Una Nueva Arquitectura para la Integración de CORBA y OODB

Discusión del Artículo

Helios Alonso - Luis Pandolfi - Pablo Rodríguez Bocca
Universidad de la República

Facultad de Ingeniería - Instituto de Ingeniería en Computación

Interoperabilidad - Electiva Técnica / Curso de Posgrado

Junio de 2002

Tabla de Contenido

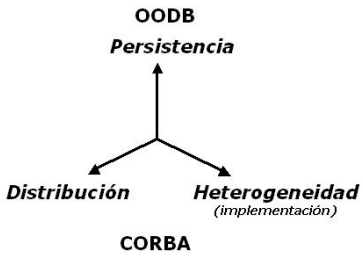
I. INTRODUCCIÓN

El artículo presenta una nueva arquitectura para combinar CORBA y OODB.

CORBA: provee distribución y heterogeneidad. Propuesto por la OMG.
OODB: provee persistencia ACID de objetos. Propuesto por la ODMG.
Ambos trabajan sobre modelos orientados a objetos.

El intento del artículo, es lograr una forma de mezclar, combinar ambos modelos, y aprovechar las ventajas de cada uno. Persistencia ACID de uno, y distribución y heterogeneidad (de implementación) en el otro.
Se puede decir que ambas propuestas se complementan.
Cada una le agrega a la otra la característica "que le falta":

- "CORBA con persistencia"
- "OODB con distribución".



Ya hay algunas propuestas y algunos productos (wrappers y adapters). El artículo plantea ciertas desventajas y limitantes de los mismos (programar en uno, acoplamiento en otro) y propone una nueva propuesta, intentando superar esos problemas. Por lo tanto **es un artículo técnico, que busca llegar a un producto**.

CORBA

Su sigla significa: Common Object Request Broker Architecture. Es una especificación, existen implementaciones de varios proveedores (ej: Jacorb para Java, es gratuito).
Permite la comunicación entre objetos distribuidos en ambientes heterogéneos.
Objetos distribuidos – objetos que corren en distintos procesos, eventualmente en distintas maquinas de una red.
Ambientes heterogéneos – heterogeneidad de la arquitectura de los computadores, sistemas operativos, y lenguajes de programación.
Pertenece al Reference Model de la OMG (Object Managment Group).

REFERENCE MODEL

Algunos de los componentes de este modelo son:

- Object Request Broker (ORB)

es el núcleo de CORBA, quien hace posible la comunicación transparente, que los objetos generen y reciban pedidos y respuestas en un ambiente distribuido.

- Object Services

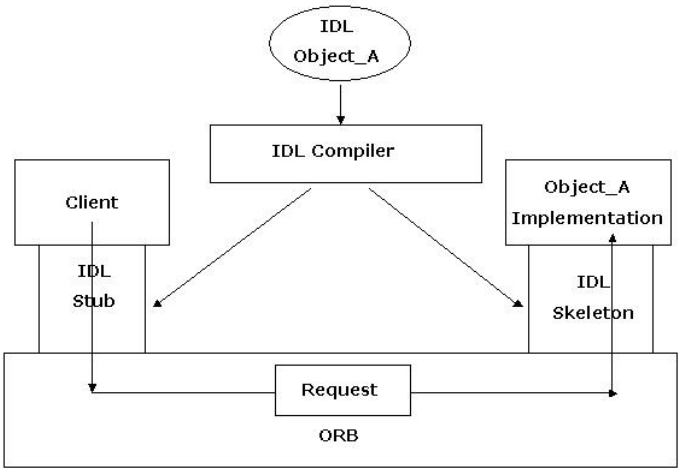
Son un conjunto de servicios, en ellos se definen operaciones y su semántica. Brindan funciones básicas necesarias e independientes del dominio de aplicación.
Ejemplos:

- Naming Service, es un directorio de objetos, asocia un nombre a la instancia de un objeto, así cualquier otro objeto puede obtener una referencia a él mediante ese nombre.
- POS, Persistent Object Service, su objetivo es brindar persistencia a los objetos CORBA, pero no ha sido aceptado por los proveedores de OODB debido a su semántica poco clara y a las debilidades de su modelo transaccional.

