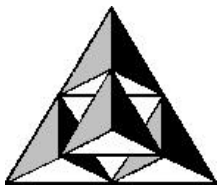
CORBA vs. DCOM:

Solutions for the Enterprise



META Group Consulting

March 20, 1998

Introduction

At the enterprise level, expectations for distributed computing technologies are very high and often outstrip the capabilities of available vendor offerings. IT professionals at many levels are asking, "Which middleware best fits the short and long-term requirements of the enterprise?" Without a clear mapping of enterprise needs to the strengths of various technologies, no clear answer exists. Moreover, vendor marketing and differentiation efforts often serve only to confuse the situation further.

The enterprise middleware battle is raging. Businesses no longer can resist the promise of middleware in providing a nimble framework upon which business applications can be integrated, cost-effectively built, and flexibly extended. For the enterprise, the attractiveness of middleware lies in its tremendous flexibility potential, which gives middleware the ability to accommodate numerous design paradigms, implementation strategies, and exploit maximum levels of reuse. As the glue that connects business applications together, middleware is now seen as the essential ingredient for component development and deployment in the distributed enterprise environment. The following table indicates typical relationships between business change, IT needs, and the inherent features of middleware technology.

BUSINESS IMPERATIVES	IT REQUIREMENTS	MIDDLEWARE STRENGTHS
Increase customer intimacy by providing unified views	Integrate customers, departments, vendors, and suppliers	Business application integration
Scale business costs closely with market changes	Grow IT capacity incrementally	Component based, N-tier environment
Increase value by lowering business costs	Preserve and leverage existing IT investments	Heterogeneous IT asset coexistence support
Deliver services faster to changing market	Reduce application development time	Shared services infrastructure

Unfortunately, the war has raged for so long that it is often difficult to distinguish the important features from the marketing hype. Business leaders do not particularly care about the underlying details of distributed object technologies. The realities of product differentiation will require organizations to carefully match the strengths of middleware offerings with pressing business needs.

The Common Object Request Broker Architecture (CORBA) and the Distributed Component Object Model (DCOM) represent a new class of software known as middleware. The purpose of this white paper is to highlight the current needs of the enterprise and compare CORBA's and DCOM's capabilities in supporting these needs. The paper will also explore synergies between emergent middleware technologies such as CORBA/Java and DCOM/Java. The paper will take the following steps in "demystifying" the fundamental differences between CORBA and DCOM:

- Lay the foundation of enterprise demands upon distributed computing technologies;
- Describe the key elements of CORBA and DCOM that support these demands; and
- Objectively assess the strengths and weaknesses of these elements for enterprise deployment.

META Group will evaluate these standards and their associated implementations across a range of environments, with specific emphasis on "enterprise" needs. This paper compiles what META Group believes to be a realistic set of "enterprise-critical" measurements and then differentiates between the leading middleware paradigms and products (CORBA and DCOM) in assessing how they support the current needs of enterprise computing.

Enterprise Computing Trends

Infrastructure Emphasis

Business priorities currently are dominated by the need to deploy new systems faster. Consequently, business decision-makers are emphasizing "application purchase" over "application build." However, purchased applications can offer only competitive parity, never differentiating advantage. IT-based competitive tactics generally take three forms:

1. Speed deployment of purchased applications (thereby accruing returns prior to competitive implementations);
2. Lower costs and risks of implementing internally developed applications (thereby generating more certain, sustainable differentiated application systems); and
3. Cut distributed computing costs (thereby enabling the business to allocate capital to non-IT uses).

In a world of distributed computing, these tactics share one objective: *infrastructure optimization*. Indeed, META Group believes the only certain way for IT to generate sustainable strategic advantage for business is through focused infrastructure development in support of strategic application objectives.

The traditional approach to IT infrastructure; the common IT hardware, software, and support has been as a pure cost center — i.e., a tactical response to support and integrate the current business application portfolio together. Infrastructure was purchased or developed in response to specific application requirements. However, when the business environment changed, followed closely by changes in applications, the infrastructure remained constant. Since the infrastructure was tightly coupled with legacy applications, the infrastructure itself restricted business-driven change.

In the current IT environment, it is virtually impossible to piece these evolving applications together, because these brittle infrastructures are highly intermingled with the applications. To gain competitive advantage, the infrastructure must be viewed as a strategic asset that no longer simply accommodates, but anticipates key changes in the business. Such a preemptive approach implies more than a tactical response. It involves a streamlined process that promotes business growth instead of prohibiting it, and enables the enterprise to:

- Achieve maximum leverage across multiple distributed applications; and
- Establish long-term flexibility in response to imminent changes in the business.

Given these needs, IT must develop a phased strategy for delivering both near-term application services and long-term strategic common technology services. This strategy is achieved by maximizing the ratio of common IT assets and services available to a single application. It should be understood that common services are not restricted to technical infrastructure (e.g., messaging, TCP/IP transport services), but also include business services (e.g., workflow, calendar/scheduling, and sales tracking). Put another way, *infrastructure* implies technical infrastructure *plus* common services. Middleware provides the basis for common services.

Legacy Integration

The ability to integrate the enterprise's legacy IT assets is perhaps the most overriding consideration for middleware adoption. Millions of lines of code and megabytes of data represent the business intelligence that drives the present and future survival of the enterprise. For the majority of organizations that are considering middleware, wholesale replacement of all IT assets at once is simply not possible. The need to maximize previous investments by extending the lifetime of legacy IT assets is critical. Controlling the pace of transition in a way that minimizes impact on the operational environment is also a paramount consideration.

CORBA and, to a lesser extent, DCOM are middleware standards. To test each of the critical requirements described above, we must compare more than just standards. For many of these requirements, comparing standards is meaningless. For example, only "live" implementations of these standards can generate meaningful performance and scalability data. On the other hand, standards can present tangible and comparable value to the enterprise, specifically in the area of interoperability.

