TECNOLOGÍA CORBA (Common Object Request Broker Architecture)

Crisman Martínez Barrera*

A todos, gracias por sus pensamientos positivos y oraciones inconmensurables. Brachel.

Este artículo pretende introducir al lector en el mundo de una de las tecnologías de punta contemporáneas, mediante la presentación de las características del estándar de la plataforma Corba, que se ha convertido en soporte a multitud de aplicaciones abiertas y es un punto de referencia inevitable para la intercomunicación entre componentes de software heterogéneos. "CORBA es el proyecto de middleware *más importante y ambiciosos emprendido por la industria hasta el momento".*

This article introduces the reader to the universe of CORBA platform, a leading technology that has become «support for a multitude of open applications» and the inevitable reference point for the intercommunication between the components of heterogeneous software. "CORBA is the most important and ambitious middleware project that the industry has undertaken up to now. Surf the Web without missing calls! Get MSN Broadband".

* Ingeniero de Sistemas y profesor de la Escuela de Ingeniería de la Universidad Central, Bogotá, coordinador del grupo de computación móvil. Candidato a Magíster en Teleinformática de la Universidad Distrital Francisco José de Caldas (Bogotá) e integrante del grupo de investigación de agentes de software móviles. Analista de sistemas y consultor de tecnologías de punta. www.mcrisman.telecomunicaciones.com

Introducción

Actualmente las telecomunicaciones son uno de los sectores más activos y con tasa más alta de crecimiento, principalmente en los países desarrollados. Colombia podría dar un salto tecnológico pasando directamente a las nuevas tecnologías, si se implementaran soluciones de hardware y software que permitieran la integración de sistemas más recientes.

El software tiene un nuevo enfoque: el desarrollo de componentes, que depende de la capacidad de integración para comunicarse entre ellos según las interfaces estandarizadas. Las especificaciones de estandarización son descritas por CORBA, que permite el desarrollo de programas de software fácilmente expansibles, reemplazables y que es el inicio para "conectar todo lo que hay en el mundo a Internet[1]", sin poner en riesgo la funcionalidad de los elementos y las aplicaciones en su totalidad.

Este artículo pretende introducir al lector en el mundo de una de las tecnologías de punta, mediante la presentación de las características del estándar de la plataforma CORBA, que se ha convertido en "soporte a multitud de aplicaciones abiertas"[2] y es un punto de referencia inevitable para la intercomunicación entre componentes de software heterogéneos. "*CORBA es el proyecto de middleware más importante y ambicioso emprendido por la industria hasta el momento*"[3]

Aproximación a la Tecnología CORBA

Los seres vivos desde su aparición hasta nuestros días buscan comunicarse con su propia especie. Los procesos de intercambio de emociones, símbolos, ideas, ilusiones, creencias, conquistas, temores, avisos o sueños son emitidos al receptor. Cuando emisor y receptor intercambian información se está utilizando un conjunto de reglas y símbolos preestablecidos, los cuales gobiernan la comunicación.

Así, el ser humano modelando el comportamiento y los procesos involucrados en la comunicación de los seres vivos logra que los computadores y otros dispositivos puedan intercambiar información en todos los niveles.

Para lograr comunicar dos dispositivos del mismo tipo se debe conocer el idioma (sistema operativo) que manipulan cada uno de ellos. Si el emisor desea intercambiar símbolos (etc.) con otro ser humano (receptor) que no habla el mismo idioma tiene dos alternativas de solución:

- Que emisor o receptor aprendan el otro idioma
- Que emisor y receptor utilicen un intermediario que domine los dos idiomas

Si emisor, receptor o intermediario conocen dos idiomas, están manipulando perfectamente las reglas que gobiernan dicha comunicación, lo que se conoce como estándar de comunicación para los componentes de una red que deseen intercambiar información de diferente tecnología y diferente proveedor.

CORBA es una arquitectura de comunicaciones que soporta la construcción e integración de tecnologías de diferente fabricante independientemente del tiempo de creación, así como pueden intercambiar información personas que dominan diferente idioma, sin importar que no sea usado actualmente.

En el futuro podrán comunicarse diferentes tipos de seres vivos, así como trasladar todo a Internet.

