Instituto de Computación
Facultad de Ingeniería
Universidad de la República Oriental del Uruguay

PROYECTO DE GRADO

de la Carrera Ingeniería en Computación

Titulo del Proyecto:

Integración de herramientas CASE usando Internet, Corba y Repositorios de Metainformación

Presentado por:

Claudia Murialdo
Enrique Delfino
Cristian Mussini

En la fecha: 17 Abril 2002

El proyecto fue supervisado por:

Raul Ruggia
Fernando Rodriguez

TABLA DE CONTENIDO

1. Introducción

1.1. Contexto

Las ***herramientas CASE*** asisten en el análisis y diseño de proyectos y facilitan la generación de aplicaciones finales, brindan facilidades en una amplia gama de actividades que van desde la asistencia gráfica en el diseño hasta la generación automática de código. Constituyen un tipo de software fundamental en el desarrollo de proyectos, sobre todo de gran porte.

Durante el ciclo de vida de un proyecto puede ser necesario utilizar varias herramientas CASE por diferentes razones, entre ellas:

- una sola herramienta puede no cubrir todas las necesidades
- no cubre todo el ciclo de vida de desarrollo del proyecto
- el proyecto se reparte entre varias empresas y no es posible que todas estén de acuerdo en utilizar la misma herramienta.
- la diversidad de lenguajes y metodologías existente ha hecho que no exista un ambiente CASE integrado para las mismas.

Sin un ambiente integrado, con una forma de intercambio de información, al momento de mover información de una herramienta a otra, es necesario ingresar nuevamente los datos en el nuevo ***modelo***. Esto es una solución costosa, que puede introducir nuevos errores, y que requiere un conocimiento completo de las herramientas con las que se esta intercambiando la información.

Una forma de integración de las mismas consiste en programar interfaces para la interacción entre cada par de componentes utilizados, debiendo reprogramar una parte importante de las interfaces si alguno de los componentes cambia.

Una forma de integración menos costosa y mas escalable consiste en definir una interface estándar independiente de los productos, de forma que integrar una nueva herramienta a un conjunto ya existente o el cambio en una de ellas, requiera sólo la programación de la interfaz para el nuevo componente (o componente modificado) que realice la comunicación con un manejador central. Este último se basa en contar con un repositorio de modelos en común y un mecanismo de conexión que permita integrar las funcionalidades.

1.2. Objetivos

El objetivo global de este proyecto es la construcción de un manejador de repositorio de modelos centralizado, definiendo un modelo estándar como modelo canónico para el almacenamiento de los mismos.
Concretamente el repositorio abarca todos los ***meta-modelos*** especificados en dicho modelo canónico, pero el proyecto se centra en el modelo ***OLAP*** para datawarehouse.

Los objetivos primarios a cumplir son:

1. **Permitir el almacenamiento de modelos en un repositorio que sea común y accesible a todas las herramientas.**
2. **Proveer los mecanismos de invocación de servicios del repositorio a través de Internet independientemente del lenguaje e implementación** (tales que no requieran necesariamente que las herramientas tengan una interfaz *CORBA* con el repositorio como se había realizado hasta ahora en proyectos relacionados).

Como objetivos secundarios se definen:

3. navegación de los modelos contenidos en el repositorio a través de Internet
4. aplicación practica de este proyecto a una herramienta CASE dada.

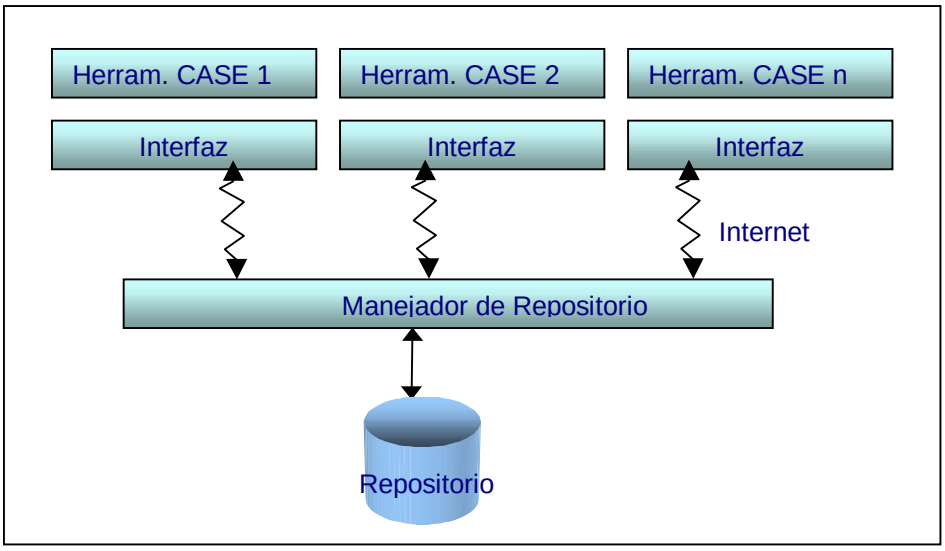


Fig. 1.1 Esquema que resume los objetivos primarios del proyecto

1.3. Descripción general

El presente proyecto implementa un manejador de repositorio centralizado para modelos OLAP, donde las herramientas CASE pueden realizar altas y modificaciones de modelos (OLAP) a través de Internet.

Los modelos son almacenados en forma persistente y cualquiera de ellos esta accesible a cualquier herramienta CASE que acceda al repositorio.

Las herramientas CASE que se integren con el manejador pueden estar implementadas en cualquier plataforma y lenguaje, lo único que requieren es implementar una interfaz (en el lenguaje y plataforma que se desee) para acceder a la funcionalidad del mismo. En caso de que dicha interfaz se implemente en el lenguaje Java, se puede utilizar una serie de objetos implementados en este proyecto que facilitan las transformaciones entre cierto modelo X (el que maneje la herramienta), y el OLAP.

El repositorio puede ser navegado a través de Internet, utilizando un browser. Se puede obtener un reporte en HTML de cada modelo almacenado, e incluso cada modelo puede ser obtenido (downloading) a través del browser. Esta navegación es de solo consulta.

La siguiente figura muestra la funcionalidad de este proyecto en forma global.

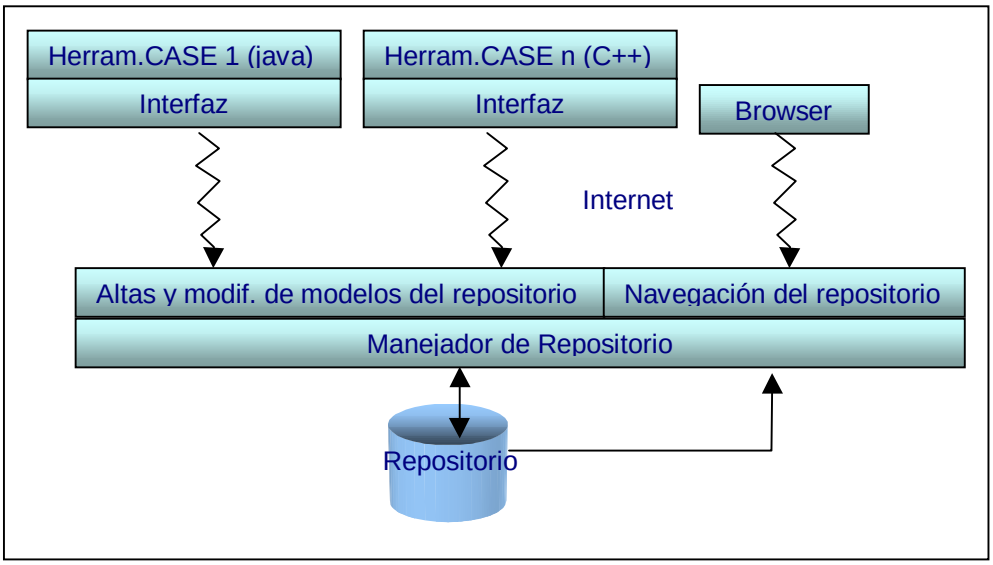


Fig. 1.2 Funcionalidad global del proyecto

Finalmente se realizo la aplicación practica del manejador a una herramienta CASE en particular, implementando la interfaz correspondiente.

Antes de dar una descripción más completa de las partes que se observan en la figura anterior, es importante mencionar dos temas que son claves en la resolución de los objetivos primarios y determinan las principales características de este proyecto: ***CWM*** y ***SOAP***.

CWM (Common Warehouse Meta-model) es el meta-modelo canónico que se maneja en el repositorio (en particular el OLAP de CWM). La especificación CWM es establecida por la ***OMG*** (basada en los estándares ***UML***, ***MOF***, y ***XMI***), consiste de un conjunto de meta-modelos llamados paquetes (como por ejemplo los paquetes OLAP, Relacional, Transformacional, Jerárquico), y define como modelar, manipular e intercambiar metadata entre herramientas. (De ahora en mas los términos paquete y meta-modelo se usaran indistintamente). MOF es un estándar de la OMG para definir, representar y manipular metadatas (por mas detalle ver Anexo C dMOF-MOF). XMI es un estándar para codificar colecciones de meta data basadas en MOF como documentos ***XML*** (ver Anexo B-MetaData).

SOAP (Simple Object Access Protocol) es el mecanismo de comunicación a utilizar entre las herramientas y el manejador de repositorio (para el alta y modificación de modelos). El protocolo SOAP se basa en XML para hacer llamadas a procedimientos remotos usando HTTP como protocolo de transporte.

Por mas detalle sobre SOAP y CWM ver apéndice A-Conectividad y B-MetaData respectivamente.

Manejador de Repositorio.

La implementación del manejador de repositorio "núcleo" (representado en la figura 1.2 por un recuadro con el nombre "manejador de repositorio") es una implementación CORBA, generada con la herramienta dMOF.

DMOF permite, básicamente, generar manejadores de repositorios para meta-modelos basados en MOF. Los modelos son almacenados en el repositorio como objetos CORBA (se almacena su ***IOR***).

Este manejador tal cual es generado puede ser accedido únicamente por clientes CORBA. La funcionalidad (para clientes CORBA) abarca todo lo que es creación y manejo de modelos OLAP de CWM. Permite además trabajar con el documento XMI correspondiente a cada uno de esos modelos (el cual contiene la misma información). Por ejemplo a partir de un objeto CORBA que representa un modelo permite obtener el XMI del mismo y viceversa (procesos que de ahora en mas llamaremos Producer –producir XMI- y Consumer –consumir XMI- respectivamente).

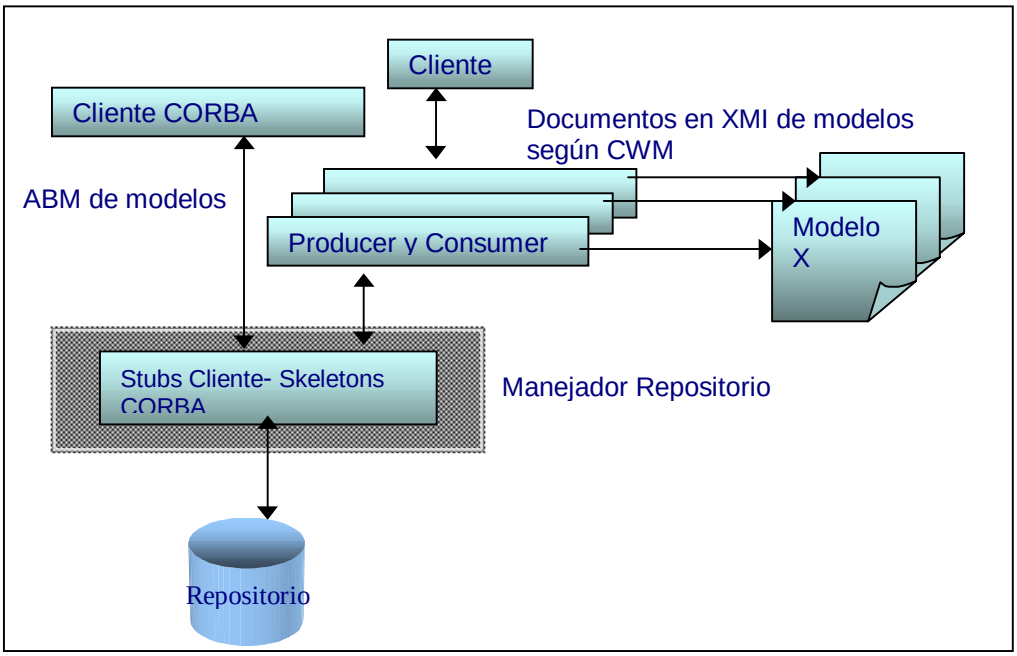
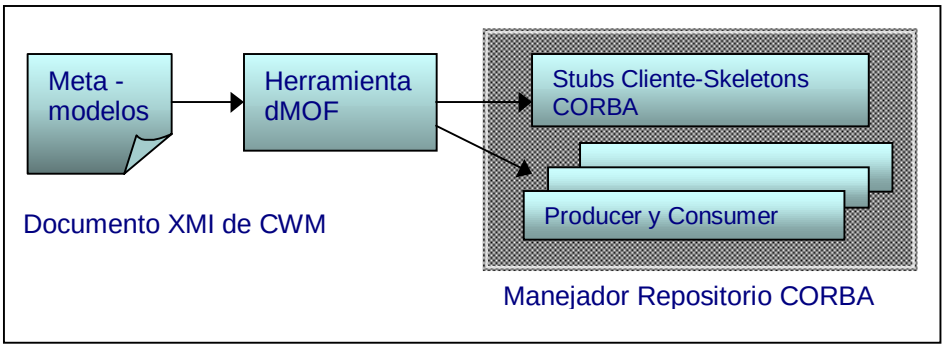


Fig. 1.3 Funcionamiento del manejador del repositorio

Para la generación del repositorio con dMOF se tomó como entrada el documento XMI de la especificación de CWM (como todo meta-modelo basado en MOF, CWM tiene una especificación en XMI).