Enterprises face real risks in providing interoperability between their IT resources. Enforceable standards provide a "contract of interoperability" between these resources, and help prevent the need for wholesale replacement following each small change in a vendor product. Cross-platform and cross-language support are both key enablers in achieving an incremental transition to middleware in a legacy-rich environment.

Enterprise Criteria

The benefits of middleware are alluring, but few organizations can afford to adopt a technology that is unproven or incapable of supporting a minimum set of criteria. Accordingly, the enterprise battle has many fronts. To truly support the needs of enterprise computing, middleware must satisfy certain essential elements. Indeed, enterprise-grade middleware should be grounded in same functionality that previous successions of systems have demanded:

- *Performance and reliability* form the baseline from which the enterprise must evaluate middleware technologies;
- *Interoperability* is on equal ground for enterprises that represent the reality of a mixed-system environment; and
- *Market viability* stands as the final test in measuring the maturity and longevity of vendor support during the lifetime of the middleware.

The following table details specific measurement criteria for each of the above middleware elements.

Interoperability	
	Cross-Language Support Cross-Platform Support Network Communications Common Services
Reliability	
	Transactions Messaging Security Directory Fault Tolerance
Performance	
	Scalability
Viability	
	Product Maturity Vendor Outlook

Historical Perspective

It is important to understand how these middleware technologies have emerged. The original objectives of CORBA and DCOM provide critical insight into their current strengths and weaknesses, and sets the stage for their future directions.

A Brief History of CORBA

The Common Object Request Broker Architecture (CORBA) was established in 1991 by the Object Management Group, a consortium dedicated to removing the intricacies of the network from individual components and applications. The members of the OMG recognized early that, in order for objects to be useful at an enterprise level, several obstacles had to be overcome.

First, the differences between programming languages, operating systems, platforms, and networks were likely to continue for the foreseeable future. Individual projects within organizations were expected to continue benefiting from the freedom of selecting tools that best suited their needs. The differences between projects, however, did not end there. The IT portfolio continued to diversify with custom and packaged mainframe and client/server applications. The growing need to integrate these disparate systems revealed many shortcomings. A comprehensive architecture did not exist to mask the differences across projects so those software assets in each could be shared and reused.

Second, the benefits of object-orientation (OO) could never be realized at an enterprise level without overcoming this complexity. Key principles of OO such as encapsulation (hiding) had to be applied to this "middle" area to reduce the complexity barriers that resulted from the lack of agreement between vendors of infrastructure hardware (platforms and networks) and application software (languages and components).

The OMG released CORBA to address these issues; since then, many important contributions have emerged from the OMG in support of component interoperability.

- OMG Interface Definition Language (IDL) — Introduced in 1991, IDL is CORBA's primary tool to insulate language differences. It has been heralded by some as the universal standard for software interfaces and has been fully adopted by the ISO (International Standards Organization DIS 14750).

- CORBA Interoperability – The CORBA 2.0 specification in 1996 defined the General Inter-ORB Protocol (GIOP) as the basis for ORB-to-ORB communication. At the 1995 Object World Show in San Francisco, the OMG unveiled ORB-to-ORB interoperability via the Internet in a showcase demonstration titled CORBAnet. The demonstration, involving seven ORBs and 12 vendors, is currently accessible to the public via the Internet.

- GIOP — This protocol requires the use of a TCP/IP based Internet Inter-ORB Protocol (IIOP) as the underlying common communications mechanism that all CORBA 2.0-compliant ORBs must support. It also permits the use of other protocols such as DCE.

The OMG continues to expand its presence horizontally and vertically through the development of CORBAservices, CORBAdomains, and CORBAfacilities. The current Object Management Architecture (OMA) defines 15 fundamental CORBAservices in four high-level (CORBAfacilities) categories: Information Management (e.g., Query, Persistence), Task Management (e.g., Transactions, Concurrency), System Management (e.g., Lifecycle, Trader), and Infrastructure Services & Elements (e.g., Security, Messaging).

At another level, CORBAdomains are more vertically defined and deal with such areas as the Internet and various industries (e.g., financial, medical, manufacturing, and telecommunications). These domains are further driven into sub-domains such as banking and insurance within the larger financial domain. The entire architecture forms a comprehensive model for distributed business object applications and infrastructure.

A Brief History of DCOM

The Distributed Component Object Model (DCOM) has evolved from a number of fielded technologies, starting in 1990 with Object Linking and Embedding (OLE). OLE started its life as a simple desktop cut-and-paste mechanism using a technology called Dynamic Data Exchange (DDE). OLE was later extended to OLE2 with the introduction of Microsoft's Component Object Model (COM) to provide inter-application communication and document embedding across Microsoft applications. OLE2 was also extended to support "drag and drop" as well as scripting capabilities to enable one application to perform simple work in another (OLE Automation).

Around the same time, Microsoft's Visual Basic product was gaining credibility as general-purpose model for assembling visual components (Visual Basic custom controls — VBXs). The limitations of VBXs became apparent. VBX architecture was not "open" and Microsoft did not disclose a standard for the VBX controls. Without this standard, vendors found it difficult to incorporate VBXs in their offerings. This motivated Microsoft to provide a more generalized infrastructure for inter-application communication and control.

This framework was defined by the combination of COM and OLE Controls (OCXs). OCXs became COM's first application-level implementation for supporting multiple applications. In other words, COM was the system-level standard and OLE was the application service built upon the COM standard (i.e., OLE is to COM as a specific ORB service is to CORBA). This clarification is not entirely true however, because COM is also the actual communications technology used to connect components.

Distributed COM (DCOM) emerged to address COM's shortcomings in supporting remote components. Microsoft has described DCOM as "COM with a longer wire." As Roger Sessions most appropriately states, "The demarcation between the two (COM and DCOM) is much more a matter of historical accident than technical foresight." Microsoft continues to differentiate COM (most recently COM+), OLE Automation (now simply Automation), and DCOM as separate entities.

