# Image Cover Sheet

| CLASSIFICATION | SYSTEM NUMBER |
| --- | --- |
| UNCLASSIFIED | 516853 |

**TITLE**

A comparison of COM and CORBA for command and control prototypes

**System Number:**

**Patron Number:**

**Requester:**

**Notes:**

**DSIS Use only:**

**Deliver to:** DK

*This page is left blank*

*This page is left blank*

# A Comparison of COM and CORBA for Command and Control Prototypes

Gaétane Harvey
CGI Inc.

Valdur Pille
DREV

**Defence Research Establishment Valcartier**

Technical Report

DREV TR-2000-219

2001-12-17

Author

Gaétane Harvey, Valdur Pille

Approved by

Guy Vézina

AH/CCISE

Approved for release by

Gilles Bérubé

Chief Scientist

## Abstract

This document contains the results of a brief analysis of the most popular technologies currently used with respect to component-based development over a distributed architecture. These technologies are COM (Common Object Model) from Microsoft and CORBA (Common Object Request Broker Architecture) from the OMG (Object Management Group), a consortium of more than eight hundred (800) companies. This study aims at highlighting the strengths and weaknesses of each technology and should help determine where to use COM and/or CORBA. This study was conducted in the scope of the Wing And Squadron Prototype project.



## Résumé

Ce rapport présente les résultats d'une analyse sommaire des technologies les plus couramment utilisées en ce qui concerne le développement à base de composants dans une architecture répartie. Ces technologies sont le COM (Common Object Module) de Microsoft et l'architecture CORBA (Common Object Request Broker Architecture) du groupe OMG (Object Management Group), un consortium de plus de 800 organisations membres. Cette étude vise à exposer les points forts et faibles des technologies COM et CORBA, et à faciliter le choix de la technologie à utiliser dans des projets précis. Cette étude a été effectuée dans le cadre du projet Wing And Squadron Planner.

This page intentionally left blank.

# Executive summary

In the area of distributed applications, two architectures that now lead the market for building distributed object infrastructures have emerged. These are: DCOM, now called COM+ and CORBA. COM+ (which is a merger of COM, DCOM, and MTS, Microsoft Transaction Server) is Microsoft's proprietary distributed object technology oriented towards the Windows platform, while CORBA is an open standard developed by the Object Management Group as a distributed object infrastructure for heterogeneous platforms. Both architectures are based on the concept of objects to distribute processes and data and have much in common, but each architecture also has different perspectives.

This study was performed by looking at some of the fundamental features of distributed object architectures from general and high-level aspects. Both COM and CORBA address technical details in different ways and again each has strengths and weaknesses. Comparisons are made from the viewpoints of specifications, implementation, services, platform and tool support, and maturity.

The major weakness for Microsoft is its platform limitations. COM usage within Java requires the Microsoft Java Virtual Machine that is not supported on non-Windows platforms. Another negative point is the reliance on one single vendor for COM support, i.e. Microsoft on Windows platforms and Software AG on Unix.

CORBA provides cross-language, cross-platform, cross-vendor support for creating servers on a wide range of operating system platforms. However, appropriate development component tools are still not as mature as on other integrated development environments

CORBA appears to be a good choice for creating middle-tier servers because middle-tier server development relies on a solid remoting architecture. In contrast, COM is more appropriate to create client-tier components.

Harvey,G., Pille, V. 2001. A Comparison of COM and CORBA for Command and Control Prototypes. DREV-TR-2000-219 Defence Research Establishment Valcartier.

# Sommaire

Dans le domaine des applications réparties, deux architectures se partagent le marché dans le développement des infrastructures à objets répartis. Ce sont DCOM, un produit commercial de Microsoft maintenant appelé COM+, et CORBA. L'architecture COM+, qui est en réalité un amalgame de COM, DCOM et MTS (Microsoft Transaction Server), est une technologie des objets répartis adaptée à l'environnement Windows, tandis que l'architecture CORBA est une norme ouverte mise au point par le groupe OMG (Object Management Group) dans le but d'offrir une infrastructure à objets répartis pour un réseau hétérogène d'ordinateurs. Les deux architectures ont beaucoup de points en commun, dont le concept des objets pour répartir les processus et les données, mais leurs perspectives sont différentes.

Cette étude porte sur certains traits fondamentaux des architectures à objets répartis, plus particulièrement les aspects généraux et à haut niveau. Les architectures COM+ et CORBA abordent différemment les mêmes détails techniques, chacun ayant ses points forts et faibles. Les deux architectures ont été comparées selon leurs caractéristiques, leur implantation, leurs services, le soutien disponible pour les plates-formes et les outils, ainsi que leur maturité.

La principale faiblesse de l'architecture COM+ de Microsoft est son usage limité aux plates-formes Windows. Pour pouvoir être utilisée avec Java, l'architecture COM+ nécessite le produit Java Virtual Machine de Microsoft, lequel ne fonctionne (n'est soutenu) que dans un environnement Windows. Un autre point négatif de l'architecture COM+ est qu'un seul fournisseur en assure le soutien technique, soit la firme Microsoft pour les plates-formes Windows et la firme Software AG pour les plates-formes Unix.