Proceso Fig. 1.4 Proceso de generación del manejador repositorio con la herramienta dMOF

Altas y modificación de modelos en el repositorio vía Internet

El alta y modificación de modelos vía Internet (ver Fig. 1.2) implementa un "puente" SOAP-CORBA. Es una "capa" sobre el manejador de repositorio en CORBA que permite a cualquier tipo de clientes (herramientas) acceder a los modelos OLAP almacenados en el repositorio vía SOAP. Permite agregar un modelo nuevo al repositorio u obtener y actualizar uno existente, **en los tres casos el modelo es dado en XMI**. Es decir que las herramientas CASE que interactúen con el manejador a través de SOAP deben enviar y/o recibir modelos OLAP (de CWM) en XMI. Adicionalmente almacena datos de cabecera de los modelos como fecha de creación y descripción.

Esta interfaz SOAP ofrece una funcionalidad restringida con respecto a la totalidad que brinda el manejador en CORBA, ya que solo permite actualizar un modelo "entero", no permite modificar partes como con CORBA. Ofrece los métodos atómicos: LoadXMI (obtención de un modelo) y StoreXMI (inserción y actualización). Para estos utiliza además la funcionalidad Producer y Consumer.

La siguiente figura muestra mas en detalle la interacción entre los recuadros titulados como "Altas y modif. de modelos del repositorio" y "Manejador de Repositorio" de la Fig. 1.2.

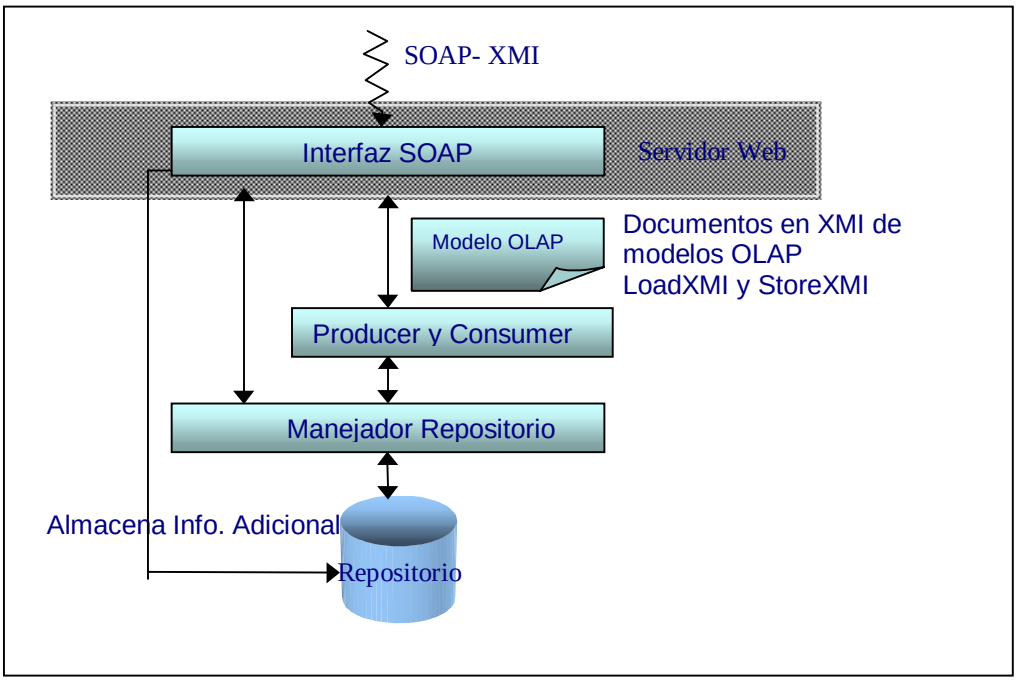


Fig. 1.5 Interacción de la Interfaz SOAP con el manejador de repositorios en CORBA generado por dMOF

Navegación del repositorio

El módulo para el acceso web al repositorio (recuadro titulado como Navegación del repositorio en la Fig. 1.2) implementa la navegación y consulta del repositorio a través de Internet, las funcionalidades que ofrece son:

- Navegación del repositorio de modelos, obteniendo un listado de los modelos existentes en el repositorio, con datos de cabecera como fecha de creación, descripción e identificador (información adicional del modelo).
- Obtención del documento XMI de cualquier modelo almacenado en el repositorio
- Generación de un reporte en HTML de cualquier modelo, en un formato más "legible" resultado de aplicar una transformación ***XSL*** al XMI mencionado.

Adicionalmente este módulo implementa la administración de servicios en forma remota, para facilitar el control de los servicios que estén corriendo del lado del servidor en forma remota (por ejemplo el servidor de repositorio).
Tanto este modulo como la interfaz SOAP corren bajo el mismo Servidor Web. Ambas pantallas se muestran mas adelante, en el capitulo 3 de Especificación del producto.

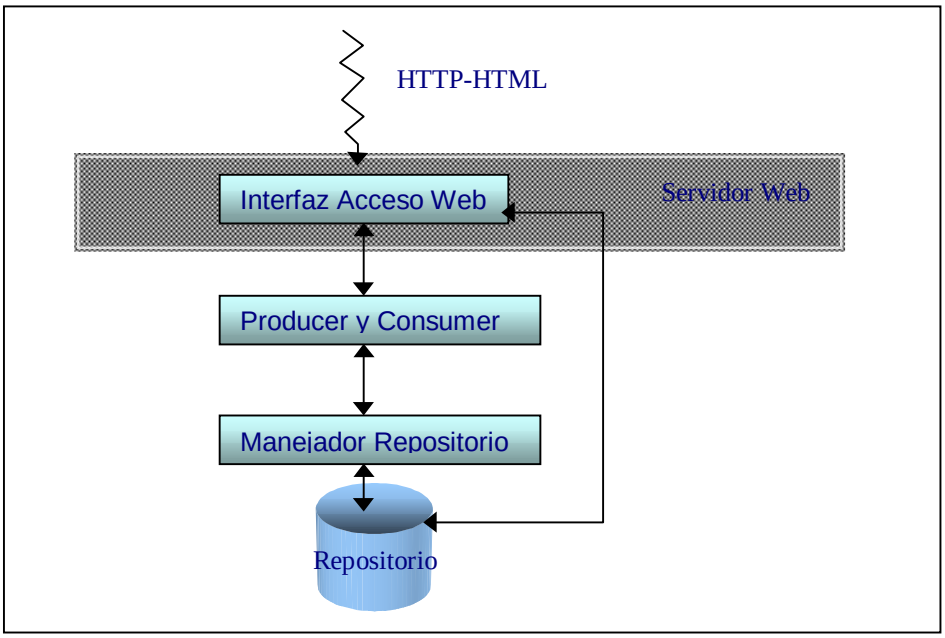


Fig. 1.6 Acceso Web al repositorio

Interfaz entre cliente (herramienta CASE) y objetos de "ayuda al cliente"

Los recuadros de la figura 1.2 con el nombre "interfaz" representan la implementación necesaria del lado cliente para invocar los métodos LoadXMI y StoreXMI (proveídos en la interfaz SOAP de alta y modificación de modelos) a través de SOAP. Como se mencionó antes lo que el cliente envía y recibe es el XMI de un modelo OLAP según CWM, por lo cual la "interfaz" también debe hacer la transformación entre el modelo manejado por el cliente (digamos modelo-X, compatible con OLAP), y el OLAP de CWM (en XMI).

Para facilitar la transformación de un modelo-X a OLAP (de CWM) es que se implementaron clases de "ayuda al cliente". La misma consiste de objetos Java que en conjunto modelan los principales componentes del paquete OLAP de CWM, y traductores que transforman esos objetos a un documento XMI de acuerdo a la especificación CWM y viceversa. La conversión se ve facilitada porque transformar un modelo-X a objetos java que modelan OLAP (los que luego generan el XMI) es mucho mas simple que transformar un modelo-X al XMI (según CWM) directamente, el cual tiene un tamaño y complejidad muy considerables.

En general el mecanismo de comunicación entre una herramienta CASE y el manejador de repositorio a través de SOAP sigue el siguiente esquema:

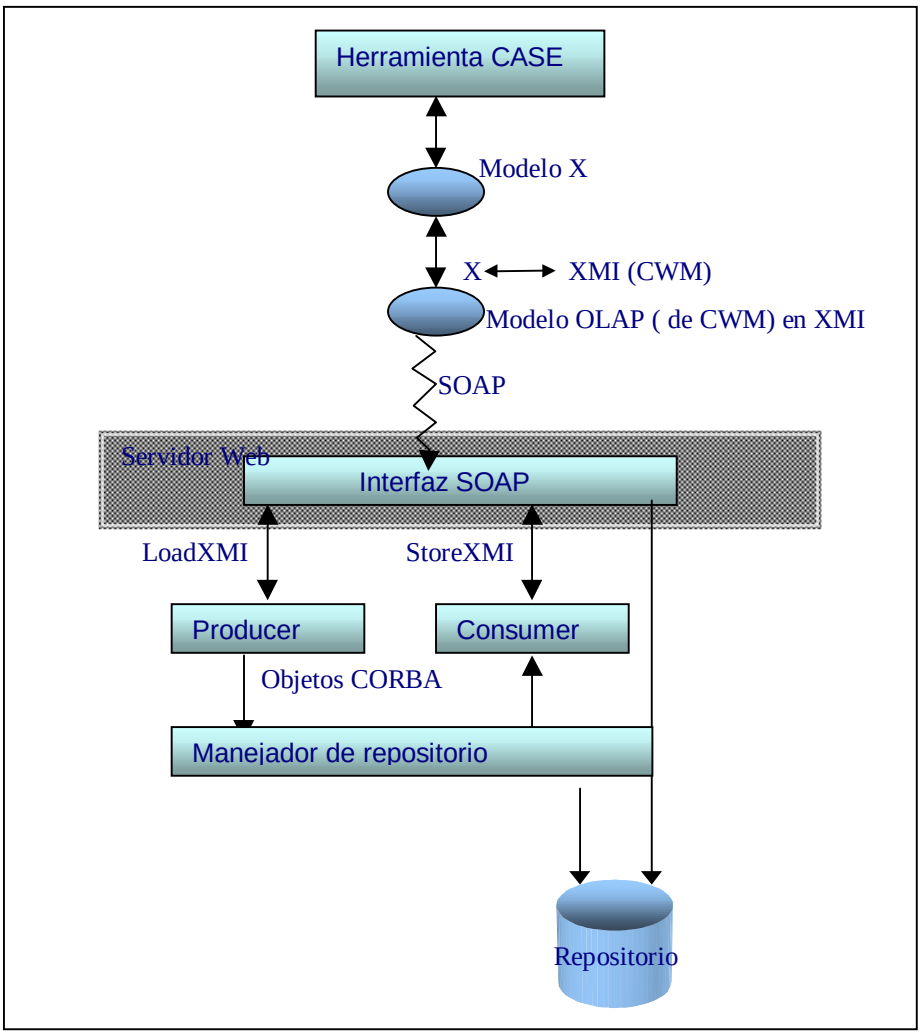


Fig. 1.7 Aplicación del manejador de repositorio a una herramienta CASE en general

Como lo muestra la figura, la herramienta CASE que utiliza un meta-modelo X primero transforma el modelo a intercambiar al correspondiente de OLAP (CWM) en XMI (si es java lo puede hacer utilizando las clases de ayuda). Envía ese XMI a través de SOAP a la interfaz SOAP, invocando el método StoreXMI para insertar (o actualizar) ese modelo en el repositorio. Este método agrega (o actualiza) ese modelo en el repositorio utilizando el Consumer que a partir del XMI obtiene un objeto CORBA equivalente. Luego para obtener un modelo existente se invoca LoadXMI el cual retorna el XMI correspondiente previo invocación al Producer. Para mas detalle se pueden observar los diagramas de secuencia en la Documentación Técnica.

Aplicación práctica a una herramienta CASE particular

La herramienta CASE particular con la que se realizó una aplicación practica de este proyecto es el editor de modelos ***CMDM***[1]. Para realizar la transformación entre modelos CMDM y OLAP según CWM se hizo previamente un estudio de correspondencias entre los mismos que se detalla en la Documentación Técnica.

[1] Herramienta desarrollada en el proyecto del año 1999 por Picerno y Fontán denominado “Un CASE para OLAP”. Es un prototipo de editor de modelos CMDM, que permite trabajar con modelos CMDM, y guardar los mismos en un formato particular.

La siguiente figura se corresponde con la Fig. 1.2 pero muestra mas en detalle el esquema global del producto obtenido diferenciando las partes mencionadas.