ActiveX became the successor to OCXs and was described as; "COM enabled for the Internet." For some time, it was used as an umbrella term covering many of the underlying technologies such as DCOM and Automation. ActiveX also describes an Automation object, be it visual or non-visual.

Finally, enter the Microsoft predator products: Viper, Falcon, and Wolfpack. These products were aimed at extending Microsoft NT from a simple file and print server to a more robust application server. Viper, now Microsoft Transaction Server (MTS), has emerged as a general server environment for ActiveX components. Falcon, or the Microsoft Message Queue Server (MSMQ), is Microsoft's answer to reliable messaging between applications. Finally, Wolfpack is Microsoft's code name for an NT-based clustering technology aimed at improved availability, manageability, and scalability. Combined, these services — COM, DCOM, OLE, and ActiveX — form Microsoft's newly coined Distributed interNet Architecture (DNA).

CORBA and DCOM: A Feature Comparison

In the following sections, we define each "enterprise" quality and compare the levels of support currently available in CORBA and DCOM specifications and products. Although this list is not comprehensive, it stands as a reasonable baseline for middleware comparison. These features are not necessarily listed in order of priority. Instead, each is treated independently, though many are highly interdependent. Finally, individual ratings are given at the end of each section to indicate the relative levels of enterprise readiness. A "+" implies full readiness, "0" connotes marginal status, and "-" indicates a failure to meet the overall needs of the enterprise.

Interoperability

Cross-Language Support

Cross-language support is one part of the critical interoperability capabilities required of enterprise systems. While languages such as C++, Visual Basic, and Java are on the rise, COBOL is still often cited as the most widely used programming language, with an estimated three million active programmers.

CORBA

CORBA was designed from the ground up to be language and platform independent through the use of a common Interface Definition Language (IDL). Now an ISO standard, OMG IDL provides a common notation for describing cross-platform, cross-language application program interfaces (APIs). IDL is used to define the "interface" of the component, not the inner workings. For this, other standard programming languages are used. IDL interfaces are translated to standard languages through mappings. Currently, IDL has been mapped to C, C++, Smalltalk, Ada, OLE (Visual Basic, PowerBuilder, Delphi, etc.), Java, and soon to Eiffel and Objective C.

The benefits of interoperability are not without costs however. IDL was never meant to substitute for a general-purpose language. Instead, it was designed to express generalized interfaces. IDL limits the language data types to a least common denominator that can be supported by all languages. Although some of the language-specific data types are not directly usable, IDL does permit an "any" type to overcome this obstacle.

DCOM

DCOM's language portability (heterogeneity) is based upon a "binary standard." Binary compatibility is accomplished at the ones-and-zeros level, an area that has previously been the domain of computer language compilers and interpreters. To guarantee compatibility at this level, the way each language is translated to machine binary code must be controlled. This can present a few obstacles, but also has its benefits. First, there are fundamental differences in how languages are translated. Since some languages are compiled and others are interpreted, "binary compatibility" requires that components support all possible translation variations. Second, there are many compilers/interpreters for a given language, each taking unique approaches to code translation. Finally, specifying compatibility at such a low hardware level creates vulnerabilities due to advances in hardware itself.

Microsoft has been successful in controlling the mainstream development tools in the desktop arena for DCOM's predecessor, COM. COM is currently supported by the popular array of Microsoft products as well as Java, PowerBuilder, Delphi, and Micro Focus COBOL. Distributed COM, however, requires additional support from Microsoft or a third party that ports DCOM (see Software AG below).

Summary: Both CORBA and DCOM provide extensive support for multiple programming languages, though they use different techniques.

Enterprise Criterion	Ratings	
Interoperability	CORBA	DCOM
Cross-Language Support	+	+

Cross-Platform Support

The "middle" in middleware often refers to the synergistic connection between disparate enterprise IT resources. Until it is feasible for all enterprise resources to be hosted entirely on homogenous hardware platforms, middleware must support new and legacy platforms and the freedom to mix them as required.

CORBA

Cross-platform support has always been a central focus of CORBA. ORBs currently exist for more than 30 platforms and supports even more Microsoft operating systems than DCOM. Orbix, one of the leading ORB products, supports 20 platforms itself.

DCOM

DCOM has approached cross-platform support as an afterthought. In 1993, Microsoft approached a German company, Software AG, to port DCOM to other platforms. Software AG has ported DCOM to several Unix variants; however, the port does not include many of the components of DCOM. For example, many critical supporting technologies have not been ported, the most important of which are MTS and MSMQ. Without MTS and MSMQ, DCOM is simply not a viable enterprise middleware. DCOM has also been ported to Macintoshes and DEC Alphas that run Windows NT. Many other ports are currently in the works (Open VMS, Digital Unix, HP-UX, Solaris, IBM OS/390, IBM AIX, and Linux).

Summary: It should be clear by now that in order to cast either of these technologies into the enterprise role, a comprehensive collection of critical infrastructure services must be considered for each of the required platforms. For DCOM, this means exploiting the combined synergies of COM, MTS, MSMQ, and clustering them together to fulfill the needs of enterprise computing. Without MTS, for example, DCOM will be unable to fulfill these needs. By way of comparison, CORBA-based products typically provide each of their services on all supported platforms. As such, ORBs are much further ahead in their support for heterogeneous enterprise environments.

Enterprise Criterion	Ratings	
Interoperability	CORBA	DCOM
Cross-Platform Support	+	-

Network Communications

Robust support for enterprise network communications requires that middleware seamlessly provide interoperability with many disparate networked systems. To enable this, the middleware should be "protocol transparent."

CORBA

The predominant CORBA networking model for cross-ORB communication is based on a form of TCP known as IIOP (Internet Inter-ORB Protocol), a connection-based protocol. IIOP was specifically designed to ensure that all ORBs use a common communications protocol. Internal to a particular ORB, however, other protocols are possible. ORB products, similar to DCOM, are usually remote procedure call (RPC)-based.