L'architecture CORBA a l'avantage d'être compatible avec plusieurs langages de programmation (cross-language), de pouvoir fonctionner sur des plates-formes multiples (cross-platform) et d'être soutenue par plusieurs fournisseurs. Ces avantages permettent de concevoir des serveurs pour une gamme étendue de plates-formes et de systèmes d'exploitation. Cependant, cette architecture manque d'outils appropriés pour la mise au point de composants.

Il reste que l'architecture CORBA constitue un bon choix pour la conception de serveurs à palier moyen (middle-tier), puisque ceux-ci nécessitent une solide architecture. Par contre, l'architecture COM+ est plus appropriée pour concevoir des composants à l'intention des clients (client-tier).

Harvey,G., Pille, V. 2000. A Comparison of COM and CORBA for Command and Control Prototypes. DREV-TR-2000-219. Defence Research Establishment Valcartier.

# Table of contents

This page intentionally left blank.

# List of figures

This page intentionally left blank.

# 1. Introduction

In the area of distributed applications, two architectures have emerged that now lead the market for building distributed object infrastructures. These are: DCOM now called COM+ and CORBA. COM+ (which is a actually merge of COM, DCOM, and MTS, Microsoft Transaction Server) is Microsoft's proprietary distributed object technology oriented towards the Windows platform while CORBA is an open standard developed by the Object Management Group as a distributed object infrastructure for heterogeneous platforms. The problem that these architectures address for the development of applications is how to define, construct and deploy the software elements that comprise complex systems in distributed environments. Both architectures are based on the concept of objects to distribute processes and data and have much in common, but each architecture also has different perspectives.

The following paragraphs present an historical background and the rationale behind such a type of architecture followed by a brief introduction to both COM and CORBA with their respective strengths and weaknesses and provide some guidelines regarding the choice of the technology.

The aim of this study was to evaluate approaches for distributed architectures for the Wing and Squadron Prototype (WASP).

# 2. Background

Periodically, there is a revolution in the computing industry causing most of the time a radical shift in the way of designing, developing or deploying systems. The new buzzwords are: component-based, distributed architecture, reusability, among others. To better understand these new trends of the industry, it is interesting to go back to the origin of the computing industry and to review the major changes that have occurred during the past thirty years.

In the seventies (and before), the mainframe was the only platform available on the market. The application architecture was monolithic, that is - the interface, logic and data were all contained in one large application. End users accessed the mainframe through terminals where the entire application was running. Figure 1 presents an example of a monolithic architecture.

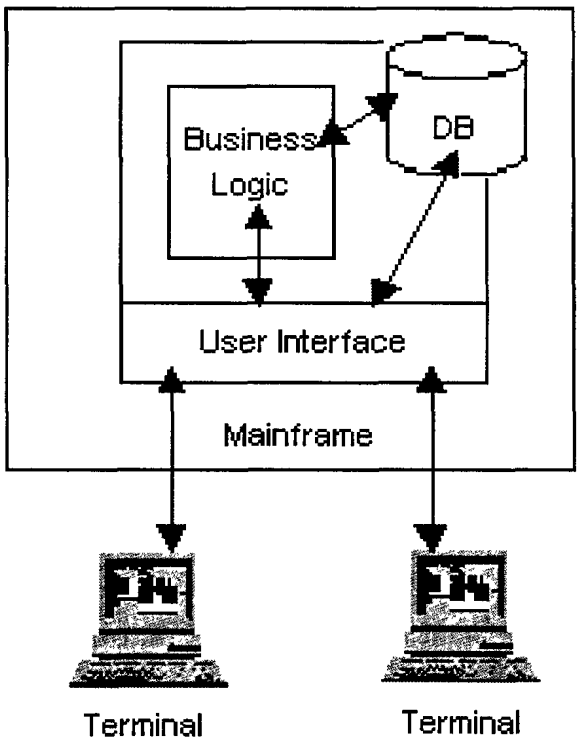


**Figure 1. Monolithic Application**

In the eighties, the advent of PCs made a significant change to the architecture of applications running on the mainframe. It was now possible to move the interface and some (or all) business logic to the PC – called the client-side while the mainframe was used to store the data and to perform some processing. It was the beginning of the client/server architecture. Another benefit of this technology is that PCs were also used to perform many other tasks without the need of a mainframe. In the same period came along the proliferation of smaller machines (namely the UNIX-based servers) which are very popular among organisations that do not require the power of a mainframe or who simply cannot afford it. Figure 2 depicts a typical client/server application where the database resides on the server-side, the user interface resides on the client-side (PC) and the business logic resides in either or both sides. This type of architecture is now commonly called a two-tier architecture.