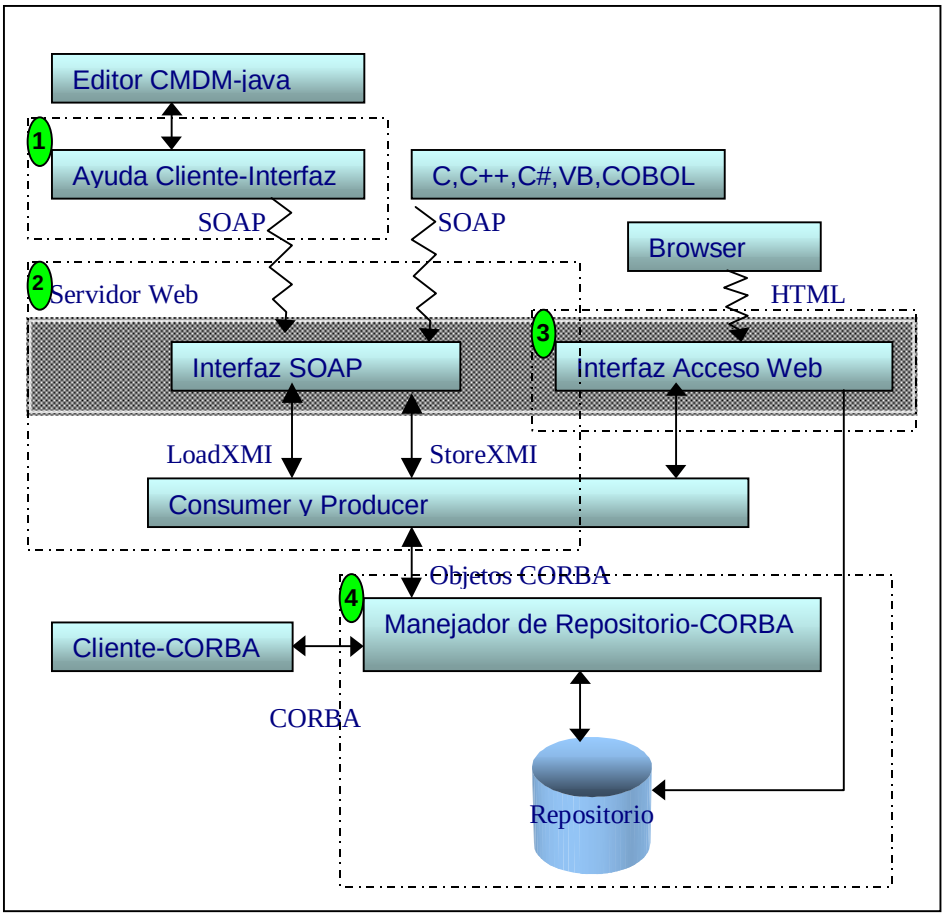


Fig. 1.8 Esquema global de las 4 partes implementadas: 1-Ayuda al Cliente, 2-Acceso SOAP al repositorio (alta y modificación de modelos), 3-Acceso web (navegación), 4-Manejador de repositorio (generado con dMOF).

1.4. Alcance

Se cumplió con el objetivo de implementar un manejador de repositorios centralizado independiente del lenguaje e implementación y accesible a través de Internet. A continuación se detallan los aspectos referentes al alcance que fueron tomados en cuenta y otros que se dejaron de lado por falta de tiempo o porque escapaban a los objetivos puntuales del proyecto.

Clientes. Se dio la mayor importancia a este aspecto, intentando no limitar por este lado. Se implemento un repositorio accesible por cualquier cliente (herramienta CASE), en cualquier plataforma y lenguaje. Se permitió también que usuarios comunes (que no necesariamente tengan acceso a una herramienta CASE que se pueda integrar), a través de un browser, puedan navegar el repositorio y entender el contenido del mismo.

Estudio y análisis del modelo canónico a utilizar. En este proyecto no se realizo un una evaluación detallada de estándares para el intercambio de metadata que incluyera un estudio comparativo, si no que se vieron tres opciones de las cuales se decidió rápidamente por CWM, ya que no era el objetivo del mismo realizar tal estudio y CWM se adaptaba a los requerimientos.

Modelos manejados. Utilizar una herramienta como dMOF que generara al manejador de repositorio amplia en gran medida el alcance potencial del proyecto, porque si bien la implementación de las interfaces SOAP, Web y de ayuda al cliente se limitan a OLAP, el manejador de repositorio puede ser generado para todos los meta-modelos de CWM por lo que bastaría con extender esas interfaces.

Funcionalidad del manejador. La funcionalidad del repositorio ofrecida por Internet a los clientes es restringida porque no permite modificar partes de un modelo, si no solamente insertar, o actualizar un modelo entero. Esta limitación se debe principalmente a que ofrecer toda la funcionalidad a través de SOAP resulta muy difícil de mantener frente a cambios que puedan surgir en la especificación de CWM. (A menos que se hubiera implementado un generador para esa interfaz, lo que escapa a los objetivos del proyecto).

Seguridad. Otro de los aspectos que esta fuera del alcance del proyecto es la seguridad y control de usuarios. Los modelos almacenados en el repositorio pueden ser consultados y modificados por todos los usuarios. No se desarrollo un sistema de lockeo de modelos, por lo que si dos usuarios modifican el mismo modelo al mismo tiempo, solo uno de ellos persistirá los cambios (quien realice último el Store) mientras el otro los perderá.

Dependencia de productos. Se tiene dependencia en cuanto al ORB utilizado del lado servidor (Visibroker), ya que es el utilizado por dMOF. Por otra parte la base de datos utilizada como repositorio es MySQL, dMOF soporta MySQL 3.22 o 3.23, PostgreSQL 7.0 y Oracle 8i por lo que también se tiene un rango limitado si se deseara cambiar de base de datos.

1.5. Organización del documento

El informe esta estructurado en cinco capítulos. En los mismos se intenta dar un enfoque global al lector de los aspectos mas relevantes del proyecto, y mencionar los principales conceptos utilizados, con la intención de dejar una idea global pero no un entendimiento total de los mismos, para ello es recomendable profundizar en los documentos Anexos y en algunos casos en la Documentación Técnica.

CAP.1. La introducción da un panorama general del contexto en que se desarrolla el proyecto y sus objetivos. Se da una breve descripción general del proyecto y su alcance.

CAP.2. En el capítulo 2 se tratan los trabajos relacionados al tema del proyecto haciendo una síntesis de los principales trabajos estudiados. Se incluyen trabajos realizados en la Facultad de Ingeniería y en el exterior.

CAP.3. En el capítulo 3 se da una descripción general del diseño del producto, incluyendo su funcionalidad y su arquitectura. Para el diseño se van describiendo las diferentes decisiones que se tomaron con el transcurso del tiempo.

CAP.4. En el capítulo 4 se realiza un estudio de la implementación poniendo énfasis en las tecnologías utilizadas y su interoperabilidad.

CAP.5. Finalmente se brinda una conclusión, la cual resume lo hecho, nuestro aporte y balance, y el futuro de este proyecto. Al final de este capítulo se encuentran las Referencias hechas tanto en este documento como en la documentación técnica y Anexos. Y por último el Glosario que da una definición concreta y “rápida” de los conceptos mas importantes mencionados en este documento (la mayoría de los mismos se extienden en los Anexos).

1.6. Convenciones tipográficas

Los términos importantes se encuentran en *cursiva* la primera vez en que se mencionan. Los mismos se detallan en el Glosario.

Las referencias bibliográficas (y web) se encuentran en [azul entre corchetes]. Al final del informe aparecen datos completos de cada referencia.

2. Trabajos relacionados

Proyectos de Grado realizados en la Facultad de Ingeniería – Universidad de la República

- **Editor de CMDM (año 1999).** [cmdm-proy] Esta herramienta es un editor genérico para modelos que cumplen con ciertas restricciones (que lo hacen un "modelo editable" por el mismo). Como caso de prueba , este editor fue instanciado para manejar el modelo de esquemas multidimensionales propuesto por el Ing. Fernando Carpani, denominado CMDM [cmdm].

  Relación con el presente proyecto. En nuestro proyecto se implementa una aplicación práctica del manejador de repositorio utilizando esta herramienta. La misma se modifica agregando las acciones necesarias que permiten al editor invocar la funcionalidad del manejador de repositorio a través de SOAP. Con esta modificación es posible crear y modificar modelos CMDM con el editor, y una vez prontos guardarlos en el repositorio de modelos OLAP (de CWM), o cargar un modelo del repositorio y editarlo como uno CMDM.
  El meta-modelo CMDM propuesto por el Ing. Fernando Carpani es tomado como base para el estudio de las correspondencias con OLAP (de CWM), y luego es aplicado al editor, debido a que el mismo hace una "adaptación" de la definición original de CMDM.

- **Integración de herramientas *I-CASE* (año 1998 y 1999)**. Se utilizó Corba y Oracle (módulo Designer) para proveer mecanismos de integración de herramientas CASE. En el año 1998 se implementó un Servidor de Repositorio de herramientas CASE que puede ser accedido desde una herramienta cliente mediante CORBA. El mismo exporta una interfase CDIF (estándar para intercambio de información entre herramientas CASE) para Modelo Entidad Relación y que almacena los datos en el repositorio de Oracle Designer/2000 (D2K).
  En el año 1999 se continúa el trabajo del proyecto de 1998 agregándole:
  - **Meta-modelo Multidimensional**, que permite el almacenamiento de esquemas MD, con la especificación del modelo MD realizada en el proyecto del año 1998.
  - **Meta-modelo Relacional**, que permite el almacenamiento de esquemas Relacionales.
  - **Meta-modelo para transformaciones entre estructuras relacionales**, que permite registrar transformaciones de diseño aplicadas a tablas en el contexto de diseño de DataWarehouses.

  Relación con el presente proyecto. Estos proyectos fueron importantes para conocer el funcionamiento de un manejador de repositorios en general, y notar ventajas y desventajas de realizar una implementación CORBA, basada en Oracle.
  Fueron estudiados en el momento de decidir la estrategia a seguir en la implementación del manejador de repositorio. Como se decidió no continuar con una implementación basada en Oracle, (y por lo tanto no extender estos proyectos), no forman parte del "código fuente" del producto final.

Trabajos externos

- **dMOF (proyecto realizado por del Distributed Systems Technology Centre)**.
  dMOF [dMOF] es una "suite" de herramientas que se resumen en:
  1. Repositorio de meta-modelos dMOF. Es una implementación persistente de la especificación de repositorios de meta-modelos MOF establecida por la OMG.
  2. Compilador MODL. Permite cargar un meta-modelo MOF (especificado en el lenguaje MODL) en el repositorio de meta-modelos.
  3. Tools para intercambio de meta-modelos MOF. Permiten intercambiar meta-modelos MOF codificados como XMI.

4. Generadores de XMI. Producen un software que codifica y decodifica ***DTD***s permitiendo el intercambio de metadata entre usuarios vía XMI.
5. Generador MOF-***IDL***. Produce IDLs CORBA para repositorios de meta data.
6. Generador de moflet. Produce código java que implementa los meta-objetos que representan la metadata del usuario del repositorio.

Relación con el presente proyecto. Este conjunto de herramientas es la base de nuestro proyecto. Las mencionadas en los puntos 1, 3, 5 y 6 se utilizaron para generar el manejador de repositorio de meta-modelos CWM en CORBA. Se puede ver el presente proyecto como una extensión al producto generado por estas herramientas. Los generadores de XMI (punto 4) se utilizaron para generar los producer y consumer (antes mencionados) , clases que transforman meta-modelos almacenados en el repositorio como objetos CORBA en documentos XMI (y sus respectivos DTDs). Esto es la clave que permitió implementar una interfaz SOAP sobre el manejador CORBA, porque justamente es el XMI de los modelos lo que se envía a través de SOAP.

3. Especificación del producto

3.1. Resumen de funcionalidades

La funcionalidad ofrecida por el producto final puede clasificarse según el usuario de la misma:

Cliente CORBA que accede al manejador de repositorio en CORBA. Este tipo de cliente es el que puede acceder a toda la funcionalidad del manejador de repositorio:

- crear un paquete OLAP, agregarle elementos (jerarquías, dimensiones, etc), guardarlo en el repositorio.
- obtener un paquete del repositorio y obtener todos sus elementos, por ejemplo puede obtener una lista de las jerarquías que contiene un paquete OLAP.

Cliente que accede vía SOAP a través de la interfaz SOAP al repositorio. La funcionalidad provista a través de la interfaz SOAP consiste básicamente en dos métodos:

- LoadXMI, que permite obtener un paquete completo desde el repositorio en XMI.
- StoreXMI, que permite guardar un modelo en el repositorio. Recibe como entrada el modelo en XMI y el id del mismo, si el id ya existe realiza el update del modelo, su descripción y fecha de actualización, si no inserta un nuevo modelo.

También provee un método para obtener el listado de todos los modelos almacenados en repositorio.

Browser con el que se **navega** el repositorio. A través de un browser se puede acceder a la siguiente funcionalidad:

- Listado tipo fichero de los modelos que existen en el repositorio, pudiendo ver la descripción, identificador y fecha de actualización de cada modelo.
- Generación y obtención del XMI de los modelos del repositorio, pudiendo bajar (download) el XMI de cualquiera de ellos.
- Reporte en HTML de un modelo, que procesa la información del XMI y la lista en forma mas legible.

Estas funcionalidades pueden ser accedidas simplemente a través de un browser, sin requerir la instalación de ningún producto adicional en el cliente. Si se quiere visualizar el XMI de cada modelo (además de bajarlo) se necesita un browser como Internet Explorer que edita el XML en una forma fácil de entender o tener un editor de XML.

Browser con el que se **administran los servicios**. A través del browser es posible administrar los servicios, pudiendo en forma remota levantar o bajar servicios que corren en el servidor.

Herramientas CASE que utilizan las clases de ayuda al cliente (en Java). Las clases de ayuda al cliente lo que ofrecen es una facilidad para armar modelos OLAP en XMI.