DCOM

Initially, DCOM utilized UDP/DCOM, a connectionless protocol that is based on the OSF's DCE RPC specification with some changes. DCOM now offers a TCP protocol configuration as an option, although by using this, many efficiency features are sacrificed.

Summary: CORBA has established the lead in common network protocol support through the de facto IIOP standard. DCOM provides protocol options, but does not support them equally.

Enterprise Criterion	Ratings	
Interoperability	CORBA	DCOM
Network Communications	+	0

Common Services

Common services form the base infrastructure of the middleware. These services are married to the individual patterns of business in an enterprise setting. For example, a banking model is highly transaction oriented and requires secure transaction support as a fundamental middleware service. To this end, most organizations require a number of key services. It should be understood, however, that not all services are equally important to all enterprises. Where more than one service is required, it should be fully compatible and interoperable with the others.

Using the OMG specification terminology, we consider the following services as a minimal set for enterprise middleware: Transactions, Directory, Messaging, Security, and interoperability. The CORBA road map provides ORB vendors with a path for service interoperability. This interoperability is required to integrate the best available third-party services across platforms. Microsoft's approach is less explicit, with service interoperability implied for the NT platform only. CORBA and DCOM products support these basic services in various degrees.

CORBA

The OMG has concentrated on the development and integration of key architectural services. Their technology adoption process is specifically aimed at ensuring that services are implemented in an interoperable manner. The CORBA specification defines 15 services, though not all commercial ORB products support the complete set. One exception to this is IBM's COS (Common Object Services), a suite of the full 15 CORBA services that is compatible with DSOM and other brokers.

DCOM

DCOM services are less defined from an architectural standpoint, though there are many CORBA equivalents. DCOM currently offers a limited naming service, transactions, and security integration with NT. Other services such as MSMQ and clustering are becoming available, but are not formally integrated into the DCOM specification.

Summary: Full-service support is not yet available from DCOM or CORBA products. At present, CORBA products have the lead in the number, maturity, and scope of enterprise-required services that are made available to both new and legacy applications.

Enterprise Criterion	Ratings	
Interoperability	CORBA	DCOM
Common Services	0	-

Reliability

Transactions

Transaction support has been the focus of both middleware technologies in recent years. During 1997, the gaps in both camps were significantly closed.

CORBA

CORBA's Object Transaction Service (OTS) specification offers a range of services for distributed transaction support. These services extend the range of traditional flat transactions to support both flat and nested transactions (since nested transactions break up transactions into hierarchies of sub-transactions, this offers developers the flexibility for failures in a sub-transaction to be retried using an alternative method, while the main transaction can succeed). OTS enables both ORB and non-ORB applications to participate in the same transaction, so that object transactions and procedural transactions (that support X/Open's DTP standard) can interoperate. It also supports transactions across heterogeneous ORBs, so that multiple ORBs can participate in the same transaction. Also, a single IDL interface supports both transactional and non-transactional implementations. To make an object transactional, developers use an interface that inherits from an abstract OTS class. Taken together, the interfaces for OTS, plus the Concurrency and Control service and Transactions, offer full commit, rollback, locking and other capabilities, enabling ORB vendors supporting it to offer distributed transaction capabilities. A number of the ORB implementers have offered links to tools from traditional TP monitors, and OTS enables them to incorporate these capabilities directly into the ORB and distribute them.

The goal of integrating best-of-breed transaction products has been widely realized in the ORB marketplace over the last year. Tuxedo, the most scalable TP monitor for highly distributed environments, has been successfully integrated with two prominent ORBs. In addition, Visigenic and Hitachi have integrated TPBroker and Iona has integrated Transarc in OrbixOTS.

DCOM

Microsoft also has been aggressively attacking transaction support in the form of its Microsoft Transaction Server (MTS). As a fully integrated transaction service, MTS has great potential for at least the Wintel environment, and is positioned by Microsoft as an extension to DCOM.

With MTS, transactions are supported implicitly, thereby freeing the developer from the complexity of dealing with transaction services directly. This enables MTS to preserve the component model. In addition, MTS provides a declarative security model. MTS is in an early state of maturity, however. Few examples are available to assess the relative scalability of MTS, and it has not been offered for the cross-platform environment to date.

Summary: We continue to believe that CORBA will remain the leading-edge middleware transaction model for networked objects, with DCOM MTS transaction support suitable for low-end processing but gaining ground quickly.

Enterprise Criterion	Ratings	
Reliability	CORBA	DCOM
Transactions	+	0

Messaging

Reliable transmission and receipt of messages is a foundational quality of distributed middleware. Without it, the electronic commerce (EC) systems of tomorrow will ultimately fail in delivering expedient and reliable services to the increasingly demanding marketplace. Effective messaging requires four important qualities: reliability, user convenience, system convenience, and performance.

In messaging, reliability means nothing less than guaranteed delivery. To guarantee delivery of anything requires a reliable middleman, not unlike the US Postal Service. Rain or shine, the postal service can be relied upon to deliver mail to its eventual destination. The operational word here is "eventual." If the weather becomes too severe, postal workers do not throw the mail away; they hold onto it until the weather permits delivery. The same quality is required of middleware.

User convenience, system convenience, and performance are highly interrelated qualities. User convenience means that the sending and receiving parties are not forced to be at a particular place and time to send and receive messages. This is known in technical terms as asynchronous communication. With asynchronous communication, a sender or system does not have to wait until the message is sent AND received before being able to do other work. This convenience enables all parties — sender, system, and receiver — to continue performing useful work, regardless of each other's current situation.

To support these needs, distributed middleware requires a robust message queuing system. Message queues support asynchronous transmission by providing a persistent queue (message queue) as a temporary message holding area. Again, CORBA and DCOM approach messaging in different ways, but both technologies are geared toward the same needs outlined above.

