MODELO ARQUITECTÓNICO PARA DESARROLLO E INTEGRACIÓN DE SISTEMAS DE INFORMACIÓN EN UN CONTEXTO ECONÓMICO

LILIANA C. BONILLA BENDECK
CECILIA E. VELASCO VIVAS

Monografía presentada como requisito para optar al título de Ingeniero en Electrónica y Telecomunicaciones

DIRECTOR: Ing. Gustavo Adolfo Ramírez

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES
DEPARTAMENTO DE TELEMÁTICA
LÍNEA DE INVESTIGACIÓN EN INGENIERÍA TELEMÁTICA
POPAYÁN
2004

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

CAPITULO 1. INTRODUCCIÓN

La información y más aún, el conocimiento que pueda tener una región sobre el estado de sus actividades económicas y productivas es un factor primordial en el desarrollo y la competitividad de dicha región. Esto se fundamenta en el soporte que brinda el conocimiento en el momento de tomar decisiones coyunturales.

Sin embargo, a pesar de la aceptación que tiene este principio en la cultura actual, en países en vía de desarrollo como Colombia y en departamentos tan atrasados como el Cauca uno de los mayores inconvenientes es la falta de información. Los mecanismos de recolección y mantenimiento de la información son aún muy primitivos y esto genera desgaste y desinterés en las instituciones y obstaculiza la realización de análisis económicos críticos que den soporte a la toma adecuada de decisiones.

Una región que actúa sin fundamentarse en su situación actual real, en sus precedentes y proyecciones es una región ciega, que da pasos sin ninguna dirección ni meta definida.

Aunque en el departamento han surgido iniciativas de recolección y distribución de la información económica como los informes del Banco de la República, las cuentas regionales, el anuario estadístico del Cauca, entre otros, no han sido fáciles de mantener y no han causado un gran impacto debido a la dificultad de ofrecer información efectivamente actualizada.

Es así como surge la iniciativa de crear un sistema de información tecno-económico para el departamento del Cauca (SITEC) con un alto valor estratégico, que permita hacer seguimiento a la evolución económica de la región y que contribuya a sustentar las decisiones por parte de las instituciones gubernamentales, empresariales y la comunidad caucana en general.[1]

SITEC es un sistema que recoge la información de distintas instituciones que por su dinámica generan y/o almacenan información relativa a su que hacer, y que es de relevancia para los análisis de la situación económica del departamento.

[1] Objetivo del proyecto SITEC. Sistema de información tecno-económico para el departamento del Cauca

Este sistema debe contener información que permita la elaboración y análisis de los primeros indicadores líderes de la actividad económica caucana y de esta forma crear un observatorio económico regional del departamento. Esto complementaría todo un esfuerzo enfocado a buscar un soporte administrativo sólido.

En este sentido, para lograr una integración de la información del departamento, se requiere conformar una estructura de coordinación con todas estas instituciones, que facilite los acuerdos administrativos para contar permanentemente con la disponibilidad de tal información.

Del mismo modo, este trabajo de desarrollo se enmarca dentro de la necesidad planteada pero con un objetivo más ambicioso que el desarrollo convencional del sistema. Este es: La realización de un modelo arquitectónico que permita lograr un acuerdo en la forma de describir los conceptos y relaciones involucrados en la realización de cualquier sistema que maneje información económica.

Es por esto que se decidió explorar MDA (Model Driven Architecture), una iniciativa de la OMG[2], como una alternativa de modelado con la cual se pueden distinguir dos fases fundamentales en el desarrollo, la primera es la concepción lógica del sistema, enfocada a los aspectos funcionales y conceptuales del dominio del sistema y la segunda, la cual permite aterrizar la concepción lógica a una plataforma real de implementación.

Para la implementación del sistema distribuido, fue necesario considerar alternativas de soluciones de integración de aplicaciones, que permitieran superar los problemas de heterogeneidad de las distintas fuentes de información.

El contenido del documento se organiza de la siguiente manera:

[2] OMG: Object Management Group. Es un consorcio sin ánimo de lucro que produce y mantiene las especificaciones de la industria del software para permitir la interoperabilidad de las aplicaciones empresariales.

Capítulo 2: MDA. Model Driven Architecture. En este capitulo se desarrolla la temática de la iniciativa MDA de la OMG relacionando los conceptos y estándares que involucra, su génesis y estado del arte. El capitulo presenta unas conclusiones que son la base para las decisiones tomadas a lo largo del desarrollo.

Capítulo 3: Integración de Aplicaciones. En este capitulo se describen las posibles alternativas de integración de aplicaciones con una perspectiva conceptual sin involucrar plataformas específicas de desarrollo, ganando la independencia necesaria para tomar una decisión de ingeniería de alto nivel. Esta concepción es fundamental para la escogencia de los estilos arquitectónicos empleados en el desarrollo del sistema

Capítulo 4: SITEC. Sistema de Información tecno - económico del Cauca. En este capitulo se describe el desarrollo del sistema involucrando los conceptos y conclusiones presentadas en los capítulos anteriores. Se describen los estilos arquitectónicos utilizados, el modelo de referencia y la arquitectura de referencia de la solución haciendo una relación con los modelos propuestos por MDA.

Capítulo 5: Conclusiones. Se presentan los principales aportes de la investigación relacionada con MDA, las opciones de integración de aplicaciones y se ponen en evidencia las gestiones más allá de la tecnología, que un desarrollo de este tipo requiere para cumplir sus objetivos.

CAPITULO 2. MDA: ARQUITECTURA DIRIGIDA POR MODELOS

Model Driven Architecture se origina en la conocida idea de separar las características funcionales de un sistema, de los detalles de su implementación en una plataforma específica.
MDA es una nueva forma de escribir especificaciones y desarrollar aplicaciones que se postula como la solución a los problemas de integración. Cuando la OMG creó los estándares de interoperabilidad de CORBA, la forma en que dichos estándares se relacionaban e interactuaban entre sí, parecía ser adecuada. Sin embargo, la necesidad de nuevos estándares y de la interoperabilidad entre ellos hizo que se ampliara la visión.
Una especificación completa en MDA consiste en un modelo independiente de la plataforma (PIM de sus siglas en ingles) más uno o más modelos específicos de la plataforma (PSM de sus siglas en ingles) y un conjunto de definiciones de interfaces que describen la manera en que el modelo base se implementa en diferentes plataformas.

2.1 CONCEPTOS BÁSICOS

2.1.1 Modelos

En MDA se define un modelo como una representación de una parte de la funcionalidad, la estructura y/o comportamiento del sistema. Se dice que una especificación es formal, cuando se basa en un lenguaje que tiene bien definida la forma (sintaxis) y el significado (semántica).
UML es un lenguaje formal. Su sintaxis está adecuada y formalmente definida, es decir, que define las reglas que se tienen que seguir para representar los conceptos y definir las relaciones entre los componentes del lenguaje. Un ejemplo concreto de un atributo es:
+integer num=0000{4 dígitos}
Como se ve, la sintaxis de un atributo es:

visibility name : type-expression = initial-value { property-string }

Donde visibility es uno de los siguientes:

\+ *public visibility*
\# *protected visibility*
\- *private visibility*

type-expression es el tipo del atributo con nombre *name*. Puede especificarse como se ve un valor inicial y un conjunto de propiedades del atributo.

Por otra parte, la semántica hace referencia al significado de dicha notación por ejemplo:Un caso de uso es un tipo de clasificador que representa una unidad coherente de funcionalidad dentro de un sistema, un subsistema, o una clase manifestada por una secuencia de mensajes intercambiados entre el sistema y uno o mas objetos externos (llamados actores) y por acciones que el sistema lleva a cabo.

2.1.2 Abstracción, refinamiento y vista

El término abstracción en MDA es usado como la supresión de detalles irrelevantes. Es muy útil caracterizar modelos según los criterios de abstracción que se hayan adoptado. A estos modelos, basados en criterios de abstracción se les conoce como una vista del sistema. Específicamente, dos modelos están en una relación de refinamiento, cuando uno de ellos -la abstracción- tiene un nivel más alto de abstracción que la realización.

Un ejemplo de abstracción- refinamiento, es un modelo de clases de análisis y un modelo de clases de diseño. En este caso, el segundo es un refinamiento del primero, o lo que es lo mismo, el primero es una abstracción del segundo.

Debe quedar claro sin embargo, que dos modelos que describan el sistema desde diferentes puntos de vista, no deben tener necesariamente una relación de refinamiento.

2.1.3 Plataforma

En MDA, el término plataforma es usado para referirse a detalles tecnológicos que son relevantes para la funcionalidad fundamental de un componente software. Entonces, un modelo independiente de la plataforma (PIM), es una especificación formal de la estructura y la funcionalidad de un sistema, que abstrae gran cantidad de detalles técnicos. El PIM es inicialmente expresado a través de un lenguaje independiente de la plataforma como UML, y luego es traducido a un modelo específico de la plataforma, mapeando el PIM a algún lenguaje de implementación o plataforma. Como plataformas específicas se pueden mencionar: Java (J2ME,J2SE,J2EE), CORBA, Servicios Web, entre otros.

2.1.4 Perfil UML

El perfil especifica conceptos de un dominio particular y permite estandarizar los términos a él asociados. El término Perfil fue adoptado por la OMG, buscando expresar la semántica del lenguaje de descripción de interfaces (IDL) de CORBA. Dicho perfil es actualmente una referencia para la definición de nuevos perfiles de diversos dominios que llegarán a estandarizarse.