Qué es CORBA

CORBA provee una infraestructura que permite la comunicación de objetos independientes de plataforma y de implementación. Uno de los componentes garantiza la portabilidad e interoperabilidad de objetos sobre redes de comunicaciones y sistemas heterogéneos[4].

Es una especificación definida por el OMG (*Object Management Group*) para la creación y uso de objetos remotos, cuyo objetivo es proporcionar interoperabilidad entre aplicaciones en un entorno distribuido y heterogéneo. Es conocido como un tipo de "*middleware*", ya que no efectúa las funciones de bajo nivel necesarias para ser considerado un sistema operativo. A pesar de que debe funcionar sobre sistemas operativos tradicionales, efectúa muchas de las operaciones que tradicionalmente se han considerado del

dominio de los sistemas operativos para entornos distribuidos[5].

CORBA es "Una arquitectura de negociación de petición de objetos comunes y que podrían ser utilizadas en capas superiores de la Red de Gestión de Telecomunicaciones (RTG) influidas fuertemente por las funciones propuestas en la industria de la información. La gestión integrada de las redes de telecomunicación tradicionales y las redes basadas en el IP son fundamentales para la creación de un marco de referencia que sirva para la gestión unificada de redes de conmutación de circuitos y redes de conmutación de paquetes constitutivos para una misma estructura".[6]

Luis Sierra afirma que CORBA *no* es una tecnología particular de Java. Es la arquitectura estándar de OMG para procesamiento distribuido. El funcionamiento es parecido a RMI (Remote Method Invocation)[7].

Desde mi punto de vista, CORBA es una arquitectura de comunicaciones entre sistemas heterogéneos que soporta construcción e integración de tecnologías de diferente fabricante. Puede agrupar antiguas y nuevas aplicaciones de software. Está basada en un gestor de peticiones a objetos comunes y permite interoperabilidad entre aplicaciones en máquinas remotas en un entorno distribuido. Es una plataforma que tiene funcionalidad de sistema abierto y que requiere para cada lenguaje soportado una interfaz estandarizada entre CORBA y la herramienta de programación.

Para construir componentes que utilicen el entorno CORBA se deben seguir los siguientes pasos:

1. Definir la interfaz remota. Se define, en primer lugar, la interfaz del objeto remoto en IDL. Dicha interfaz permitirá generar, de manera automática, el código fuente del *stub* y el *skeleton* así como todo el código necesario para comunicarse con el ORB. Si sólo se implementa el cliente porque el servidor ya existe, se tendría que proporcionar el fichero IDL correspondiente a la interfaz que expone el servidor.

2. Compilar la interfaz remota. El compilador genera todo el código fuente mencionado en el paso anterior.

3. Implementar el servidor. A partir de los esqueletos que genera el compilador *idl* es sencillo implementar el servidor. Además de los métodos que implementan la interfaz remota, el código del servidor crea un mecanismo para arrancar el ORB y esperar por la invocación de un cliente.

4. Implementar el cliente. De una manera similar al servidor, el cliente hace uso de los stubs generados en el paso 2. El cliente se basa en el stub para arrancar su ORB, encontrar el servidor utilizando el servicio de nombrado, obtener una referencia al objeto remoto e invocar sus métodos.

5. Arrancar los programas. Una vez está todo implementado, se arranca el servicio de nombrado, el servidor y finalmente, el cliente.

1. Arquitectura de Corba

Para que el cliente pueda realizar una invocación sobre un objeto, se debe tener una referencia del objeto (IOR) y conocer el tipo de objeto y la operación que desea invocar. El cliente puede iniciar la petición a través de una conexión IDL o bien construyendo la invocación de forma dinámica utilizando el DII. El ORB se encarga de encontrar el código de la implementación apropiada, transmitir los parámetros y transferir el control a la Implementación de la Interfaz a través del esqueleto IDL, o a través del esqueleto dinámico (DII) como se explica más adelante.

Las invocaciones pueden producir excepciones de diversa índole. Por ejemplo la referencia al objeto puede ya no ser válida, o la interfaz IDL del objeto ha podido cambiar. El ORB se encargará de informarnos de todas estas posibles excepciones y nuestro código deberá estar preparado para gestionar estas excepciones.