IDL

Su sigla significa: Interface Definition Language. Es parte de CORBA, no es un lenguaje de programación, es el lenguaje utilizado para definir las interfaces (atributos y métodos) de los

objetos CORBA, luego el compilador IDL mapeará esas definiciones a un lenguaje de programación determinado.
La siguiente figura muestra la interacción entre los elementos básicos de CORBA.



OTS

Es un servicio más. Es el Object Transaction Service. Define un modelo para manejar transacciones (transacción: unidad de trabajo con las propiedades ACID).

OODB

ODMG estandariza las bases de datos orientadas a objetos definiendo tres lenguajes:

- ODL, Object Definition Language, es para definir los objetos.
- OML, Object Manipulation Language, es para manipular los objets (crear, borrar).
- OQL, Object Query Language, es para recuperar objetos según criterios (consultas).

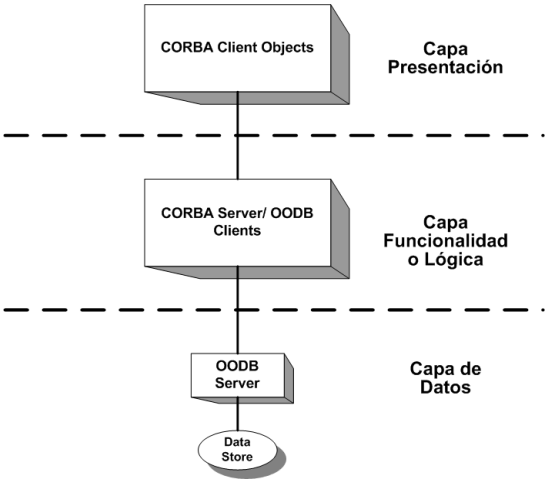
ADAPTERS vs WRAPPERS

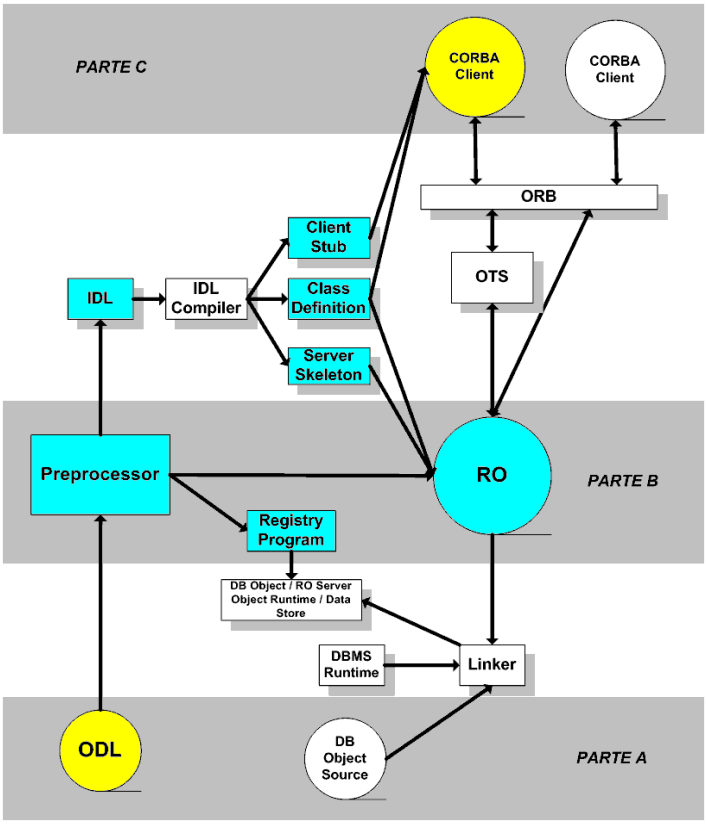
Son dos enfoques diferentes para la integración de CORBA/OODB.
El adapter es un agregado a la arquitectura CORBA, y básicamente lo que realiza es guardar la implementación CORBA del objeto en una OODB, brindando persistencia a la distribución.
El wrapper brinda a los objetos ya existentes en un almacén de datos un envoltorio (interfaz CORBA) que le brinda distribución.
Veamos las ventajas de cada enfoque:

Adapters	Wrappers
Se complementan con CORBA	Uso de los objetos de la base de datos sin modificarlos.
Integración "sin costuras", transparente al programador.	Permite OODB, BD relacionales, y aplicaciones legadas.
Implementación de la integración directa, basta utilizar un ORB con adapter, no exige gran esfuerzo de programación como el wrapper.	Mayor performance, debido a que deben ser implementados para cada aplicación.
	Uso mínimo de almacenamiento.
	Separa la interfaz del objeto (objeto CORBA) de su implementación (objeto de la BD).

II. ARQUITECTURA

La nueva arquitectura para la integración CORBA/OODB planteada en el documento se enmarca dentro de las arquitecturas de middleware. Un **Middleware** [1]es una tecnología de software diseñada para ayudar a manejar la complejidad y heterogeneidad inherente a un sistema distribuido. Por tanto, se plantea una arquitectura de tres capas (**Three-Tier Model**), donde la capa intermedia brinda todas las funcionalidades o lógica del sistema distribuido. La capa superior, o de presentación, solo se encarga de la aplicación en particular que se quiere realizar, y la capa más inferior, en este caso son las bases de OODB.





Sin intención de entrar en el detalle de la arquitectura se muestra a grandes rasgos las tres capas del modelo (partes A, B y C) donde desde el punto de **vista de los desarrolladores** solo importa concentrarse en el Cliente CORBA (aplicación realizada como cualquier otra que utilice CORBA) y en el otro extremo las base de datos en la determinación de su ODL[2].

Debe entenderse que el motivo de agregar en este caso la capa de lógica intermedia es dar portabilidad, estandarización (no depende de implementaciones particulares de los vendedores de OODB) y principalmente disminuir el tiempo de desarrollo. En este ultimo punto se piensa que el documento realiza un aporte interesante a las técnicas de integración realizadas hasta el momento, puesto que enmarcándose en las arquitecturas de wrappers, obtiene las ventajas del enfoque y elimina la principal desventaja del mismo, que es el volumen de código que debe ser generado. Para lograr esto, el planteo es desarrollar un **preprocesador** encargado de generar al momento de la integración la mayor parte del trabajo. El preprocesador se encarga de realizar la conversión entre los modelos de datos de las bases OODB (el ODL) y el modelo de CORBA (el IDL), además realiza parcialmente el **RO** que es el encargado de mantener

[1] Se puede encontrar mayor especificación de los middleware en: RFC 2768.

[2] La vista del programador se observa en amarillo.

la correspondencia entre el objeto definido en CORBA y su correspondiente en OODB y controlar las transacciones.

MAPEO ODL-> IDL

En el documento se plantea dicha conversión de forma no muy formal, donde a grandes rasgos la estrategia de transformación es transformar los tipos básicos de ODL a IDL como tales (ambos modelos tienen a los enteros, reales, etc.), para los tipos básicos definidos en ODL que no se encuentran en IDL (como timestamp) se plantea utilizar técnicas de wrappers para implementar su código.
En el caso de los tipos definidos por el usuario se plantea realizar en el IDL las operaciones como agentes de las operaciones en la base de datos. Solo en el caso de las operaciones que son relaciones se realiza un tratamiento especial: en ODL se pueden definir tres tipos de relaciones según su cardinalidad (1-1, 1-m, n-m), cada una de esas relaciones se implementa en CORBA con funciones del tipo *get* y *set relation*, donde el tipo devuelto es un objeto o una colección según la cardinalidad de la relación original.