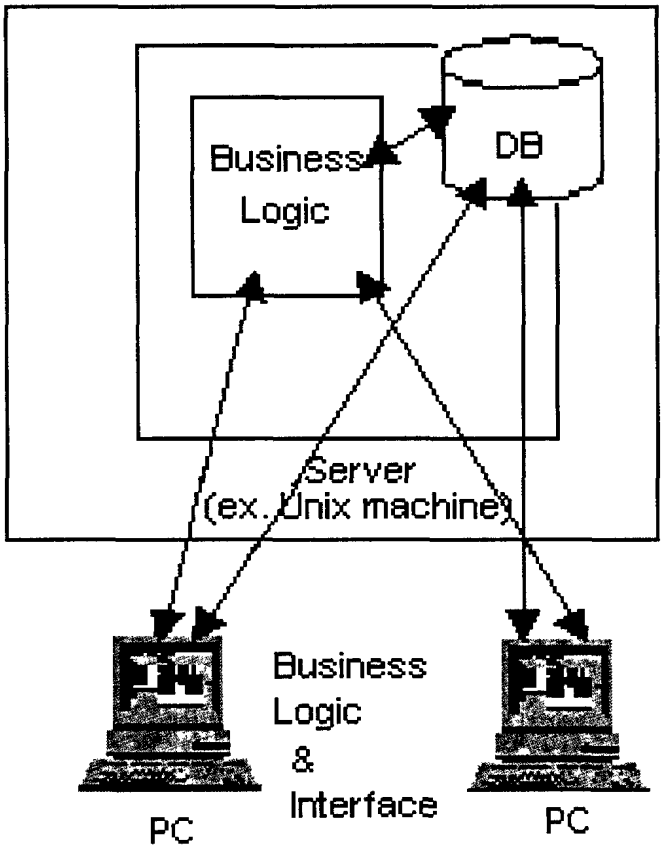


**Figure 2. Two-tier Client/Server Architecture**

The nineties was the beginning of object-oriented concepts and namely the concept of encapsulation. The two-tier client/server architecture was evolving towards a three-tier architecture. Compared to a two-tier architecture, the three-tier architecture provides an intermediate layer (e.g. located on the server-side to access the database). This evolution serves mainly to resolve the performance and deployment problems faced by the two-tier architecture where the database is accessed directly from the application that resides, in most cases, on the client-side. The three-tier architecture distributes the application as follows: the business logic resides on the server, a new layer called the database layer is created on the server side, the interface resides on the PC and accesses the data through the business logic only. This architecture resolves the problem of deployment as any change made to the business logic or the database does not require to re-deploy the application to all the PCs that need to execute it. Figure 3 provides an example of a three-tier architecture.

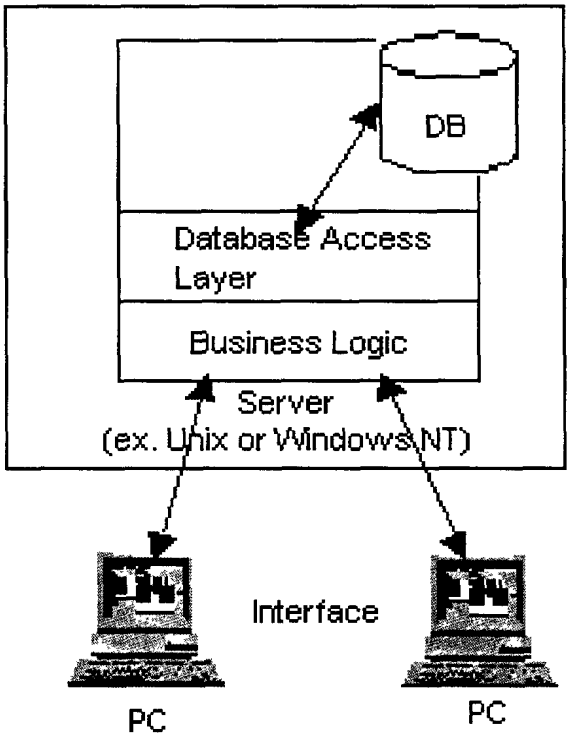


**Figure 3. Three-tier Client/Server Architecture**

The next step in this evolution is called the distributed system architecture. This is the type of architecture that we address in the current WASP project. The distributed architecture enforces the concept of objects and encapsulation and more specifically the component-based architecture. The system is built by creating business components that provide various services (ex. logic, data access, security, transaction, etc.) and can interact with each other. The implementation of the component is specified by an interface and is accessible only through this interface. As long as the interface is not changed (which is contrary to this strategy), a component can be modified (the implementation part only) without any impact on other components that use it. In a distributed environment, the components can obviously be distributed on many various platforms providing that way a great deal of flexibility.

Several technologies compete in the area of distributed applications – e.g. Java RMI, RPC, DCE, COM, and CORBA. The latter two are the subject of this study. Both provide standard mechanisms to support the development of component-based applications in a distributed environment. Many COM or CORBA-compliant products are now available on the market but the organisations that want to move to this type of architecture are facing the difficult dilemma of deciding which architecture is the best for them. The answer is not so simple. The following paragraphs introduce both architectures and then present some guidelines and recommendations for distributed prototypes

# 3. What is Com/DCOM/COM+?

COM refers to both the Microsoft Component Object Model *specification* and *implementation*. It is developed by Microsoft Corporation to provide a framework for component-based development. It is available on Windows platforms and *some* other operating system versions such as UNIX and IBM. Contrary to CORBA, COM is not platform-independent.