A continuación se describe cada una de las características fundamentales de la Arquitectura CORBA (figura 1):

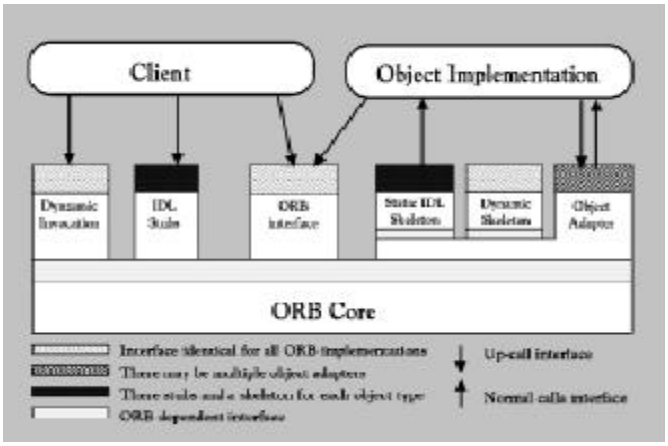

Figura 1. Arquitectura CORBA

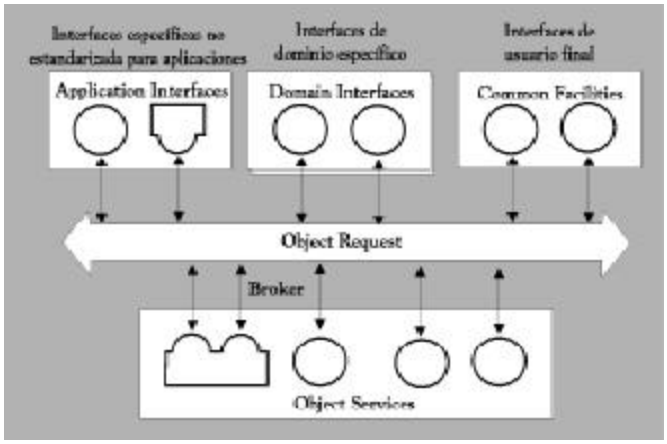

Figura 2. El ORB de CORBA

1.1. Objetos CORBA

Las implementaciones de los objetos reciben las invocaciones como llamadas hacia arriba (up-call), desde el ORB hacia la Implementación de la interfaz. La implementación de la interfaz puede elegir un adaptador de objetos entre un conjunto de ellos, una decisión que estará basada en la clase de servicios que pueda requerir dicha implementación.

Los objetos CORBA se diferencian de los objetos de los lenguajes habituales de programación en que[89]:

- Pueden estar localizados en cualquier lugar de la red.
- Pueden ejecutarse en cualquier plataforma de hardware y de sistema operativo.
- Pueden estar escritos en cualquier lenguaje.
- Pueden tener la capacidad de detectar el entorno, procesar información y además tienen la capacidad de comunicación.

1.2. ORB object request broker

Componente que permite que clientes y objetos puedan comunicarse en un ambiente distribuido como se muestra en la figura 1. Y que contempla cada una de las interfaces que el ORB manipula (figura 2).

El bus de objetos es el intermediario entre clientes y servidores que transmite las peticiones cliente-servidor y las respuestas servidor-cliente. Se necesita un ORB en cada máquina. El ORB soporta cuatro tipos de interfaces de objetos:

- Object Services: Son interfaces para servicios generales. Son usadas en cualquier programa basado en objetos distribuidos.
- Common Facilities: Son interfaces orientadas al usuario final y que se programan por la aplicación específica.
- Domain Interfaces: Son interfaces de dominio específico para las aplicaciones.
- Application Interfaces: Este tipo de interfaz acepta interfaces que no sean estandarizadas y se utilizan en aplicaciones específicas.

1.3. El adaptador de objetos (OA)

El adaptador de objetos (OA) como se muestra en la figura 1, es el módulo que permite a las implementaciones de los objetos acceder a servicios ofrecidos por el ORB, éste genera las referencias a los objetos. El adaptador de objetos exporta una interfaz pública para su uso por la implementación del objeto y una interfaz privada para ser usada por el esqueleto del objeto que depende de la implementación del adaptador de objetos (figura 3).