Actualmente no existe una definición normativa para el concepto de perfil, sin embargo, se ha adoptado la siguiente definición:

Un perfil UML es una especificación que cumple con uno o más de los siguientes puntos:

- Se identifica como subconjunto del meta-modelo[3] UML
- Especifica "reglas bien definidas" además de las que están definidas por el subconjunto del meta-modelo, las cuales expresan restricciones escritas en lenguaje natural u OCL (UML's Object Constraint Language) que contribuyen a la definición del elemento del meta-modelo.
- Especifica "elementos estándares" además de los definidos por el subconjunto del meta-modelo. "Elementos estándares" es un término utilizado en la especificación del meta-modelo UML para describir una instancia estándar de un estereotipo, valor etiquetado y restricciones.
- Especifica la semántica expresada en lenguaje natural, además de la definida por el subconjunto del meta-modelo.
- Especifica elementos comunes del modelo, es decir, instancias de UML expresados en términos del perfil.

UML permite construir nuevos elementos o modificar los existentes por medio de 3 mecanismos de extensión, que permiten una mayor precisión en la definición de un dominio. Estos mecanismos son:

- Valores adicionados (tagged values): con los cuales se añaden nuevas características o atributos a los elementos definidos por el meta-modelo. Ejemplos de valores

[3] Un metamodelo define los conceptos y relaciones necesarias para la creación de un modelo.

adicionados pueden ser el analista de negocio que creó un caso de uso, el autor de una clase, o la fecha en la que una clase fue creada o modificada.

- Restricciones (Constraints): Permiten especificar reglas y restricciones a los elementos del modelo en UML. Por ejemplo, un perfil puede especificar una restricción de no permitir establecer una asociación entre dos clases que tengan el estereotipo <<entity>>.

Estereotipos (Stereotypes): Son utilizados para clasificar, marcar elementos del modelo o introducir nuevos tipos de elementos al modelo. Cada nuevo estereotipo puede incluir valores etiquetados y restricciones.

Perfiles de Ejemplo de la especificación UML 1.4 :

La especificación de UML contiene los siguientes ejemplos de perfiles:

- El perfil UML par EJBs
- El perfil UML para el modelado del negocio
- El perfil UML para el Proceso de desarrollo de software.

Con la intención de aclarar conceptos se describirá el perfil para EJBs. En la figura 2.1 se muestra el metamodelo presentado para EJBs:

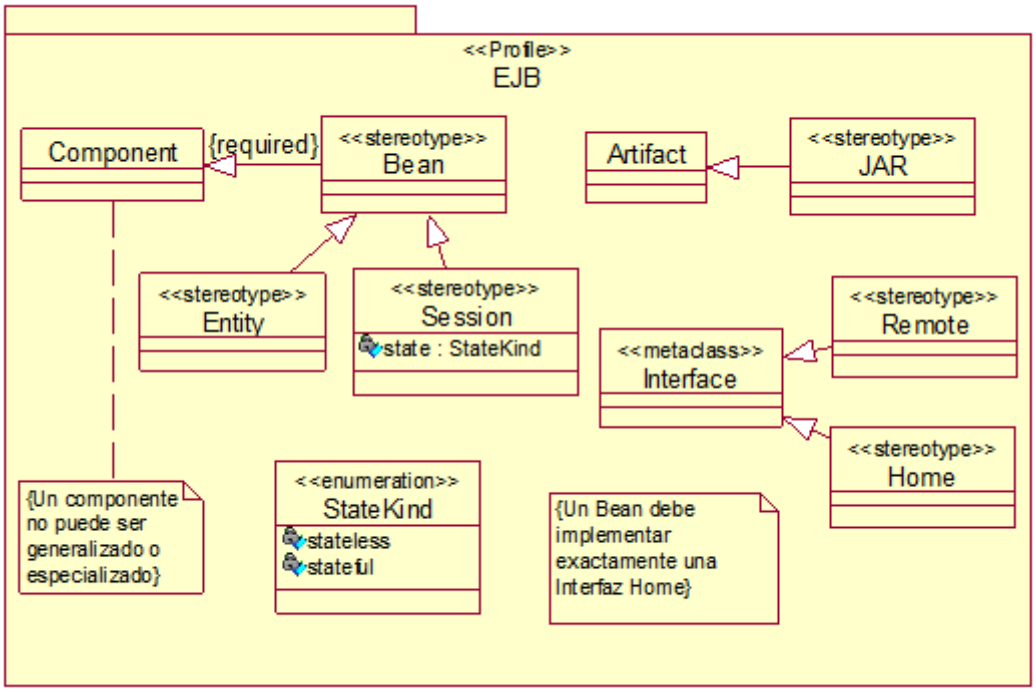

Figura 2.1. Perfil UML para EJBs

Un estereotipo Bean extiende de un Componente, lo cual es una condición necesaria. El estereotipo Bean es de tipo abstracto, con dos subtipos, Entidad y Sesión. Cada instancia de Componente debe ser extendida por una instancia ya sea del estereotipo Entity o Session. Como un estereotipo extiende de Class (dentro de UML), quiere decir que también éste puede tener propiedades. En este caso, un estereotipo de Session tiene un atributo state que corresponde a una definición adicionada cuyo valor especifica el estado de la sesión. El valor adicionado es una enumeración, StateKind, la cual tiene un valor Stateless o Stateful.

La clase Componente tiene una restricción, descrita en la nota adicionada a la clase la cual especifica que un componente no puede ser generalizado ni especializado.

El diagrama también muestra que una metaclase Interfaz es extendida por los estereotipos Remote y Home. El paquete tiene una restricción que afirma que un Bean debe implementar exactamente una interfaz Home. Cabe mencionar que la relación entre las metaclases y los estereotipos se denomina extensión, la cual muestra que las propiedades de la metaclase son extendidas a través del estereotipo. Como en el caso de Component y Bean, se puede especificar que una extensión es requerida. Esto es aplicado como una restricción, que significa que una instancia del estereotipo debe estar siempre enlazada a una instancia de la metaclase.

Los perfiles son intercambiables utilizando mecanismos existentes de XMI, ya que el principal objetivo de XMI es el intercambio de modelos UML entre diferentes herramientas de modelado. XMI puede reflejar las extensiones de UML. Esto significa que definiciones adicionales y propiedades que hayan sido añadidas a una herramienta particular pueden ser expresadas en XMI e intercambiada entre herramientas, siempre y cuando las dos herramientas soporten la habilidad de añadir extensiones UML.

Los perfiles UML no permiten los mecanismos de extensión fuertes, es decir, la modificación directa del meta-modelo. Esto implica que todos lo nuevos elementos deben extender de elementos comprendidos por el metamodelo. Por ejemplo, como se ve en la figura 2.2 un estereotipo extiende del bloque de construcción Clase del metamodelo UML.

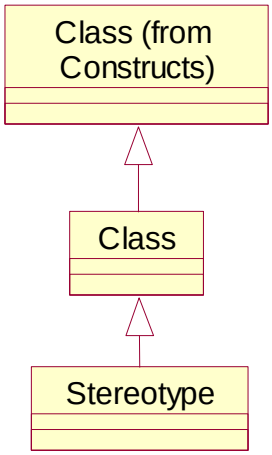

Figura 2.2. Mecanismos de extensión de UML

2.1.5 Arquitectura de software

Cuando se habla de arquitectura de software se nombran distintas ideas acerca de su definición tales como diseño, estructura o infraestructura. Pero en realidad no existe una definición precisa de lo que realmente significa. Sin embargo, debido a la relevancia que para la temática implica, se optó por apropiar la definición propuesta en [LPR 98]:

"La arquitectura de software de sistema computacional es la estructura o estructuras del sistema, la cual a su vez comprenden componentes de software, las propiedades visibles externamente de tales componentes y las relaciones entre ellos."

Profundizando en la definición previa, se pueden identificar varios aspectos importantes:

1. La arquitectura define componentes y contiene información de la interacción entre ellos. Esto implica que la información que no es relevante a la forma en que los componentes se relacionan entre sí debe ser omitida. En otras palabras, la información relevante de los componentes son sus propiedades externamente visibles.
2. La definición deja claro que un sistema puede contener más de una estructura, las cuales contienen información arquitectural relevante. Pero no son la arquitectura por sí mismas. Entonces, una arquitectura puede contener más de una clase de estructuras, más de una clase de componentes (como módulos y/o procesos), más de una clase de interacciones entre los componentes, entre otros. Intencionalmente, la definición no especifica que son, ni cuales son los componentes y relaciones arquitecturales. De tal

forma que un componente puede ser un objeto, un proceso, una librería, una base de datos, etc.

3. La definición implica que cada sistema software tiene una arquitectura ya que cada uno puede ser visto como una composición de componentes y relaciones entre ellas. En el caso más trivial, un sistema es en sí un componente.
4. El comportamiento de cada componente es parte de la arquitectura a tal punto que el comportamiento puede ser observado desde el punto de vista de otro componente.

2.1.6 Estilos arquitectónicos, modelos de referencia y arquitecturas de referencia

2.1.6.1 Estilos arquitectónicos: Un estilo arquitectónico es la descripción del tipo de componentes y patrones del control de su ejecución y/o transferencia de datos. Un estilo puede ser pensado como un conjunto de restricciones (restricciones sobre los tipos de componentes y sus patrones de interacción)- y esas restricciones definen un conjunto de familias de arquitecturas que los satisfacen. Como ejemplos se puede mencionar la arquitectura cliente-servidor, cliente servidor multinivel, arquitectura de objetos distribuidos, arquitectura de máquina abstracta, arquitectura de control centralizado, entre otras.

2.1.6.2 Modelos de referencia: Es la división de la funcionalidad junto con el flujo de datos entre las partes. Un modelo de referencia es la descomposición estándar de un problema conocido en partes que cooperativamente dan la solución. Los modelos de referencia son característicos de dominios maduros y generalmente son obtenidos a partir de un análisis de dominio.