From its original version introduced in 1993, COM has evolved to DCOM (1996) and then COM+ (2000). In its basic form, COM defines an application programming interface (API) to allow implementation of components which can interact within one or multiple processes on a single host computer. The Microsoft API is specified using Microsoft's Interface Definition Language (a binary structure). This binary structure provides the basis for interoperability between software components written in an arbitrary language such as Visual Basic, C++, Java, etc. Thus COM components can be developed in any language as long as the corresponding compiler supports Microsoft's binary structure.

DCOM (Distributed COM) is the extension of COM allowing components to interact between different host computers in a network. Among the facilities and services offered with DCOM (or COM+), there are:

- The Microsoft Transaction Server (MTS): The MTS supports the creation of transaction-oriented components allowing developers to create secure server applications and to ensure database integrity.
- The Microsoft Message Queue (MSMQ): The MSMQ allows applications to communicate in an asynchronous way. With MSMQ an application can send a message to another application without waiting for a response. The message is sent in a message queue that is handled by the MSMQ component.
- The Microsoft Internet Information Server (IIS) or Active Server Page (ASP): IIS and ASP allow the creation of Web-based application (using VBScript) combined with the creation and use of COM components and transactions.

COM+ is the new generation of technology providing a set of additional enhancements to handle security, component registration, communication, database connection pools, etc. COM+ is built on what already exists within DCOM. It enhances or extends existing services by providing:

- A publish and subscribe service: This service provides a general event mechanism that allows multiple client applications to subscribe to various published events.

- Queue components: This service allows client applications to invoke methods on COM components using an asynchronous model.
- Dynamic load balancing : This service allows the automatic spread of client application's requests across multiple equivalent COM components
- Full integration of MTS: Includes broader support for transactions, security and administration.

All these facilities and services are provided to make the work of developers easier when building COM component-based applications in a distributed environment. The developers can concentrate on business objects while aspects concerning persistence, object communication, security, database transaction, etc. are handled by the different commercial COM components.

As stated by Pritchard [4], COM evolved in a bottom-up fashion, where each step in the evolution results in increasing functionality for ongoing product development at Microsoft.

# 4. What is CORBA?

CORBA is the Common Object Request Broker Architecture defined by the Object Management Group (OMG), a consortium of now more than 800 companies representing the entire spectrum of the industry. One goal of the OMG is to agree on a standard architecture for the distribution and communication of objects across multiple platforms. CORBA is then designed specifically for distributed environments. Originally, companies such as IBM, DEC and HP were facing the problem of integrating a mix of systems for their clients with various architectures, various operating systems and various platforms. Their goal was then to address the question of interoperability and portability. Other OMG members were more concerned about defining a standard for developing distributed applications. No matter what is the rationale behind the first initiative of the OMG, the common agreement of the members was to create a standard mechanism to allow objects to communicate independently of what is the platform, the language or the operating system used. CORBA is then *platform-independent* and *language-independent.*

- The OMG allows any organisation to participate and to submit proposals either to the Platform Technology Committee or the Domain Technology Committee. The consortium uses a membership structure that includes domain contributing, influencing, auditing, government and university members. OMG members agree to adopt specifications and provide compliant products but the technology submitted must be available within one year.

The first CORBA specification was published in 1991 and it continues to evolve through the addition of new services and facilities. CORBA's specifications include:

- The Object Request Broker architecture and specification which is the foundation for building applications from distributed objects;
- CORBA Services that handle object lifecycle, naming, persistence, transactions, security, etc.
- CORBA Facilities that provide support for compound documents, workflow, etc.
- CORBA Domains that provide capabilities for vertical market such as insurance, health care, financial services, etc.

The language that supports integration of objects is called the Interface Definition Language (IDL). All CORBA object interfaces and data types are created using the IDL. CORBA objects defined with the IDL can easily be mapped through an IDL Compiler to a specific target language such as C, C++,

Visual Basic, Cobol, etc. CORBA implementations support a wider range of languages than COM and on more platforms. Surprisingly, CORBA as an architectural solution is used more often than COM on Microsoft platforms.

Pritchard [4] sees the evolution of CORBA differently form the evolution of COM. Actually CORBA evolves in a top-down fashion where the first step involved the creation of the Object Management Architecture describing high-level services and facilities and low-level object bus. Corresponding specifications were produced and then implementations followed.

# 5. COM/CORBA Comparison

A certain number of characteristics must be looked at when considering COM and CORBA. They are direct competitors and both argue about the merits of their approach but in the industry there is no clear evidence of the winner. Nevertheless, the choice is important, as it will affect the tools, applications and the development process used by an organisation.

