the business logic servers written in C++ on a Tandem platform. The fourth tier was the database itself.
The reason for putting the RMI services on an NT server and not on the Tandem server was for the lack of a Java Virtual Machine (JVM) implementation on the Tandem platform at the time of architecting the framework.
This architecture required an automatic one-time installation of a Java Plug-in on the users' machines. When the user connects to the specific website, the installation of the Java Plug-in starts automatically with minimal effort, few clicks, on the user part. Rolling out new releases is as simple as replacing jar files on the NT server where the Java servers reside.

- **DCOM architecture**.
  This architecture was a three-tiered architecture. The first tier was the Client Graphical User Interface which was a Visual Basic application that connected

Table 6
DOMs evaluation on enterprise changes.

Enterprise changes factor	CORBA	DCOM	RMI
BPR, mergers, acquisitions, partnership, and globalization	• High support for enterprise changes	• Not well supported at this point of time	• Does not support all aspects for enterprise changes well

Table 7
DOMs evaluation on time constraints.

Time constraint factors	CORBA	DCOM	RMI
Design and development time	• Reduces the overall design and development time if reuse and integration are the main goal • Also depends on the product	• Meets the time constraints by rapid application development provided the platform requirement is Microsoft	• The best among three for new applications • Fair or as good as CORBA for existing systems requiring integration
Maintenance time	• Good for integration application and depends on the product	• Good in Microsoft platforms	• Good for new applications, for existing application integration some "glue code" has to be written • Also, depends on the product

to the servers on the Tandem platform using a layer of software which implemented DCOM. This approach required another layer on the client side to facilitate the communication with the Tandem server. The second tier consisted of C/C++ servers running on a Tandem platform. The third tier was the database itself.

This architecture required a push process to install the application on all of the clients' machines (different cities in the US). Also, it required an initial configuration of the communication layer to the Tandem server. The user machine had to be rebooted after every push process.

- **CORBA-based architecture**.
  CORBA is used in this architecture instead of RMI. The client software communicates with an ORB on the Tandem platform. This is a three-tiered architecture similar to the first option, except there is no need for the NT server and Java services. All of the servers and services would exist on the Tandem platform.

The client is to be written in C++ using Microsoft Foundation Classes (MFC). The client layer requires a DCOM layer to communicate with the ORB on the Tandem platform. This requirement is similar to the DCOM architecture client layer requirement.

5.1. Goals and requirements

RTM is a new web-based system to be built to replace an erroneous manual process in maintaining DMS250 and DMS300 switch tables. Our internal customer and the production support team wanted a system that, in addition to supporting the business requirements, supported the following goals and requirements:

- Ease of deployment.
- Web-based.
- Track changes per user.
- Multiplatform support.
- Reduce the cost of maintaining the application.
- Reduce or eliminate downtime when upgrading the application.

The new system is explained in the next section.

5.2. Proposed distributed computing systems

The new RTM system is to replace an erroneous manual process in maintaining DMS250 and DMS300 switch tables. The system is to be web enabled with a capability to track changes made to records within each table. This gives the users a history log of events when troubleshooting a customer's voice and data services and features. It also provides management with activity reports per user in resolving technical tickets.

In the new RTM system, the users have the ability to change the business logic engine of the application by making data changes in the database. This is necessary in order to add new features as part of a new software switch release or an FCC mandated change.

We will now use the framework in the evaluation and selection of DOMs for the new proposed system. The models CORBA, DCOM, and RMI will be evaluated with regard to the goals and requirements in section 5.1.1 for each criterion of the framework. The factors of each criterion are analyzed in detail to see the SDP's requirements and constraints before the evaluation.

