Apéndice

B

Meta data

UML y MOF

UML es un estándar de notación muy conocido por la mayoría de la gente, utilizado para describir modelos de objetos. En una vista de alto nivel el modelo de UML revela 3 niveles:

Meta modelo. El meta modelo de UML define un número de elementos tales como Clases, Operación, Atributo, Asociación, etc. Estos elementos se llaman meta clases.

Modelo. Instancias de una Class de UML representan entidades, y procesos. Por ejemplo las Clases Cliente, Policía, etc. Se puede definir también instancias de la meta clase Association entre Clases, tal como la asociación entre Cliente y Policia. Estas instancias de meta clases de UML se denominan meta-objetos.

Objetos del usuario. Son instancias de los elementos del modelo. Por ejemplo instancias de la clase Cliente son: Cliente #123, Cliente #456. Estos son los objetos. Los modelos de objetos de UML de dominios particulares, se denominan modelos de objetos de dominio o modelos de dominio.

Existen también otros metamodelos como son los metamodelos para sistemas de bases de datos. Un meta modelo de una base de datos define elementos básicos como base de datos, tabla, columna, clave, etc. Que son usados para definir modelos de datos específicos.

Un modelo de datos basado en un meta modelo de base de datos define bases de datos especificas, tablas, columnas, claves, etc.

MOF y la unión de diferentes meta modelos.

Tantos los modelos de objetos de dominio como los modelos de datos son guardados en bases de datos especializadas llamadas repositorios de meta data, algunas veces llamado simplemente como repositorio. Los administradores de los repositorios normalmente necesitan manejar modelos de dominios y modelos de datos en una forma unificada, pero la naturaleza tan dispar de los meta modelo sobre los cuales están basados les es un gran obstáculo.

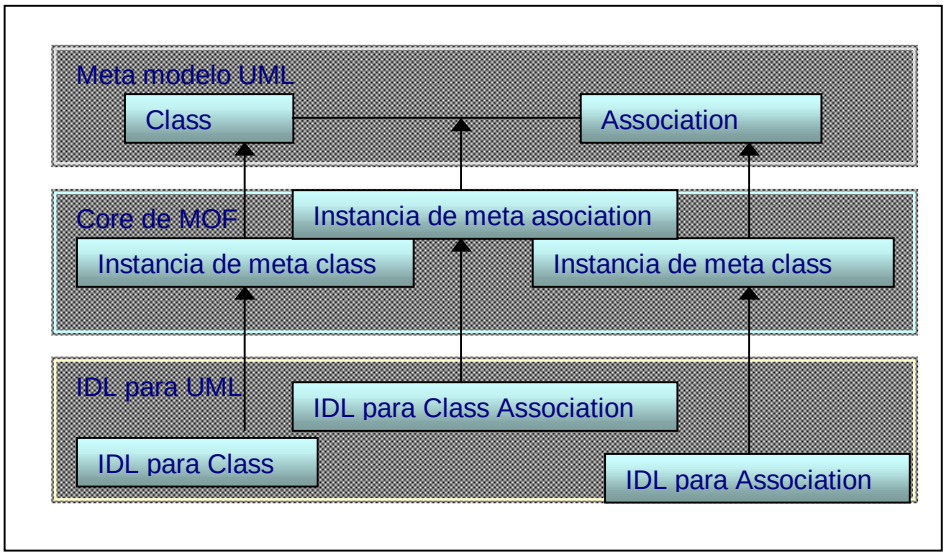
MOF (Meta-Object Facility) provee una base común para esos meta modelos. Si dos meta modelos diferentes están de acuerdo a MOF, los modelos basados en ellos pueden residir en el mismo repositorio.

El meta modelo de UML esta basado en el estándar MOF, (es decir que sus constructores estan definidos en términos de los elementos “core” de MOF),y esto es una pieza clave de la estrategia de la OMG para soportar repositorios integrados.

MOF define un conjunto de constructores que pueden ser usados para describir meta modelos.