The information that one can find in the literature provides very valuable input to take a decision but the selection of one technology or the other must certainly be guided by the existing infrastructure and the evolution of the organisation.

Tallman and Kain [3] provide a very helpful comparison between COM and CORBA. Their comparison is based on a set of characteristics addressing the documentation (specifications), the object model, error handling, persistence, transparency, scalability, services, platform, tools and maturity.

Pritchard [4] proposes a categorisation of distributed object architecture: *component* architecture and *remoting* architecture. *Component* architecture is defined as architectures that focus on component packaging and cross-language interoperability. *Remoting* architecture is defined as architectures that focus on support for remote invocation on distributed objects. Pritchard then classifies the COM architecture as being a component architecture and CORBA as being a remoting architecture. Based on this classification, Pritchard makes an assessment of strengths and weaknesses of both technologies.

The next sections provide a comparison of both COM and CORBA side by side according to various characteristics and perspectives found in the literature. This comparison does not address fundamental technical aspects of COM and CORBA (ex. object creation, destruction, data types, etc.), it provides a survey on fundamental inherent aspects such as specifications, services, implementation, etc.

As both COM and CORBA are in constant change and evolution, the reader must be aware that the information provided herein may have changed or may change in a near future. The term 'COM' used in the rest of the document does not necessarily mean COM (the first version of this architecture), it is commonly used to designate either COM, DCOM or COM+.

## 5.1 Distributed Architecture

*Distributed architecture* refers to the capability for inter-machine communication.

**COM**

- Because of limited platform support, COM is still identified as being more a component architecture than a remoting architecture. COM does support distributed objects but especially on Windows-based platforms.

**CORBA**

- CORBA was intended from the beginning to solve the problem of communication. The CORBA specification outlines an architecture for communication between processes either on the same machine or on different machines.

## 5.2 Specifications

*Specifications* refer to formal written documentation. They should be concise, authoritative and usable by developers. They provide a common framework for application development where primary goals are reusability, maintainability, portability, and interoperability of object-based software in distributed, heterogeneous environments. All components or commercial products (implementation) based on COM or CORBA architecture should adhere to their respective specifications.

**COM**

- The COM specification is incomplete, making it difficult in some cases to determine the correct behaviour.
- There is no single document describing the Microsoft Interface Definition Language (MIDL) for COM. The MIDL is an extension of the IDL for OSF DCE RPC and the COM specification includes only the extension for COM.
- Microsoft controls the COM specification and the corresponding implementation and the Microsoft intentions are unknown about their updating.

**CORBA**

- The CORBA specification is formal and concise. The information presented is clear and well organised.

- The complete CORBA specification set is distributed in multiple documents. One document presents the objectives, terminology and the conceptual models; the core of CORBA and the different languages mapping (IDL format) are described in a separate document and finally there are two other documents describing respectively the CORBA Services and the CORBA Facilities.
- The core document is available at the following address: http://www.omg.org/library
- The CORBA specification is fed by OMG members who submit requests for proposals and for comments. Specifications are adopted as standards only after acceptation (vote) by the OMG representatives.
- As the CORBA specification is augmenting, it is a challenge to maintain clarity and integrity across the large number of distinct IDL specification.
- There is a lack of basic naming standards across CORBA services.
- Certain services overlap in functionality.

COM and CORBA are both described by specifications but the documents differ substantially in authority, style, precision and purpose.

## 5.3 Implementation

Implementation refers to components and products developed by vendors and based on COM or CORBA formal specifications. These components/products are available to develop end-user applications.

**COM**

- Thousands of third-party ActiveX controls (i.e. in-process COM components) are available that can be used to quickly create end-user applications.
- Microsoft and other vendors have built many tools to accelerate development of ActiveX controls and ActiveX-based applications.
- The suite of COM APIs is growing incrementally, making it difficult to comprehend and use it effectively.

**CORBA**

- A large number of vendors and research organisations have released CORBA products rapidly.
- CORBA implementations did not achieve the expected acceptance particularly among large enterprises.
- Initial releases of several products were not complete and very robust.
- Vendors' submissions have not always been entirely mature and have not always been delivered on schedule.

## 5.4 Services

In addition to basic services allowing objects to communicate, *services* provide additional capabilities to simplify component-based development in a distributed environment.

COM and CORBA both provide a basic framework for creating and using components. An essential part of the framework is the value-added services that they provide to components. These include: naming, security, access, events, transactions, etc. It is very difficult to compare them side by side as they do not necessarily categorise their services the same way. Nevertheless, it is imperative to identify what are those services on both sides.