3.2. Arquitectura

Se trata de un sistema distribuido el cual puede ser separado en los siguientes componentes (que constituyen las tres capas del sistema):

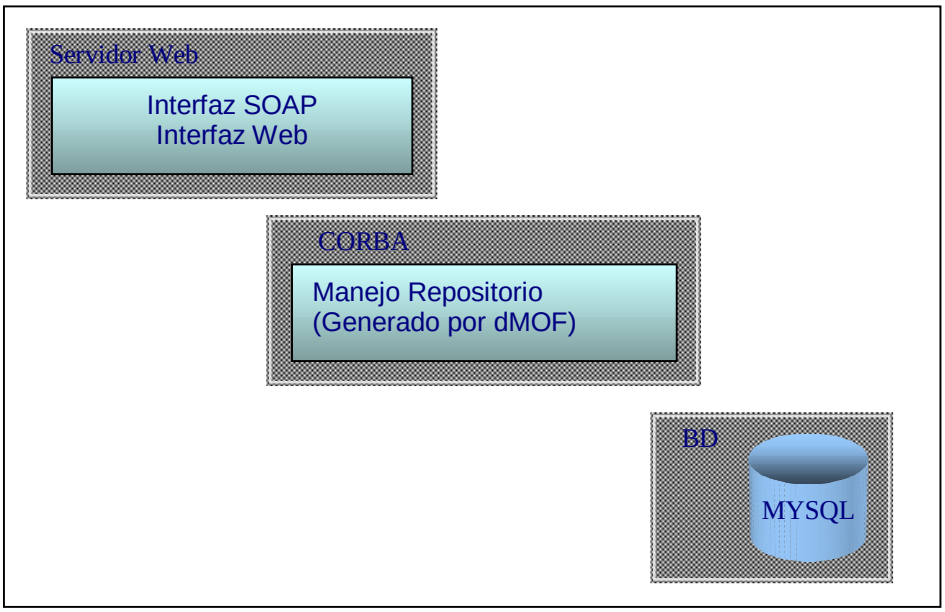


Fig. 3.1 Arquitectura del sistema

En la capa superior se tiene el servidor web con las interfaces SOAP y Web (de navegación del repositorio). A estas interfaces pueden acceder los usuarios y herramientas CASE. Están implementadas sobre el manejador de repositorios en CORBA el cual internamente tiene una arquitectura CORBA, por lo cual si es necesario también puede distribuirse a través de una Intranet por ejemplo.

El tipo de comunicación entre las herramientas CASE y la interfaz SOAP es a través del protocolo SOAP. Mientras que la comunicación con la interfaz Web del navegador es HTTP. Ambas interfaces se comunican con el manejador de repositorio a través del BUS de CORBA, por lo que dichas interfaces pueden estar corriendo dentro un servidor Web, y el manejador de repositorio en otro servidor. La base de datos (repositorio del manejador) es accedida vía JDBC.

Como componente adicional debemos incluir la parte del cliente que facilita la comunicación con el Servidor. Dichos componentes se encuentran relacionados de la siguiente forma:

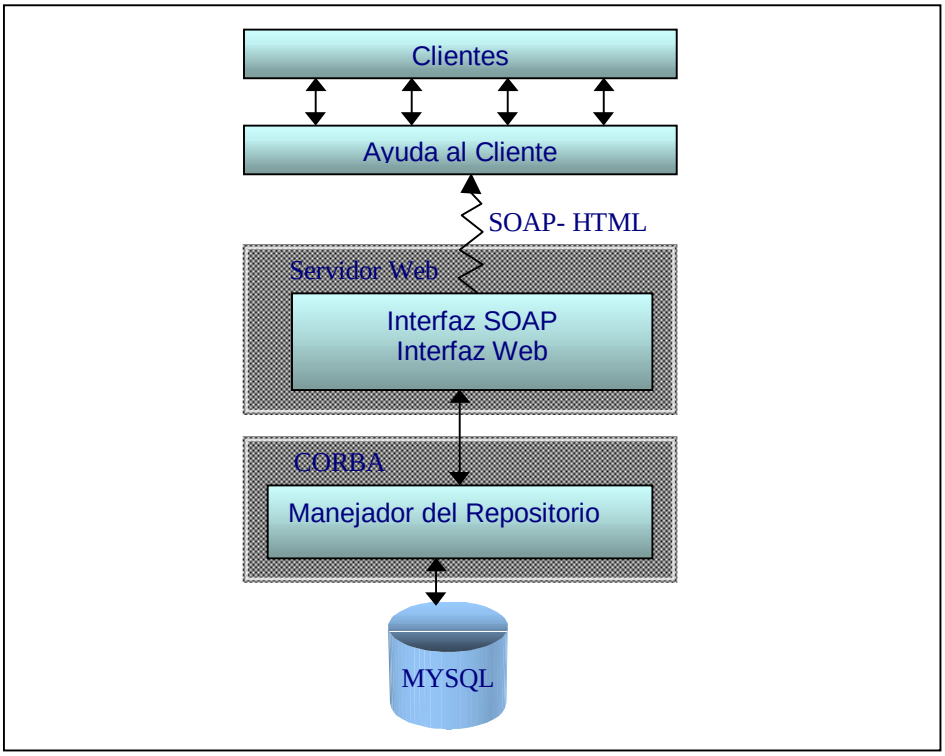


Fig. 3.2 Arquitectura del sistema

Interconexiones del Sistema

Cliente – Ayuda al cliente

La ayuda al cliente consta de una serie de clases java que hacen posible la traducción entre un meta-modelo y CWM. Por lo tanto el mecanismo de conexión entre las partes es la llamada a procedimientos java sobre los distintos objetos.

Ayuda al cliente – Interfaz SOAP

Esta comunicación es mediante SOAP. Es necesario aclarar que cualquier aplicación que quiera acceder usando SOAP lo puede hacer sin necesidad de las clases de ayuda al Cliente. Incluso puede no ser una aplicación JAVA. Esta independencia se logra por la utilización de SOAP como medio para la comunicación, cualquier aplicación existente, en cualquier lenguaje y plataforma, puede acceder a la URL en que atiende el Servidor vía SOAP y hacer Load o Store de un modelo.

Interfaz Web – Manejo del Repositorio

Esta interacción se realiza utilizando los stubs cliente generados por dMOF.

Manejo del Repositorio – Base de Datos(Repositorio)

Esta comunicación la establecen los distintos servidores CORBA con la configuración necesaria al levantarlos.
También se accede a la BD directamente desde el repositorio, para las páginas de navegación.

3.3. Análisis y Diseño

Para describir el análisis y diseño realizado en el proyecto se detallan a continuación (en el orden de suceso) las principales decisiones tomadas y las alternativas que existían a cada una de ellas, otra vez, separando en los "cuatro módulos" del proyecto comenzando por el manejador de repositorio en CORBA.

Manejador de repositorio

En el comienzo del desarrollo del manejador del repositorio existen dos opciones:

Extender los proyectos de la Facultad de Ingeniería mencionados en el capítulo 3 (basados en Oracle) para permitir el manejo de otros meta-modelos. Esta opción tiene la desventaja de ser dependiente de Oracle, lo cual es una restricción importante y de acuerdo a los objetivos planteados se decide descartar su uso.

Investigar e implementar otro tipo de solución que sea mas amplia, basada en estándares y fácilmente extensible. Se opta entonces por esta opción.

En la primer fase de investigación se evaluaron tres estándares para el manejo de metadata: ***OIM***, CWM y ***DWQ***.

OIM (Open Information Model). Esta es una especificación que actualmente esta integrada a CWM, en principio fue desarrollada por MDC (Metadata Colaition) pero posteriormente dicho consorcio se unió a la OMG definiendo una nueva versión de CWM que unificaba ambos estándares (la versión de CWM anterior y OIM). Evidentemente esta razón nos llevó a descartar la opción de OIM (principalmente porque OIM pasó a ser un modelo con poco soporte y en desuso).

CWM (Common Warehouse Metamodel). Es otra especificación que básicamente define tres cosas: la estructura y semántica de metadata compartida (definición basada en MOF), el formato de intercambio de la misma (en XMI), y las APIs de acceso a esa metadata compartida (con IDLs). Esta especificación es desarrollada por la OMG (así como los estándares MOF, XMI e IDL), y existe un buen soporte y documentación abundante.

DWQ (Data Warehouse Quality) es otra especificación que se tuvo en cuenta. Pertenece a un proyecto europeo que desarrolla técnicas para el diseño y operaciones de datawarehouse. Esta opción fue descartada porque resulto ser mas ventajoso CWM, el cual esta basado en estándares como MOF, XMI y UML. Además no se contaba con una descripción detallada de la especificación.

La razón fundamental por la que se elige CWM como meta-modelo para el repositorio es el hecho de que esté especificado en estándares y defina una interfaz para el acceso al repositorio así como una forma de intercambio de la información.
La especificación de CWM consiste de

- IDL's, cada IDL modela uno de los meta-modelos (paquetes) de CWM
- Documento XMI, es un único documento que contiene la especificación de todos los paquetes de CWM. Adicionalmente se tiene el DTD con el que valida dicho documento XMI.
- Modelo en MOF, que especifica las clases paquetes, asociaciones de los packages de CWM.

Una vez que se decide usar CWM se investigaron opciones para implementar el manejo del repositorio de meta-modelos basados en MOF y XMI. En el momento de dicha investigación solo se encuentra una herramienta que genera un manejador de repositorio persistente (en java y CORBA) de modelos basados en MOF: dMOF. Esta herramienta se adecua a CWM porque como entrada puede tomar el XMI de la especificación de CWM y como

salida genera en java y CORBA el manejador antes mencionado. (Este proceso se detalla en el Anexo C dMOF).
A través de este manejador es posible el alta, baja y modificación de los elementos de las diferentes instancias de meta-modelos. Adicionalmente tiene la ventaja de no estar ligado a una Base de Datos en particular (soporta tres bases de datos) y su distribución cuenta con una licencia del tipo universitaria.
En cuanto a la Base de Datos utilizada para almacenar el repositorio de modelos se analizaron dos bases de datos no comerciales: MySQL y PostGreSQL. Se eligió MySQL por ser mas sencilla de instalar y configurar y estar disponible de forma nativa para una amplia gama de sistemas operativos.

El esquema del manejador de repositorio generado por dMOF es el siguiente:

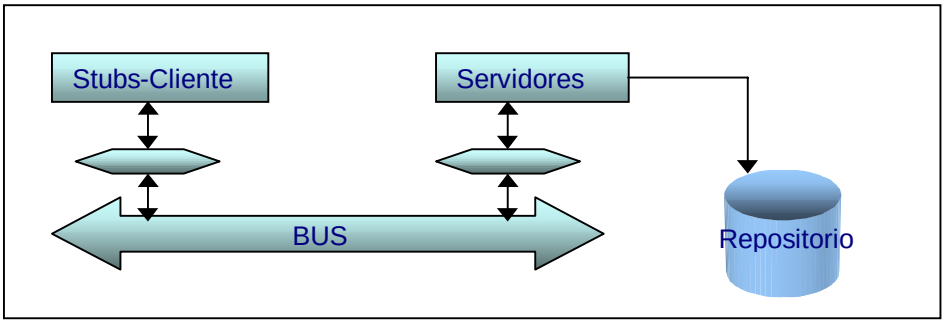


Fig. 3.3 Manejador de repositorio generado con dMOF

Acceso al repositorio vía SOAP

Una vez que contamos con el manejo del repositorio en CORBA para la especificación de CWM, es necesario poder accederlo remotamente. Se plantean entonces 2 opciones:

Usar un mecanismo CORBA. En esta opción se debe tener en cuenta los aspectos de seguridad intrínsecos de los clientes como ser firewalls de los mismos, en el cual generalmente se encuentran habilitados un numero reducido de puertos. Para solventar este problema se investigó la herramienta Gatekeeper de Borland Visibroker que permite la comunicación entre clientes CORBA y el/los Servidor/es CORBA utilizando un único puerto. Igualmente la opción de clientes CORBA fue descartada porque actualmente esta tecnología se esta viendo desplazada por otras.

Usar un mecanismo basado en XML. Aquí se estudiaron las alternativas XML-RPC y SOAP (por información detallada sobre ambas tecnologías ver anexo A –Conectividad).

Se decide utilizar SOAP por dos razones fundamentales, primero SOAP es una extensión de XML-RPC, lo cual hace que el mecanismo SOAP sea mas usado, y en general mas conocido. Y segundo SOAP permite enviar datos de tipos definidos por el usuario, lo cual no es posible a través de XML-RPC. Por ejemplo utilizando XML-RPC para Java los clientes solo pueden enviar datos de tipos predefinidos como ser String, Integer, y otros. En cambio con SOAP es posible además enviar un tipo de datos definido por el usuario, por ejemplo objeto de tipo dirección (que tenga atributos como calle, numero, ciudad).

Una vez que se elige SOAP hay que establecer que implementación se utilizará y como establecer el puente entre SOAP y los Stubs CORBA generados en java. La implementación de SOAP para java que se utilizó es la de Apache SOAP. Se eligió simplemente por ser open source, tener buen soporte y ser fácil de instalar y configurar.

Para establecer la conexión entre la interfaz SOAP y el manejador CORBA se investigaron proyectos relacionados, y se encontró el proyecto de SoapCorba Bridge [soapBridge]. El mismo permite de forma genérica acceder a cualquier interfaz CORBA desde SOAP y viceversa. Pero debido a que utiliza el ***ORB*** Orbacus , y para llevar la configuración a otro ORB (Visibroker) en ese momento había que investigar en detalle su código fuente se descarto esta opción, y se decide realizar un desarrollo a medida.