Las funciones que realiza este adaptador son:

- Generación e interpretación de las referencias a objetos.

- Invocación de métodos.
- Seguridad en las interacciones.
- Activación y desactivación de objetos e implementaciones.
- Traducción de referencias a objetos con sus correspondientes implementaciones.
- Registro de las implementaciones.

Debido a que las implementaciones de los objetos dependen del adaptador de objetos, se deben definir la menor cantidad de adaptadores de objetos.

1.4. IDL (Interface Definition Language)

Para poder especificar los servicios que ofrecen los objetos que forman parte de un sistema abierto y distribuido, se necesita contar con algún lenguaje preciso, bien definido, e independiente de cualquier posible representación de los datos o estructuras que él define, así como la futura implementación de los objetos que especifica. La norma ISO/IEC 14750 (ITU-T X.920) define dicho lenguaje, al que se conoce como *lenguaje de definición de interfaces* de ODP, o ODP IDL por su acrónimo en inglés. Su principal objetivo es describir la signatura de los objetos que especifica, en términos de las estructuras de datos que se manejan y el perfil de las operaciones que definen sus servicios. De esta forma se consigue la ocultación necesaria para el desarrollo de aplicaciones abiertas[10].

En IDL, una interfaz es una descripción de un conjunto de posibles operaciones que un cliente puede solicitar de un objeto. El objeto satisface una interfaz si este puede satisfacer una solicitud de otro objeto. La interfaz provee mecanismos compuestos que le permiten a tal objeto soportar múltiples interfaces.

Las operaciones que se realizan denotan servicios que pueden ser atendidos y ejecutados para cambiar de valor y adquirir un valor. Una operación es reconocida por un identificador de operación. Una operación no es un valor.

Los tipos de datos que manipula CORBA en IDL son:

- Tipos básicos : long, short, ushort, ulong, float, double char, boolean, enum, string, octect, any
- Tipos compuestos: struct, union, array
- Tipos derivados: sequence <*tipo*>
- Tipos de objeto: interface, referencia a objetos

Un tipo es una entidad con predicados asociados y definidos con valores en un objeto. Un valor satisface un tipo si el predicado es verdadero para la variable.

Los tipos son usados para restringir los posibles valores, parámetros, o para identificar un posible resultado.

La Interfaz IDL se compone del repositorio de interfaces y la interoperabilidad de la Interfaz de invocación dinámica:

- El repositorio de interfaces

El repositorio de interfaces (IR) es un servicio que ofrece objetos persistentes que representan la información IDL de las interfaces disponibles en CORBA, de una forma accesible en tiempo de ejecución (runtime). Esta información puede ser utilizada por el ORB para realizar peticiones. Y además, el programador de aplicaciones puede utilizar esta información para acceder a objetos cuya interfaz no se conoce en tiempo de compilación, o para determinar que operaciones son válidas en un objeto.

- La interfaz de invocación dinámica

El DII (Dynamic Invocation Interface) es una interfaz que nos permite la construcción dinámica de invocaciones para un determinado objeto. Ello garantiza que el cliente pueda especificar el objeto, la invocación y los parámetros que se pasan al servidor. La invocación es idéntica a la que llega a través de la interfaz estática pero que ya dentro del cliente, logra una flexibilidad fundamental en arquitecturas complejas y dinámicas.

Una invocación dinámica se compone, de una referencia al objeto, una operación y una lista de parámetros. Todos estos datos se obtienen del Repositorio de Interfaces (IR).

1.5. Stub

Es el intermediario entre el cliente y el ORB (figura 1). El Stub recoge del cliente llamadas a métodos y las transmite al ORB. Se requiere una clase de stub por cada clase remota (ver detalles en la figura 3).