2.1.6.3 Arquitectura de referencia: Es el modelo de referencia mapeado a componentes software (que cooperativamente implementarán la funcionalidad definida en el modelo de referencia) y el flujo de datos entre dichos componentes. Si un modelo de referencia divide la funcionalidad, una arquitectura de referencia es el mapeo de dicha funcionalidad a la descomposición de un sistema.

Los modelos de referencia, estilos arquitectónicos y arquitecturas de referencia no son arquitecturas. Son pasos previos, útiles, hacia una arquitectura.

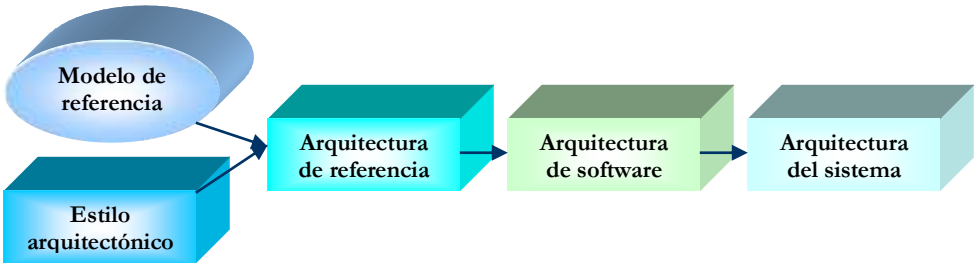

Figura 2.3. Relación entre Modelos de referencia, estilos arquitectónicos, arquitectura de referencia y arquitecturas de software implementadas

2.1.7 Patrones

Los patrones permiten que los desarrolladores definan aproximaciones estándares para solucionar problemas comunes. Un patrón es una solución de diseño de software a un problema, aceptada como correcta, a la que se ha dado un nombre y que puede ser aplicada en otros contextos. [TP 03]

2.2 LAS CUATRO CAPAS DE MODELOS DE LA OMG [AJW 03]

Como se pudo notar, el concepto de metamodelo juega un papel fundamental en la integración e interoperabilidad. Enfrentando el peligro de que gran variedad de metamodelos no compatibles se fueran definiendo y evolucionando de manera independiente, se generó la necesidad de una estructura de integración global para todos los metamodelos en el escenario del desarrollo de software. La respuesta fue proporcionar un lenguaje para la definición de metamodelos, un meta-metamodelo. Cada metamodelo define su propio lenguaje para describir el dominio específico de su interés. Por ejemplo, UML describe todos los aspectos relacionados con un sistema de software orientado a objetos. Otros metamodelos pueden orientar dominios como el almacenamiento de datos (CWM), procesos de software, organización, evaluación, calidad del servicio, etc.

Para entender la relación entre los estándares de la OMG que juegan un rol dentro de MDA, se debe tener una concepción de cómo están definidas las capas de modelos:

2.2.1 Capa M0: Las instancias

En esta capa se encuentra el sistema en ejecución. Las instancias son representaciones software de entidades del mundo real. Tomando como ejemplo el contexto de un restaurante, se encuentran dos clientes, Pedro y María que piden la orden número 200, los cuales son únicos e irrepetibles.

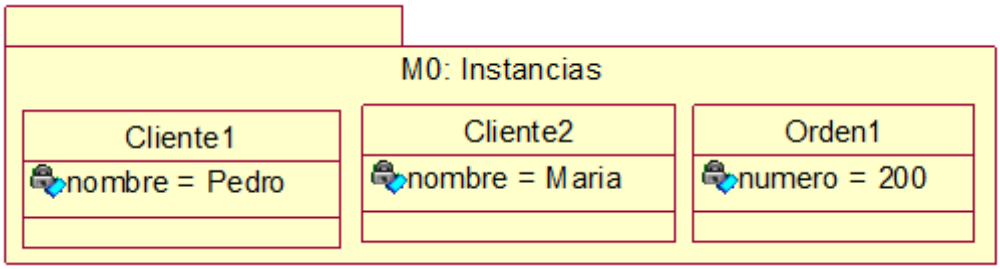

Figura 2.4 Capa M0. Instancias

2.2.2 Capa M1: El modelo del sistema

Esta capa contiene los modelos en los cuales se definen conceptos y propiedades que clasifican las instancias de la capa M0, es decir, los elementos de la capa M1 especifican como deben ser las instancias en la capa M0. Continuando con el ejemplo anterior, este modelo define los conceptos Cliente y Orden con sus respectivos atributos.

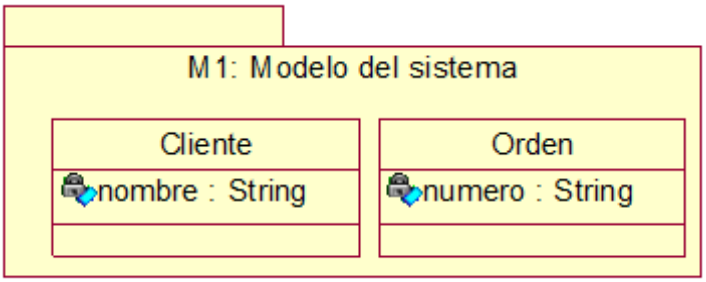

Figura 2.5 Capa M1: Modelo del Sistema

2.2.3 Capa M2: El modelo del modelo

Los elementos que existen en la capa M1 (clases, atributos y otros elementos del modelo) son a su vez, instancias de clases de M2, así también, todo elemento de M2 clasifica los elementos de M1. M2 contiene los conceptos necesarios para razonar sobre los conceptos de la capa M1, tales como una Clase, una Asociación. Este modelo se utiliza para definir modelos por lo que

el modelo que reside en M2 es denominado un metamodelo. UML y CWM son ejemplos de metamodelos. Retomando el ejemplo, los conceptos Cliente y Orden heredan del concepto *UML Clase* que define este metamodelo y sus atributos nombre y numero, heredan de la clase *UML Atributo*.

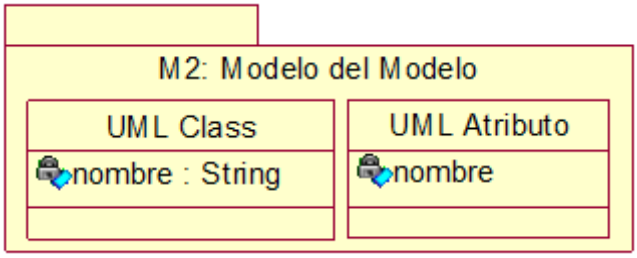

Figura 2.6 Capa M2: El modelo del modelo

2.2.4 Capa M3: El modelo de M2

Siguiendo la misma línea, se puede deducir que esta capa contiene los conceptos necesarios para definir los elementos de M2 y por lo tanto, cualquier elemento de M2 es una instancia de M3. Como este es un modelo que define otro metamodelo, se denomina meta-metamodelo. Dentro de la OMG, el MOF[4] es el lenguaje estándar en M3, y todos los lenguajes de modelado como UML[5] y CWM[6] son instancias de MOF.

En principio, se podrían añadir más capas pero en la práctica no tiene ninguna utilidad, considerando que la separación es puramente conceptual. En vez de definir una capa M4, la OMG definió que todos los elementos de la capa M3 deben definirse como instancias de conceptos de la misma M3. El concepto *UML Clase* y *UML Atributo* al que se hace referencia en la capa anterior, son instancias del elemento *MOF Clase* del meta-*metamodelo*.

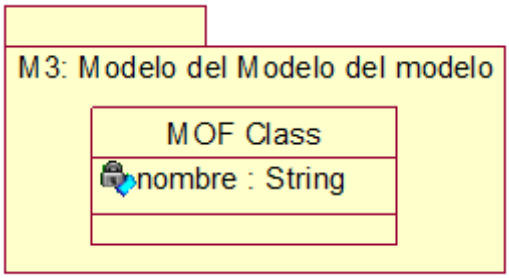

Figura 2.7 M3. El modelo del metamodelo

[4] MOF Meta-Object Facility
[5] UML Unified Model Languaje
[6] Common Warehouse Metamodel

En la siguiente figura se puede observar las relaciones entre las capas:

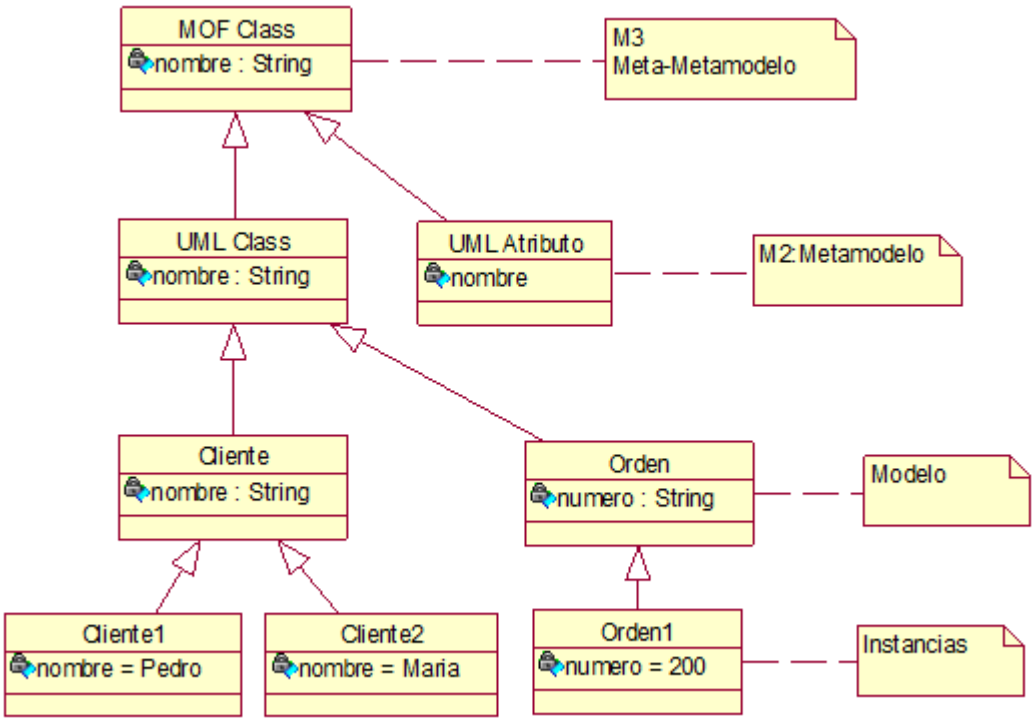

Figura 2.8 Niveles MOF

Existen dos razones por las cuales el metamodelado es importante en MDA. Primero, se necesita un mecanismo para definir lenguajes de modelado, sin ninguna clase de ambigüedades. De tal forma que una herramienta de transformación pueda leer, escribir y entender los modelos.