RO-COMPONENT

El RO-Component busca trasladar las propiedades ACID de la capa inferior, **a los objetos de la capa media**. Es así que tiene que ofrecer persistencia y transacciones.

Usa el patrón típico de wrapper para ofrecer la durabilidad. Cada objeto de la OODB tiene un objeto Corba que lo enmascara y otro objeto llamado Factory que representa la colección. Las inserciones, los borrados y las consultas se llevan a cabo mediante el Factory, mientras que las modificaciones son a través de los get y set de los objetos. Para representar las relaciones se usan referencias o conjuntos de referencias entre objetos (dependiendo de la multiplicidad).

El otro tema es implementar las transacciones. Ya las bases de datos OODB ofrecen propiedades ACID, por lo que aparentemente el tema de las transacciones sería automático. Pero esta propuesta **pretende usar varias bases de datos a la vez, y hacer transacciones globales**.

Propone usar el OTS (Object Transaction Service) del modelo de Corba, **para implementar dichas transacciones globales**. Es así, que cada componente de la transacción (cada objeto y colección persistentes) implementan una interfaz llamada *Resource*. Es mediante esta interfaz que se comunican con el OTS.

Así, los sub-componentes serían:

Factory del tipo A	la colección
Objeto Corba del tipo A	la entidad
Implementación de la interfaz Resource en el objeto A	sincronización interna

Acá se plantea el problema de la autonomía vs. transacciones distribuidas. El artículo no menciona nada de cómo llevar a cabo las transacciones locales –más allá del uso del OTS-, pero si se quiere mantener la autonomía debería usarse el enfoque de “unlabeled box” visto en el curso.

Para eliminar la principal desventaja de los wrappers -que era el programar la persistencia a mano- se vale de un preprocesador que genera la mayor parte de ese componente.

PREPROCESADOR

Este nuevo enfoque resuelve el problema de la integración CORBA/OODB mediante una tercer parte: el procesador, es por eso que es el componente clave. Su objetivo es la generación automática de código. Su entrada es la especificación en ODL, de esto genera el IDL, parte de la implementación del RO, y otros componentes.
El preprocesador debe conocer la diferencia de los códigos entre las funcionalidades propietarias de los proveedores de OODB y de ORB.
Mediante este componente se reduce el tiempo necesario para la integración.

En el artículo no queda claro como implementarlo.

III. DISCUSIÓN

Veamos los puntos fuertes y débiles de la propuesta del artículo:

Puntos Fuertes

- Brinda una arquitectura abierta, posibilitando la integración de cualquier proveedor de OODB que cumplan con el estándar de ODMG.
- Uso de un estándar en las transacciones, el OTS.
- Soporta múltiples bases de datos. Los clientes no necesitan conocer ninguna característica específica de las OODB, en especial el modelo de transacciones. Nos permite realizar transacciones múltiples (entre objetos de distintas bases de datos).
- Resuelve los problemas del POS
- Integración con otros servicios CORBA. Es fácil de agregar alguna integración con otro servicio, implementándola en el RO_Factory (por ejemplo con el Object Security Service).
- Integración automatizada mediante el preprocesador, de esta forma toma lo mejor de los dos enfoques de integración (ver Adapters vs Wrappers).

Puntos Débiles

- Implementación del Preprocesador poco definida
- Falta Integración Semántica, o sea que objetos de diferentes OODB pero con igual semántica se mapeen al mismo objeto CORBA.
- Restringida a OODB que cumplan con el estándar de ODMG, no permitiendo BD relacionales que son ampliamente utilizadas ni aplicaciones legadas.