Hasta este momento se tiene el manejador de repositorios en CORBA y una interfaz SOAP que se quiere implementar sobre el anterior. Resta definir que tipo de datos se van a enviar en los llamados SOAP (un llamado SOAP esta en XML, pero el tipo de datos que se envía puede ser Strings, números, XML, y otros).

Lo que debe "viajar" dentro de los llamados SOAP son instancias de meta-modelos, es decir que las herramientas CASE van a necesitar enviar y recibir modelos a nuestro manejador de repositorios. Estas instancias de modelos se pueden modelar con:

- **los objetos CORBA** que las representan, que son parte de lo generado por dMOF ó
- **el XMI** correspondiente. Es posible obtener el documento XMI equivalente a cualquier modelo.

De acuerdo a esto, podemos entonces enviar y recibir modelos como objetos CORBA serializados, o como XMI. Veamos las ventajas y desventajas de cada caso:

Enviar objetos CORBA a través de SOAP implica implementar los serializadores y des-serializadores de cada uno debido a que no son java beans (en caso de serlo no es necesario, porque se pueden utilizar los serializadores de objetos que vienen en la implementación de SOAP). Por ejemplo el Objeto Dimension que modela una dimensión asociada a un Cubo del paquete OLAP, es un objeto CORBA que esta relacionado a una serie de objetos como nivel, jerarquía, restricción, esquema, y no es un java bean (como tampoco lo son nivel, jerarquía, restricción, esquema). Para poder enviar un dato de tipo Dimensión a través de SOAP, es necesario tener un serializador y des-serializador para el tipo Dimensión (y por lo tanto para nivel, jerarquía, restricción, esquema), es decir que se deben implementar a mano dichos serializadores. Debido a la gran cantidad de objetos que se tienen, implementar los serializadores para cada uno de ellos resultaba casi imposible, y muy poco flexible si después surge un pequeño cambio en la especificación de CWM ya que requeriría re-implementar la parte correspondiente.
Existe la serialización de objetos CORBA a través del método string_to_object que proveen los ORB, pero desde luego que para esto las llamadas deben ser CORBA y por lo tanto clientes no CORBA no podrían serializar un modelo y enviarlo al repositorio, solo podrían recibir modelos serializados en forma de un String (ilegible) que representa el IOR. Por otra parte se estarían enviando referencias a objetos del lado del servidor lo cual no es bueno desde el punto de vista de la seguridad.

Enviar el XMI de las instancias de los meta-modelos es sin duda la opción que resultó viable, ya que en este caso el XMI se envía como String dentro del llamado SOAP, evitando tener que implementar serializadores.

Pero para enviar los modelos en XMI debe ser posible la transformación de modelos en XMI a modelos como objetos CORBA y viceversa.
Precisamente esta conversión es otra de las funcionalidades que provee dMOF. Siguiendo el proceso que se detalla en el apéndice correspondiente a dMOF , se logran generar dos clases java para cada meta-modelo que permiten realizar los procesos de carga de una instancia del meta-modelo referido desde un archivo XMI al repositorio y viceversa. Estas clases se llaman de la forma <Nombre del Package>XMIConsumer y <Nombre del Package>XMIProducer respectivamente (OLAPXMIConsumer y OLAPXMIProducer en el caso de OLAP). El mecanismo que utilizan ambas clases es de tipo batch para leer el XMI completo obteniendo los elementos del modelo en el caso del Consumer , y para obtener todos los elementos del modelo con sus respectivos componentes y armar un archivo XMI en el caso del Producer.
Estas clases tienen los métodos consume y produce respectivamente, los cuales hacen el manejo entre un archivo y el repositorio. Lo que necesitamos para poder usar SOAP

directamente es cargar en el repositorio una instancia desde un String que contiene el XMI. Y esto es relativamente sencillo, basta con cambiar unas pocas líneas del código generado. Este procedimiento se puede extender para cualquier otro package .

Una vez que se decide enviar el XMI de los modelos a través de SOAP, se implementan en la clase denominada InterfazSOAP dos métodos para la carga y obtención de modelos: LoadXMI y StoreXMI. Dichos métodos serán las dos funcionalidades del repositorio a las que podrán acceder las herramientas CASE vía SOAP.

Adicionalmente para cada modelo se almacenan datos de cabecera como descripción y fecha de actualización.

En definitiva se logra tener:

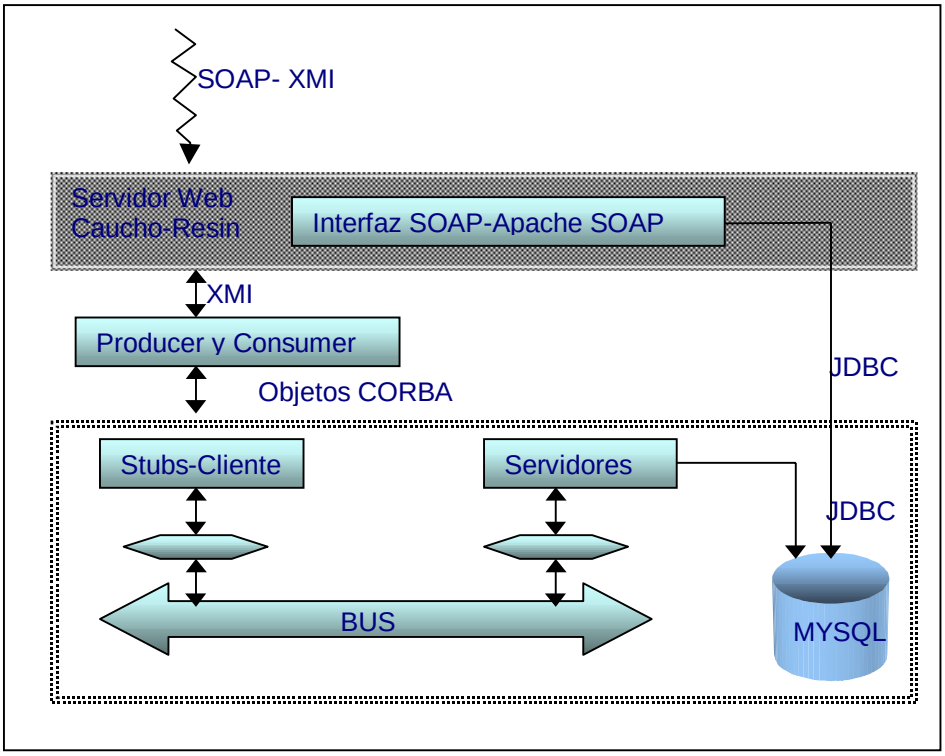


Fig. 3.4 Manejador de repositorio generado con dMOF

Navegador del repositorio y administración de servicios

Hasta aquí los requerimientos que establecen implementar el manejador de repositorios accesible desde cualquier herramienta CASE están cubiertos. Como requerimiento adicional, se planteó la necesidad de poder navegar el repositorio obteniendo información referente a los modelos allí existentes. La información que se quiere obtener consiste de:

Un listado tipo ficha de cada uno de los modelos mostrando datos de cabecera de cada uno como identificación, descripción y fecha de creación del modelo.

El modelo propiamente dicho haciendo un download del mismo y visualizarlo completamente de alguna forma.

En primer lugar el mecanismo de acceso a esta funcionalidad debe poder ser simple y remoto, por eso es que se decide implementar una interfaz web que permita navegar el repositorio a través de un browser. Para el tipo de información requerida en el listado, basta con acceder a la información guardada por la InterfazSoap en la BD para obtener los datos de cabezal de cada modelo almacenado.

El segundo punto no es tan directo, ya que en la base de datos (repositorio) se tienen almacenados objetos serializados, tanto el manejador de repositorios como la interfaz SOAP almacenan Strings correspondientes a instancias (objetos) serializadas, y por lo tanto los modelos no existen en la BD como datos legibles.
El navegador de repositorios debe listar información legible para el usuario, no un String que represente un objeto y el cual carece de sentido para un usuario. Entonces la alternativa es mostrar el XMI de cada modelo, lo cual sí es entendible y es posible obtener a través de la clase Producer. Pero el XMI de un modelo a pesar de ser legible, no es fácilmente entendible a simple vista, y peor aún si el usuario no conoce XML. Por esta razón es que se decide agregar una segunda forma de edición de modelos, entendible fácilmente por cualquier usuario, y obtenible a partir del XMI. Esa forma es generar un reporte en HTML a partir del XMI aplicándole una transformación XSL. Ese reporte debe mostrar el XMI ya procesado, por ejemplo no debe mostrar las referencias internas del XMI entre dos elementos si no reemplazar las referencias por el elemento al que se hace la referencia.

Para la administración de servicios se decide agregar a este módulo la opción de levantar y bajar servicios que corren en el servidor de repositorio en forma remota (pantalla de "Administración del sistema"). Esta opción permite una forma mas simple de levantar los servicios necesarios para el funcionamiento del servidor. Dichos servicios son por ejemplo: MySQL, el servidor de OLAP (generado por dMOF).

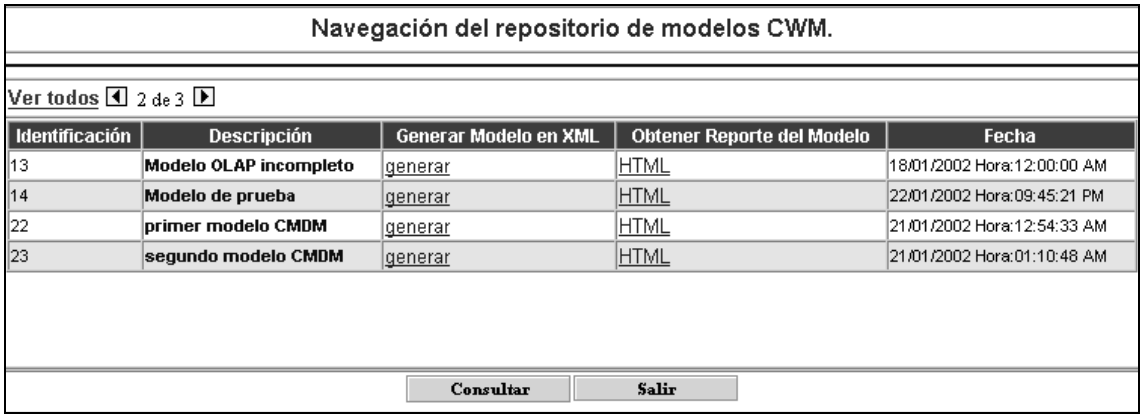


Fig 3.5 Pantalla para la navegación del repositorio de modelos

Fig. 3.6 Pantalla para la administración de servicios

Resumiendo se tiene entonces el esquema de este modulo:

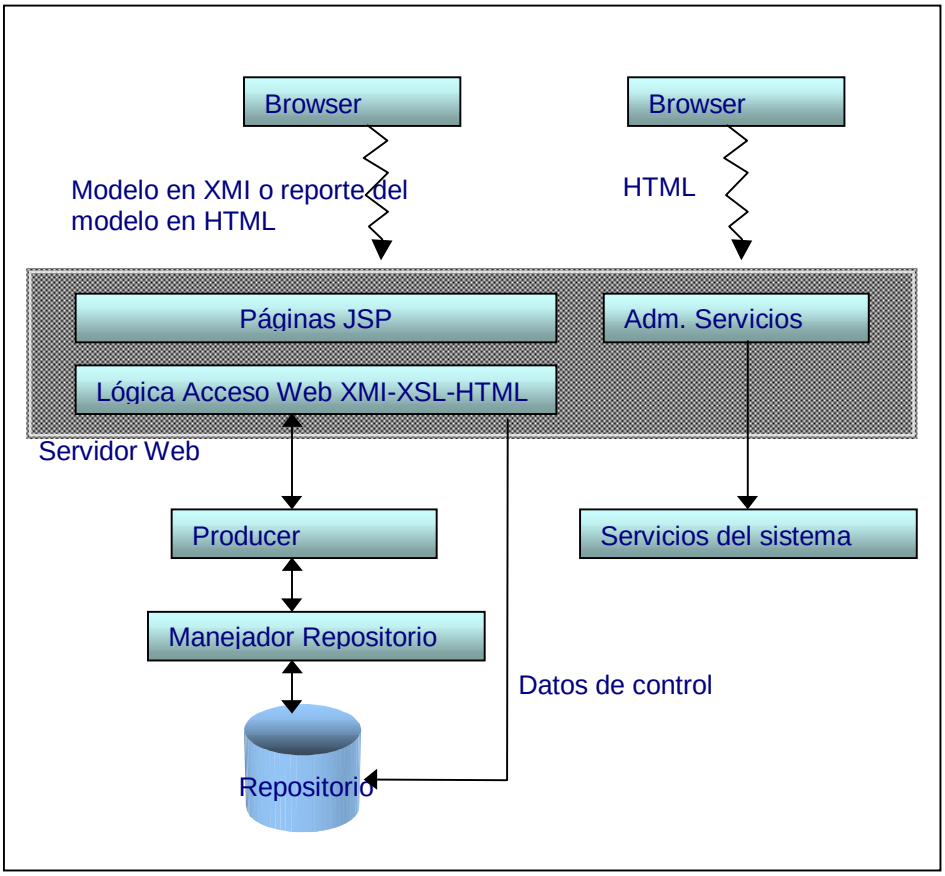


Fig. 3.7 Interfaz de acceso Web al repositorio y administración remota de servicios

Aplicación: editor CMDM y clases de ayuda al cliente

Con los módulos de Manejo del Repositorio y la interfaz SOAP se logra que cualquier aplicación se pueda comunicar usando SOAP obteniendo y guardando instancias de los meta-modelos del repositorio y con el navegador de repositorio acceder a través de un browser a la información y listado de dichas instancias.
Queda por tratar la aplicación del repositorio al prototipo de herramienta CASE para CMDM. Este punto involucra varios pasos:

1. **Investigación del modelo CMDM**, y establecimiento de correspondencias con el package OLAP de CWM. El estudio de correspondencias se realizó en primer lugar entre el paquete OLAP CWM y la especificación de CMDM realizada por el Ing. Fernando Carpani. (Por mas detalle ver Documentación Técnica).
   Pero esto no fue suficiente porque el editor de CMDM implementa un subconjunto de la definición de CMDM mencionada, e incluso se modifican algunas características, por ejemplo se agregan restricciones locales a los elementos que contienen información sobre la presentación gráfica de los mismos (color, posición, ángulo). Por eso se realizó después un segundo estudio del mapeo entre CWM y el CMDM que implementa el editor.