Segundo, las reglas de transformación que constituyen una definición de transformación describen la forma en que un modelo en un lenguaje fuente puede ser transformado en un modelo en un lenguaje objetivo.

2.3 VISIÓN GENERAL DE MDA [RS 00]

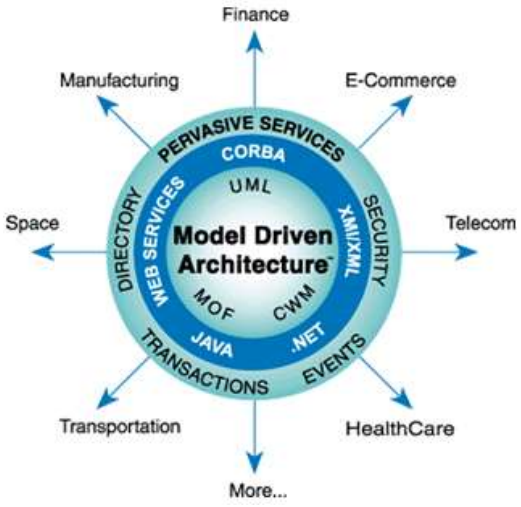

Figura 2.9. Visión General de MDA

2.3.1 Núcleo

El núcleo de la arquitectura de MDA está basado en los estándares de modelado de la OMG: UML, MOF, XMI y CWM. Estas tecnologías son usadas para describir PIMs. Un PIM puede ser refinado muchas veces hasta que el nivel deseado de descripción del sistema se haya obtenido. Después una infraestructura en particular es tomada en cuenta y el PIM es transformado en un PSM. Entonces, el PSM es refinado tantas veces como se necesite.

2.3.1.1 UML (Lenguaje Unificado De Modelado)[UML]

UML orienta el modelado de la arquitectura, objetos, interacciones entre objetos, aspectos de modelado de datos del ciclo de vida de la aplicación, como también, aspectos de diseño de desarrollos basados en componentes incluyendo construcción y ensamblaje.

Los artefactos capturados en modelos de UML (en términos de clases, interfaces, casos de uso, diagramas de actividad) pueden ser fácilmente exportados a otras herramientas utilizando XMI.

2.3.1.2 XMI (XML Metadata Interchange) [XMI]

En su nivel más simple, XMI define un mapeo desde UML a XML, proporcionando formatos estándares y Definiciones de Tipos de Documentos (DTDs) para capturar modelos UML (y metamodelos). Esto hace posible convertir un modelo UML en XML, distribuirlo

prácticamente a cualquier sitio, y luego convertirlo de nuevo a UML. Este mapeo hace posible el intercambio de modelos UML entre diferentes herramientas.

Técnicamente, el mapeo de XMI utiliza MOF y no UML. Sin embargo, debido a que UML esta basado en los metadatos de MOF, cualquier modelo definido con UML es compatible con las características de mapeo de XMI. Se está adelantando trabajo para extender XMI para soportar el estándar de la W3C, XML schema. El siguiente código muestra una instancia de un documento XMI 1.1 para un caso de uso "Ordenar Producto".

Figura 2.10 Caso de Uso Ordenar Producto

```
<?xml versión= "1.0" encoding="ISO-8859-1" ?>
-<XMI xmi.versión= "1.1" xmlns:UML= "//org.omg/UML/1.3"
+<XMI.header>
+<XMI.content>
</XMI>
```

El documento contiene una sección de cabecera y una sección de contenido.

La sección de cabecera especifica el proveedor que generó el XMI (TogetherSoft) y la versión de la utilidad exportada (6.0)

```
-<XMI.header>
      -<XMI.documentation>
            <XMI.exporter> TogetherSoft</XMI.exporter>
            <XMI.exporterVersion>6.0</XMI.exporterVersion>
      </XMI.documentation>
      <XMI.metamodel xmi.name= "UML" xmi.versión= "1.4"/>
</XMI.header>
```

Toda la información acerca del modelo de UML se encuentra en la sección Contenido:

```
<UML:UseCase xmi.id = 'S.10'
      name = 'Ordenar Producto'
      visibility = 'package'
      isSpecification = 'false'
      isAbstract = 'false'
</UML:UseCase>
```

2.3.1.3 MOF (Meta Object Facility) [MOF]

MOF es un estándar de la OMG que define un lenguaje común y abstracto para la especificación de metamodelos. MOF es un ejemplo de un Meta-metamodelo, o modelo del metamodelo. MOF es indiscutiblemente orientado a objetos por naturaleza. Éste define elementos esenciales, sintaxis y la estructura de los metamodelos que son usados para construir modelos orientados a objetos de sistemas discretos. MOF es el modelo común para los metamodelos CWM y UML. MOF es el corazón de MDA. Es el punto de partida, el estándar que define lenguajes usados para describir sistemas. MOF es también parte de la estrategia a largo plazo de la OMG para soportar la creación e intercambio de una variedad de metamodelos a través de diversos repositorios. Los principales constructores de metamodelos proporcionados por MOF son:

- Clases: Las clases pueden tener atributos y operaciones. Los atributos se utilizan para la representación de metadatos. Las operaciones soportan las funciones específicas sobre los metadatos. Las clases pueden heredar de otras clases.
- Asociaciones: Soportan los enlaces binarios entre instancias de clases. Cada asociación tiene dos finales de la asociación (AssociationEnds) que pueden especificar semántica de ordenamiento o agregación y restricciones de cardinalidad.
- Paquetes: Son colecciones de Clases relacionadas y asociaciones. Los paquetes pueden incluir paquetes importados y pueden heredar de otros paquetes.

2.3.1.4 CWM (Common Warehouse Metamodel) [CWM]

CWM es el estándar de almacenamiento de datos de la OMG. Este cubre el ciclo de vida completo de diseño, construcción y manejo de aplicaciones de almacenamiento de datos. Es probablemente el mejor ejemplo de aplicar el paradigma MDA a un área de aplicación ya que CWM permite el mapeo de PIMs de MDA a esquemas de bases de datos.

CWM cubre el diseño, construcción y gestión de aplicaciones de almacenamiento de datos y soporta la gestión de su ciclo de vida. La colección de metadatos proporcionada por CWM es lo suficientemente comprensible para modelar un sistema entero de almacenamiento de datos. Usando las herramientas de CWM, una instancia de almacenamiento de datos puede ser generada directamente del modelo.

CWM es una extensión de MOF. Cualquier metaclase dentro de CWM hereda de alguna metaclase del modelo de Objetos. Por ejemplo, considérese el paquete relacional de CWM. Como se ve en la figura, el metamodelo relacional define una metaclase llamada "Table" que representa cualquier tabla de una base de datos relacional. Esta metaclase se deriva de la metaclase del modelo de Objetos "Class". De manera similar, la metaclase relacional "Column" deriva de la metaclase "Attribute" del modelo de objetos. Esto define formalmente la relación semántica entre los conceptos Tabla y columna: Una tabla es "algo" que tiene propiedades (o atributos) y sirve como una plantilla para una colección de cosas.

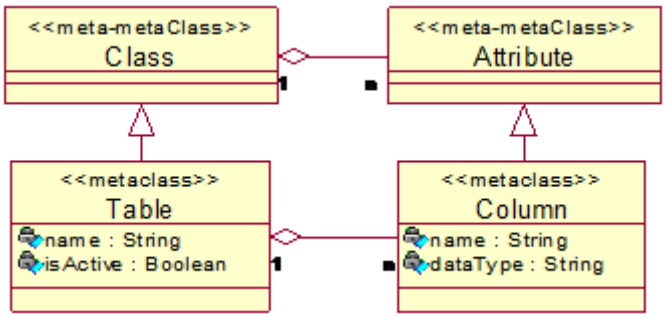

Figura 2.11 Relación MOF – CWM

2.3.2 Segundo Nivel - Ambientes Mediadores

En el anillo siguiente al núcleo, se muestran algunos de los ambientes mediadores que son actualmente objetivos para MDA, tales como: Servicios Web, CORBA (en particular el Modelo de Componentes de CORBA CCM); Java (incluyendo EJBs); C#/.NET; XML/SOAP4.