5.3. Cost

The overall goal of this new architecture is to reduce the maintenance cost of deploying, enhancing, porting and to web-enable all the new applications. From the cost evaluation summary of the DOMs as shown in table 3, DCOM software cost is the least expensive as long as the platform is Microsoft, which in not the case in our large Enterprise

Systems. RMI is less expensive than CORBA because RMI is bundled with SDK; however, enterprises have to buy IDE tools for RMI, which costs either less or the same as CORBA IDE tools. Skilled engineers for RMI cost more than CORBA and DCOM. It is easy to get skilled resources and hardware resources for DCOM and they are cost-effective in Windows platforms. CORBA and RMI hardware resources cost more than DCOM in Windows platforms. CORBA training and support costs are more expensive than DCOM and more or less equivalent to RMI. Hence DCOM would be a better choice than CORBA or RMI.

5.4. Training

The department has to train their software engineers and SDP is willing to face the cost and any other challenges with the training. Let us consider CORBA for SDP's training requirements. From table 4 it is clear that SDP has to train their upper and middle management, and it has to train their existing developers for the design and development of the application with CORBA. The developers or administrators have to be trained on installing the runtime software. SDP has to train the developers on additional tools. The advantage of DCOM over CORBA is that there is no need for upper management training and there is no need for extensive developer training provided the platform is restricted to Microsoft. Training on tools such as MTS is not a big issue. Comparing RMI to CORBA, there is no need for upper management training and installation of RMI runtime separately compared with CORBA. SDP's requirements for integration with existing system need security training with RMI and train their developers on tools such as RMI over IIOP for integration with legacy systems, which comparatively consumes less resource than CORBA. This is because of the reason that CORBA needs management training, extensive developers training, and other training required for installation. Hence, based on the above arguments DCOM or RMI would suit for SDP's tracking system as far as training is considered.

5.5. Resources

SDP is a C++ IT shop based on client/server architecture. This project had eight developers. Four are C++ developers and four are Java developers. Two out of the four Java developers had no previous Java experience. SDP was in the middle of a huge project, migrating old applications written in C to C++ using object-oriented techniques. The RTM project in SDP could not use the developers with more experience because of other more demanding projects. There is no room for outsourcing with SDP, as it is an organizational policy not to outsource IT development work.

Let us apply the framework and evaluation summary on resource criteria from table 5 to choose DOMs that meet SDP's requirements and goals. Considering the overall resource evaluation for CORBA, the resources available for CORBA are plenty and matured, and it satisfies SDP's goal of support for multiplatform. The major disadvantage for SDP using CORBA is that it needs sophisticated skills to develop distributed object applications using CORBA. SDP's goal is to use the existing developers, and most of

them do not have the level of sophistication required by CORBA. DCOM would be a better choice compared with CORBA because it does not require as skilled a resource as CORBA does. Also, the DCOM resources are easy to find and cost less.

For further comparisons between RMI and CORBA, Curtis [1999] has given a simple "mantra" for selecting CORBA vs. RMI:

> "Use RMI when you are developing new Java distributed application; use CORBA when you need to access existing non-Java applications."

Since RTM is a new web-based system, RMI gets an edge over CORBA. Also, learning Java would be easy for SDP's C++ developers. Hence DCOM or RMI would be suitable for SDP's RTM system.

5.6. Enterprise changes

The Enterprise changes criterion has a high impact on SDP. Every company goes through reorganization. After reorganization, departments may assume responsibility for new systems that are different in technology, implementation language, and architecture. The SDP IT shop was created to migrate a legacy system from the procedural world into object-oriented client/server, to web-enable most of its systems for its internal customers, and to integrate with other legacy and new systems in other departments inside the company. It has to keep up with mergers, acquisitions, and partnership changes. It already does some business in countries other than the United States. Legacy integration, interoperability, reuse, scalability, and expandability are the requirements of SDP. The systems to be integrated vary from desktop applications to mainframe and web-enabled or distributed applications involving a wide range of languages, tools, and technologies. CORBA seems to be a leader in the integration of large-scale applications in a heterogeneous environment. For high-volume, heavily loaded server environment CORBA would be the best choice because it is predictable and reliable. CORBA has proven its support in large-scale, heterogeneous, distributed environments [Neff 1998]. It would be the best fit for the integration of complex systems and legacy systems. DCOM is quite efficient for distributed applications in Windows platforms. DCOM may not be a good choice for complicated integration of legacy systems in multiple platforms [Bollinger *et al.* 1998]. RMI is not a good choice for the integration of complex legacy systems [Curtis 1999]. Hence CORBA would best suit SDP's enterprise change needs.