**COM**

Microsoft Services are categorised as follows:

Microsoft Transaction Server (MTS):

- Automatic transactions
- Configurable Security
- Database connection
- Automatic thread support
- Process isolation through packages
- Integration with mainframe transactions

Microsoft Message Queue (MSMQ):

- Com-based access
- Automatic message journalising

- Automatic notification
- Data integrity
- Message priority support

COM+ adds-ons:

- Publish and Subscribe Service
- Queue components
- Dynamic Load Balancing

**CORBA**

CORBA Services are specified by the OMG. They are available in the form of components through different CORBA-compliant products. The CORBA services include the following :

- Naming Service
- Event Service
- Life Cycle Service
- Persistent Object Service
- Transaction Service
- Concurrency Control
- Relationship Service
- Query Service
- Licensing Service
- Property Service
- Time Service
- Security Service
- Object Trade Service
- Object Collections Service

The complete specifications of these services are available at the following address:
http://www.omg.org/corba/sectrans.html

CORBA's main strength, i.e. allowing developers to use the best services provided from different vendors, is also its biggest weakness: it is very difficult to make one vendor service work with another vendor service.

Both COM and CORBA provide a set of additional services and facilities that evolve as long as new versions become available. COM has evolved to DCOM and now COM+. Each version is built on top of the other with enhancements to existing services or add-ons of new services.

## 5.5 Platform and Tool Support

*Platform* and *Tool Support* refer to: 1) the range of desktop and server platforms 2) the range of development tools (design tools, compilers, debuggers, class libraries, etc. 3) the range of object-oriented and procedural programming languages

**COM**

- Prior to 1998, the platform was limited to Windows NT and Windows95, and now supports the implementation of COM on some other platforms notably UNIX and IBM, but the version implemented on UNIX is apparently two (2) versions earlier than the one implemented on Windows NT.
- On Windows development platforms, COM objects can be built by using popular Microsoft languages such as Visual Basic, Visual C++ and Java.
- Huge numbers of tools exist for creating COM objects on Windows platform. However very few tools exist for creating COM server objects on non-Windows platforms.
- Integration of mixed Windows environments (NT, 95, 98) using DCOM is not simple, as the level of functionality offered on each platform is not the same.
- The commitment of Microsoft to implement COM on multiple platforms is not clear.

**CORBA**

- CORBA is platform-independent making it inherently multi-platform.
- One area where most CORBA vendors are weak is in development tools. This is partially explained by the fact that CORBA products tend to cover a wide range of programming languages and platforms rather than focusing on a single platform.
- CORBA is designed to support widely-used programming languages without requiring their modification. The OMG has produced standard CORBA IDL mappings for Java, C, C++, Smalltalk, Cobol, and Ada. In addition, mappings for other languages have been implemented by some vendors. For example, Inprise Corp. made a port to Delphi Object Pascal language.

## 5.6 Maturity

*Maturity* is the capability to demonstrate the technology on software development projects in a wide range of industries and the capability to prove the stability of specification and commercial implementations.

As the market is evolving so rapidly, it is very difficult to measure exactly the maturity of a technology. Tallman and Kain tried to find out the number of software projects in the industry that have been using the COM or CORBA technology. They came up with the conclusion that CORBA has a clear advantage in two aspects. First of all, they identified a significant number of major projects in various domains developed by companies around the world and using CORBA as an architecture framework. Although they have not been able to identify major project references for COM they strongly believe that there are significant COM-based development projects occurring at the desktop level. Another important factor of the maturity of CORBA is the knowledge acquired by the community of CORBA developers and according to Tallman and Kain, CORBA is at least two years in advance.

What has been observed by people working on some projects here at DREV, and using the COM technology (notably the Coyote Prototype project), the COM product available for the Unix platform was at least one or two versions behind the one available for the Windows platform. In the near future it is not quite obvious that COM versions on platforms other than Windows will always be late according to the last COM specification but it is certainly something that one must consider. According to Tallman and Kain, a likely scenario is that Microsoft will always offer a much more sophisticated set of COM features on its own platforms. This would guarantee second-class citizenship for all other implementations.

## 5.7 Server-side and Assessment Strategy

An interesting perspective examined by Pritchard [4] is the use of COM and CORBA on the server side of the organisation. Deciding where to use COM and CORBA depends on many factors such as platform, tools and high level services. In distributed object system environments, the *server side* is defined as everything but the client. The client that is generally called the thin-client has to deal with the presentation layer. The server that includes the middle-tier logic (business logic, business data, etc.) is spread across a diverse range of objects which may reside on several different physical locations. The server-side must then provide for high availability, performance, scalability and security in order to support different sets of clients. Using COM or CORBA architecture on the server side aims to build (COM/CORBA) middleware components that constitute the glue of the whole system. Pritchard does

not provide a direct answer on what is best or not from the server side but he underlines the risk of mixing multiple products from different CORBA vendors although a certain level of interoperability exists. The mix of COM and CORBA products is possible as long as the server-side system is partitioned into reasonable domains. If transaction boundaries or security contexts cross multiple domains it is suggested to use the same COM or CORBA middleware product across those domains.