Como trabaja MOF

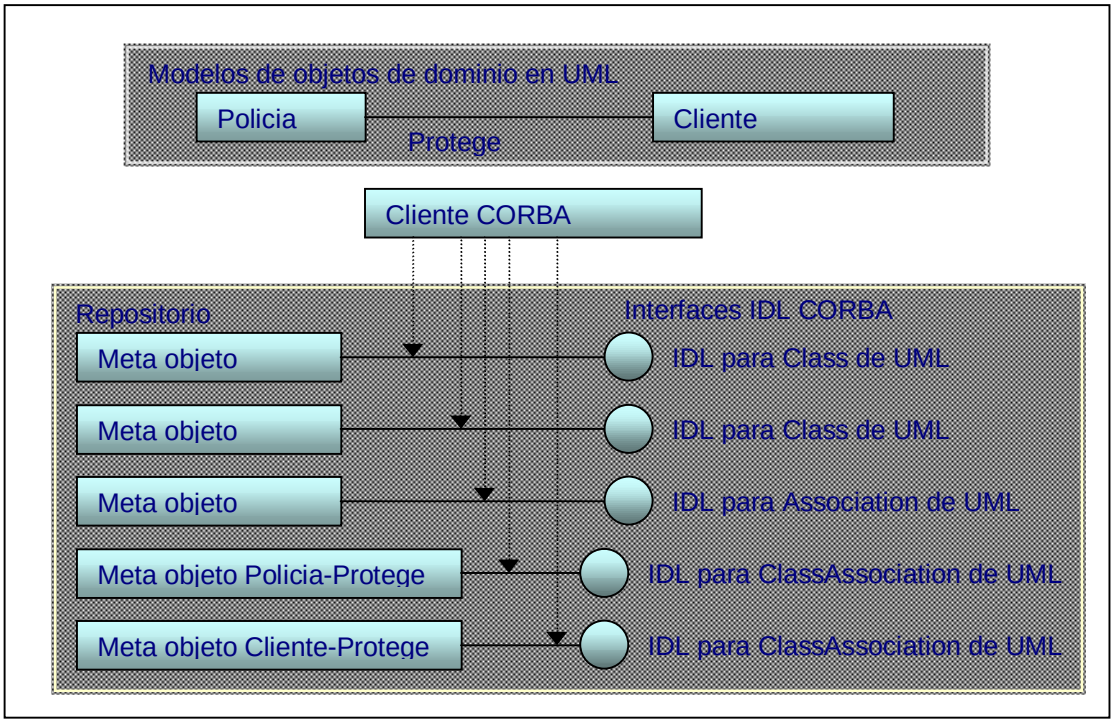
Los elementos “core” de MOF pueden ser mapeados a IDL. Por ejemplo aplicando el mapeo MOF-IDL al meta modelo de UML, se obtiene una representación de dicho meta modelo en IDL. La existencia de esta IDL, representación de UML, significa que cada elemento en el meta modelo de UML tiene una descripción en IDL. Por ejemplo dos elementos del meta modelo de UML son Class y Association. La aplicación del mapeo MOF-IDL produce IDLs para la Class y Association de UML.



Las IDLs para los elementos del meta modelo de UML son determinadas por el mapeo MOF-IDL

Desde el punto de vista del core de MOF, la Class o Association de UML son instancias del constructor MetaClass de MOF. Lo que es una asociación del punto de vista de UML es una MetaClass del punto de vista de MOF. Por otra parte, existe una MetaAssociation entre una Class (UML) y Association (UML), la cual se llama Class-Association.

Por ejemplo, supongamos que un analista usa una herramienta de UML compatible con MOF para definir una Class llamada Policia y otra Cliente. Esta herramienta guarda las clases Policia y Cliente en su repositorio como meta objetos que exponen una IDL CORBA para Class. Supongamos que se define la Association “Proteje” entre Policia y Cliente. Esto se guarda como la IDL CORBA para Association en el repositorio.



Store de un modelo de dominio en UML en un repositorio compatible con MOF

Por lo tanto los mapeos UML-MOF y MOF-IDL permiten guardar modelos de dominios basados en UML como conjuntos de meta objetos que exponen interfaces IDL estandarizadas.

Las IDLs para los elementos de UML pueden ser implementadas en Java. Por lo tanto los meta objetos pueden ser objetos CORBA basados en Java.

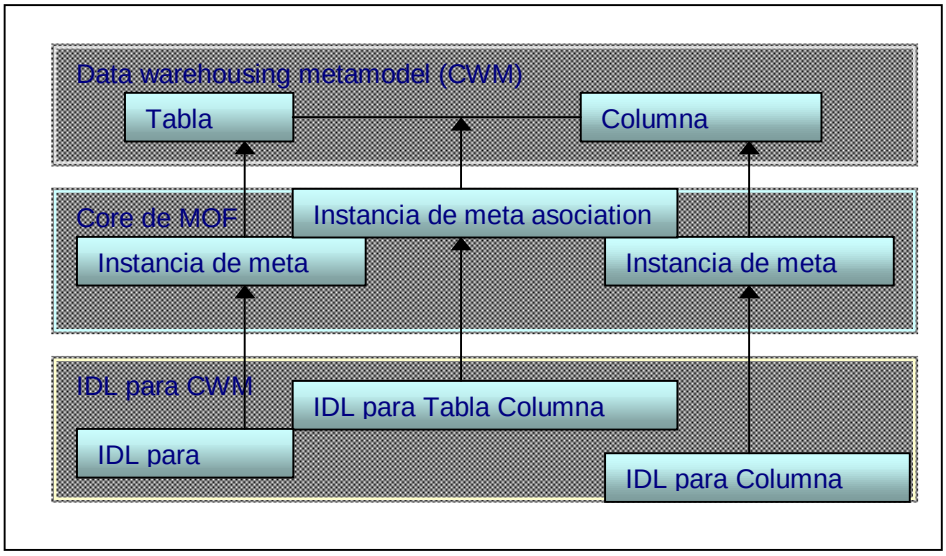
Los niveles de MOF

MOF agrega un cuarto nivel a la clasificación que mostramos antes. MOF llama a estos niveles: M0, M1, M2, y M3.