5.7. Time constraints

Another major factor impacting SDP is time constraints. To be competitive in the market, to maintain SDP company's ranking among other telecommunication companies, and to increase its revenue, SDP is pressured to deliver its IT products to its internal customers on time. Any slippage in deliverable time would bring a noticeable loss in revenue for the company or incompliance with FCC mandated deadlines. The company was loosing millions of dollars because of charges settlements with other telecommunication companies because the current system is manual. From the time constraints

evaluation summary table, CORBA meets SDP's requirements for reuse, integration, interoperability, and multiplatform. We have to keep in mind that time constraints depend on the particular CORBA product too. CORBA may not be a good choice for SDP, because the learning curve required would impact its time-constraint needs. DCOM development tools exist in a user-friendly environment, which aids the development work and saves time. DCOM would be a good choice that meets SDP's time constraint requirements. RMI is very good for new application development especially for the proposed RTM system, as it has to be web-enabled. It would be easy for the SDP's C++ developers to learn Java. Hence, DCOM or RMI would be a good selection for transition into distributed computing.

5.8. Summary

The conclusion for the selection of DOM for SDP is based on the above discussions and summarized below. This conclusion is purely a case-by-case decision and the organization goals of SDP were considered in making this decision.

From the preceding table, CORBA meets 1 out of 5 framework criteria requirements of SDP while RMI and DCOM score 3 and 4, respectively. As far as SDP's organizational goals are concerned CORBA meets the requirement only for the enterprise changes criterion of the framework. This eliminates CORBA from the picture for SDP and the choice narrows to DCOM and RMI.

In section 5.1 we mentioned that SDP's RTM web-based system must support different platforms. This is a very vital point to consider in making a final decision on narrowing down to DCOM or RMI. It is undisputable to say that DCOM is the best and most matured model if all of the platforms involved in the company are Microsoft Windows operating systems [Bollinger *et al.* 1998]. However, as mentioned in section 4.1.1 DCOM on other platforms is not well-supported and not mature enough. DCOM on other platforms requires sophisticated interface tools and might affect important issues such as application performance, development time, training on interface tools and cost of interface tools. DCOM's successor COM+ seems to have addressed these problems and provided good solutions, but COM+ is still an evolving technology and not matured. Hence DCOM is not a good choice for organizations operating on different platforms. Because SDP operates on multiple platforms, SDP should eliminate DCOM as the choice on the basis of the above argument, although DCOM might suit 4 out of 5 framework criteria requirements of SDP. This leaves only the choice of RMI.

From table 8, we see that RMI meets SDP's requirements for the training, resources, and time constraints factors, while it lags for the Cost and Enterprise Changes factors. On the basis of the cost evaluation summary, we note that RMI costs more only when compared with DCOM, while it costs less or equivalent for SDP when compared with CORBA. Though SDP has to shell out some extra money for choosing RMI over DCOM, it has to trade off that extra cost, because DCOM does not integrate into SDP's existing systems, which operate on multiple platforms.

Table 8
DOM selection summary for SDP's web-based system.

Framework criteria	Supports SDP's requirement		
	CORBA	DCOM	RMI
Cost	No	Yes	No
Training	No	Yes	Yes
Resources	No	Yes	Yes
Enterprise changes	Yes	No	No
Time constraints	No	Yes	Yes

Now the last factor to consider is enterprise changes. From the summary table, it is clear that CORBA is best suited for the integration of large legacy systems and it has the upper hand in tackling problems related to enterprise changes. Also, CORBA has proven its support in large-scale, heterogeneous, distributed environments [Neff 1998]. However, CORBA fails to meet SDP's requirement for 4 out of 5 categories. Hence SDP needs to compromise on using RMI over CORBA in addressing Enterprise Changes.

