Common Object Request Broker Architecture (CORBA) es un estándar definido por Object Management Group (OMG) que permite que diversos componentes de software escritos en múltiples lenguajes de programación y que corren en diferentes computadoras, puedan trabajar juntos; es decir, facilita el desarrollo de aplicaciones distribuidas en entornos heterogéneos.

CORBA fue el primer producto propuesto por OMG. Su objetivo es ayudar a reducir la complejidad, disminuir los costes y acelerar la introducción de nuevas aplicaciones informáticas, promoviendo la teoría y la práctica de la tecnología de objetos en los sistemas distribuidos.

Es una tecnología que oculta la programación a bajo nivel de aplicaciones distribuidas. No obstante también brinda al programador una tecnología orientada objetos; las funciones y los datos se agrupan en objetos y estos objetos pueden estar en diferentes máquinas, pero el programador accederá a ellos a través de funciones normales dentro de su programa.

CORBA es más que una especificación multiplataforma, también define servicios habitualmente necesarios como seguridad y transacciones. Y así este no es un sistema operativo en sí, en realidad es un middleware.

Su primera versión se lanzó en 1991.

En 1995 aparece CORBA 2, con nuevas normas que permiten: que puedan cooperar implementaciones de diferentes fabricantes, que pueda ser implementado sobre cualquier nivel de transporte y que pueda funcionar en Internet sobre TCP/IP, creando un protocolo: IIOP (Internet IOP).

CORBA 3 se muestra en 2002, como intento de plantar cara a Microsoft y su modelo de programación de objetos distribuidos DCOM. Entre otras cosas, se introdujo el CORBA Component Model (CCM), con el que se pasó de un modelo de objetos distribuidos (EJB, restringido a Java) a un modelo distribuido orientado a componentes.

Características

Entre las principales características de CORBA nos encontramos con:

- Independencia en el lenguaje de programación y sistema operativo: CORBA fue diseñado para liberar a los ingenieros de las limitaciones en cuanto al diseño del software. Actualmente soporta Ada, C, C++, C++11, Lisp, Ruby, Smalltalk, Java, COBOL, PL/I y Python.
- Posibilidad de interacción entre diferentes tecnologías: uno de los principales beneficios de la utilización de CORBA es la posibilidad de normalizar las interfaces entre las diversas tecnologías y poder así combinarlas.
- Transparencia de distribución: ni cliente ni servidor necesitan saber si la aplicación está distribuida o centralizada, pues el sistema se ocupa de todo eso.
- Transparencia de localización: el cliente no necesita saber donde ejecuta el servicio y el servicio no necesita saber donde ejecuta el cliente.
- Integración de software existente: se amortiza la inversión previa reutilizando el software con el que se trabaja, incluso con sistemas heredados.
- Activación de objetos: los objetos remotos no tienen por qué estar en memoria permanentemente, y se hace de manera invisible para el cliente.
- Otras como: el fuerte tipado de datos, la alta capacidad de configuración, libertad de elección los detalles de transferencia de datos, o la compresión de los datos.

Elementos

La arquitectura CORBA está orientada a objetos. En CORBA los objetos poseen muchas de las características de otros sistemas orientados a objetos, como son la herencia en cuanto a interfaces y el polimorfismo. Pero lo más interesante de este aspecto, es la posibilidad de proporcionar estas características a lenguajes no orientados a objeto como C o COBOL, aunque CORBA trabaja más eficientemente con lenguajes orientados a objeto como son Java o C++.

Object Request Broker (ORB)

El Object Request Broker (ORB), es un componente fundamental de la arquitectura CORBA y su misión es facilitar la comunicación entre objetos. Éste se encarga de enviar las peticiones a los objetos y retornar las respuestas a los clientes que las invocan por el proceso de serialización.

El ORB tiene como principal característica la transparencia. Facilita la comunicación entre cliente y servidor ocultando:

- La localización de los objetos: El cliente no tiene porque saber donde se encuentra el objeto destino, puede estar en su propia máquina o en un proceso en una máquina remota.
- La implementación de los objetos: El cliente ignora el lenguaje de programación con el que se ha escrito el objeto remoto, su implementación o el sistema operativo en el que se encuentra.
- Estado de ejecución del objeto: Al enviar una petición sobre un objeto remoto, el ORB se encarga de inicializar el objeto en caso de ser necesario, antes de enviarle la petición.
- Mecanismos de comunicación de los objetos: El cliente no sabe qué mecanismos de comunicación utiliza el ORB para enviar las peticiones y retornar el resultado.