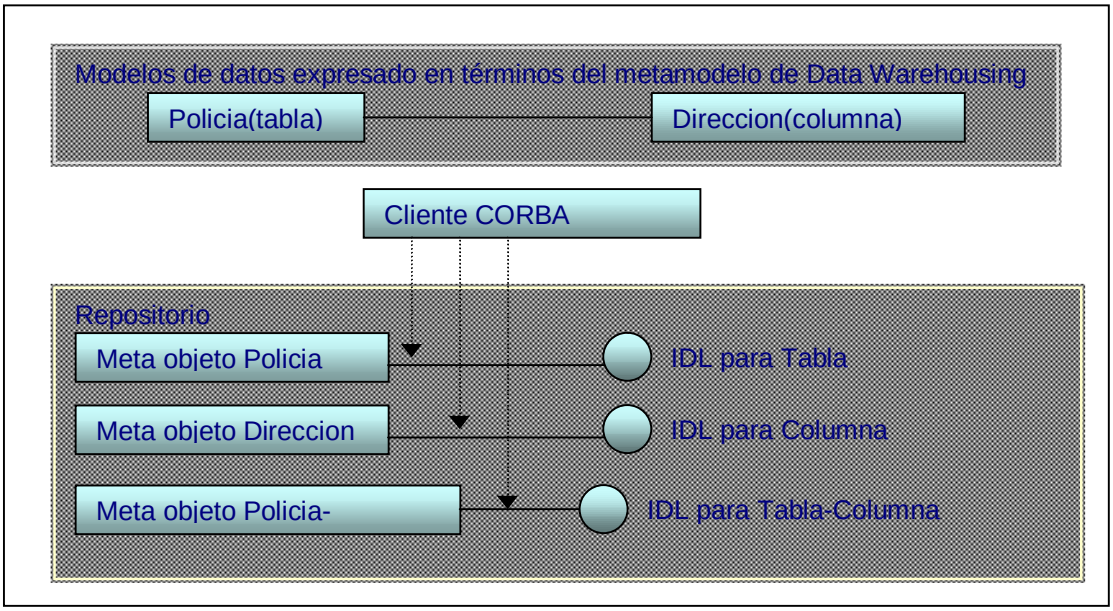
Nivel	Descripción	Elementos
"**M3**" MOF Core	Define los elementos usados para especificar meta modelos	MetaClass, MetaAttribute, MetaAssociation Mof::Class, Mof::Attribute, Mof::Association
"**M2**" Meta-Model	Un meta modelo definido en términos de los elementos core de MOF consiste de MetaClasses, MetaAttributes, etc.	UML: Class, Attribute, Operation Data Warehousing: Base de datos, tabla, fila. UML Classes: Cliente, Policía UML Associations: Protege
"**M1**" Model	Un modelo expresado en términos de un meta modelo. Modela un dominio de información especifico. Consiste de instancias de elementos de un meta modelo (es decir meta objetos).	Data Warehousing Table: Policía Data Warehousing Row: Dirección
"**M0**" User Objects or User Data	Instancias de los elementos de un modelo.	Cliente #123, Cliente #456, Policía #ABC, Policía #DEF, Association entre Cliente #123 y Policía #ABC

La siguiente figura muestra como el meta modelo de un data warehouseing es definido en términos de elementos del core de MOF y como el mapeo MOF-IDL es aplicado para derivar las interfaces IDL para los elementos del meta modelo.

Luego se muestra como un modelo particular expresado en términos de un meta modelo de data warehousing compatible con MOF es guardado en un repositorio compatible con MOF y accedido por los clientes.



IDLs para los elementos del meta modelo de data warehouseing son determinadas por el mapeo MOF-IDL



Store de un modelo de datos en un repositorio compatible con MOF

MOF también especifica interfaces para un repositorio. Esas interfaces son llamadas Facility interfaces. Lo que hacen es permitir a un cliente del repositorio interactuar a nivel de repositorio, esto es, provee una manera estándar para que un cliente obtenga la primera referencia al repositorio al mas alto nivel y obtenga una imagen a alto nivel de su contenido. El cliente a su vez, puede usar interfaces específicas para browsear el repositorio.

Existen repositorios integrados basados en una arquitectura común para los meta modelos, pero compatibles con MOF solo hemos encontrado a dMOF (que justamente implementa la especificación de repositorios de meta modelos en MOF).

Elementos de MOF

Los tres principales constructores para el modelo de metadata provisto por MOF son las Clases, Asociaciones y Paquetes. Estas son similares a las contrapartes en UML con algunas simplificaciones

- **Clases** pueden tener Atributos y Operaciones a nivel de “objeto” y “clase”. Atributos tienen el uso obvio, ej: representación de metadata. Operaciones son provistas para soportar funciones especificas del metamodelo sobre la metadata. Atributos y Operaciones pueden ser definidas como “ordenadas”, o tener restricciones estructurales sobre su cardinalidad y unicidad. Clases pueden tener herencia múltiple de otras Clases.