CORBA

The early CORBA specifications addressed messaging from a more primitive standpoint. The Event Service was the basis of many messaging protocols such as push-pull and pull-push. ORBs typically provided two avenues for messaging: the Event Service primitives or a proprietary mechanism. The OMG recently addressed a more robust messaging model in the CORBAservices specification. This specification addresses the asynchronous communication option that is required by enterprise-grade applications; however, it has not been adopted yet. Many ORB products have implemented extensions to the CORBA Events service that provide reliable messaging. For example, in early 1996, Orbix announced their OrbixTalk Reliable Multicast Protocol, which provides reliable sequencing and delivery of messages.

Some ORB implementations have integrated an enterprise-grade messaging service on par with standalone Message-Oriented Middleware (MOM). IBM's ComponentBroker is one example of integration with MQSeries, a leading MOM product. Iona has been successful in demonstrating GIOP over MQSeries. BEA has also announced intentions to integrate its newly acquired MessageQ into the Iceberg product.

DCOM

Formally, DCOM does not directly support asynchronous communication. Microsoft's answer to reliable messaging is a separate offering titled Microsoft Message Queue Server (MSMQ) or Falcon. On the plus side, MSMQ promises to support each of the important qualities of reliable messaging and more. Unfortunately, MSMQ is not a fully integrated part of DCOM at this time and has the same interoperability limitations as MTS.

Software AG's EntireX product, a cross-platform port of DCOM, is integrated with the proprietary EntireX Message Broker service. This service does not rely upon MSMG and provides persistent storage of messages to enable asynchronous communication between clients and servers.

Summary: Reliable messaging is now being recognized by both CORBA and DCOM as a critical service for the enterprise. CORBA has been augmented with leading MOM products, but full inter-service integration has not yet been achieved. DCOM has also been augmented with early MOM functionality, but also lacks full integration with other complementary services and is again, not available across a wide range of platforms.

Enterprise Criterion	Ratings	
Reliability	CORBA	DCOM
Messaging	0	-

Security

Clearly, security is one of the key considerations for enterprise computing. Most organizations cringe at the prospect of opening up the mainframe to the Internet. Distributed applications that are exposed to the Web simply cannot tolerate security breaches.

CORBA

The CORBA Security service is one of the most comprehensive security specifications available for distributed computing. The 262-page specification was jointly adopted with the Time Service and covers nearly every conceivable aspect of security, including integrity, accountability, availability, confidentiality, and non-repudiation. It also recognizes that differing levels of security needs exist in an enterprise environment. The service defines three (0-2) levels of security compliance, ranging from non-aware ORB products to those that require the entire range of services (access control, delegation, auditing, authentication, and policy implementation).

ORB products differ widely in their support for security. For example, ICL's DAIS product was the first ORB to offer CORBA security conforming to Kerberos and the GSS API standards. Orbix provides both the SSL-IIOP standard (secure encrypted communications over the Internet) and an implementation of the CORBA Security Level 1 service. Finally, Visigenic has recently partnered with MITRE Corporation spin-off Concept Five to deliver the first ORB complying with CORBA Level 2 security.

DCOM

DCOM utilizes NT mechanisms as the basis of its security support. NT Version 3.5 has been rated level C2 by the National Computer Security Center and ensures a comprehensive array of security controls such as discretionary access, authentication, and auditing. DCOM also provides a CryptoAPI to enable advanced encryption of information. This service requires the support of a Cryptographic Service Provider (CSP) that is provided with NT.

Without question, the combination of NT, MTS, and COM can provide a comprehensively secure environment; however, because DCOM's security managers are NT dependent, this support is limited to Windows/NT platforms.

Summary: CORBA and DCOM are both building comprehensive security mechanisms. To CORBA's credit, the recognition of a wide variety of security services will provide more solutions to the differing needs of the enterprise. For DCOM, the cooperation of the operating system is paramount to providing high levels of security. Although from different directions, both middleware technologies are reaching critical mass in their support for secure distributed computing.

Enterprise Criterion	Ratings	
Reliability	CORBA	DCOM
Security	0	0

Directory Service

An essential feature of any middleware is the ability to keep track of the location of key services in the distributed network space. This lessens the burden of each application (provides location transparency) and, more importantly, provides for load balancing and failover services. Examples of working directory services include; DNS, X.500, Novell NDS, and Microsoft NTDS, though each is accessed by a specialized interface.

CORBA

The OMG has specified the Naming Service for just this purpose. Similar to a "white pages" directory, the Naming Service permits a component to look up a service by name. The Naming Service was designed to allow the use of conventional directory services such as those identified above. These services are wrapped by a higher-level service interface that masks idiosyncrasies from the developer.

VisiBroker offers a CORBA-compliant naming service that is fault tolerant (self-recovering) and persistent (survives shutdowns and abnormal failures), and supports federated name spaces. Orbix also provides a fault-tolerant naming service.

DCOM

Microsoft's answer to this need is called the Active Directory Service (ADS). This service is said to combine the best features of X.500 and DNS. Like the OMG Naming Service, ADS intends to abstract differences between various directory services by providing one standardized interface. ADS Version 1.0 is offered with NT 4.0, with the full ADS capability to be integrated in NT 5.0. The ADS Interface (ADSI) is based on DCOM with specific offerings from directory service providers being implemented as DCOM objects.

Summary: Both CORBA and DCOM are beginning to support sophisticated directory services on par with previous "enterprise tested" incarnations such as NDS and DNS.

Enterprise Criterion	Ratings	
Reliability	CORBA	DCOM
Directory Service	0	0

Fault Tolerance

Middleware's ability to "heal itself" in the event of reasonable failure is essential for most enterprise applications. There are many supporting mechanisms that contribute to this capability. Asynchronous messaging (discussed under Messaging) is one example. Service pools and redundant failover mechanisms also enable graceful recovery and increase the fault tolerance and reliability of middleware. Finally, a reliable directory service is needed to find and connect backup services in the event of failure.