En MDA, la especificación base de cualquier servicio, facilidad o aplicación es un modelo independiente de la plataforma. En este ambiente, los arquitectos pueden especificar enlaces desde una aplicación hasta servicios y facilidades requeridas, y/u otras aplicaciones como parte de sus modelos. Trabajando con esta estructura de modelos enlazados, las herramientas de MDA automáticamente construyen **puentes** (Bridges) permitiendo conectar las implementaciones realizadas en diversas plataformas.

2.3.3 Tercer Nivel – Servicios

Gracias a la experiencia de la OMG con OMA (Arquitectura de Gestión de Objetos) se define en el tercer nivel, conociendo la necesidad de ofrecer los servicios que se requieren para soportar la computación distribuida. En MDA, estos servicios han sido denominados Servicios Transversales (Pervasive Services) debido a que una única implementación de éstos, independientemente de la plataforma en la que corran, puede servir a cualquier aplicación o cliente que los requiera valiéndose de los puentes entre plataformas generados por MDA. Entre los servicios definidos para MDA, tenemos:

Servicios de directorio, transacción, seguridad, notificación y eventos distribuidos, servicios adicionales, como los sugeridos inicialmente por la lista de servicios de CORBA, se irán añadiendo otros, a medida que se desarrollen especificaciones al respecto. [RS 00]

2.3.4 Cuarto Nivel – Dominios

El anillo exterior representa los mercados verticales o dominios. Desde Enero de 1996 un creciente porcentaje de los miembros de la OMG se han integrado en comités (Domain Task Forces) enfocados en la estandarización de servicios y facilidades en mercados verticales específicos. Hasta ahora, esas especificaciones se componen de interfaces escritas en OMG IDL acompañadas de una descripción semántica en Texto.

Un servicio o facilidad bien concebida debe estar basada en un modelo semántico independiente de la plataforma objetivo. Debido a que estos modelos están escondidos, es decir, no han sido llevados a un lenguaje de modelado como UML sino que están implícitos por las IDLs, estos servicios y facilidades no han sido ampliamente implementadas por fuera del ambiente CORBA. Buscando maximizar el uso de estas especificaciones, se entregarán artefactos como modelos en UML independientes de la plataforma, Modelos UML específicos e interfaces IDL de por lo menos una plataforma [RS 00].

2.4 MODELOS DE MDA

2.4.1. PIM - Modelo Independiente De La Plataforma [JS 01]

Todos los proyectos de desarrollo con MDA empiezan con la creación de un Modelo Independiente de la plataforma (PIM), expresado en UML. Un modelo MDA tendrá múltiples

niveles de PIMs. Aunque todos deben ser independientes de cualquier plataforma particular, todos exceptuando el modelo base incluyen aspectos independientes de la plataforma de comportamiento tecnológico.

El PIM base expresa únicamente funcionalidad y comportamiento. Construido por expertos del negocio y modelado, este modelo expresa las reglas del negocio y funcionalidad, distorsionado lo menos posible por la tecnología.

Debido a la independencia de la tecnología el PIM base conserva su aplicabilidad a través de los años, requiriendo cambios únicamente cuando las condiciones del negocio se modifiquen.

En [OMG 03] se denomina al PIM base como el Modelo Independiente de la Computación o CIM (de sus siglas en ingles), y se define como el modelo que muestra al sistema en el ambiente en el cual operará, lo cual ayuda a visualizar lo que se espera que el sistema realice.

PIMs en el siguiente nivel, incluyen algunos aspectos tecnológicos aunque los detalles específicos de las plataformas se mantengan ausentes.

Por ejemplo, todos los ambientes de componentes permiten a los desarrolladores especificar patrones de activación. (Muchos utilizan los términos comunes de *session* y *entity*). Conceptos adicionales- persistencia, transaccionalidad, niveles de seguridad e incluso información de configuración- pueden ser tratados análogamente, adicionándolos a un PIM de segundo nivel, que permita mapearlos más precisamente a un modelo específico de la plataforma (PSM).

Las herramientas de modelado de aplicaciones de MDA contendrán representaciones de los Servicios Transversales ("Pervasive Services") y las facilidades de dominio permitiendo que sean usadas o incorporadas en las aplicaciones en el nivel del modelo a través de su selección en un menú. Cualquier facilidad definida en UML puede ser importada en la herramienta, y ser usada por la aplicación de la misma manera.

Adicionalmente, si el servicio o la facilidad corre sobre otra plataforma, la herramienta de desarrollo generará invocaciones entre las plataformas automáticamente, evitando la codificación a mano para la integración entre plataformas.

2.4.2 PSM - Modelo Dependiente de La Plataforma:

Para generar el Modelo Dependiente de la Plataforma que se haya seleccionado como objetivo, las características de ejecución y la información que se diseñó en el modelo de la aplicación de una forma general, son convertidas a las formas específicas requeridas por dicha

plataforma.. Por ejemplo, al seleccionar J2EE como la plataforma de implementación, el PSM debe incluir elementos como interfaces Home y Remote para los EJBs de Entidad y Sesión descritos en el PIM.

2.5 TRANSFORMACIONES DE MODELOS

PIM, PSM y las técnicas de mapeo, están basados en metamodelos expresados preferiblemente en con las tecnologías del núcleo de OMG tales como: MOF, CWM o UML.
El mapeo es un conjunto de reglas y técnicas usadas para modificar un modelo buscando conseguir otro modelo. Las técnicas de mapeo son usadas para transformar:

2.5.1 PIM a PIM

Esta transformación es usada cuando los modelos son mejorados, filtrados o especializados durante el ciclo de vida del desarrollo sin necesitar ninguna información dependiente de la plataforma. Uno de los ejemplos más obvios de esta clase de mapeo es la transformación de los modelos de análisis a diseño.

2.5.2 PIM a PSM

Esta transformación es usada cuando el PIM es lo suficientemente refinado para ser proyectado a la infraestructura de ejecución. La proyección está basada en las características de la plataforma. Pasar de un modelo lógico de componentes a un modelo comercial existente de componentes (como EJB para la plataforma de J2EE o CCM para la plataforma CORBA) es una clase del mapeo de PIM a PSM.

2.5.3 PSM a PSM

Esta transformación es generalmente usada para el refinamiento de modelos dependientes de la plataforma.

2.5.4 PSM a PIM

Esta transformación se requiere para abstraer modelos de implementaciones existentes en una tecnología particular a un modelo independiente de la plataforma y debe ser soportada en herramientas.

2.6 MÉTODOS DE TRANSFORMACIÓN DE MODELOS

Para realizar las transformaciones pueden utilizase combinaciones de transformaciones manuales y automáticas. A continuación se describen 4 posibles transformaciones, descritas en [OMG 03]:

2.6.1. Transformación manual

La transformación manual no es muy diferente al proceso de diseño que se ejecuta normalmente en un proceso de desarrollo. MDA añade valor en 2 sentidos:

- La diferenciación explícita entre el modelo independiente de la plataforma y el modelo dependiente de la plataforma surgido de la transformación.
- El registro (record) de la transformación, que incluye un mapa desde el elemento del PIM hasta los correspondientes elementos del PSM y muestra cuales partes del mapeo fueron usadas por cada transformación.

2.6.2. Transformación de un PIM preparado usando un perfil UML

Un PIM puede ser generado a partir de un perfil UML. Seguidamente, este modelo puede ser transformado a PSM usando un segundo perfil UML, específico de una plataforma. Esta transformación puede involucrar la utilización de marcas proporcionadas por el perfil de la plataforma, sobre el PIM.

2.6.3 Transformación usando patrones y marcas

Los patrones pueden ser utilizados en la especificación del mapeo. Las marcas se utilizan para preparar un "PIM marcado" cuyos elementos marcados se transformarán acorde al patrón para producir un PSM.

2.6.4 Transformación automática

Hay contextos, o situaciones en los cuales el PIM proporciona toda la información requerida para la implementación, y por lo tanto no existe la necesidad de añadir marcas o utilizar datos de perfiles adicionales, para poder generar el código. En estos casos, es posible que el desarrollador de una aplicación construya un PIM donde esté completamente definida la estructura del sistema incluyendo pre y post condiciones. Para modelar el comportamiento directamente en el modelo, utiliza un lenguaje de acción, completando la totalidad de características requeridas del PIM, es decir, contiene toda la información necesaria para generar código. La herramienta de transformación es capaz entonces de interpretar el modelo y transformarlo en código.

2.7 DEFINICIONES DE TRANSFORMACIONES [AJW 03]

Esta parte del documento busca profundizar en la forma en que las transformaciones son definidas. Sin embargo, no debería ser requerimiento que los usuarios de MDA definieran sus propias transformaciones ya que una de las ventajas que ofrece MDA es que al soportarse en herramientas, muchas de estas transformaciones, particularmente las de las plataformas más requeridas, deberían ser realizadas de manera automática.
No obstante, se considera pertinente conocer los principios básicos que brinden al usuario de MDA las bases suficientes para discernir entre una herramienta u otra.

Para lograr una transformación de un modelo origen a un modelo objetivo, se debe relacionar cada elemento del primero con uno o más elementos del segundo. La forma más directa de llevarlo a cabo es relacionar la metaclase del modelo origen con la metaclase del modelo objetivo.
El siguiente es un ejemplo que ilustra lo descrito, en donde el lenguaje del PIM y del PSM es el mismo (UML).

- **Regla 1:** Por cada clase denominada *className* en el PIM hay una clase denominada *className* en el PSM.