- **Asociaciones** soportan link dinámicos entre instancias de Clases. Cada Asociación tiene dos Asociaciones Finales que pueden especificar semántica de “orden” o “agregación” y restricciones estructurales sobre cardinalidad y unicidad. Cuando una Clase es de tipo Asociación Final, la clase puede contener Referencias que permiten navegar entre links de Asociaciones de instancias de Clase.

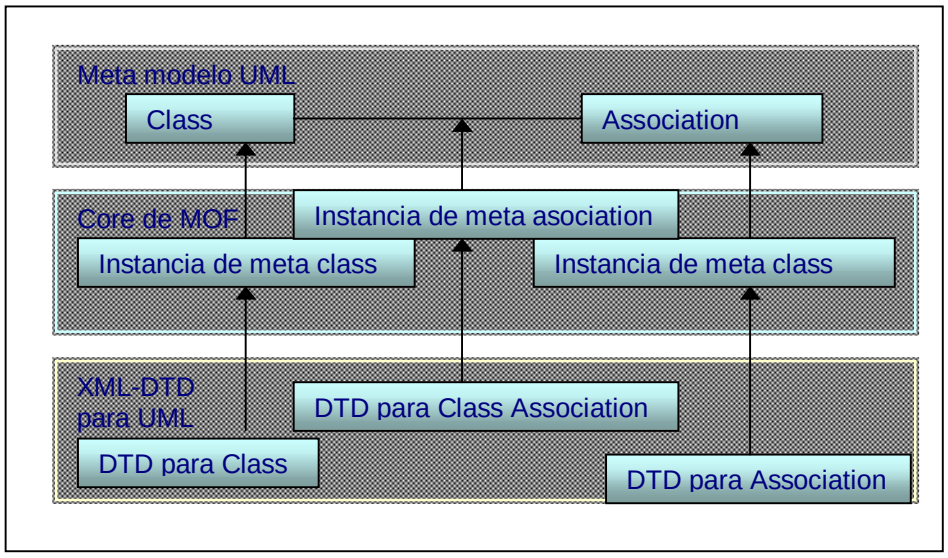
- **Paquetes** son colecciones de Clases y Asociaciones relacionadas. Pueden ser compuestos por otros Paquetes importados o por heredar de ellos. Pueden también ser anidados, aunque esto provee una forma de información oculta mas que de reuso.

- Otro constructor del modelo MOF son **Tipos de Datos y Restricciones**. Tipos de Datos permite el uso de tipos que no sean objetos para Parámetros o Atributos. En la especificación de la OMG de MOF, estos deben ser tipos de datos o tipos de interfaces expresables en CORBA IDL. Las restricciones son usadas para asociar restricciones semánticas con otros elementos en un meta modelo MOF. Esto define las reglas para la metadata descripta por un metamodelo. Cualquier lenguaje puede ser utilizado para expresar Restricciones.

XMI

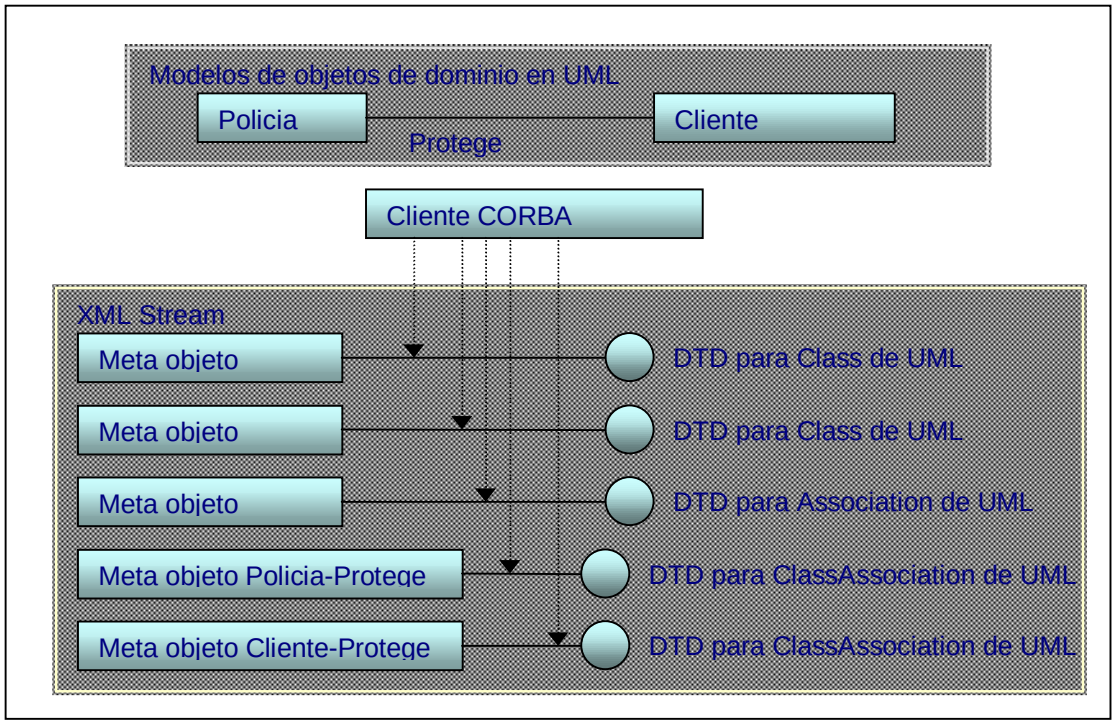
Cuando la OMG publicó un RFP (Request For Proposal) para un Formato de intercambio de modelos basado en Stream (Stream-Based Model Interchange Format -SMIF) se presentaron Unisys, IBM, Oracle, y otras con la propuesta de XMI. XMI (XML Meta data interchange) define un mapeo desde el core de MOF a un formato XML Data Type Dictionary (DTD).