Además, es un componente que actúa como servidor, puede estar ejecutándose en cualquier máquina conectada a la red que recibe peticiones por parte de clientes que pueden ser locales o remotos. Indistintamente de ello, el cliente siempre tendrá la ilusión de que la llamada se ejecuta localmente. En otras palabras el stub logra que el programador no se ocupe de las instrucciones de programación remotas ya que son objetos que residen en el cliente y que representan objetos remotos instalados en un servidor. En él se identifica: Host, puerto e identificador del objeto.

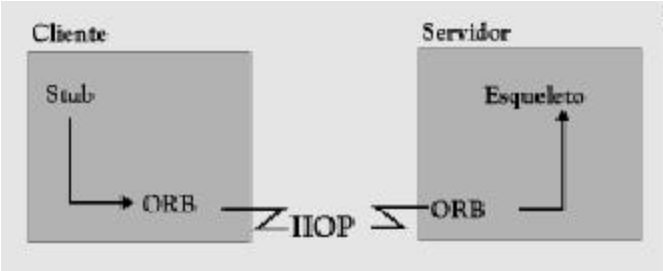

Figura 3. Ubicación del Stub

1.6. Esqueleto

Es el intermediario entre ORB y los objetos del servidor (figura 1). Recibe llamadas del ORB y ejecuta los métodos correspondientes en el servidor sobre el objeto que corresponda. Cuando el cliente establece un objeto local (con servicio remoto), la petición se realiza por intermedio del protocolo de comunicaciones IIOP a través del ORB. El servidor recibe la petición, busca el objeto definido (compara el esqueleto del método en el módulo esqueleto) lo ejecuta y retorna la respuesta al cliente (figura 3).

2. Ventajas al utilizar CORBA

- Heterogeneidad

Un sistema heterogéneo consiste en conjuntos de elementos interconectados de hardware y software de diferente fabricante y que puede integrar aplicaciones de diferente tecnología (figura 4).

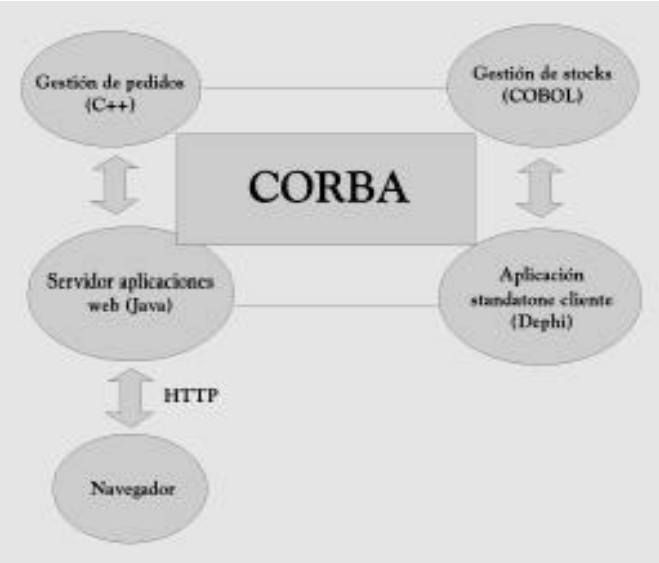

Figura 4. Integración de aplicaciones

La infraestructura de sistemas de información antiguos que poseen las compañías no son fácilmente reemplazable, debido al costo de desarrollo y al tiempo de implantación, una de las mejores alternativas es integrar antiguas tecnologías con nuevas para así obtener un completo beneficio.

- Movilidad[11]

La migración de procesos en sistemas distribuidos tradicionales es muy útil para mejorar el reparto de carga de los diferentes computadores. Tiene como fin garantizan el rendimiento global y ciertas restricciones de administración o seguridad.

- Eficiencia

  - La red lleva menos mensajes.
  - El servidor realiza más trabajo.
  - Se evita la latencia/inestabilidad de la red en los procesos.

- Adaptación al cliente

  - El cliente puede extender la funcionalidad del servidor.
  - Fácil instalación para el usuario.

- No se requiere instalación de servidor.
- No se acuerdan los procedimientos entre los clientes y los servidores.
- Instalación dinámica de los procedimientos del cliente en el servidor.

• Tiempo de desempeño

Además, la ejecución asíncrona permite que los procesos controlen la gestión y terminación de tarea y que el cliente pueda finalizar o continuar haciendo otras cosa en su sistema, por otro lado se reduce el tráfico en la red y la capacidad de cómputo del cliente (figura 5).