- **Regla2:** Por cada atributo público denominado *attributeName* : *Type* de la clase *className* en el PIM, en el PSM se deben definir los siguientes atributos y operaciones para la clase *className* :
    - Un atributo privado con el mismo nombre *attributeName*: *Type*
    - Una operación pública denominada como el nombre del atributo con un prefijo "get" y el tipo del atributo como el tipo de retorno de la operación. *getAttributeName(): Type*
    - Una operación pública denominada como el nombre del atributo con un prefijo "set" y el atributo como parámetro y sin retorno. *setAttributeName(att: Type)*

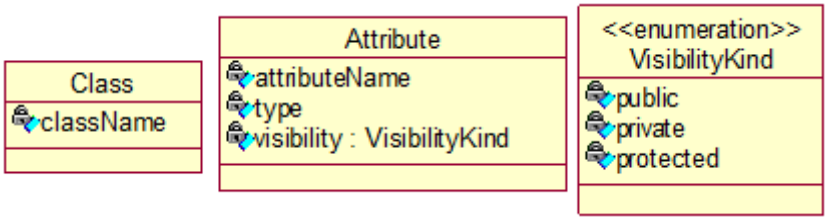

Figura 2.12. Metaclases Class y Attribute en el lenguaje del PIM

La primera regla hace referencia a Class, que es un elemento tanto del lenguaje del PIM y del PSM. En otras palabras, el metamodelo de dichos lenguajes de modelado incluyen la metaclase Class.

La segunda regla se refiere a otra metaclase denominada Attribute, que contiene 3 atributos, nombre del atributo, tipo y visibilidad.

Las metaclases Class y Attribute se ubican en el nivel M2.

De la segunda regla se puede inferir que el lenguaje PSM debe incluir una metaclase *Operation*, con atributos como nombre, parámetro, tipo de retorno y visibilidad. Ya que parámetro tiene un nombre y un tipo es también una clase.

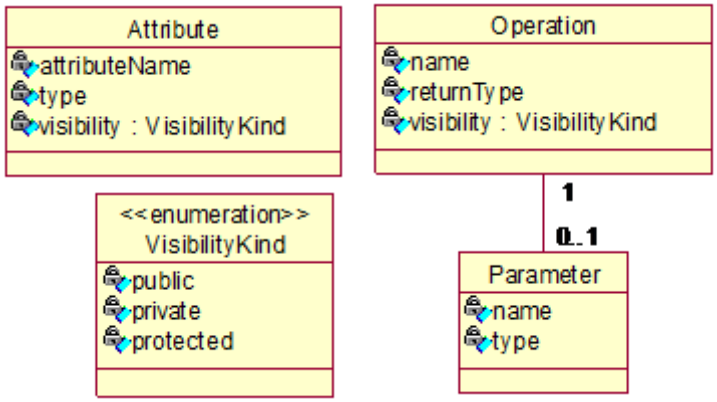

Figura 2.13. Las metaclases del lenguaje PSM

Dentro del marco de MDA, no existe todavía un lenguaje estándar para escribir definiciones de transformaciones. Sin embargo, ya existen grupos trabajando en la definición de un lenguaje denominado QVT (Query, Views and Transformations) que han publicado revisiones iniciales de la propuesta.

2.8 LENGUAJES DE MDA

2.8.1 QVT (Query, Views and Transformations) [QVT]

La propuesta de QVT es proporcionar un estándar para expresar las trasformaciones de modelos. QVT requiere que las transformaciones de modelos estén definidas precisamente en términos de la relación entre un metamodelo MOF fuente y un metamodelo MOF objetivo. Una transformación en este contexto consiste de los siguientes conceptos: definición de patrones, términos MOF, reglas de transformación, y relaciones de seguimiento y correspondencia.

Ejemplo: Definición de una transformación de UML a Java

TRANSFORMATION uml2java(SOURCE UML, TARGET Java, Tagging) TRACKING TModel;	Defini-ción de la Transfor mación
RULE umlClassifierToJavaClass(X, Y) FORALL UMLClassifier X WHERE X.name = N	TRegla

```
MAKE JavaClass Y,
Y.name = N
LINKING X, Y BY JavaClassFromUMLClassifier;
...
```

Figura 2.14 Definición de una transformación de UML a Java

Las reglas de transformación son utilizadas para describir las cosas que deberían existir en la extensión objetivo, basada en las cosas que existen en la extensión fuente. Las reglas de transformación pueden ser extendidas, permitiendo una descripción modular e incremental de la descripción de las transformaciones. Además, una regla puede ser remplazada por otra facilitando la optimización y reutilización. Las reglas de transformación consisten de un Término que identifica las fuentes de la regla y un conjunto de términos simples que identifican los objetivos de la regla.

En el ejemplo, las reglas son utilizadas para expresar mapeos desde conceptos en el modelo UML a conceptos en el modelo Java. En este caso, cada UML classifier es mapeada a una clase Java con el mismo nombre. Este mapeo es expresado utilizando la regla de transformación "umlClassifierToJavaClass" mostrada en la Figura 2.14.

Un término MOF permite hacer afirmaciones acerca de los elementos de la fuente y del objetivo. Por ejemplo, se utilizan las MOFInstances para hacer afirmaciones acerca del tipo de objetos de la fuente y el destino, que están involucrados en el mapeo expresado en la regla y MOFfeatures para hacer afirmaciones acerca de los valores de los atributos de los objetos. En el ejemplo anterior, se utiliza una MOFinstance para emparejar UMLClassifiers a la fuente y establecer que la variable X es un tipo de UMLCLassifier. Una MOFfeature es utilizada para enlazar el valor del atributo nombre de la clase Classifier a la variable N. En la siguiente parte de la regla, se utiliza una MOFInstance para establecer que la variable Y es del tipo JavaClass, y un MOFfeature para afirmar que el valor del atributo nombre de la clase Java debe ser el valor de la variable N.

Una correspondencia es una afirmación de una dependencia funcional entre elementos objetivos y el conjunto de elementos de la fuente que es caracterizada en el modelo de transformación por clases rastreadas y representadas en tiempo de ejecución por instancias de las clases rastreadas. Se utilizan TrakingUses que referencian clases rastreadas con reglas de

transformación para establecer correspondencia entre los elementos de la fuente y los del destino. Una clase rastreada es clase MOF normal que incluye un atributo extra "KEY" el cual referencia al conjunto de atributos de la clase que son atributos identificados.

Un TrakingUse en el objetivo de la regla de transformación afirmará la correspondencia entre los elementos de la fuente y el destino que están determinados por la variables proporcionadas, mientras que en la fuente actuará como una consulta sobre todas las afirmaciones de la correspondencia, y atará las variables a los resultados de dicha consulta.

En el ejemplo, se utiliza una TrakingUse "JavaClassFromUMLClassifier" en el objetivo de la regla "umlClassifierToJavaClass" para establecer una correspondencia entre los UMLClassifiers y las clases java a las cuales son mapeados.

Se hace uso de la correspondencia establecida en "umlClassifierToJavaClass" en la regla de transformación "umlAttributeToJavaField", mostrada en la figura 2.15, la cual mapea los atributos propios de cada UMLClassifier a campos de Java pertenecientes a clases Java mapeadas desde los UMLClassifiers.

El TrakingUse "JavaClassFromUMLClassifier" es utilizado para consultar la correspondencia entre los UMLClassifiers en la fuente y las clases Java en el objetivo y enlazar "JC" a la clase Java que fue mapeada desde el UMLClassifier que es el poseedor del UMLAttribute acoplado.

La variable JC es entonces utilizada en la MOFfeatures en el objetivo de la regla, para proporcionar un valor para el atributo del campo de Java "Y" que es mapeada desde el UMLAttribute X.

```
CLASS JavaClassFromUMLClassifier {
UMLAttribute a;
JavaClass c;
KEY (a);
}
RULE umlAttributeToJavaField
FORALL UMLAttribute X
WHERE JavaClassFromUMLClassifier LINKS X.owner, JC
MAKE JavaField Y,
   Y.owner = JC
```

```
LINKING X, Y BY FieldFromAttr;
```

Figura 2.15 Correspondencia de una clase UML a una clase Java

Las definiciones de patrones son utilizadas para definir estructuras comunes que pueden ser repetidas a lo largo de la transformación. Los patrones son utilizados generalmente para simplificar y modularizar la construcción de reglas de transformación, ya que un patrón puede ser utilizado por múltiples reglas.

Para el ejemplo particular, es posible que se declaren muchas reglas que hagan referencia a un UMLClassifier y su nombre, como también a una clase Java y su nombre, por lo cual se crea una definición de patrón que represente esos atributos comúnmente usados. La regla de transformación "umlClassifierToJavaClass" puede ser re-escrita haciendo referencia a dichos patrones, usando un PatternUse. Ver figura 2.16.

```
DEFINE PATTERN umlClassifierAndName(X,N)
FORALL UMLClassifier X
WHERE X.name = N;
DEFINE PATTERN javaClassAndName(X,N)
MAKE JavaClass X,
X.name = N;


RULE umlClassifierToJavaClass(X, Y)
FORALL umlClassifierAndName(X, N)
MAKE javaClassAndName(Y, N)
LINKING X, Y BY JavaClassFromUMLClassifier;
```

Figura 2.16 Regla de transformación de una clase UML a una clase Java utilizando definiciones de patrones

2.8.2 OCL Lenguaje de restricciones de Objetos [TP 03]

El Lenguaje de restricciones de (constraint) de Objetos (OCL) está incorporado en UML como un estándar para especificar detalles adicionales, o precisar detalles en la estructura de los modelos.

OCL es más funcional que el lenguaje natural, pero no tan preciso como un lenguaje de programación. Puesto que OCL es un lenguaje para la expresión pura, sus declaraciones están garantizadas de no tener efectos laterales - simplemente transportan un valor y nunca pueden cambiar el estado del sistema.

Una restricción puede ser especificada como un invariante, una precondición o una post condición. Esta descripción implica que OCL es usado para especificar requerimientos estáticos.