IV. CONCLUSIONES

Se considera que se alcanzaron los objetivos iniciales del articulo: Lograr una arquitectura que integre CORBA y OODB. La especificación de dicha arquitectura es clara pero poco profunda, principalmente se desarrolla muy poco el preprocesador (y como realiza la automatización) y no es explicita la solución de las transacciones globales, en particular el detalle de la implementación del RO y que tanto es generado automáticamente.
En contraparte los autores realizan una implementación inicial de todo el sistema, logrando mostrar el funcionamiento completo de la idea.

Desde un punto de vista más general surge el cuestionamiento de sí la integración planteada puede ser utilizada para interoperar, con todos las cuestiones que esto implica.
En primera instancia es importante mencionar que la arquitectura esta pensada para interactuar con varias fuentes OODB, pero no permite otras bases de datos relacionales o legadas. Por ser una arquitectura de wrapper es posible desarrollar el código especial para otros tipos de bases de datos, pero esto no es planteado por los autores.
Para resumir las limitantes principales desde el punto de vista de la interoperabilidad se consideran los principales aspectos en la siguiente tabla:

Distribución	Autonomía		Heterogeneidad	
Buena	Diseño	NO	Sistemas	SÍ
	Comunicación	SÍ	Sintáctico	NO
	Ejecución	SÍ	Semántico	NO
	Participación	NO		

Donde se resalta que es una solución potente en cuanto a la distribución lograda, muy mala desde el punto de vista de la autonomía puesto que no existe diseño independiente en las fuentes (solo se permite elegir el vendedor de la base OODB), ni flexibilidad en la participación de la federación (se comparte lo que se encuentre definido en la ODL). Pero principalmente presenta falencias si se considera la heterogeneidad, esta bien trabajada la heterogeneidad de sistemas permitiendo distintos hard/software, sin embargo carece de heterogeneidad sintáctica (solo acepta el modelo de datos OODB) y mucho mas importante no intenta abordad el problema de la integración semántica, relegando entonces este problema completamente a las aplicaciones de la capa presentación.

A FUTURO

Para solucionar esta carencia de integridad semántica los autores plantean a futuro realizar una Vista Global de Esquemas utilizando un servicio de CORBA de Naming Service para esquemas. Hasta que no se tenga implementado el mismo, el sistema solo puede funcionar como un repositorio de datos con transacciones globales y no un sistema de datos distribuidos.

También se plantean a futuro mejorar en otras áreas como: Seguridad, Manejador de Eventos, Control de Versiones.

Finalmente se realiza una discusión sobre la performance del sistema al ser desarrollado de forma automática por del Preprocesador, comparándolo con otros wrappers desarrollados manualmente. Se concluye que la performance es buena, pero es posible optimizar en algunas áreas al preprocesador.

V. REFERENCIAS

- Ruey-Kai Sheu, Kai-Chih Liang. Shyan-Ming Yuan, *Member, IEEE,* and Win-tsung Lo, “A New Architecture for Integration of CORBA and OODB”, IEEE Transaction on Knowledge and Data Engineering, Vol. 11, No. 5, September/October 1999.
- J. Kleindienst, F. Pasil, and P. Tuma, “What We Are Missing in the CORBA Persistent Object Service Specification”, *Proc. OOPLSA ’96 Workshop*, URL: http://www.infosys.tuwien.ac.at/Research/Corba/archive/special/missing-persistence.ps.gz.
- V. Vasudevan and R. Anthony, *Aproaches for the Integration of CORBA with OODBs,* URL: http://www.infosys.tuwien.ac.at/Research/Corba/archive/special/ORB_OODB.ps.gz, Aug. 1994.
- V. Amirbekyan and K. Zielinski, “What CORBA/ODB Integration Technique to Choose: Adapter vs. Wrapper”, *Proc. OOPSLA ’97 Workshop,* http://galaxy.uci.agh.edu.pl/~vahe/ad_vs_wr.htm.