2. **Investigación del código fuente del editor**. Se tiene entonces un editor de modelos CMDM en java, y se quiere aplicar el manejador de repositorio de modelos CWM a esta herramienta, es decir se quiere que dado un modelo en CMDM construido con el editor sea posible guardarlo en forma remota (en el servidor de repositorio, vía SOAP) como

un modelo CWM. Para eso es necesario transformar ese modelo CMDM a uno CWM y guardarlo a través de la interfaz SOAP en el repositorio.
El proceso "de ida" es entonces: se crea un modelo CMDM en el editor, se mapea a CWM, se envía como XMI y se guarda en el repositorio.
El proceso de vuelta es: se obtiene un modelo del repositorio en XMI desde el editor, se hace el mapeo de CWM a CMDM y se carga en el editor (ver el esquema que aparece mas adelante).
Este editor implementado en java representa internamente la estructura de un modelo CMDM como un árbol de objetos. Y permite guardar cada modelo en un archivo de extensión .emd que contiene la información en un formato definido por este editor (basado en las marcas begin-end).
Se necesita convertir un modelo de CMDM a una "estructura que este según el modelo CWM" y que sea fácil de mapear a XMI. Para eso se puede:

- Convertir el archivo .emd a una estructura para CWM
- Convertir la estructura de árbol a una estructura para CWM

La primera opción tiene la desventaja de que resulta mas lenta que la segunda, porque implica transformar la estructura de árbol a un String en formato emd (proceso que ya hace la herramienta cuando va a guardar un modelo), escribir ese String en un archivo (.emd), luego leer ese archivo y transformarlo a XMI. O bien se puede tomar el String en emd y generar el XMI. En cambio la segunda opción ahorra un paso: el de transformar el árbol a emd, porque se transforma la estructura de árbol directamente a XMI. Se elige entonces esta opción.

3. **Implementación de clases de ayuda al cliente** para realizar los mapeos entre modelos. En general la "estructura según CWM" se va a necesitar siempre que se quiera hacer un mapeo de un modelo X a CWM, y debe ser independiente del modelo X de que se trate. Por eso es que se decide implementar un conjunto de objetos en java para modelar la estructura de un OLAP CWM como "ayuda a los clientes" ya que lo que se tiene generado con dMOF es una estructura de OLAP pero en CORBA. Se suma a esta ayuda al cliente los parsers que transforman los objetos para representar el OLAP de CWM a XMI y viceversa. Esos objetos consisten de java beans que se corresponden con los componentes del paquete OLAP de CWM, por ejemplo tenemos los beans DimensionModel, JerarquiaModel, NivelModel. (Por mas detalle ver diagrama de clases en la documentación técnica). Cada uno mantiene referencias a los objetos con que se relaciona para facilitar después la generación del XMI, por ejemplo DimensionModel contiene una lista de referencias a las jerarquiasModel que contiene, y al EsquemaModel al que pertenece. Los parsers son dos clases muy parecidas al producer y consumer generados con dMOF en cuanto a funcionalidad, salvo que en este caso se trata de la transformación entre objetos (java beans) y XMI (y no entre objetos CORBA y XMI). Aquí se utilizan los paquetes de java para el manejo de documentos XML.

4. **Adaptación del editor.** Teniendo entonces esas clases de ayuda al cliente lo que resta por hacer es integrarlas al editor. Para esto, se agrega al editor un método que toma como entrada la estructura de árbol de un modelo CMDM y construye el modelo OLAP de CWM con los java beans mencionados, e inversamente uno que dada la estructura del OLAP construya el árbol correspondiente. Luego se agregan los métodos de invocación vía SOAP al repositorio. Las acciones que se agregan al menú del editor CMDM son
    - Insertar un modelo nuevo en el repositorio
    - Actualizar un modelo existente en el repositorio
    - Cargar un modelo desde el repositorio

    Las dos primeras acciones invocan al método StoreXMI. En el caso de actualización se realiza un llamado SOAP previo para obtener un listado de los modelos que existen en el repositorio (con sus respectivas descripciones) para que el usuario seleccione uno. La ultima acción invoca el método LoadXMI de la InterfazSOAP, el cual retorna el id asignado.

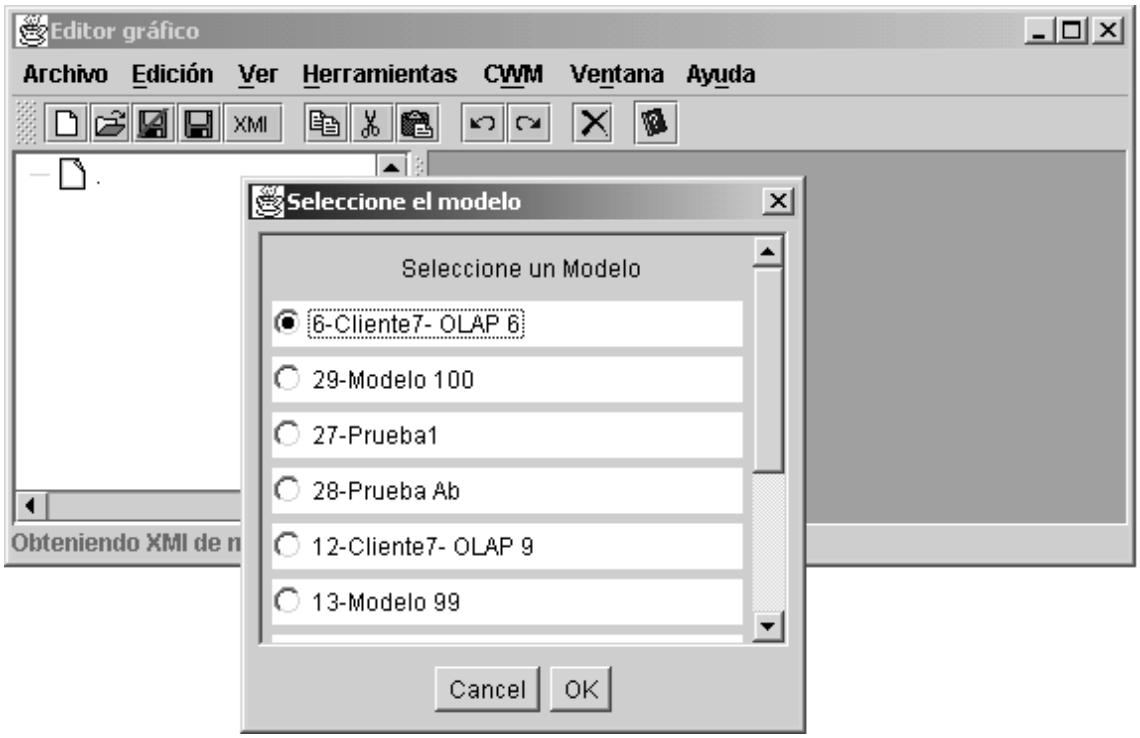


Fig. 3.8 Acciones agregadas al menú de l editor CMDM que permiten integrar el editor con el manejador de repositorios

Fig. 3.9 Acciones agregadas al menú de l editor CMDM que permiten integrar el editor con el manejador de repositorios

Se agregó además la opción a este editor de guardar un modelo en CMDM como XMI (modelo en CWM) en forma local, y viceversa. Esto es parte de la funcionalidad antes descripta, realiza el mismo proceso, salvo que a la hora de guardar no hace un llamado SOAP si no que lo guarda en un archivo local de extensión XML. Al cargar un archivo .XML realiza el proceso inverso pero obteniendo el XML desde el archivo local y no a través de la invocación SOAP a la función LoadXMI.

Fig. 3.10 Acciones agregadas al menú de l editor CMDM que permiten integrar el editor con el manejador de repositorios

Gráficamente se puede representar como:

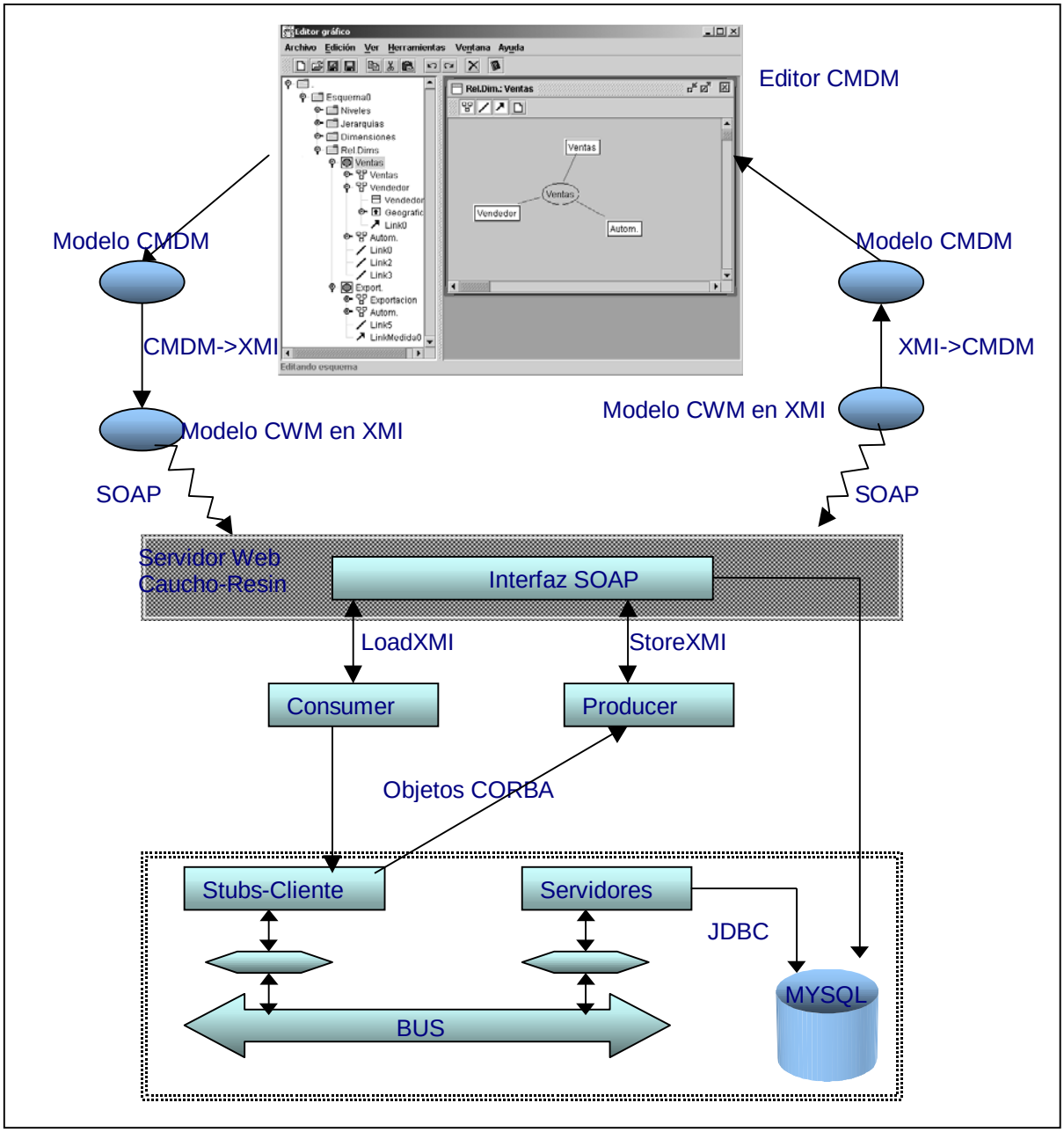


Fig. 3.11 Aplicación de repositorio a una herramienta CASE

En cualquier otra herramienta que trabaje con un modelo X y se quiera implementar su acceso al repositorio CWM el proceso a seguir es el mismo:

1. primero estudiar la correspondencia entre el modelo X y CWM
2. luego estudiar la estructura interna de representación de X
3. implementar los dos métodos de conversión de X (en la estructura de 2.) a XMI (en CWM) según la correspondencia definida en 1.
4. agregar los métodos como acciones en el menú que invocan los de 3. y envían el llamado a través de SOAP al repositorio.

4. Implementación

4.1. Arquitectura Tecnológica

Podemos separar las diferentes tecnologías utilizadas según los "módulos" diferenciables del proyecto.

El manejador de repositorios es una implementación CORBA en java. Accede a la base de datos MySQL (repositorio) vía JDBC. Los clientes (tanto clientes CORBA como la interfaz SOAP) acceden a él con llamados CORBA.

Por otra parte la interfaz SOAP esta implementada también en java, y la comunicación con las herramientas es utilizando el protocolo SOAP.