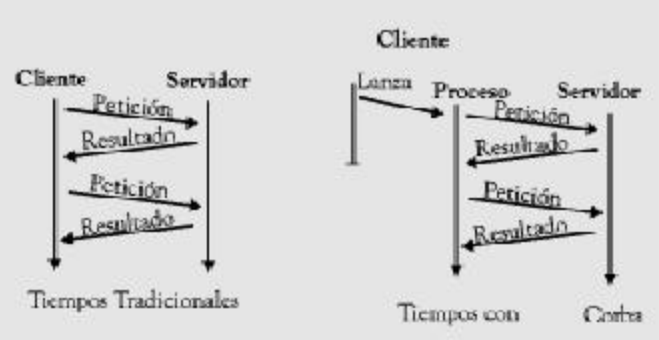

Figura 5. Tiempos en la red utilizando CORBA

• Robusto

- Reducción de la dependencia de la disponibilidad de la red y del cliente/servidor.

- Los procesos migrados al sistema servidor no se ven afectados por los fallos del cliente o de la red.

- Los procesos se ejecutan realizando tareas específicas en lugares diferentes.

- Automatización de las tareas distribuidas.

3. Desventajas al utilizar CORBA

El problema fundamental de los sistemas de integración es el software. Aún no existe mucha experiencia en el diseño, implantación y uso de software como CORBA. Precisamente, éste es un campo de investigación actual.

Las redes son indispensables para la comunicación entre máquinas; sin embargo, pueden plantear problemas de saturación, embotellamiento, interrupción o pérdidas de mensajes.

El posible acceso a todo el sistema por parte de los usuarios plantea el inconveniente de la necesidad de un sistema de seguridad adecuado y estándar, aunque CORBA maneja la seguridad.

Conclusiones

CORBA proporciona una infraestructura y un modelo común desde donde los requisitos expresados en diferentes lenguajes (las diferentes metodologías de desarrollo), pueden ser integrados para formar un sistema globalmente consistente.

CORBA ofrece un conjunto de mecanismos muy útiles a la hora de desarrollar aplicaciones distribuidas, junto con un soporte tecnológico suficientemente maduro como para construir aplicaciones robustas, eficientes y competitivas, a la vez que integrables con otros sistemas que cumplan estos estándares.

Los sistemas que son desarrollados con tecnologías antiguas pueden ser integrados con las nuevas a través de CORBA. Esto es, construyendo interfaces para que intercambien información local o remota a través de la red para resolver problemas en forma parcial e incremental.

Ya, algunas tecnologías incorporan interfaces para intercambiar información a través de CORBA, así como desarrollos adicionales que facilitan la integración de servidores y clientes con filosofía CORBA.

Java como herramienta también integra interoperabilidad con CORBA siempre y cuando los objetos estén usando un ORB compatible con las especificaciones y que se apoyen con IIOP como protocolo de comunicaciones.

Finalmente, el protocolo de comunicación IIOP, establecido por la especificación CORBA para la

interoperabilidad entre distintas plataformas, se ha convertido en el protocolo por defecto utilizado en los estándares para asegurar la interoperabilidad entre todos los sistemas.

4. Glosario

Aplicación

Suministra uno o más programas, es una colección de objetos que interactúan para realizar un objetivo común.

Estado

Corresponde a la situación previa y actual que determina el comportamiento futuro de la información.

IDL(Interface Definition Language)

Lenguaje de programación de forma independiente para especificar objetos de interfaz.

IIOP (internet-inter-ORB Protocol)

Protocolo de comunicaciones que está diseñado para permitir la interacción entre ORB.

Interfaz

Descripción de un conjunto de posibles usos de un objeto. Una interfaz describe un conjunto de respuestas potenciales en el cual un objeto puede participar. Es el dispositivo o elemento que comunica dos entornos que operan con diferente lenguaje.

Interoperabilidad:

Habilidad para intercambiar peticiones y respuestas. Un objeto es interoperable si los métodos ofrecen y/o evalúan servicios de otros.