To emphasize it time and again, for further comparisons between RMI and CORBA, Curtis [1999] has given a simple "mantra" for selecting RMI over CORBA:

> "Use RMI when you are developing new Java distributed application; use CORBA when you need to access existing non-Java applications."

Because the RTM system is web-enabled and most of the related applications will be developed new in Java, it would be a wise decision to choose RMI over CORBA. Also with RMI, SDP can choose to use Java for any new program that has to be written for the proposed system and it should consider JNI and RMI over IIOP for integration of existing non-Java systems. Considering all of these arguments, Curtis's recommendation, the summary in table 8, and most of all SDP's long-term goals, it is obvious that RMI would be the best choice for SDP.

6. Conclusions

This paper presented the framework based on managerial aspects, which can be used by organization willing to transition to DOC. DOM's CORBA, DCOM and RMI are evaluated using the framework suggested in section 3. Also, a web-based case study has been presented in applying the framework for evaluation and selection of DOM(s) for an organization with its goals and requirement. This section presents the conclusion with observations and further areas for research.

CORBA is oldest, matured and most widely used DOM of all [Linthicum 1998]. According to the report of Harman [1997] CORBA is the preferred DOM and 61% of the companies using CORBA were happy with CORBA. DCOM was used by 50% out of 200 companies surveyed. Also he has stated that, "... larger companies are more likely to have adopted CORBA ..." as his survey data revealed that large companies are generally ahead of smaller companies in adopting object technology. It is undeniable that

DCOM is the leading model for desktop and distributed application in Microsoft's platform [Tallman and Kain 1999]. For non-Microsoft platforms, RMI seems to be the leader in building new distributed object applications considering time constraints. CORBA is the leader in the integration of legacy systems and large-scale, robust applications in heterogeneous environment [Neff 1998]. Orfali, Harkey, and Edwards [1997] have said that CORBA with Java is the best fit for object-oriented web projects. To repeat it here again, the simple "mantra" for selecting CORBA vs. RMI given by Curtis [1999] is:

> "Use RMI when you are developing new Java distributed application; use CORBA when you need to access existing non-Java applications."

It is very difficult to say which models would be best for an organization even though we have arrived at a conclusion favoring RMI for the RTM System of SDP. This is because different models have their own advantages and disadvantages. So to achieve a perfect balance, an organization might have to consider tools that would integrate two or more of these models [Bollinger *et al.* 1998; Norman 1998]. Examples for such integrated tools are listed below.

- *WebLogic*. It is a product of BEA Systems. It is for Java language and supports RMI seamlessly. Also, supports DCOM and CORBA ORBs. The capabilities of WebLogic Enterprise is documented by BEA Systems, Inc. as follows [BEA Systems 2000]:

  > "With BEA WebLogic Enterprise, you can build, deploy, and manage component-based solutions for your enterprise. WebLogic Enterprise brings together the Object Request Broker and online transaction processing functions with industry programming models such as Enterprise JavaBeans, CORBA, and ATMI. The result is a platform that enables you to deliver scalable, secure, and transactional e-commerce applications in a well-managed environment."

- *RMI over IIOP*. Developed by Sun and IBM. RMI over IIOP delivers CORBA compliant distributed computing capabilities to the Java 2 platform and SDK [Sun 200].

Using a mix of all these tools and models will not be the ideal solution always. Selection of DOM is case sensitive. In a nutshell organizations have to follow a case-by-case approach in selecting DOMs and tools that would integrate its managerial goals and project goals. The guideline for making such a selection is made easy by the framework presented in this paper.

The area of DOC is still evolving. There is more room for future and further development on the topics related to DOC. It would be necessary and more appropriate to evaluate CORBA 3, COM+/.NET, and EJB/J2EE. The remaining of this section discusses the areas for further research.