Así como el mapeo MOF-IDL proporciona una formula para obtener IDL para todos los constructores de un meta modelo compatible con MOF, XMI provee una formula para derivar un XML-DTD representando los constructores de un meta modelo compatible con MOF.



El mapeo MOF-XML mapea elementos de un meta modelo en UML a elementos XML-DTD

Una herramienta compatible con MOF puede por lo tanto representar y enviar un modelo de dominio basado en UML en la forma de XML cuya estructura esta de acuerdo a un XMl DTD. El objetivo de quienes envían un XMI es que el XML sea usado para importar y exportar modelos desde y a un repositorio persistente. Algunos desarrolladores están usando archivos de texto XML como repositorios, pero emplear estos archivos XML sin parsear como repositorios online, puede que no escale con el numero y tamaño de modelos en empresas de gran tamaño.



El mapeo MOF-XML especifica como representar modelos de dominio basados en UML, en XML

Obteniendo el XML DTD para los meta objetos vía MOF se incrementa la interoperabilidad entre meta objetos basados en IDL por un lado, y las representaciones en XML de meta objetos por otra.

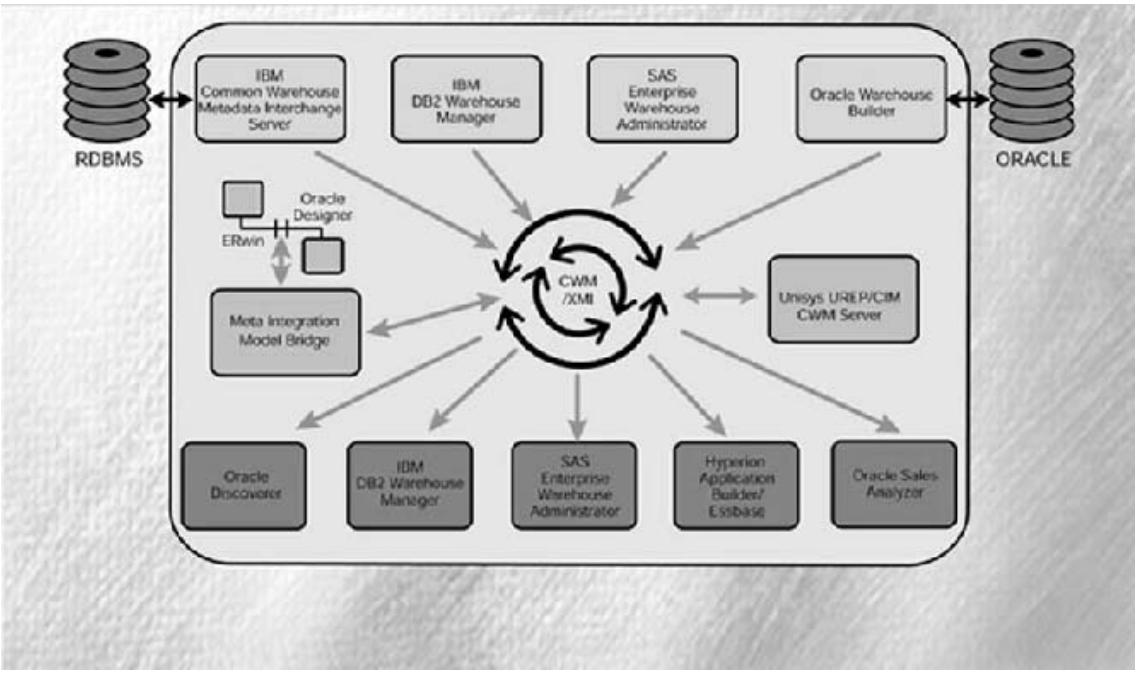
CWM

El intercambio de metadata es uno de los problemas mas críticos que afronta la industria de data warehouse. Hasta 1998 no existe una solución común, existe MDIS (Meta Data Interchange Specification) publicado por MDC (Meta Data Coalition) pero es implementado por pocos vendedores. Fue entonces en Mayo de 1998 que IBM, Oracle y Unisys comienzan los esfuerzos por estandarizar CWM (Common Warehouse Metamodel) dentro de la OMG, el cual finalmente es publicado en Junio de 2000, poco después de publicar los estándares UML , MOF (1997) y XMI (1999).

Las herramientas de análisis y administración de datos requieren diferentes meta-datas y un modelo de meta-data diferente (conocido como meta-modelo) para resolver el problema de la meta-data del data warehouse, es imposible tener un único repositorio de meta-data que implemente un único meta modelo para todas las meta-datas en la organización. Por eso es necesario un estándar para intercambio de meta-data de warehouse.