ITU-T

International Telecommunication Union. Unión Internacional de Telecomunicaciones, también conocida como CCITT.

Método

Código desarrollado en un lenguaje de programación orientado a objetos que puede ser ejecutado para realizar un objetivo. Componente de una clase.

Objeto

Combinación de estados y conjunto de métodos que personifican las características abstractas y el comportamiento de cualquier cosa. Un objeto es una instancia de una clase.

ODP (Open Distributed Processed)

Procesamiento abierto y distribuido. Modelo de referencia que proporciona normas para el desarrollo de aplicaciones abierta.

ORB(Object Request Broker)

Provee la forma para que cualquier objeto reciba peticiones y ofrezca respuestas.

Sistema distribuido

Formado por un conjunto de elementos de computador autónomos unidos por una red de comunicaciones y equipados con software que soporten el intercambio de componentes.

Sistema operativo

Puede ser definido como aquella parte del sistema que da vida al hardware. El desarrollo de los sistemas operativos va siempre detrás del desarrollo del hardware, pero permiten mejorar el rendimiento de este y en el peor de los casos, ocultarán todas sus particularidades.

Citas

1 Mattern Friedemann. Instituto Federal de Tecnología Suiza. Revista Novotica No. 15. Sept de 2001.

2 Documento 8-S. Unión Internacional de Telecomunicaciones. Oficina de Desarrollo de las Telecomunicaciones. 14 de noviembre de 2000. Reunión preparatoria regional para la conferencia mundial de desarrollo de las telecomunicaciones Sofía (Bulgaria), 28-30 de noviembre de 2000.

3 *Castells, Pablo*. *Programación orientada a objetos*. *E*.T.S. Informática, Universidad Autónoma de Madrid. 23 de mayo de 2002.

4 A Discussion of the Object Management Architecture. Copyright 2000, Object Management Group., Inc. (OMG). Año 2000.

5 García Álvarez, Fernando. *Objetos distribuidos y agentes móviles*. Universidad de Oviedo Departamento de Informática, junio 2001.

6 Documento 8-S. Unión Internacional de Telecomunicaciones Oficina de desarrollo de las Telecomunicaciones. 14 de noviembre de 2000. Reunión preparatoria regional para la Conferencia Mundial de Desarrollo de las Telecomunicaciones Sofía (Bulgaria), 28-30 de noviembre de 2000.

7 Sierra, José Luis. *Laboratorio de Programación III*. Curso 2001 – 2002

9 García Álvarez, Fernando. *Objetos distribuidos y agentes móviles*. Universidad de Oviedo. Departamento de Informática, junio 2001

10 Vallecillo Moreno, Antonio. RM-ODP: *El modelo de referencia de ISO para el Procesamiento abierto y distribuido.* Año 2000.

11 García Álvarez, Fernando. *Objetos distribuidos y agentes móviles*. Universidad de Oviedo. Departamento de Informática, junio 2001.

Bibliografía

Anónimo. *Introducción a las tecnologías e integración de aplicaciones*. Agosto -1999.

H. KILOV, B. Rumpe, I. Simmonds (Eds.). *Behavioral Specifications of Business and Systems*. Kluwer Academic Publishers, 1999.

MESTRAS PAVÓN, Juan. *Agentes móviles*, *Departamento de Sistemas Informáticos y Programación Universidad Complutense Madrid*. 2000.

Normas de estandarización de sistemas e integración de componentes.

ODP-protocol Support for Computational Interactions (ISO/IEC 14752; ITU-T X.931)

ODP-Type Repository Function (ISO/IEC 14769; ITU-T X.960)

ODP-Reference Model: Enterprise Viewpoint (ISO/IEC 15414; ITU-T X.911)

ODP-Reference Model: Quality of Service (ISO/IEC 15935; ITU-T X.905)

http://www.cs.wustl.edu/~schmidt/corba-products.html

Productos comerciales de CORBA.

http://adams.patriot.net/~tvalesky/freecorba.html

Productos libres de CORBA.

http://www.itu.int

Unión Internacional de Telecomunicaciones.

Autorretrato en los Alpes de Carabaya. Puno, 1930