Pritchard [4] proposes some criteria that could be used for determining when to use COM or CORBA. These criteria relate to three (3) main areas: Platform, Essential Services and Intangibles and are subdivided as follows:

PLATFORM

- Legacy system support
- Operating system support
- Programming language support
- Availability of development tools

ESSENTIAL SERVICES

- Transaction support
- Security and privacy
- Messaging support
- Distributed object management

INTANGIBLES

- Vendor perception
- Vendor commitment and viability
- Vendor lock-in
- Availability of product
- Availability of development staff

## 5.8 Other Considerations

The selection of either COM or CORBA, aside from the above requires the consideration of several additional points, for a specific project:

- Development effort
- Efficiency
- Size of code
- Speed, Network overhead
- Scalability
- Complexity
- Integration and management
- Security services
- Quality of Service
- Costs

# 6. Conclusion

This study was performed by looking at some of the fundamental traits of distributed object architecture. Technical details (also fundamental) such as object handling, marshalling/unmarshalling, data types, object creation and destruction, etc. have not been considered in the comparison. Both COM and CORBA address these technical details in different ways and again each has strengths and weaknesses. Given the time allowed for this study and the objectives of it, the parallel was limited to more general and high-level aspects. The next paragraphs summarise the comparison, followed by recommendations with respect to the specific WASP project.

The major weakness for Microsoft is its platform limitations. The position of Microsoft to support multiple platforms in the future is unknown. Another problem is that COM usage within Java requires usage of the Microsoft Java Virtual Machine that is not supported on non-Windows platforms. The final negative point is to rely on one single vendor for COM support i.e. Microsoft on Windows platform and Software AG on Unix platform.

CORBA provides cross-language, cross-platform, cross-vendor support for creating servers on a wide range of operating system platforms. One of the greatest strengths is that it is an open standard issued from the consensus of a large number of vendors. The fact that a large number of parties are involved in CORBA is also its major weakness. Specification proposals often can take twelve (12) months before getting approval and then it takes another six (6) months to develop products based on the specifications. Appropriate development component tools have not achieved a high level of maturity.

The study does not reveal a clear winner or loser. As stated before, each technology possesses some strengths that the other technology lacks. According to Pritchard [4], CORBA appears to be a good choice for creating middle-tier servers because middle-tier server development relies on a solid remoting architecture. In contrast, COM is more appropriate to create client-tier components, this is where CORBA is more deficient with less support and tools.

This comparison of COM and CORBA does not identify the technology that is most appropriate to a specific situation, as it is not a black and white situation. Hopefully, this document should help the reader to understand the main differences between COM and CORBA and should provide some guidelines with respect to selection criteria.

With respect to the Wing and Squadron Prototype and in a larger sense for all applications that agree with the Army Integrated Management Environment (AIME) and information exchange through the common ODS, the architecture for the construction of distributed objects should be based on CORBA.

AIME proposes a global architecture to improve data exchange amongst different DND entities through a common ODS (Object Data Store) based on ATCCIS (Army Tactical Command and Control Information System). For the purpose of AIME, a number of key technologies were identified to support the development, implementation and evolution of the ODS. The key technology elements identified are:

- Component Development approach;
- CORBA (used for the construction of business objects);
- XML (used for data transfer between the ODS and the application databases).

A complete description of the AIME architecture as well as the description of the technological foundation can be found in [6].

The coexistence of both technologies could be considered to leverage the advantages of both COM and CORBA or for any other valuable reasons. An approach which consists of creating components that bridge the two distributed object environments may present some advantages but there are also some disadvantages related to the potential costs associated with design, implementation, and maintenance of bridging components. Pritchard [4] considers that the trade-off between advantages and disadvantages is dependent on the number of bridging components that must be created. The OMG CORBA Architecture and Specification Version 2.4.2 document {Reference 7], Chapter 18, discusses CORBA – COM mapping.

As a final comment, during this study, more major applications successfully implemented with CORBA were discovered than with COM.

# 7. References


1. Component Object Model (COM), DCOM and Related Capabilities, Ed Morris, Emil Litvak http://www.sei.cmu.edu/str/descriptions

2. Microsoft Component Services, Server Operating System, A Technology Overview http://www.microsoft.com/com/wpaper

3. COM versus CORBA: A Decision Framework, Owen Tallman, J.Bradford Kain http://www.quoininc.com/quoininc

4. COM and CORBA Side by Side, Pritchard Jason, Addison Wesley, 1999

5. CORBA, Teach Yourself in 14 Days, Rosen Berger Jeremy, Ed Sam Publishing, 1998

6. Army Integrated Management Environment (AIME), DLIR, DND, Oct. 1999

7. http://cgi.omg.org/cgi-bin/doclist.pl

# List of symbols/abbreviations/acronyms/initialisms

| | |
|---|---|
| AIME | Army Integrated Management Environment |
| API | Application Program Interface |
| ATCCIS | Army Tactical Command and Control Information System |
| COM | Common Object Model |
| CORBA | Common Object Request Broker Architecture |
| DB | Database |
| DCE | Distributed Component Architecture |
| DCOM | Distributed Common Object Model |
| DND | Department of National Defence |
| IDL | Interface Definition Language |
| MIDL | Microsoft Interface Definition Language |
| MSMQ | Microsoft Message Queue |
| MTS | Microsoft Transaction Server |
| ODS | Object Data Store |
| OMG | Object Management Group |
| OSF | Open Systems Foundation |
| PC | Personal Computer |
| RPC | Remote Procedure Call |
| WASP | Wing and Squadron Planner |
| XML | Extensible Markup Language |