Interface Definition Language (IDL)

El lenguaje de definición de interfaz o IDL (Interface Definition Language), es un lenguaje que se utiliza para definir las interfaces entre los componentes de una aplicación y es el elemento que soporta la interoperabilidad de la tecnología. El IDL sólo puede definir interfaces, no implementaciones. Al especificar las interfaces entre objetos en CORBA, IDL es el encargado de asegurar la independencia del lenguaje de programación utilizado.

Para que esto sea posible, es necesario asegurar que los datos son correctamente intercambiados entre dos lenguajes distintos, para lo cual IDL define los tipos de datos de

una forma independiente del lenguaje con las correspondencias necesarias. Por ejemplo, un tipo *long* en C++ de 32 bits puede corresponderse con un *int* de Java. OMG ha definido bastantes correspondencias con lenguajes de programación populares, como: C, C++, COBOL, Java, Smalltalk o Ada.

Tendremos principalmente dos tipos diferentes de IDL en CORBA:

- *Stub* IDL: es la interfaz estática a los servicios declarados en las interfaces IDL, para el cliente todas las llamadas parecen locales. Actuará como proxy del objeto remoto, realizando la invocación de métodos remotos, incluyendo la serialización, la recepción de respuestas y la deserialización. El cliente puede tener tantos stubs como interfaces IDL existan. Es generado a partir del IDL en el lenguaje de programación del cliente (C, C++ Java, Smalltalk, Ada,…) por un compilador IDL.
- *Skeleton* IDL: es el representante estático del cliente en el servidor. Para el servidor todas las llamadas parecen locales. Es generado a partir del IDL por un compilador IDL y realiza ladeserialización de las invocaciones del cliente.

Objetos por referencia

La referencia de los objetos se obtiene a través de un campo Uniform Resource Identifier (URI) en formato cadena, una búsqueda NameServer (similar a la del Sistema de Nombres de Dominio (DNS)) o pasada como argumento durante la llamada a un método.

Los objetos referenciados son objetos mucho más livianos que implementan la interfaz del objeto real, ya sea remoto o local. Las llamadas a métodos por referencia consisten en una sucesión de llamadas al ORB que se encarga de bloquear los hilos y esperar por una respuesta, ya sea de éxito o fracaso. Los parámetros, información de retorno (en caso de haberla), y las excepciones son serializadas por el ORB internamente según el lenguaje de programación y sistema operativo.

Datos por valor

El IDL proporciona el lenguaje de definición de intercomunicación entre objetos o sistemas operativos. Los objetos de CORBA se pasan por referencia, mientras que los datos (*integers*, *doubles*,*structs*, *enums*, etc...) se pasan por valor. Esta combinación hace que podamos complementar datos fuertemente tipados al compilar clientes y servidores, y aun así preservar la flexibilidad que CORBA nos facilita.

Objects by value (OBV)

Además de objetos remotos, CORBA y RMI-IIOP definen el concepto de *Objects by value* u OBV (Objetos por valor) y *Valuetypes*. El código de los métodos de objetos *Valuetype* es ejecutado localmente por defecto. Si el OBV ha sido recibido por el lado remoto, el código necesario debe ser conocido a priori por ambas partes o descargado dinámicamente por el emisor. Para hacer que todo esto sea posible, el registro, definido en el OBV, contiene la base de código que contiene una lista, separada en diferentes máquinas, de URLs desde donde este código debe ser descargado. Además el OBV puede tener métodos remotos.

Los OBVs pueden tener campos que se transfieren cuando los OBVs son transferidos. Estos campos pueden ser los OBVs por sí mismos, formando listas, árboles o gráficos arbitrarios. Los OBVs poseen una jerarquía de clases, incluyendo herencia múltiple y clases abstractas.

CORBA Component Model (CCM)

CORBA Component Model o CCM (Modelo de Componentes de CORBA) es un añadido a la familia de definiciones de CORBA. Fue introducido en la versión CORBA 3 y describe un frameworkpara los componentes de CORBA. Aunque no depende de language dependent Enterprise Java Beans (EJB) es una forma más general de éste, proporcionando cuatro tipos diferentes de componentes en vez de los dos que el EJB define. Además proporciona la abstracción de entidades que pueden proveer y aceptar servicios a través de unas interfaces propias bien definidas llamadas ports (puertos).