Una precondición define el estado que un sistema (u otro subsistema) debe asumir antes de que una acción determinada se ejecute. De la misma forma, una post condición define el estado en el que un artefacto debe estar cuando la acción se completa. Y un invariante define un estado que el artefacto debe mantener durante su ciclo de vida.

El contexto de una expresión OCL especifica la entidad del modelo para la cual dicha expresión es definida. Usualmente un contexto es una clase, interface, tipo de dato o componente.

Para ejemplificar lo anterior se supone una clase persona con los siguientes atributos:

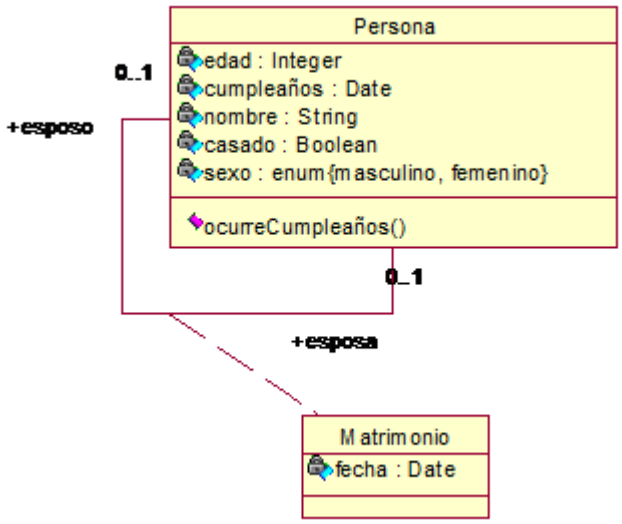

Figura. Modelo ejemplo OCL

Invariantes de Persona:

- Una persona debe tener 1 o más años de edad

```
context Persona inv: -- atributo
self.edad>0
```

- Un matrimonio debe ser entre un hombre (maculino) y una mujer (femenino)

```
context matrimonio inv:
self.esposa.sexo = #femenino and
self.esposo.sexo = #masculino
```

- Una persona no puede tener a la vez un esposo y una esposa

```
context persona inv:
not ((self.esposa->size=1) and (self.esposo->size=1))
```

Precondiciones y postcondiciones:

- Cuando una persona cumple años, su edad es igual a la edad anterior + 1.

```
@pre, postconditions
context Persona::ocurreCumpleaños()
post: edad = edad@pre+1
```

La especificación que existe actualmente de OCL se describe en el documento "UML 1.5 with Action Semantics". Para la especificación OCL 2.0 ya salió el RFP y se encuentra en proceso de discusión.

OCL 2.0 incluye unas mejoras al estándar existente:

1. OCL 2.0 tiene un metamodelo consistente con MOF 2.0, entonces OCL utiliza y extiende el mismo metamodelo utilizado para crear los lenguajes UML y CWM. Consecuentemente, a cualquier modelo que haya sido creado a partir del MOF es posible aplicar OCL, incluyendo el mismo MOF.
2. OCL 2.0 define el lenguaje a dos niveles, una sintaxis abstracta (o metamodelo) y una sintaxis concreta. El metamodelo define los conceptos y reglas para aplicar dichos conceptos. La sintaxis concreta implementa la sintaxis abstracta. La sintaxis concreta proporcionada por la especificación de UML 2.0 es solo una de las muchas implementaciones del metamodelo OCL.

2.9 PROCESO DE DESARROLLO CON MDA [MTR 01]

En este apartado se describe una aproximación que fue desarrollada por el proyecto MASTER (IST-2001-34600), cuyo objetivo es validar el concepto de MDA. Esta validación consiste en aplicar conceptos de MDA para el dominio de la gestión del tráfico aéreo.

Para llevar a cabo el proyecto, se definen una secuencia de modelos que permiten estructurar la organización del desarrollo basado en MDA. Esta secuencia incluye el proceso desde sus inicios (captura de requerimientos), hasta la generación del código a partir del PSM.

2.9.1 Modelos de requerimientos y de contexto

Permiten ver el sistema como una caja negra. Estos modelos describen la especificación frontal del sistema y detalla los requerimientos

2.9.1.1 PIM De Contexto

El PIM del contexto:

- Introduce el sistema y sus objetivos (Que se espera del sistema)
- Describe los principios del negocio que estructuran el sistema
- Describe los actores externos que interactúan con el sistema
- Introduce servicios de alto nivel requeridos para interactuar con el sistema, como comportamientos claves del sistema
- Define objetos del negocio.

El PIM de contexto es la entrada para elaborar el modelo de requerimientos.

El modelamiento del PIM del Contexto incluye 3 actividades:

- **Definir el sistema:** Se identifica el sistema. Las metas, los principios de negocio y los servicios de alto nivel o eventos ofrecidos por el sistema son descritos.
- **Definir los límites del sistema:**

  Identificación de los actores.

  Identificación de los objetos del negocio (como conceptos claves)

  Identificar los flujos de datos entre los actores y el sistema.
- **Definir las principales funcionalidades del sistema:**

  La técnica usada para esta descripción son los casos de uso de alto nivel. Cada caso de uso representa un área funcional. En este modelo, solo se tienen en cuenta las principales interacciones de los actores con el sistema.

2.9.1.2 PIM De Requerimientos

Describe como se requiere que el sistema se comporte en el momento de ser simulado. El objetivo es doble:

- Construir un modelo de las expectativas del cliente con un vocabulario claro.
- Contar con una descripción única de requerimientos que pueda ser usada por los modelos subsecuentes.

El modelo de requerimientos contiene dos clases de requerimientos: Funcionales y no funcionales. Para tener una aproximación completa del modelo, cada requerimiento textual debe estar relacionado con un elemento del modelo.

El PIM de requerimientos incluye 3 actividades mayores:

- **Especificar los requerimientos funcionales**

  Refinar la descripción del sistema, los actores, servicios, eventos y objetos del negocio del PIM del contexto.

  Identificar y organizar las capacidades. Una capacidad es un caso de uso funcional. Ellos se organizan acorde con las áreas funcionales identificadas en el PIM de contexto.

- **Especificar los requerimientos no funcionales**

  Identificación y organización de las fuerzas. Una fuerza expresa un requerimiento no funcional de alto nivel.

- **Relacionar los requerimientos funcionales y los no funcionales**

2.9.2 PIM de análisis

Especifica una visión interna del sistema (vista de caja blanca) sin ninguna consideración tecnológica o software. Contiene la especificación funcional del sistema enfocada en el dominio y áreas de aplicación y obedece al principio de separación de los requerimientos funcionales de los no funcionales. La parte funcional describe los elementos del sistema (con clases, atributos, paquetes, etc), sus funciones (con operaciones), y sus fronteras (interfaces).

Cada elemento del modelo de análisis encaja con el modelo de requerimientos. Esto es esencial para poder seguir el rastro (traceability).

Este modelo es visto como el más duradero estando libre de cualquier plataforma e incluso de cualquier estilo arquitectónico.

El modelado del PIM de análisis involucra 2 actividades mayores:

- **Mantenimiento de interfaces externas**

  Buscando garantizar consistencia con las especificadas en el PIM de requerimientos.

- **Realización del análisis de dominio**

  Aquí se construye el núcleo de la arquitectura funcional del sistema. Se crean las realizaciones funcionales, refinando una por una las capacidades identificadas en el PIM de requerimientos.

2.9.3 PIM de diseño

El PIM de diseño representa una solución independiente de la plataforma expresada en términos tecnológicos. Es decir, utiliza conceptos utilizados en paradigmas de desarrollo y estándares de notación que no se relacionan estrechamente con ninguna plataforma particular. Este modelo, siendo todavía un PIM especifica el conocimiento requerido para producir los subsecuentes modelos dependientes de la plataforma escogida.

2.9.4 PSM

El modelo dependiente de la plataforma es el último paso antes de la generación automática de código. El PSM llena el vacío entre un diseño genérico y una implementación específica. El PSM tiene en cuenta tanto aspectos propios del negocio, como característicos de la plataforma. Además, debe incluir tanto los requerimientos funcionales como los no funcionales. Para diseñar un PSM se deben tener en cuenta algunos artefactos, a saber:

- **Metamodelo De La Plataforma:** Una plataforma proporciona un conjunto de conceptos de programación con los cuales se deberá mapear el PIM.
- **Perfil de la plataforma:** Para construir un PSM que utiliza conceptos de una plataforma se debe mapear el metamodelo de la plataforma al mundo UML. El propósito de los perfiles de plataforma, dar una representación UML de cada concepto de dicha plataforma.
- **Metamodelo del PSM:** Define todos los artefactos de una capa específica de la plataforma.

Para el desarrollo de modelos basados en MDA deben tenerse en cuenta dos consideraciones:

- La arquitectura debe ser transversal a todos los modelos.
- Los modelos son los primeros elementos generados durante el desarrollo. Los demás artefactos como archivos de código, de configuración, documentos, entre otros se espera que se generen a partir de los modelos.

2.10 DE LOS REQUERIMIENTOS DEL CLIENTE AL PIM [AJD 01]

Uno de las principales metas y ventajas de MDA es la automatización de la derivación de modelos, que permitan pasar de los requerimientos al código.

Basándonos en la experiencia del proyecto [MTR 01], se considera que la automatización propuesta por MDA alcanza su mayor potencialidad haciendo uso de conceptos de ingeniería de líneas de productos, ya que es necesario contar con un dominio estable en el que una arquitectura genérica del PIM para tales sistemas ha sido definida y desarrollada. La ingeniería de líneas de productos es una aproximación del desarrollo de software que busca, específicamente, valerse de las características comunes y variables de las funcionalidades de sistemas que pertenezcan a un mismo dominio en términos de la reutilización a gran escala.