CORBA

The CORBA specification does not directly support fault-tolerance services; however, many ORB vendors have provided this support. For example, Visigenic's VisiBroker provides symmetric failover support to automatically bind to another object server on a separate host in the event of service failure.

Most ORBs provide a simple timeout mechanism for detecting dead or disconnected clients. This approach alone is not sufficient for highly fault-tolerant applications.

DCOM

Basic support for fault tolerance is provided at the protocol level. DCOM utilizes reference counts augmented by "keep alive" messages or pinging as an essential component of the DCOM object life cycle. It requires the successful transmission and receipt of a heartbeat every two minutes between a client and server. If three consecutive heartbeats are missed, the server declares the client dead and decrements the reference count. According to a recent Web FAQ[1] maintained by AT&T Labs (updated Nov. 5, 1997), DCOM does not support configurable times, so clients may not detect problems for a considerable period of time (six minutes). Further, if a distributed component gets into a continuous loop, there is no automated way to detect a problem, because the

[1] COM Reliability FAQ; http://akpublic.research.att.com/~ymwang/resources/COM-R-FAQ.htm

heartbeats will still be sent. Finally, this approach utilizes significant network resources and may not scale well for large numbers of connections. Microsoft has taken positive steps to streamline this approach and has employed piggybacking, grouped pings, and delta pinging to reduce network traffic. Anything beyond this generally requires extensive customization on both the middleware and application side.

Summary: Both DCOM and CORBA do not directly support robust fault tolerance; however, with sophisticated customization, it can be provided. For DCOM, it is not clear whether such customization is possible across a heterogeneous platform environment, because most workarounds currently require the support of NT or Windows 95 components.

Enterprise Criterion	Ratings	
Reliability	CORBA	DCOM
Fault Tolerance	0	0

Performance

Scalability

We generally define scalability as the middleware's ability to perform when the size of the problem increases. Middleware performance can be highly variable depending on how it is used. For example, component granularity is one of the most significant drivers of performance stress. In other words, as the pieces get smaller, so does sheer volume — causing the middleware substrate mechanisms to work harder. As this occurs, the need arises for middleware mechanism tuning in ways that conventional database products have supported. Finally, middleware performance is very costly to measure, since only in vitro modeling can provide a reasonable capacity estimation.

There must be compelling evidence, via current implementations or anecdotal data, that indicate the middleware's ability to scale through various scenarios. These scenarios may include numbers of objects or users. Key areas where impacts are most likely are found in services that commonly aggregate components such as naming services or interface repositories. Finally, middleware must able to support threads to allow parallel processing of work.

CORBA

As a specification, CORBA does not address specific scalability services aside from providing for the transparent distribution of processing. Instead, individual ORBs deal with this problem in one of two ways:

Threading — Many ORBs provide thread-safe libraries that use each operating system's native thread model. This enables threads to be created for clients, objects, or even specific method calls of an object. In addition, several ORB products also support thread pools. Filters can also be used to balance processing based on current loading.

Tuning — ORB products provide various internal tuning mechanisms to enable optimization for specific situations. For example, internal memory representations can be changed to order references by "most frequently used" or other criteria that suit the specific conditions.

DCOM

DCOM offers similar scalability mechanisms such as parallel processing and threading. As with CORBA, DCOM features are not transparently supported, and require detailed knowledge of client/server interactions to implement.

Thread pools — DCOM utilizes thread pool managers to maximize scalability however, Windows NT symmetric multiprocessing is required to support this feature.

Summary: Both middleware products are relatively nascent in their support for highly scalable enterprise applications. There are, however, a growing number of large-scale ORB implementation examples in the investment, aerospace, and telecommunications industries. In addition, indirect evidence of scalability can be inferred when combining ORBs with enterprise-grade products such as commercial TP monitors and MOM. Concrete evidence of large-scale DCOM enterprise applications is not readily available at this time.

Enterprise Criterion	Ratings	
Performance	CORBA	DCOM
Scalability	0	-

Viability

Product Maturity

Despite the current fragmented condition of middleware offerings, we believe IT organizations should be consumers of middleware components, not producers. As of this writing, the best way to manage the massive complexity of middleware is through the purchase and customization of commercial frameworks that organize their flexibility into structured application packages. Such frameworks go beyond the primitive and complex services that CORBA and DCOM can provide but still require a minimum maturity level from each.

CORBA

Many commercial ORB products are in their third generation of development. As such, we are beginning to see a critical mass of services (Directory, Messaging, Transactions, and Security) in several leading products. Although the OMG specification is explicitly designed to insure these services are well integrated, no single ORB vendor has brought them all together in strict CORBA compliance. Irrespective of this, ORBs are now being used for enterprise systems in many demanding industries, including telecommunications, aerospace, and investment.

DCOM

The arrival of the predator services (Falcon, Viper, Wolfpack, and Active Directory) represents Microsoft's recognition of what must be in place in an enterprise setting. Like the leading ORB products, these services are not fully integrated (even in the "NT only" environment) and are not explicitly part of DCOM. What is worse, platform interoperability is only just appearing on the DCOM radar screen and will likely be the last piece to fall into place.

Summary: Products from both DCOM and CORBA are only just beginning to aspire to the so-called "heavy lifting" demands of the enterprise. At the end of the day, representative products from both middleware camps require a tremendous amount of financial fortitude and technical expertise from the adopting enterprise to be successful.

Enterprise Criterion	Ratings	
Viability	CORBA	DCOM
Product Maturity	0	-

Vendor Outlook

Clearly, the trick is to buy and extend frameworks that are based on the likely winners of the middleware framework wars, not the losers.

CORBA

CORBA differs from DCOM in an important way. While the CORBA specification is controlled by the OMG standards body, ORBs are produced by a variety of vendors (though most belong to the OMG). This separation has caused — and always will cause — a natural tension between the need to differentiate product offerings and the need for interoperable standards compliance. The OMG currently enjoys membership and backing from some 760 members. This large membership will continue to uphold the OMG's emphasis on interoperable and standardized solutions but is often to blame for slow progress.