El CCM tiene un contenedor de componentes donde los componentes software pueden ser depositados. El contenedor ofrece una serie de servicios que los componentes pueden usar. Estos servicios incluyen (aunque no están limitados solo a éstos) notificación, autenticación, persistencia y proceso de transacciones. Son los servicios más utilizados que cualquier sistema distribuido necesita y, moviendo la implementación de estos servicios de los componentes del software al contenedor de componentes, la complejidad de los componentes se reduce drásticamente.

Interceptores portables

Los interceptores portables (portable interceptors) son los ganchos usados por CORBA y RMI-IIOP para proporcionar una de las funciones más importantes del sistema CORBA, que define los siguientes tipos de interceptores:

- Los interceptores de Referencia del Objeto Remoto (IOR) permiten la creación de las nuevas referencias de los objetos remotos presentes en el actual servidor.
- Los interceptores clientes proporcionan la llamada a métodos remotos desde el lado del cliente. Si el objeto servicio existe en el mismo servidor donde el método es invocado, además permiten llamadas locales.
- Los interceptores servidor proporcionan la tramitación de las llamadas de métodos remotos en el lado del servidor.

Los interceptores pueden incluir la información específica de los mensajes al ser enviados y los IORs al ser creados. Esta información puede ser leída más adelante por el interceptor correspondiente en el lado remoto. Los interceptores pueden a su vez lanzar excepciones de reenvío, redirigiendo la petición a otro objetivo.

General InterORB Protocol (GIOP)

El GIOP es un protocolo abstracto por el cual los ORBs se comunican. Los estándares asociados con el protocolo son mantenidos por OMG. La arquitectura de GIOP proporciona una serie de protocolos entre los que se encuentran:

- Internet InterORB Protocol (IIOP): implementación de GIOP para su uso por internet, proporcionando una interfaz entre los mensajes GIOP y la capa TCP/IP.
- SSL InterORB Protocol (SSLIOP): protocolo IIOP sobre SSL, proporcionando encriptación y autenticación.
- HyperText InterORB Protocol (HTIOP): protocolo IIOP sobre HTTP, proporcionando transparencia remota.
- Zipped IOP (ZIOP): una versión comprimida de GIOP que reduce el uso de ancho de banda.

Localizaciones de CORBA (CorbaLoc)

Las localizaciones de CORBA (CorbaLoc) hacen referencia a un objeto referenciado CORBA que tiene una estructura similar a la de una URL.

Todos los productos CORBA deben soportar dos URLs definidas: CorbaLoc y CorbaName. El propósito de ambas es proporcionar una forma que pueda ser leída y editada por el ser humano para especificar la localización donde el IOR puede ser obtenido. Un ejemplo de CorbaLoc se puede ver más abajo:

```
corbaloc::160.45.110.41:38693/StandardNS/NameServer-POA/_root
```

Un producto CORBA puede opcionalmente soportar el uso de formatos HTTP, FTP y FILE. La finalidad de estos es proporcionar detalles sobre cómo descargar los campos IOR.

Cómo funciona

Todos los componentes CORBA son objetos, cada cual tendrá una interfaz y una identidad única, cada uno de los objetos se podrá implementar en un lenguaje de programación distinto y ejecutarse sobre cualquier sistema operativo.

El servidor crea objetos remotos, hace accesibles referencias a esos objetos remotos y espera a que los clientes invoquen a estos objetos o a sus métodos. Por otro lado el cliente

obtiene una referencia de uno o más objetos remotos en el servidor e invoca a sus métodos.

La manera de realizar la invocación por parte del cliente es usando stub, una interfaz de comunicación con el servidor generada a partir del IDL, usando la invocación dinámica para acceder a los objetos del servidor gestionando a su vez las excepciones producidas por su llamada.

Necesita para ello una referencia del objeto remoto o IOR (Interoperable Object References), el tipo del objeto y el nombre de la propia operación que desea invocar. Describiendo las interfacesIDL, un ORB genera automáticamente código en el lenguaje seleccionado para realizar la integración de las aplicaciones distribuidas. Evidentemente, puesto que sólo describe interfaces, todas las tareas complejas relacionadas con los lenguajes de programación, como control de flujo, gestión de memoria, composición funcional, no aparecen en IDL.

El ORB, a partir de la petición del cliente encuentra el código de la implementación apropiada y transmite los parámetros, el control a la implementación de la interfaz a través del *skeleton* IDL e informa de excepciones en el proceso (como referencias IOR o IDL no válidas).