CWM usa UML, MOF, y XMI para modelar, manipular e intercambiar respectivamente warehouse metadata. CWM es una especificación de sintaxis y semántica en la que herramientas de data warehousing y Business Intelligence pueden apoyarse para intercambiar metadatas. También se dice que es un framework para especificar representación externa de metadata de data warehouse con propósitos de intercambio.

Los diferentes tipos de flujos que se pueden dar cuando se intercambia metadata se muestran en la figura. Horizontalmente las herramientas están colocadas según el rol que cumplen, en la parte superior como metadata producers, en la parte inferior como metadata consumers, y en el medio como ambos. Verticalmente las herramientas están ordenadas según manejan metadata relacional (izquierda) , OLAP (derecha) o warehousing metadata (en el centro).



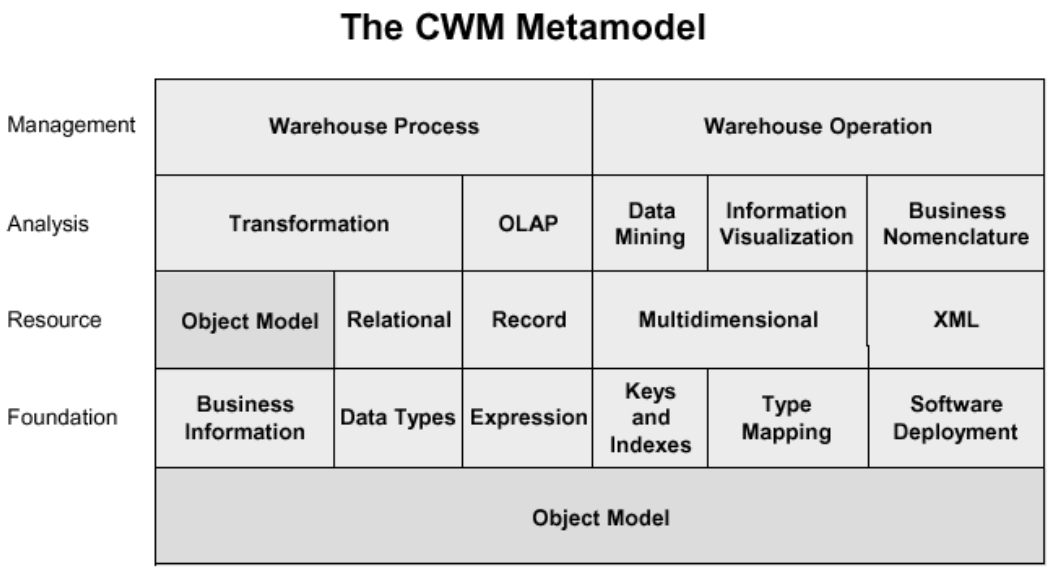
Por ejemplo se puede tener los siguientes intercambios:

- Intercambio de metadata relacional entre DB2 UDB (producer, usando CWM interchange server), Oracle Discoverer (consumer), y Meta Integration Model Bridge (consumer y producer).
- Intercambio de metadata entre Oracle Warehouse Builder (producer) y Hyperion Application Builder de Oracle Sales Analyzer (consumer).

El Meta Modelo de CWM

El Meta-modelo de CWM consiste de un número de sub meta-modelos la cual representan la "common warehouse meta-data" en las siguientes áreas de interés para data warehouse:

- **Data Resources.** Estos incluyen meta-modelos que representan fuentes de datos orientados a objetos, relacionales, de registros, multidimensionales, y XML. En el caso de fuentes de datos orientados a objetos, CWM reusa y depende sobre el UML Foundation.
- **Análisis de datos.** Esto incluye meta-modelos que representan transformación de datos, OLAP (On-line Analytical Processing), data mining, visualización de información, y nomenclatura de negocios.
- **Administración de Warehouse.** Esto incluye meta-modelos que representan procesos de warehouse y resultados de operaciones de warehouse



Organización de CWM

El meta-modelo CWM usa paquetes y una estructura de paquetes jerárquica para controlar complejidad, dar entendimiento, y soportar reuso. Los elementos del modelo están dentro de los siguientes paquetes:

Paquetes bases

- Paquetes de información del negocio. Contiene clases y asociaciones que representan información del negocio acerca de elementos del modelo
- Paquete de tipos de datos. Contiene clases y asociaciones que representan constructores que modeladores pueden usar para crear tipos de datos específicos que ellos necesiten.
- Paquete de expresiones. Contiene clases y asociaciones que representan árboles de expresiones.

- Paquetes de claves e índices. Contiene clases y asociaciones que representan claves e índices.
- Paquetes de desarrollo de software. Contiene clases y asociaciones que representan como el software es desarrollado en un data warehouse.
- Paquetes de mapeos de tipos. Contiene clases y asociaciones que representan mapping de tipos de datos entre diferentes sistemas.