La lógica que implementa el acceso (consulta) web al repositorio esta desarrollada en java, mientras que las páginas que componen la vista se basan en la tecnología ***JSP*** (java server pages) y utilizan además el framework de *Struts* que implementa el modelo MVC para el web, y consiste de una serie de librerías y meta-tags de HTML que facilitan el desarrollo de las paginas JSP y la lógica que existe por detrás de las mismas. En este modulo también se utilizan los paquetes de java de manejo de XML para el parseo de estos documentos, y XSL para la transformación de dicho documentos XMI en HTML.

Tanto la interfaz SOAP como la de navegación del repositorio corren en un servidor web, y acceden a la base de datos vía JDBC.

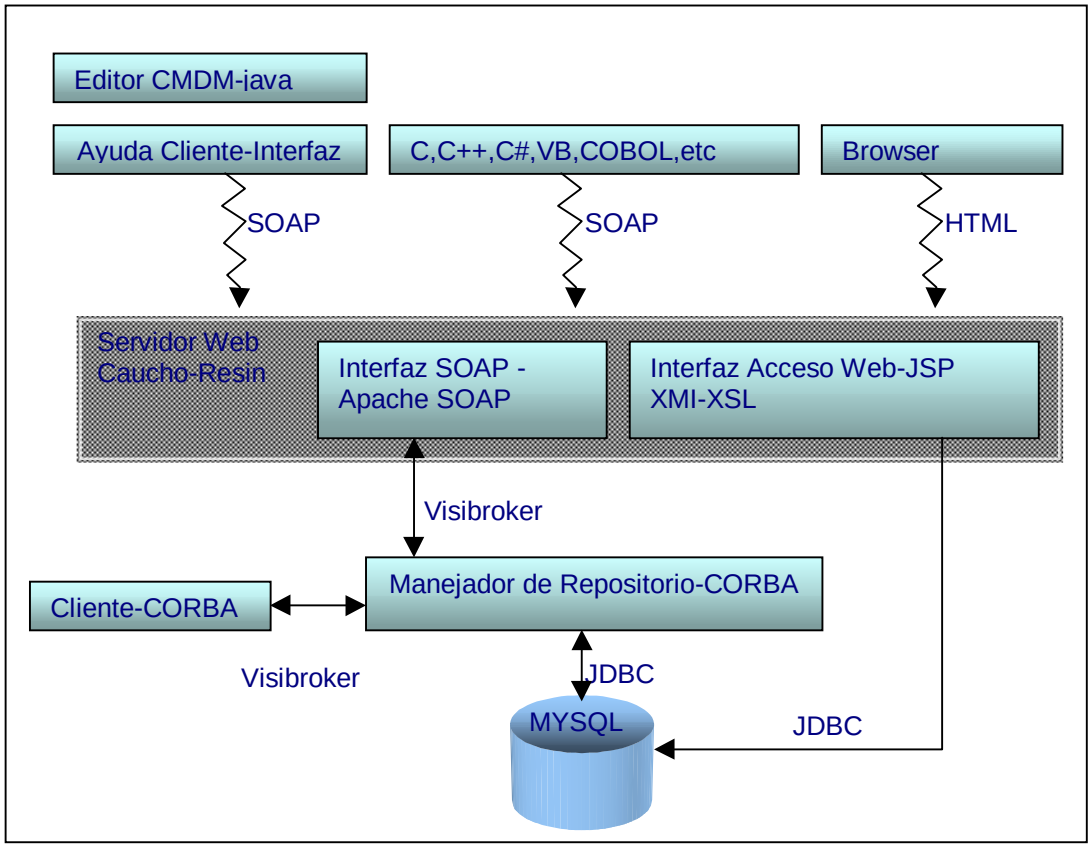


Fig. 4.1 Arquitectura Tecnológica

4.2. Productos empleados

Los productos utilizados para implementar la arquitectura mencionada son los siguientes:

- **JDK 1.3.0.** Java Development Kit de Sun Systems versión 1.3.0. para todo lo implementado en java. [sun]
- **VisiBroker for Java 4.5** de Borland Inprise como implementación de ORB para la parte implementada en CORBA.[inprise]
- **dMOF 1.1** del Distributed Systems Technology Centre (DSTC) de la Universidad de Queensland Australia.[dstc]
- **MySQL 3.23.38** como base de datos para el repositorio.[mysql]
- **Resin 2.0** de Caucho Developer Source License como servidor web el cual soporta Java Server Pages versión **(JSP)1.2** [caucho] [jsp-resin]
- **Struts 1.0** de Apache Software Foundation como framework para la implementación del navegador del repositorio y administración de servicios. [struts]
- **Apache SOAP** de la Apache SOAP Community como implementación de SOAP versión 1.1. [soap]
- **Xerces Java Parser 1.4.4** de Apache Soap el cual soporta la especificación de XML version 1.0 [xerces]
- **JAXP** [jaxp]
- **Xalan 2.2** como procesador de XSLT para transformar documentos XML en HTML [xalan]
- **XMLSpy 3.5** como editor de XML
- **Jbuilder 4 Enterprise** de Borland como editor de código java.

4.3. Estructura de clases

Las clases generadas por dMOF siguen una estructura por paquetes. Para cada paquete de CWM existen dos packages, uno que contiene los stubs y skeletons de los objetos CORBA que modelan ese paquete y otro que contiene el servidor de ese paquete (meta-modelo). Por ejemplo para el paquete OLAP de CWM se tienen dos packages: Olap y OlapImpl , el segundo es el que contiene el servidor de OLAP y la implementación de los stubs y skeletons que se encuentran en el package Olap. Las clases de Olap son generadas a partir de las IDL de Olap, las cuales a su vez se obtienen del XMI de CWM con la tool mof2idl de dMOF. Las clases de OlapImpl se obtienen a partir del XMI con la tool de dMOF mof2moflet. La siguiente figura muestra los 2 paquetes generados para OLAP, y algunas de las clases que existen en cada uno de ellos.

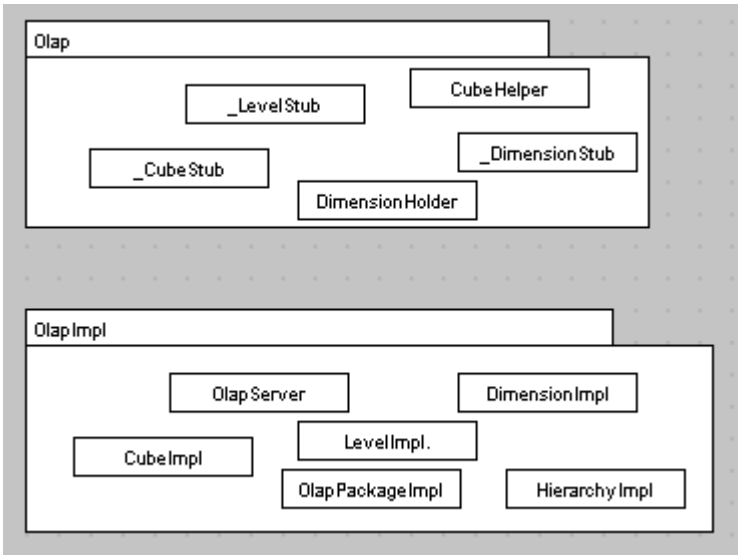


Fig. 4.2 Algunas de las clases generados por dMOF para OLAP

Las clases desarrolladas en el proyecto se encuentran distribuidas en los siguientes paquetes (para ver interrelación entre las clases ver diagramas de clases en documentación técnica).

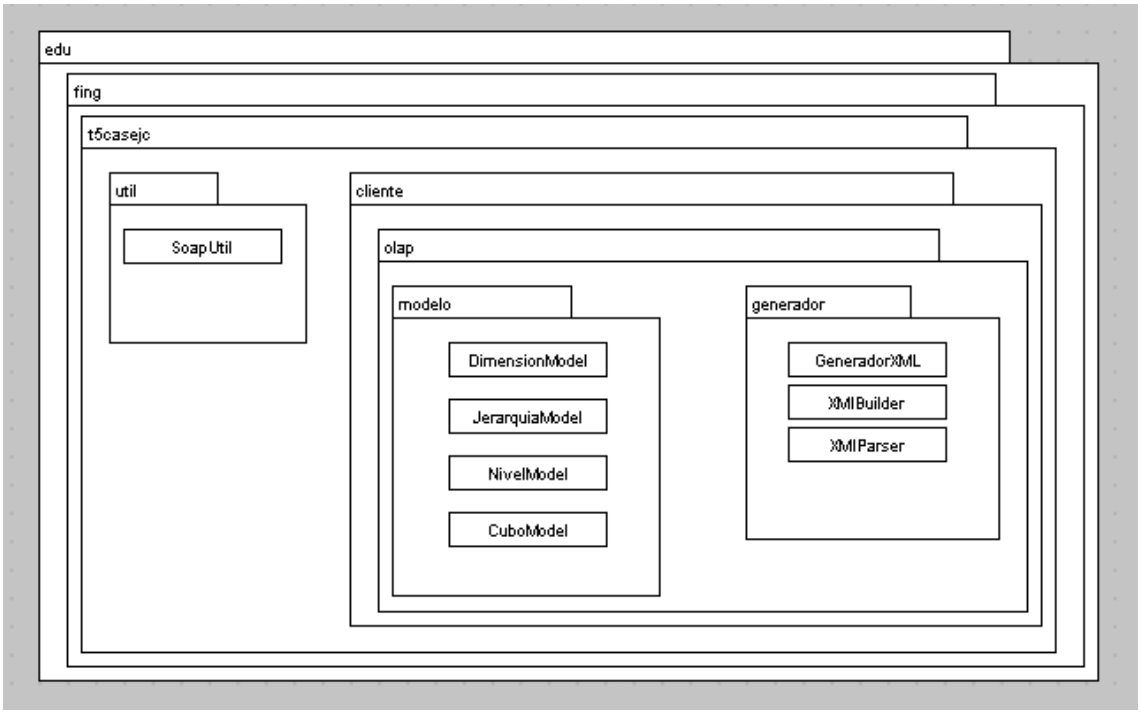
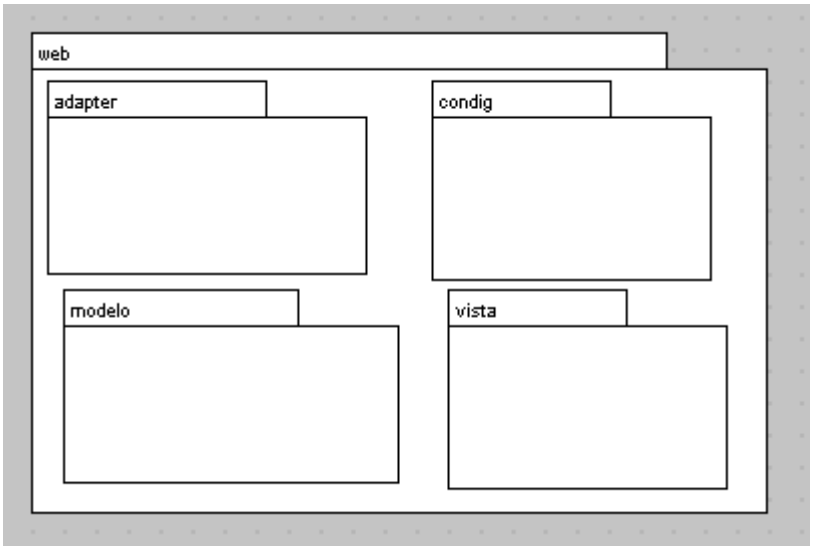


Fig. 4.3 Estructura de paquetes

Los paquetes util y cliente bajo edu.fing.t5casejc contienen las clases de “ayuda al cliente”, la figura anterior muestra algunas de las clases que contiene el paquete modelo. Las mismas son las que modelan los elementos de CWM comunes con CMDM.

La clase GeneradorXML implementa los métodos para convertir un modelo (en CWM, formado con los objetos en el paquete modelo) a XMI. Y utiliza para esto las clases XMIBuilder y XMIParser.

El paquete útil contiene la clase SoapUtil que tiene métodos para facilitar al cliente el armado de un llamado SOAP.



En el paquete web se encuentran las clases que implementan el navegador de repositorio y administración de servicios. En el paquete adapter se encuentran las clases que manejan los eventos del usuario (en las paginas JSP). En config y modelo estan las clases de configuración de la base de datos, y modelado de la información que se despliega en las paginas. Vista contiene las clases que mapean con las paginas JSP (con los forms).

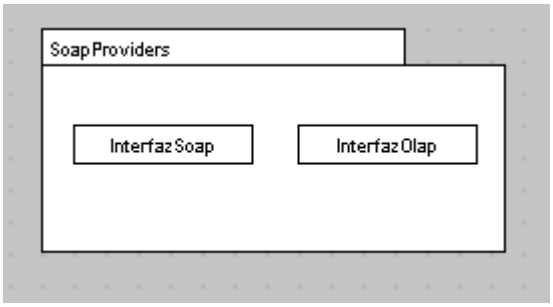


Fig. 4.5 Estructura de paquetes (SOAP)

Se tiene también otra estructura de clases para el proyecto del editor de CMDM, la cual es la original del mismo mas una clase que fue agregada la cual agrega las acciones relacionadas con CWM.

5. Conclusiones

5.1. Síntesis de lo hecho

En síntesis, a lo que se llegó con este proyecto es a una solución al problema de intercambio de metadata que afronta la industria de datawarehouse, de herramientas de análisis y administración de datos en general. Se implementó un repositorio de meta-data que implementa un único meta-modelo. Elegir CWM como dicho meta-modelo, canónico, presenta una gran ventaja por estar basado en estándares como UML, MOF y XMI. En particular UML para el modelado resulta ser simple de usar y entender. Que este diseñado de acuerdo a MOF nos permitió usar otras especificaciones dependientes de MOF las cuales fueron fundamentales en el desarrollo del proyecto como son:

- XMI para intercambiar meta data warehouse que esta representada usando el meta-modelo de CWM e
- IDL para programar acceso a meta data de warehouse basada en el meta-modelo de CWM.