Para recibir la petición, recibe la invocación de uno de sus métodos como llamadas desde el ORB hacia la implementación de la interfaz, la llamada puede venir de un cliente que haya utilizado los stubs IDL; los esqueletos de la interfaz son específicos de cada interfaz y del adaptador de objetos que exista en la implementación de CORBA. Una vez completada la invocación el control y los valores de retorno son devueltos al cliente.

Puede utilizar los servicios que proporciona el adaptador de objetos e incluso que proporciona el ORB, mientras es procesada la petición que ha recibido el cliente, para ese caso puede elegir un adaptador de objetos entre un conjunto de ellos, para tomar esa decisión se basa en la clase de servicios que pueda requerir la implementación de la interfaz.

Problemas y críticas

Aunque CORBA prometió mucho en cuanto a la forma de codificar y construir software, ha sido objeto de muchas críticas.

Algunos de los fallos se deben a la implementación y el proceso por el que CORBA fue concebido como un estándar y los problemas que se surgieron en referencia a la política de negocio al implementar un software estandarizado. Éstos derivaron en un rechazo en el uso y adopción de CORBA en nuevos proyectos y áreas.

Implementación e incompatibilidades

La especificación inicial de CORBA definía tan solo el IDL, y no la interoperabilidad entre los lenguajes. Esto significa que la compatibilidad del código fuente no era mejor que la que había disponible en el momento. Con CORBA 2 y posteriores, este problema se resolvió.

Transparencia en la localización

La noción de CORBA en cuanto a transparencia ha sido motivo de crítica, y esto es debido a que los objetos que residen en el mismo espacio de direcciones y que son accesibles con una simple llamada a una función son tratados como objetos que residen en cualquier parte del sistema distribuido. Esto hace que los accesos locales sean tan complicados como serían en el escenario remoto más complejo. Aunque cabe decir que CORBA no ofrece restricciones en la complejidad de las llamadas.

Deficiencias en el diseño y proceso

La creación del estándar CORBA es citado a menudo por su proceso de diseño por comité. No hubo proceso de arbitraje entre las propuestas conflictivas o de decisión de la jerarquía de los problemas a resolver. Así, el estándar se creó teniendo en cuenta todas las propuestas sin importar la coherencia entre éstas. Esto hizo que la especificación fuera muy compleja, cara de implementar y a menudo ambigua.

Problemas con las implementaciones

A lo largo de su historia, CORBA se ha visto afectada por las deficiencias en sus implementaciones. Ha habido pocas implementaciones que hayan incluido todos los elementos críticos de la especificación, y las implementaciones existentes estaban incompletas o eran inadecuadas. Al no existir requisitos a la hora de proponer nuevas características, los miembros incluían propiedades que no habían sido probadas nunca en cuanto a usabilidad o implementación. A su vez las implementaciones se vieron obstaculizadas por la tendencia general para detallar las normas y la práctica común de comprometerse a adoptar todas ellas, lo que normalmente generaba APIs incoherentes y de difícil utilización. En la actualidad este punto ha cambiado considerablemente a mejor y podemos encontrar implementaciones de CORBA de mejor calidad.

Cortafuegos

CORBA (más precisamente, IIOP) utiliza conexiones TCP/IP crudas con el fin de transmitir los datos. Sin embargo, si el cliente está detrás de un cortafuegos muy restrictivo o un entorno con servidor proxy transparente que sólo permite conexiones HTTP hacia el exterior por el puerto 80, la comunicación puede llegar a ser imposible, salvo que el servidor proxy en cuestión permita el método CONNECT HTTP o conexiones con sockets. Hubo un momento, en el que era difícil hasta forzar implementaciones para utilizar un solo puerto estándar, en vez de esto tenían que escoger varios puertos al azar. Hoy en día, los ORBs actuales no tienen estas deficiencias. Debido a estas dificultades, algunos usuarios han preferido el uso de los servicios web en lugar de CORBA.

Estos se comunican utilizando XML/SOAP través del puerto 80, que normalmente se deja abierto o se filtra a través de un proxy HTTP dentro de la organización, para la navegación web a través de HTTP. Implementaciones CORBA recientes, sin embargo, soportan SSL y pueden ser fácilmente configurados para trabajar en un solo puerto. La mayoría de los ORBs de código abierto más populares como son TAO y JacORB, soportan también GIOP bidireccional, lo que da a CORBA la ventaja de ser capaz de utilizar la comunicación de retorno de llamada en lugar del enfoque de sondeo característica de

implementaciones de servicios web. A su vez, cada vez se comercializan más cortafuegos que soportan CORBA sin problemas.