Design by contract support

One of the major advantages of DOMs is their support for reuse [Neff 1998]. For effective reuse the reuse tools and technologies should support design by contract. The importance of design by contract for reuse is given by [Jézéquel *et al.* 1997] as below:

> "Effective reuse requires design by contract. Without a precise specification attached to each reusable component precondition, post condition, invariant no one can trust a supposedly reusable component. Without a specification, it is probably safer to redo than to reuse."

There are lessons learned from the past on how and why reuse projects failed without the support for design by contract. An example would be the European Ariane 5 launcher crash event on June 4, 1996. The European Ariane 5 launcher crashed about 40 seconds after its take off causing half-billion dollars loss, which was caused mainly by software reuse error. For further details refer to Jézéquel et al. [1997].

These lessons learned do not seem to be incorporated into the DOMs. The IDL specification provided by the DOMs define only types, names and signature of operations and attributes. It does not provide any semantic specification mechanism [Jézéquel *et al.* 1997; Cicalese and Rotenstreich 1998]. It would be interesting to see the support for design by contract in DOC arena and how DOMs facilitate design by contract.

Real-time and embedded distributed object computing

DOM's support for embedded systems would be another area to continue the research. Many organizations have been pressured by the high demand for the support for embedded distributed systems. Some of the areas where embedded systems are on demand include wireless, mobile computing, consumer products, etc. DOMs support for real-time and embedded distributed systems would be the next focal point. It would be interesting to see the DOC tools and technology support in this area.

References

Agha, G.A., M. Astley, J.A. Sheikh, and C. Varela (1998), "Modular Heterogeneous System Development: A Critical Analysis of Java," In *Proceedings of the Seventh Heterogeneous Computing Workshop*, Orlando, Florida.

Bacon, J. (2000), "Expectations and challenges in large-scale distributed systems," *IEEE Concurrency 8*, 1, 2–3.

BEA Systems, Inc. (2000), "Documentation for the E-Generation," available at `http://e-docs.bea.com`.

Bollinger, T., D. Diskin, and S. Chubin (1998), "Recommendations for Using DCE, DCOM, and CORBA Middleware," MITRE Document MITRE-DAS-C1, Release 1.6, MITRE Corporation.

Bradley, M. (1997), "IIOP: OMG's Internet Inter-ORB Protocol: A Brief Description," Object Management Group, `http://www.omg.org/library/whitepapers.html`.

Burghart, T., "Distributed Computing Overview," Quoin Incorporation, available at `http://www.quoininc.com/quoininc/rtivcles.html`.

Cicalese, C.D.T. and S. Rotenstreich (1998), "Behavioral Specification of Distributed Software Component Interfaces," *IEEE Computer 31*, 3, 126–128.

Curtis, D. (1997), Java, RMI and CORBA, "Object Management Group," available at `http://www.omg.org/library/wpjava.html`.

Curtis, D. (1999), "RMI, IIOP, and EJB," *Distributed Computing 2*, 6, 18–20, 32.

Devarakonda, M. (1998), "The Practical Aspects of Object-Oriented Programming," *IEEE Concurrency 6*, 3, 30–33.