There are several leading vendors whose viability is sound for at least the next several generations of enterprise technology. Users must insist on enterprise players that will not only survive the middleware wars but also remain committed to ORBs as part of their long-term, strategic direction. Such vendors as IBM, BEA, and Orbix appear to fit this category.

DCOM

Clearly, Microsoft will be one of the survivors. Microsoft stands as testament to the difference between technological elegance and marketing leadership, a distinction that should never be overlooked. While there is no question about Microsoft's intention to support its value proposition — *optimized product integration on lower-cost NT platforms* — overwhelming support for competing platforms would not be a logical assumption. That being said, Microsoft will continue to focus its attentions on real scalability, manageability, and low cost in the NT environment.

Summary: Both technologies will continue to have significant market backing and support. DCOM will continue to enjoy heavy independent software vendor (ISV) tool support while ORBs will continue to be supported by corporate customers. It is important to note, however, that non-DCOM integrated services such as MTS and MSMQ have not been widely tested, and ISV acceptance for these is yet to be determined.

Enterprise Criterion	Ratings	
Viability	CORBA	DCOM
Vendor Outlook	+	+

Middleware Synergy with Java

The Internet has heralded in a new era of distributed computing. The ubiquitous nature of the Internet demands platform-independent component solutions. To this end, Java has emerged as one of the dominant technologies for Internet-based computing. By combining Java effectively with middleware, businesses can gain maximum market exposure to valued enterprise IT resources.

CORBA + Java

In many respects, CORBA and Java address similar goals. Both support distributed business applications, and both attempt to insulate the specifics of different platforms from the business applications. CORBA supports distributed objects in a heterogeneous (cross platform, language, and network) server environment. Java supports component relocation across platforms, but not across languages.

The differences between these middleware technologies are subtle and diminishing each day. For example, Remote Method Invocation (RMI) enables Java components to be easily relocated across the network. The CORBA specification is now addressing object "pass-by-value" to accomplish a similar goal. Further, both Sun and the OMG have been actively involved in combining these models where advantageous as described below. It is the often in the areas of distinct differences, however, where CORBA and Java can be combined to gain the best of both worlds.

Differences and Synergistic Opportunities

CORBA is a distribution and integration technology — not a general-purpose development language. CORBA assumes that various general-purpose languages will continue to be used and integrated. Java on the other hand, is a full-fledged language and set of services used to build distributed components. Java is positioned as a language that *directly incorporates* distribution as well as a comprehensive computing platform with a set of infrastructure services. CORBA, on the other hand, is positioned to provide distribution features for *existing* languages and applications.

Java has evolved in many areas that are complementary to CORBA. On the client side, Java provides client functionality and portability. The visual features of Java applets and JavaBeans components have been widely recognized in the marketplace. By combining Java's graphical features on the front end with CORBA's access to legacy applications on their native back-end platforms, developers can integrate new technology with the old and gain the best aspects of each. In a sense, Java can provide the component model that is missing from CORBA.

Java has also been aggressively positioned to support the server side. For example, Sun's Enterprise JavaBeans (EJB) now provides a collection of services for database access, transactions, directory, and messaging. Although many of these services have equivalents to well-established ORB offerings, several services have emerged that add significant value to the enterprise middleware milieu.

In many ways, CORBA and Java have been engineered to collaborate. The following sections provide examples of how Java and CORBA have been or are now being designed to work together.

Language Interoperability

Since 1996, the OMG has worked to provide a comprehensive mapping between the Java language and CORBA. Even today, JavaBeans language elements are being extended by Sun to work with CORBA-based services. The reverse is also true as the OMG pursues the adoption of JavaBeans as part of its until now, lacking component model support.

Java Layering

With EJB, Java is being layered above CORBA services as well as others. Using this approach, Java developers can gain several advantages. First, the layering hides many of CORBA's details and frees the programmer to concentrate on the business application. Support for transactions and distribution becomes implicit thereby relieving the need for low-level programming. Without this layer, service complexity is fully exposed to the developer. Second, it permits services other than CORBA to be plugged in on the server side. For example, the Java Transaction Service (JTS), a Java platform API, allows bindings to multi-vendor transaction processing solutions, including the CORBA Object Transaction Service (OTS).

Network Communications

In June of 1997, Sun announced that IIOP was to be adopted as a way for Java RMI to communicate across the network in addition to its native protocol, Java Remote Method Protocol (JRMP). This effectively extended Java with CORBA support for standards-based interoperability and connectivity. Java applications can transparently invoke operations on remote CORBA services using IIOP or JavaIDL, a Java-based ORB that also allows communication between Java and CORBA components.
In addition, Java components can be automatically configured as CORBA objects and accessed through IIOP by other non-Java clients.

Back-End Services

The Java Database Connectivity (JDBC) API provides a uniform interface to a wide range of relational databases. IIOP-based JDBC implementations can reap the benefits of combining client-neutral Java components with server-neutral CORBA connectivity to distributed and heterogeneous data stores.

The Java Naming and Directory Interface (JNDI) is another example of EJB-based layering. JNDI is a set of interfaces that supply distributed naming and directory services. It can be mapped to the CORBA Naming service as well as mature directory services such as X.500, LDAP, NIS, and NDS.

Summary: In many ways, Java and CORBA have been architected to collaborate in the enterprise environment. Most importantly, Java components can be used as an extension to CORBA services for thin-client Internet computing. The synergistic opportunities outlined above enable CORBA services and Java components to be mixed and matched as needed without sacrificing platform interoperability.

DCOM + Java

Several vendors have taken steps to combine DCOM and Java. Microsoft has focused primarily on *integration* by allowing Java components into the Windows environment. Sun and other third-party vendors have provided products that enable Java to *interoperate* with ActiveX on Windows platforms.

Client Services

On the client side, Microsoft has blurred the distinction between Java and ActiveX by permitting both components to be viewed in the same browser. The benefits of this integration are primarily enjoyed by users of ActiveX-supported browsers, however.