Paquete de fuente de datos

- Paquete Relacional. Contiene clases y asociaciones que representan meta-datos de fuente de datos relacionales.
- Paquetes de Registros. Contiene clases y asociaciones que representan meta-datos fuente de datos de registros.
- Paquete Multidimensional. Contiene clases y asociaciones que representan meta-datos de fuente de datos multidimensional.
- Paquete XML. Contiene clases y asociaciones que representan meta-datos de fuentes XML.

Paquete de Análisis

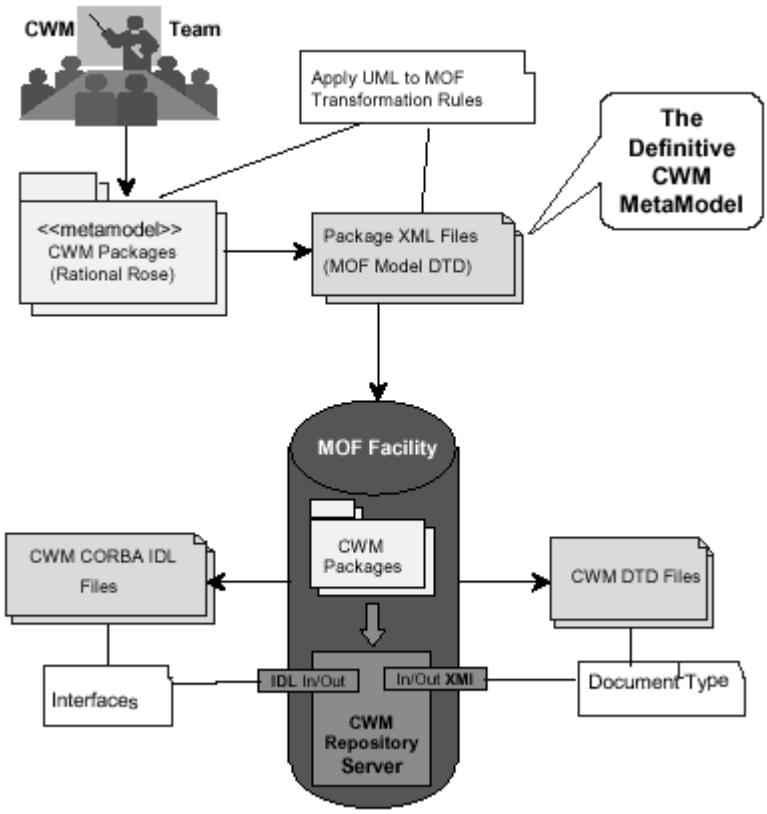
- Paquete Transformacional. Contiene clases y asociaciones que representan metadatos de herramientas de datos transformacionales.
- Paquetes OLAP. Contiene clases y asociaciones que representan metadatos de herramientas OLAP
- Paquete de Data Mining. Contiene clases y asociaciones que representan meta-datos de herramientas de data mining.
- Paquete de Visualización de Información. Contiene clases y asociaciones que representan meta-datos de herramientas de visualización de información
- Paquete Business Nomenclature. Contiene clases y asociaciones que representan meta-datos sobre el negocio.

Paquete de Administración

- Paquete del proceso de Warehouse. Contiene clases y asociaciones que representan meta-datos del proceso de warehouse.
- Paquete de operaciones de Warehouse. Contiene clases y asociaciones que representan meta-datos de operaciones de warehouse

El meta modelo de CWM esta diseñado de acuerdo al estándar MOF. Esto permite a CWM usar otras especificaciones que son dependientes de MOF. En particular permite usar XMI para intercambiar meta data warehouse que esta representada usando el meta modelo de CWM, y permite usar IDL para programar acceso a meta data de warehouse basada en el meta modelo de CWM.

La siguiente figura esquematiza la relación que existe entre MOF, XMI y CWM.



Un DTD estándar para el metamodelo CWM es generado usando reglas de producción DTD de XMI's. Metadata de warehouse pueden ser codificadas con un documento XML usando reglas de producción de documentos XMI.

Un documento estándar XML para el metamodelo CWM es también generado usando reglas de producción de documentos XMI, basado sobre el MOF DTD.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- <!DOCTYPE XMI SYSTEM 'Model.dtd' > -->
<XMI xmi.version="1.1" xmlns:Model="org.omg.mof/Model/1.3" timestamp="Sat Jul 07 19:58:02 2001">
   <XMI.header>
      <XMI.documentation>
         <XMI.exporter>Unisys.JCR.1</XMI.exporter>
         <XMI.exporterVersion>1.3.2</XMI.exporterVersion>
      </XMI.documentation>
      <XMI.metamodel xmi.name="org.omg.mof.Model" xmi.version="1.3"/>
   </XMI.header>
   <XMI.content>
      <!-- ==================== p1    [Package] ==================== -->
      <Model:Package xmi.id="a3B47904400DD" name="p1" annotation="" isRoot="false"
isLeaf="false" isAbstract="false" visibility="public_vis">
         <Model:Namespace.contents>
            <!-- ==================== p1.c1    [Class] ==================== -->
            <Model:Class xmi.id="a3B47930B028E" name="c1" annotation="" isRoot="false"
isLeaf="false" isAbstract="false" visibility="public_vis" isSingleton="false">
               <Model:Namespace.contents>
                  <!-- ==================== p1.c1.att1    [Attribute] ====================
-->
                  <Model:Attribute xmi.id="a3B479319037E" name="att1" annotation=""
scope="instance_level" visibility="private_vis" isChangeable="true" isDerived="false" type="G.4">
                     <Model:StructuralFeature.multiplicity>
                        <Model:MultiplicityType lower="1" upper="1" is_ordered="false"
is_unique="false"/>
```