Dick, K. (1999), "Introduction to Enterprise JavaBeans," *Distributed Computing 2*, 1, 26–38.
Emmerich, W. (1997), "An Introduction to OMG/CORBA," In *Proceedings of the 19th International Conference on Software Engineering*, pp. 641–642.
Fritzinger, J.S. and M. Mueller (1996), "Java™ Security," Sun Microsystems, available at `http://java.sun.com/security/whitepaper.txt`.
Gray, D.N., J. Hotchkiss, S. LaForge, A. Shalit, and T. Weinberg (1998), "Modern Languages and Microsoft's Component Object Model: Programming COM Made Simple," *Communications of the ACM 41*, 5, 55–65.
Griffin, W.G. (1995), "Lessons Learned in Software Reuse," *IEEE Software 12*, 4, 11.
Guttman, M. and J. Matthews (1998), "Transitioning to Enterprise Components," *Object Magazine 8*, 4, 38–69.
Guttman, M. and R. Appelbaum (1998), "The Next Generation of CORBA," *Component Strategies 1*, 2, 40–51.
Harman, P. (1997), "The Corporate Use of Object Technology, RP13U," Cutter Information Corporation, available at `http://www.cutter.com/itgroup/reports/corpuse.htm`.
Haughey, W.P. (1999), "Strategic Approach to Value Chain Integration," Object Management Group, available at `http://www.omg.org/omg/strategy.html`.
Horstmann, M. and M. Kirtland (1997), "DCOM Architecture, Microsoft Corporation," available at `http://msdn.microsoft.com/library/backgrnd/html/msdn_dcomarch.html`.
IONO Technologies (2000), "IONO Orbix 2000 Product Information," available at `http://www.orbix.com`.
Jézéquel, J.M. and B. Meyer (1997), "Design by Contract: The Lessons of Ariane," *IEEE Computer 30*, 1, 129–130.
Kirtland, M. (1997), "The COM+ Programming Model Makes it Easy to Write Components in Any Language," Microsoft Corporation, available at `http://www.microsoft.com/msj/1297/complus2/complus2.htm`.
Kochikar, V.P. (1998), "The Object-Powered Web," *IEEE Software 15*, 3, 57–62.
Leppinen, M., P. Pulkkinen, and A. Rautiainen (1997), "Java and CORBA-Based Network Management," *IEEE Computer 30*, 6, 83–87.
Linthicum, D.S. (1998), "Multitiered Application Integration," *Component Strategies 1*, 6, 52–58.
Maffeis, S. (1997), "Piranha: A CORBA Tool for High Availability," *IEEE Computer 30*, 4, 59–66.
McArthur, K.M. (1999), "Component-Based Architectures and their Impacts on Reuse," Masters Thesis, University of Nebraska at Omaha.
Microsoft Corporation (1996), "DCOM Technical Overview," available at `http://msdn.microsoft.com/library/backgrnd/html/msdn_dcomtec.html`.
Microsoft Corporation (1999), "MTS," available at `http://www.microsoft.com/com/tech/mts.asp`.
Microsoft Corporation (2000), "Product and Technology Catalog," available at `http://shop.microsoft.com/Products`.
Mowbray T.J. and W.A. Ruh (1997), "*Inside CORBA*," Addison-Wesley, Reading, MA.
Murphy, M.F. (1995), "Enterprise Integration Extends to People," *IEEE Software 12*, 1, 16.
Nakamura, H. (1998), "Enterprise Middleware," *Distributed Computing 1*, 6, 38–41.
Neff, K. (1998), "DCOM and CORBA: Why Are We Still Fighting?," *Distributed Computing 1*, 5, 29–34.
Nester, C., M. Philippsen, and B. Haumacher (1999), "A More Efficient RMI for Java," In *Proceedings of the ACM 1999 Conference on Java Grande*," 152–159.
Norman, R.J. (1998), CORBA and DCOM: Side by Side," *Distributed Computing 1*, 5, 41–45.
Oberndorf, P., L. Brownsword, E. Morris, and C. Sledge (1997), CMU/SEI-97-SR-019, Software Engineering Institute, pp. 3–4.
OMG (1997), "A Discussion of the Object Management Architecture, OMA: formal/00-06-41," Object Management Group, available at `http://sisyphus.omg.org/technology/documents/formal/object_management_architecture.html`.

OMG (1999), "CORBA 2.3 Overview, formal/99-07-06," Object Management Group, chapter 2, available at ftp://ftp.omg.org/pub/docs/formal/99-07-06.pdf.

OMG (2000), "CORBA Academy," Object Management Group, available at http://www.omg.org/gettingstarted/training.htm.