Finally, for developers interested in having their components interoperate with existing ActiveX components, Sun provides the JavaBeans Bridge for ActiveX. This Bridge provides users of legacy COM containers with the ability to use JavaBeans components, but again, only in the Wintel client-side environment .

Back-End Services

On the server side, Microsoft's version of the Java Virtual Machine (JVM) provides the ability to make Java objects appear as DCOM objects, giving them full access to DCOM system services. The JVM seamlessly integrates many underlying DCOM mechanisms such as reference counting and object introspection. Integration with Java class libraries also exposes key DCOM functions such as monikers and structured storage.

Unfortunately, the Microsoft JVM does not support many fundamental Java services such as RMI, the Java Native Interface (JNI), or EJB services such as JTS. In one example, there is no equivalent RMI layering over DCOM as there is between RMI and IIOP. In effect, this relegates Java to a programming language and limits synergy between Java and DCOM. Further, ActiveX wrapped Java components lose their cross-platform abilities. For example, JNI allows Java programmers to combine native code with Java code and retain portability. Microsoft's JVM does not currently support JNI and thereby nullifies Java's portability capabilities.

Summary: On the surface, DCOM and Java appear to fit well together. The full enterprise benefits of this combination, however, are very limited. Although Java can access a limited number of DCOM services through the Microsoft JVM, full access to enterprise-grade DCOM services across platforms is not available. Until substantive interoperability has been demonstrated between DCOM services on platforms other than NT, enterprise synergy between the two will be greatly limited.

Summary of Differences between CORBA and DCOM

Both middleware technologies provide integration frameworks for object-based, distributed client/server development. As such, both technologies provide full support for component distribution. In pure technology terms, this paper has demonstrated that most of these challenges will eventually become resolved. The key remaining difference lies in support for cross-platform interoperability.

Historically, CORBA has focused on enterprise-wide interoperable solutions, whereas many of the underlying technologies that make up DCOM have focused on the desktop. Since 1989, CORBA has followed the path of developing the architecture first, then the implementation. Microsoft, on the other hand, follows the opposite path of "build first, then architect." Unfortunately, CORBA's "architect first" distinction does not necessarily hold with the vendors of ORB products. As a result, ORB vendors often provide services that are not in full compliance with the CORBA specification.

From the outset, CORBA has concentrated on providing an open architecture to support interoperability with both existing and new technologies. As evidence, neither Java nor DCOM were defined at the time CORBA was originally specified; however, significant progress has been made toward integrating these into the standard. This is a testament to open nature of the CORBA reference architecture and the collaborative nature of the OMG as a standards body. The inherent problem with this approach is market timeliness. The collaborative nature of the OMG is often cited as an obstacle for getting functionality "on the glass."

DCOM has approached distribution and interoperability as an afterthought. This is evident in the way that DCOM has evolved from DDE to OLE to OLE2 to ActiveX and so on. However, this makes DCOM's significance as a limited platform middleware no less important. The often-cited problem with this approach is disruption to previous iterations of technology and lack of interoperability.

Finally, CORBA has traditionally been heavily concentrated toward the mixed platform server domain. The recent collaborations with Java have given CORBA products a way to realize support for the client tier also. With ActiveX, DCOM incorporates both client-side and server-side features. The homogenous nature of the desktop has given DCOM less motivation for cross-platform support. With NT's maturation in the middle-tier, DCOM will most likely continue this trend.

Conclusions

Bottom Line: CORBA continues to be seen as the middleware of choice where enterprise systems are made up of a wide variety of languages, platforms, and operating systems. Although DCOM is rapidly evolving into a viable middleware for NT platforms, it has yet to become practicable for real-world, heterogeneous enterprise environments. Microsoft's relentless displacement of middle-tier platforms, however, demands that enterprise middleware adopters actively employ DCOM-friendly approaches, because both technologies will likely coexist and mature in the foreseeable future.

Interoperability	CORBA	DCOM
Cross-Language Support	+	+
Cross-Platform Support	+	-
Network Communications	+	0
Common Services	0	-
Reliability		
Transactions	+	0
Messaging	0	-
Security	0	0
Directory Service	0	0
Fault Tolerance	0	0
Performance		
Scalability	0	-
Viability		
Product Maturity	0	-
Vendor Outlook	+	+

Note that there is no overall grade at the bottom of each middleware category. This is because we recognize that not all organizations require all levels of enterprise support. Further, since the grading was conducted on the best implementations currently available in each category, it is impossible to make such broad generalizations. This condition is equally true with both CORBA and DCOM. In the future however, we believe most ORBs will provide a complete set of critical services and Microsoft will eventually integrate their separate services (e g. MTS, MSMQ) into DCOM or another consolidated offering such as DNA.

For now and the foreseeable future, it is unlikely that any one middleware offering will dominate as the "universal solution." This is particularly true at the enterprise level, where IT diversity is greatest. META Group believes organizations must view the move toward middleware as a set of discrete steps. Instead of an all-at-once approach, companies must develop a three- to five-year middleware strategy that supports targeted business needs in a prioritized manner.

This report has defined several critical needs that the enterprise demands of middleware products. It then decomposed the offerings of the competing middleware models, CORBA and DCOM, and described their current progress in meeting these needs. This report has concluded that between the two models, CORBA-based middleware remains the dominant technology for enterprise needs that demand multiple-platform/operating system interoperability. Conversely, DCOM-oriented middleware may be the obvious choice for homogeneous Wintel-based environments although, several scalable CORBA ORB implementations have been demonstrated on NT platforms. DCOM's dependence on the Wintel platform and lack of interoperability are pervasive inhibitors for the enterprise. This condition continues with the DCOM/Java relationship as described above. For the time being, it is unlikely that any single middleware will solve the disparate needs of the enterprise, particularly as long as the enterprise-computing environment requires a mix of languages, networks, and platforms to solve business problems.