(CORBA - Comprehensive Report)

Name : Mootaz Medhat Ezzat Abdelwahab **Id :** 20206074
Program: Software Engineering **Group:** B (S4)
Delivered To: Prof. Fatma Omara

The evolution of distributed systems necessitated a standardized communication protocol. This led to the development of **CORBA**, or Common Object Request Broker Architecture, a middleware technology that has a significant role in the field of **distributed computing**. It enables communication between **objects** in a **distributed environment regardless** of their **implementation details** such as specific **operating systems**, **programming languages**, and **hardware platforms**. In this report, I will provide a comprehensive discussion on aspects related to CORBA, following the outlines below**:**

➢ Overview

• Definition & Purpose:

CORBA is a standard defined by the Object Management Group **(OMG)** designed to facilitate the communication between distributed objects across different platforms and programming languages.

• Main Features:

- **Supporting** multiple programming languages and platforms, promoting interoperability.
- **Facilitating** communication between objects residing on different machines through the use of Object Request Brokers **(ORBs)**.
- **Using IDL** to define interfaces that enable objects to communicate with each other.

➢ Architecture (Main Components)

The CORBA architecture consists of four main components**:**

• Object Request Broker (ORB):

The ORB acts as a mediator between clients and servers, handling communication and interaction between distributed objects. It provides services such as object activation, object location, and object persistence.

• Interface Definition Language (IDL):

The IDL is a language-independent specification used to define interfaces between objects. It serves as a bridge between different programming languages, allowing objects to communicate seamlessly.

• Object Services:

CORBA provides various object services, including naming, trading, security, event, and transaction services. These services enhance the functionality and reliability of distributed systems.

• Object Adapters:

Object Adapters are responsible for mapping the CORBA object model to the underlying programming language's object model. They provide a bridge between the ORB and the object implementation.

➢ How CORBA Works

• When a CORBA client wants to invoke a method on a CORBA object, it first obtains a reference to the object from the naming service. The client then sends a request to the ORB, specifying the object reference and the method to be invoked. The ORB marshals the request parameters and sends it to the CORBA server that hosts the object.

• The CORBA server unmarshals the request parameters and invokes the method on the CORBA object. The server then marshals the result parameters and sends them back to the ORB. The ORB unmarshals the result parameters and returns them to the client.

➢ Advantages (Pros)

• **Enabling** objects written in different programming languages to interact seamlessly, promoting interoperability **(Interoperability)**.
• **Supporting** the reuse of existing components, allowing organizations to leverage their investments in legacy systems **(Reusability)**.
• **Providing** a scalable architecture, allowing systems to handle increasing loads by distributing objects across multiple servers **(Scalability)**.
• **Allowing** objects to be deployed on different platforms, reducing the dependency on a specific operating system or hardware **(Portability)**.

➢ Limitations

• CORBA's extensive features and capabilities can make it complex to implement and maintain, requiring expertise and careful planning **(Complexity)**.
• CORBA's middleware layer introduces additional overhead, which may impact system performance compared to direct communication between objects **(Performance Overhead)**.
• Developers need to familiarize themselves with CORBA's concepts, IDL, and associated tools, which can require time and effort **(Learning Curve)**.

➢ CORBA Applications

Several companies and organizations have successfully utilized CORBA in their systems, reaping benefits such as improved interoperability and efficient communication between different platforms. As a simple example of using CORBA is a **distributed banking application**. The application could consist of two main components**:**

• Client Component (Runs on The User's Computer):
The client component could use CORBA to invoke methods on the server component to perform tasks such as checking account balances, transferring money, and withdrawing cash.

• Server Component (Runs on The Bank's Computer):
The server component could use CORBA to communicate with other components of the banking system, such as the database and the ATM network.

➢ CORBA & Other Distributed Computing Technologies

CORBA is not the only middleware technology available. Other popular middleware technologies include Java RMI, DCOM, and Web Services. However, CORBA remains a popular choice for developing distributed object applications, especially in enterprise environments due to its unique features and advantages, including support for multiple programming languages and its ability to facilitate communication between different platforms.