Orfali, R., D. Harkey, and J. Edwards (1997), "CORBA, JAVA, and the Object Web," *Byte*.

Pattison, T. (1998), "*Programming Distributed Applications with COM and Microsoft Visual Basic 6.0*," Microsoft Press.

Putman, L.H. and W. Myers (1997), "How Solved Is the Cost Estimation Problem?," *IEEE Software 14*, 6 105–107.

Quoin Inc., (1998), "COM versus CORBA: A Decision Framework," version 1.3.

Roy, M. and A. Ewald (1996), "Choosing between CORBA and DCOM," *Object Magazine 6*, 8, 24–30.

Scallan, T., "A CORBA Primer," Segue Software Incorporation, available at http://www.omg.org/library/whitepapers.html.

Schneidewind, N.F. (1998), "How to Evaluate Legacy System Maintenance," *IEEE Software 15*, 4, 34–42.

Schneidewind, N.F. and C. Ebert (1998), "Preserve or Redesign Legacy Systems?," *IEEE Software 15*, 4, 14–17.

Semaphore S., "Integrating, Distributed Applications via CORBA," Object Management Group, available at http://www.omg.org/library/whitepapers.html.

Siegel, J. (1996), "CORBA: Getting to the Fundamentals," *Object Magazine 6*, 8, 52–55.

Siegel, J. (1998), "OMG Overview: CORBA and the OMA in Enterprise Computing," *Communications of the ACM 41*, 10, 37–43.

Siegel, J. (1999), "A Preview of CORBA 3," *IEEE Computer 32*, 5, 114–116.

Siegel, J., "What's Coming In CORBA 3," Object Management Group, available at http://sisphus.omg.org/technology/corba/corba3releaseinfo.html.

Sneed, H.M. (1995), "Planning the Reengineering of Legacy Systems," *IEEE Software 12*, 1, 24–34.

Stewart, R., D. Rai, and S. Dalal (1999), "Building Large-Scale CORBA-Based Systems," *Component Strategies 1*, 7, 34–59.

Sun Microsystems, Incorporation (1998), "JavaTM Remote Method Invocation Specification, Revision 1.50, JDK 1.2," available at http://java.sun.com/products/jdk/1.2/docs/guide/rmi/spec/rmi-title.doc.html.

Sun Microsystems, Incorporation (1999), "Java Remote Method Invocation – Distributed Computing for Java," available at http://java.sun.com/marketing/collateral/javarmi.html.

Sun Microsystems, Incorporation (2000), "RMI over IIOP," available at: http://java.sun.com/products/rmi-iiop.

Sun Microsystems, Incorporation (2000), "Java 2 Platform, Enterprise Edition," available at http://java.sun.com/j2ee.

Tallman, O. and B. Kain (1998), "COM vs. CORBA: A Decision Framework," *Distributed Computing 1*, 12, 41–43.

Tallman, O. and Kain, B. (1999), "COM vs. CORBA: A Decision Framework," *Distributed Computing 2*, 1, 46–50.

Vonoski, S. (1998), "New Features for CORBA 3.0," *Communications of the ACM 41*, 10, 44–52.

Waldo, J. (1998), "Remote Procedure Calls and Java Remote Method Invocation," *IEEE Concurrency 6*, 3, 5–7.

Watson, A., R. Soley, and M. Bradley (1997), "Comparing ActiveX and CORBA/IIOP," Object Management Group, available at http://www.omg.org/library/activex.html.

Wolf, H., F. Sauer, and K. Luc (1998), "How Java and CORBA Delivered," *Component Strategies 1*, 5, 55–60.

Wollrath, A., J. Waldo, and R. Riggs (1997), "Java-Centric Distributed Computing," *IEEE Micro 17*, 3, 44–53.

Yee, A. (1999), "Making Sense of the COM vs. CORBA Debate," *NetworkMagazine*, available at http://www.performancecomputing.com/features/9906dev.shtml.