Basados en el manejador de repositorios de meta-modelos de CWM en CORBA se logró implementar un acceso vía SOAP que amplia el rango de clientes que pueden acceder al mismo.

5.2. Aportes

Si bien existen otros manejadores de repositorios desarrollados en proyectos anteriores, éstos están desarrollados en CORBA o están basados en las herramientas de Oracle. Por esta razón uno de los aportes fundamentales de este proyecto es permitir que las herramientas CASE que utilicen el manejador de repositorio no deban implementar una interfaz CORBA, dando la libertad de que puedan estar implementadas en el lenguaje y plataforma que se quiera.

Se introdujeron estándares para la representación e intercambio de información como MOF y XMI. MOF para la especificación de modelos, XMI para el intercambio entre herramientas CASE y el repositorio .

Otra característica importante a destacar del proyecto es que permite navegar el repositorio de modelos e incluso obtener un modelo (download) en XMI a través de un browser, lo que no limita a obtener información del repositorio sólo a través de una herramienta CASE, si no que se puede acceder aunque no se utilice un CASE.

En cuanto al aspecto tecnológico se utilizó SOAP, XMI, XML, JSP que hasta el momento no se habían empleado en los proyectos similares y son muy importantes debido al gran uso que tienen actualmente en aplicaciones en general.

Se estudió el producto dMOF que forma parte de un proyecto en la Universidad Queensland Australia y se mantuvo una buena relación con los participantes del mismo, pudiendo intercambiar también conocimientos, y dejando abierta la posibilidad de continuar haciendo trabajos en conjunto.

5.3. Balance

Este proyecto tuvo una importante dedicación a la investigación y estudio de tecnologías así como de herramientas y productos. La implementación de lo que sería el

código final fue una parte menor. El aporte más importante de este proyecto es la experiencia en

las tecnologías estudiadas que se adquirió, y el entendimiento de los conceptos relacionados al mundo de modelos, meta-modelos, y datawarehouse en general, llegando a entender los diferentes niveles de abstracción que se encuentran en cada uno, y las diferentes posturas que existen para modelar un datawarehouse, y manejar repositorios de modelos.

En cuanto al meta-modelo utilizado, CWM, si bien es “muy nuevo” aún, aparenta ser “el estándar” para el intercambio de metadata en el futuro. En primer lugar porque esta basado en XML y en UML que es muy usado actualmente por ser un lenguaje muy poderoso para el modelado de todos los tipos de metadata (como ser relacional, basado en registros, orientado a objetos, XML, etc).
Por otra parte CWM es ampliamente suficiente para modelar un datawarehouse integro y se pueden utilizar herramientas visuales para construir modelos en CWM (por ejemplo cualquiera que soporte UML). Es también una especificación muy sólida, ya que esta diseñada por expertos en metadata, bases de datos y data warehouse, como son IBM, Unisys, Oracle, NCR, Union Bank of Switzerland y Sun Microsystems.

Por otro lado actualmente CORBA se ve desplazado por otro tipo de tecnologías, por lo cual es dudoso (a futuro) el uso de la interfaz CORBA en un repositorio de modelos como objetos CORBA. Es decir el acceso al repositorio por parte de clientes CORBA no aparenta ser una forma muy utilizada.

Si bien en este proyecto se utilizó la herramienta dMOF que genera código CORBA, y se agrego una capa SOAP, es de notar que una solución similar en cuanto a funcionalidad se habría logrado con un generador de repositorios en java (local, sin utilizar CORBA, eliminando la dependencia de Visibroker y resultando mas performante) y una capa SOAP similar (que podría formar parte del código generado, para ser mas mantenible). Evidentemente esta solución requiere mas tiempo que el disponible para un proyecto de grado.

5.4. Trabajos futuros

A medida que se realizo el proyecto surgieron los siguientes puntos a tener en cuenta para los trabajos futuros:

Editor grafico de modelos CWM.

Implementar un editor que gráficamente permita construir modelos de CWM en XMI, ya que manejar el XMI de un modelo en CWM con un editor de XML es poco natural para un usuario común. Hasta ahora lo que se tiene son las clases de ayuda al cliente, las cuales como se mencionó podrían requerir algún agregado para cubrir todos los elementos, esto seria entonces el punto de partida para implementar el editor de modelos en XMI. Existen editores de MOF que generan XMI, pero seria bueno poder trabajar a nivel gráfico directamente con XMI.

Extensión del repositorio

En la implementación actual se llegó a implementar y a “poner en producción” el paquete OLAP de CWM lo que involucra las partes de invocación SOAP y de acceso por navegador.

Una posible extensión del presente proyecto es implementar el servidor y navegador de repositorios de todos los meta-modelos de CWM generando los generadores de XMI y extendiendo la interfaz SOAP para el resto de los meta-modelos (lo cual es un proceso bastante sencillo partiendo de que ya existe para OLAP).

Manejador de repositorio en java (sin utilizar CORBA).

Sería muy interesante implementar un generador de código muy similar al de la herramienta dMOF pero que genere código java, para acceso local al repositorio, no a través de un ORB como en CORBA. Incluso seria muy bueno hacer esta implementación en conjunto con el grupo de desarrollo de dMOF, porque es una implementación muy similar a la que ellos ya realizaron, con la misma funcionalidad, mas simple aún (si se conoce la que ya existe para CORBA).
Incluso, si se desea que el manejador siga teniendo una arquitectura distribuida (pero no CORBA, como el de dMOF), se podría generar el mismo con ***EJB***'s.

Extensión del manejo del repositorio a través de SOAP

Se implementó una capa que brinda una conexión entre los clientes y el repositorio. El manejo de las instancias es indivisible, o sea , que se manejan instancias completas y no elementos. La extensión de la Interfaz Soap podría ser en dos aspectos:

- manejo de seguridad. La seguridad implementada actualmente es nula en cuanto al control de accesos a las instancias. Se podría agregar ese control, tanto para las actualizaciones, consultas o inserciones.
- manejo de elementos. Es posible realizar el manejo de elementos que forman parte de la instancia si se agregan componentes para manejar la sesion, los objetos, y la seguridad entre los objetos utilizados y los clientes que están accediendo.

Como sugerencias se pueden establecer los siguientes puntos:

- Implementar una serie de interfaces para los distintos elementos a ser utilizados las cuales serán las encargadas de hacer el manejo del almacenamiento y del llamado a los distintos objetos corba.
- Reutilizar las clases generadas por dMOF para el manejo de XMI.
- Establecer un sistema de independencia de los objetos utilizados entre clientes.
- Estudiar en detalle el funcionamiento de SOAP BRIDGE.

5.5. Referencias

[activation] http://java.sun.com/products/javabeans/glasgow/jaf.html
[apacheSoap] http://xml.apache.org/soap/
[caucho] http://www.caucho.com
[cmdm] http://www.fing.edu.uy/~csi/Proyectos/DwDesigner/index.html
[cmdm-proy] http://www.fing.edu.uy/inco/cursos/ingsoft/web2001/index2.htm ->entrar a Documentos -> Requerimientos del Sistema
[cwm] http://www.omg.org/technology/cwm/index.htm
[dMOF] http://www.dstc.edu.au/
[dstc] http://www.dstc.edu.au/AboutDSTC/index.html
[especCORBA] http://www.omg.org/technology/documents/specifications.htm
[idlCompiladores] http://www.omg.org/technology/documents/formal/corba_language_mapping_specs.htm
[implCORBA] http://www.cetus-links.org/oo_object_request_brokers.html
[implCORBAFree] http://www.omg.org/technology/corba/corbadownloads.htm
[implSoap] http://www.xmethods.net/ve2/ViewImplementations.po
[inprise] http://www.inprise.com
[instSOAP] http://xml.apache.org/soap/docs/index.html
[JavaMail] http://java.sun.com/products/javamail/
[jaxp] http://java.sun.com/xml/jaxp/index.html
[JMS] http://java.sun.com/products/jms/index.html
[jsp-resin] http://www.caucho.com/products/resin/ref/jsp.xtp
[mysql] http://www.mysql.com/
[resin] http://www.caucho.com/products/resin/
[RMI] http://java.sun.com/products/jdk/rmi/index.html
[soap] http://xml.apache.org/soap/index.html
[soapBridge] http://soap2corba.sourceforge.net/
[soapCauca] http://comdist.ucauca.edu.co/resultados.htm
[struts] http://jakarta.apache.org/struts/
[sun] http://java.sun.com/
[tomcat] http://jakarta.apache.org/tomcat/index.html
[uml-mof-xmi] http://www.javareport.com/html/from_pages/oldarticles.asp?ArticleID=1246
[xalan] http://xml.apache.org/xalan-j/index.html
[xerces] http://xml.apache.org/xerces-j/

5.6. Glosario

Herramienta CASE

Computer Aided Software Engineering. Un CASE es un conjunto de estándares, especificaciones, recomendaciones dedicadas a: métodos (por ejemplo análisis e ingeniería en si mismos), metodologías (tales como SADT, SSADM), notaciones (como BNF, DFD, STD para esquemas, diagramas y cuadros de flujo), y configuración, y administración de proyectos (que ayuda a los desarrolladores dentro del diseño (de un modelo), desarrollo (codificar programas) y soporte. Un software que implemente todas las técnicas de CASE es demasiado grande y complejo, por eso la mayoría de las herramientas CASE permiten implementar solamente algunos estados, como por ejemplo ERwin y BPwin que implementan solo algunos estados como modelado de datos (diagramas entidad relación con ERwin) o modelado de procesos del negocio (diagramas de transición de estados usando BPwin). Algunos otros ejemplos de herramientas CASE son: Embarcadero ER/Studio 5 (para el modelado de datos), Embarcadero ER/Studio 5 (para el modelado de datos orientados a objetos), Oracle Designer 2000, Oracle Designer 2000 y Popkin' System Architect (modelado de proceso y datos)

DTD

Un Document Type Definition define elementos de metadata, su orden, estructura, reglas, y relaciones. Un DTD permite procesar automáticamente en una forma uniforme diferentes instancias de documentos del mismo tipo.

DWQ

Data Warehouse Quality es un proyecto europeo que desarrolla técnicas para el diseño y operaciones de datawarehouse.

EJB

Enterprise JavaBeans es una especificación realizada por Sun, que define una arquitectura para que componentes escritos en el lenguaje Java, corran en la parte servidor de una red que usa el modelo cliente/servidor.

IDL

Interface Definition Language. En CORBA la interfaz de un objeto es definida en IDL. Dicha definición especifica los métodos que el objeto esta preparado para realizar, sus parámetros de entrada, su resultado y cualquier excepción que pueda generarse durante la ejecución. Toda la información necesaria para construir un cliente del objeto es proporcionada por la interfaz.

IOR

Interoperable Object Reference. Contiene información acerca del protocolo y ubicación de un objeto remoto.

I-CASE

Integrated Computer Aided Software Engineering

JSP

Java Server Pages es una tecnología especificada por Sun que permite desarrollar y mantener paginas web dinámicas en forma fácil. Las JSP utilizan tags y XML scriptles en Java para encapsular la lógica que genera el contenido de las paginas.

Meta-modelo

Un modelo que define el lenguaje para expresar un modelo. Una instancia de un meta-meta-modelo.

Modelo

Una abstracción semánticamente consistente de un sistema. Ver: sistema

MOF

Meta Object Facility es una especificación de la OMG para estandarizar la representación de repositorios de meta data. Define como representar y manipular metadatas.

CMDM

Modelo conceptual para la especificación de bases multidimensionales.

CORBA

Common Object Request Broker Architecture es la especificación de la OMG para la interoperabilidad de objetos distribuidos.

CWM

Common Warehouse Metamodel es un estándar para la representación de metadata.

OIM

Open Information Model es un estándar para la representación de metadata.

OLAP

On-Line Analytic Processing es una tecnología que permite a los usuarios de bases de datos multidimensionales generar resúmenes comparativos o descriptivos en forma “online” de los datos.

OMG

Object Management Group

ORB

Object request Broker es la especificación central de CORBA. Provee los mecanismos con los cuales los objetos invocan y reciben respuesta en forma transparente, proporcionando interoperabilidad entre aplicaciones de diferentes maquinas en ambientes heterogéneos distribuidos.

Sistema

Una colección de unidades conectadas entre sí, que están organizadas para llevar a cabo un propósito específico. Un sistema puede describirse mediante uno o más modelos, posiblemente desde puntos de vista distintos.

SOAP

Simple Object Access Protocol protocolo basado en XML que permite comunicar componentes y aplicaciones mediante HTTP a través de Internet e independientemente de la plataforma.

UML

Unified Modeling Language es un lenguaje estándar definido por la OMG para análisis y diseño orientado a objetos.

XMI

XML Metadata Interchange. Es una especificación de la OMG basada en XML para el intercambio de metadata . Permite el intercambio fácil de metadata entre herramientas de modelado (basadas en UML) y repositorios de metadata (basados en MOF) en ambientes heterogéneos distribuidos. XMI integra tres estándares XML, UML y MOF.

XML

Extensible Markup Language es una especificación que define una forma estándar de agregar marcas a un documento, es en definitiva un lenguaje para documentos que contienen información estructurada.

XSL

Extensible Stylesheet Language es un lenguaje para expresar style sheets. Un XSL es un archivo que describe cómo desplegar un documento XML de un tipo dado.