# Distribution list

**INTERNAL DISTRIBUTION**

| | |
|---|---|
| 1 | - Director General |
| 1 | - Deputy Director General |
| 1 | - Chief Scientist. |
| 6 | - Document Library |
| 1 | - Mr. V. Pille (Author) |
| 1 | - Mr. É. Dorion |
| 1 | - Dr. G. Vézina |
| 1 | - Mr. D. Gouin |
| 1 | - Mr. J.-C. St-Jacques |
| 1 | - Mr. R. Fortin |
| 1 | - Dr. A. Gibb |
| 1 | - Mr. D. Thibault |
| 1 | - Mr. G. Thibault |
| 1 | - Mr. A. Bergeron |
| 1 | - Mr. R. Charpentier |
| 1 | - Mr. D. Demers |
| 1 | - Dr. É. Bossé |
| 1 | - Mr. P. Labbé |
| 1 | - Mr. J. Roy |
| 1 | - Mr. R. Carling |
| 1 | - Mr. S. Paradis |
| 1 | - Mrs. M. Bélanger |
| 1 | - Mr. M. Blanchette |
| 1 | - Mr. J. Berger |
| 1 | - Maj M. Gareau |

## EXTERNAL DISTRIBUTION

1 - DRDKIM
1 - DRDKIM (unbound copy)
1 - DRDC
1 - DGRDP
1 - 1 Assoc DGRDP
1 - CLFCSC Library
1 - Canadian Forces College Library
1 - Cdn Div HQ
1 - DSTL
1 - DSTL 6
1 - DSTL 7
1 - DSTA
1 - DSTM
1 - DSTCCIS
1 - DREO
1 - DCIEM
1 - DREA
1 - DRES
1 - DDCEI
1 - DDCEI 2
1 - DDI
1 - DGIMOSD
1 - DIMSD
1 - DISOA
1 - DLCSPM
1 - DLCSPM 3
1 - DLCSPM 4
1 - DLIR 3
1 - DLFR 4
1 - DMSS
1 - DMSS8-2-5
1 - Gaétane Harvey, CGI

UNCLASSIFIED
SECURITY CLASSIFICATION OF FORM
(Highest Classification of Title, Abstract, Keywords)

## DOCUMENT CONTROL DATA

| 1. ORIGINATOR (name and address) | 2. SECURITY CLASSIFICATION (Including special warning terms if applicable) |
|---|---|
| Valdur Pille<br>DREV | Unclassified |

3. TITLE (Its classification should be indicated by the appropriate abbreviation (S, C, R or U)
Comparison of COM and CORBA for Command and Control Prototypes (U)

4. AUTHORS (Last name, first name, middle initial. If military, show rank, e.g. Doe, Maj. John E.)
Harvey, Gaetan
Valdur Pille

| 5 DATE OF PUBLICATION (month and year) | 6a. NO. OF PAGES | 6b .NO. OF REFERENCES |
|---|---|---|
| | 34 | 7 |

7. DESCRIPTIVE NOTES (the category of the document, e.g. technical report, technical note or memorandum. Give the inclusive dates when a specific reporting period is covered.)
Technical Report

8. SPONSORING ACTIVITY (name and address)
DAR
101 Col By Drive
Ottawa

| 9a. PROJECT OR GRANT NO. (Please specify whether project or grant) | 9b CONTRACT NO. |
|---|---|
| Project 13dm15 | W7701-9-3009/001/XSK |

| 10a. ORIGINATOR'S DOCUMENT NUMBER | 10b. OTHER DOCUMENT NOS |
|---|---|
| DREV TR-2000-219 | N/A |

11. DOCUMENT AVAILABILITY (any limitations on further dissemination of the document, other than those imposed by security classification)

- ☒ Unlimited distribution
- ☐ Restricted to contractors in approved countries (specify)
- ☐ Restricted to Canadian contractors (with need-to-know)
- ☐ Restricted to Government (with need-to-know)
- ☐ Restricted to Defense departments
- ☐ Others

12. DOCUMENT ANNOUNCEMENT (any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in 11) is possible, a wider announcement audience may be selected.)

13. ABSTRACT (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual).

This document contains the results of a brief analysis of the most popular technologies currently used with respect to component-based development over a distributed architecture. These technologies are COM (Common Object Model) from Microsoft and CORBA (Common Object Request Broker Architecture) from the OMG (Object Management Group), a consortium of more than eight hundred (800) companies. This study aims to highlight the strengths and weaknesses of each technology and should help to determine where to use COM and/or CORBA. This study was done in the scope of the Wing And Squadron Prototype project

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus, e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

Comparison
COM
DCOM
COM+
CORBA
Distributed Architectures
Component Services
Component Object Model