```
                                    </Model:StructuralFeature.multiplicity>
                                </Model:Attribute>
                            </Model:Namespace.contents>
                        </Model:Class>
                        <!-- ==================== p1.Byte    [DataType] ==================== -->
                        <Model:DataType xmi.id="G.4" name="Byte" annotation="" isRoot="false" isLeaf="false"
isAbstract="false" visibility="public_vis">
                            <Model:DataType.typeCode>
                                <XMI.CorbaTypeCode>
                                    <XMI.CorbaTcString xmi.tcLength="0"/>
                                </XMI.CorbaTypeCode>
                            </Model:DataType.typeCode>
                        </Model:DataType>
                    </Model:Namespace.contents>
                </Model:Package>
            </XMI.content>
        </XMI>
```

OIM

MDC (Meta Data Coalition) se fundó como un consorcio de 50 vendedores (entre ellos BMC, Informatica, Microsoft, y SAS) y usuarios finales, con el objetivo de proporcionar una solución para el intercambio de metadata. Esta alianza es quien desarrolló el MDC OIM (Open Information Model), un estándar de metadata que es una especie de merge entre el MDIS (Meta Data Interchange Specification) y el OIM propuesto originalmente por Microsoft. En septiembre de 2000, MDC y la OMG (Object Management Group), anunciaron la unión de MDC con OMG. Como resultado MDC no continuó con sus operaciones en forma independiente sino que su trabajo continuo en la OMG para integrar los estándares desarrollados por cada una. Hasta ese momento existían 2 estándares principales para metadata y modelado en las áreas de data warehousing y desarrollo basado en componentes: OIM de MDC y CWM de la OMG.

El consorcio de MDC con OMG llegó a un acuerdo entre los mas importantes vendedores de metadata y data warehousing para converger en un único estándar, incorporando lo mejor del OIM de MDC con lo mejor del CWM de la OMG.

La especificación resultante es entonces liberada como una versión siguiente de CWM. Como un único estándar que permite a los usuarios intercambiar metadata entre diferentes productos de diferentes vendedores. OIM es entonces uno de los estándares en que esta basado CWM.

OIM está especificado en UML y está organizado en áreas fáciles de usar y extender. La especificación de OIM se en enfoca en cumplir los siguientes **objetivos**:

- Soportar interoperabilidad de herramientas entre tecnologías y compañías mediante un modelo de información común
- Abarcar todas las fases del desarrollo de sistemas de información.
- Ser un modelo extensible, expresivo y fácil de usar de tipos de meta-datos
- Proveer mecanismos para especializar y extender los tipos de meta-datos en vez de modificar o reemplazar conceptos
- Permitir agregar nuevos conceptos al core de manera consistente
- Proveer una especificación independiente de la tecnología y de vendedores
- Soportar implementaciones heterogéneas usando diferentes tecnologías lenguajes de programación.
- Ser escalable desde herramientas individuales a repositorios de meta-datos de la empresa

Las **áreas** que cubre OIM son las siguientes:

- Modelos de Análisis y Diseño
- Modelos de Objetos y Componentes
- Modelos de Bases de datos y Data warehouses
- Modelos de Administración del conocimiento
- Modelos de Business Engineering

Durante cada paso del ciclo de vida de diseño, desarrollo, e instalación de software, los profesionales usan herramientas de análisis y diseño por razones diferentes- como herramientas de entrada, para documentación, o para análisis y validación de resultados.
Esto requiere que las Herramientas de análisis y diseño estén fuertemente integradas con el resto de las aplicaciones a través del intercambio de meta-datos o compartiendo un repositorio común.
Modelos de datos orientados a objetos, modelos de datos de las organizaciones, y cualquier otro tipo de meta-data evoluciona individualmente. Es necesario entonces que las relaciones entre un modelo y los elementos del sistema puedan ser expresados y mantenidos.
El modelo de Análisis y Diseño debe entonces proveer no solo elementos de modelado sino también mecanismos para referenciar elementos fuera del alcance del modelo. Esta capacidad y los conceptos genéricos en el modelo lo hacen un adaptador natural para servir como un "core model" desde el cual otros modelos más especializados heredarán conceptos más generales como los de package, containment o dependency.
El modelo de Análisis y Diseño cubre los dominios de modelado orientado a objetos y diseño de sistemas de software. El modelo provee conceptos para describir problemas y soluciones durante todo el ciclo de vida.
El modelo de Objetos y Componentes define "componente" como "un paquete de software que ofrece servicios a través de interfases." Esto está planificado para capturar las perspectivas de un componente como la unidad de empaquetado y reparto, proveedor de servicios.