La captura de requerimientos de un sistema software es una práctica que se lleva a cabo en todos los procesos de desarrollo. Sin embargo, esta práctica debe ser desarrollada de una manera diferente cuando los requerimientos corresponden a un sistema a la medida, que cuando hacen referencia a una familia de sistemas, donde cada sistema miembro comparte características comunes con los demás.

Para el último caso la aplicación de ingeniería de familias de sistemas permite aprovechar muchas ventajas de las características comunes de los miembros.

La ingeniería de familia de sistemas incluye dos disciplinas principales:

Ingeniería del dominio: Comprende el conjunto de practicas de ingeniería para la definición de dominios (la familia de sistemas), establecer los aspectos comunes y variables entre los miembros del dominio, y crear una base de aspectos reutilizables del dominio.

Ingeniería de la aplicación: Comprende el conjunto de prácticas de ingeniería para capturar los requerimientos específicos de un cliente y hacer uso de las características del dominio para construir la solución software para dicho cliente.

El PIM del dominio debe comprender entonces las partes comunes y variables del sistema, las cuales deben tenerse en cuenta cuando se defina un PIM específico para un cliente específico. Es por esto que existe la necesidad de identificar que partes del PIM son comunes a todos los posibles sistemas finales y que partes son dependientes de los requerimientos del usuario.

Esta actividad es parte del análisis del dominio en la cual se identifican y especifican las partes variables y comunes a la línea de productos conforme a un modelo del dominio de dicha línea.

Las características comunes identificadas en el análisis del dominio formarán la arquitectura básica del PIM genérico del dominio. Y las variaciones determinarán un PIM de un sistema específico para unos requerimientos de usuario específicos.

2.11 HERRAMIENTAS DE MDA

Las herramientas que en este momento dan soporte a MDA de manera formal son pocas. Existen varias aproximaciones que intentan seguir el paradigma del desarrollo orientado por modelos y la generación automática de código que si bien facilita la implementación de una aplicación particular, no cumplen rigurosidad de una especificación completa.

En esta sección se describirán algunas de las herramientas estudiadas que se consideraron de mayor relevancia por su pertinencia y cercanía al paradigma. Entre estas herramientas encontramos: ArcStyler 4.0, Metanology y Objectering.

Además, se añade una comparación entre ellas y otras herramientas del mercado.

2.11.1 ARCSTYLER

ArcStyler ofrece un IDE para la generación de software basado en MDA. ArcStyler soporta el ciclo completo de desarrollo: la creación de los modelos de negocio independientes de la plataforma, la transformación automática a un modelo dependiente de la plataforma y una generación de código optimizada para gran variedad de plataformas como J2EE, CORBA, .NET, EAI entre otras.

Plataformas Soportadas

ArcStyler soporta la creación de software para múltiples plataformas. Para esto, utiliza cartuchos extensibles para MDA. Los cartuchos de MDA empaquetan los ítems y mecanismos específicos de la tecnología requeridos para las transformaciones (reglas de transformación de modelos, reglas de verificación de modelos, información de perfiles UML). ArcStyler incluye cartuchos de referencia y estándares para diferentes arquitecturas (e.g. C#/.NET incluyendo aplicaciones web basadas en ASP.NET, Java/J2EE incluyendo aplicaciones web basadas JSP).

Transformación de Modelos

Las funciones de mapeo que transforman modelos a otros modelos o artefactos de texto, tales como fuentes de código o Scripts, constituyen el núcleo de MDA. Una función de mapeo puede usar uno o más modelos como entrada y producir un modelo de salida.

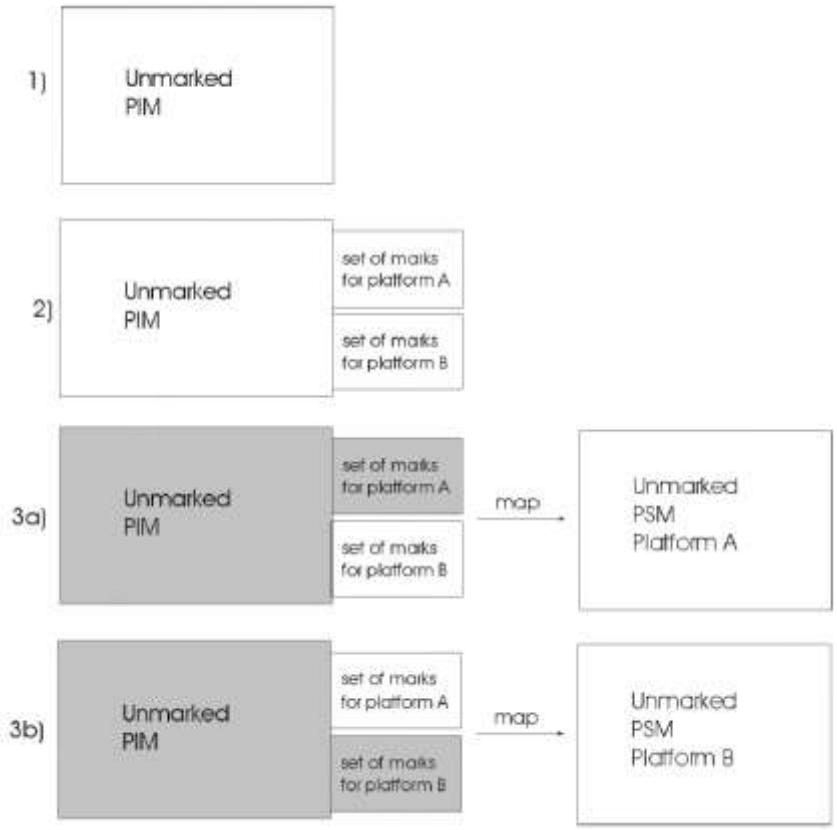

Figura 2.17. Marcación de un modelo para funciones de mapeo diferentes para dos plataformas distintas

En ArcStyer, en adición a los modelos de entrada, las funciones de mapeo pueden usar marcas como entradas. Las marcas son pequeñas anotaciones a los elementos del modelo que pueden contener cualquier información que sirva como entradas adicionales a los cartuchos MDA. Estas marcas pueden contener la información específica de la plataforma. Las marcas pueden ser adheridas y removidas fácilmente del modelo, no contaminan el modelo con especificaciones dependientes de la plataforma. En lugar de esto, conjuntos separados de marcas pueden coexistir simultáneamente en un modelo, haciendo posible mapear el mismo modelo a diferentes plataformas sin haber cambiado el modelo propiamente dicho.

ArcStyler define las Marcas independientemente de su representación en UML. Cualquier cartucho de MDA de ArcStyler define un conjunto de marcas que él espera en un modelo de entrada. Las definiciones de dichas marcas pueden ser añadidas o removidas de un repositorio.
El conjunto de definiciones de las marcas proporcionan tipos para todas las marcas, especifica a que tipos de los elementos del modelo son aplicables y sus valores por defecto.
ArcStyler define la arquitectura de cartuchos CARAT (Cartridges Architecture) la cual está basada en la idea de aplicar MDA a la creación de funciones de mapeo. En otras palabras: ArcStyler utiliza modelos para especificar las reglas de mapeo y usa un cartucho MDA para derivar la implementación del modelo de reglas.
Si es requerido modificar las reglas de mapeo, ArcStyler ofrece un soporte para el desarrollo de cartuchos, incluyendo un IDE de cartuchos MDA. Diversos conceptos como la herencia, permiten un desarrollo rápido de cartuchos simplemente extendiendo un cartucho y refinando el sub-cartucho. También se pueden desarrollar cartuchos desde cero si los existentes no resuelven un problema particular.

2.11.2 METANOLOGY (MDE)

MDE es un ambiente integrado de desarrollo dirigido por modelos. MDE transforma un modelo UML de la aplicación a gran parte de la implementación a través de metaprogramas.

Plataformas Soportadas

Metanology se utiliza únicamente para generar aplicaciones de la arquitectura J2EE.

Transformación de Modelos

Con MDE se modelan las aplicaciones software utilizando UML. Se desarrollan metaprogramas que interpretan modelos para una arquitectura y plataforma específica. Se ejecutan los metaprogramas con los modelos para crear la implementación de una aplicación en una arquitectura y plataforma específica. Se completa la aplicación utilizando los editores integrados u otro IDE.

Los modelos se construyen utilizando los diagramas de componentes, clases y casos de uso de UML. Los casos de uso y las clases se manipulan en la vista de paquetes (Packages View). Los componentes se manipulan en la vista de Subsistemas (Subsystems View).

2.11.3 OBJECTEERING

Objeectering/UML Profile Builder es un editor de perfiles, aplica MDA, utilizando los perfiles UML estándares de la OMG o especializados, de acuerdo a las necesidades del cliente.

Plataformas Soportadas

Objectering soporta transformaciones automáticas a código C++ y Java. Por medio de la herramienta de Objectering Profile Builder permite definir perfiles que permiten extender estas posibilidades.

Transformación de Modelos

Objecteering/UML es utilizado para expresar y gestionar requerimientos, compilar modelos en UML 1.4, generar documentación y automatizar la generación de código para Java, C++, VB, C# and CORBA IDL.

La herramienta CASE contiene unos módulos que proporcionan servicios particulares sobre modelos construidos. Un módulo es un grupo de servicios, empacados independientemente, que pueden ser seleccionados. Técnicamente, un módulo es un grupo de perfiles UML que pueden ser definidos a través de la herramienta Objecteering/UML profile Builder. Los siguientes son los módulos disponibles:

Objecteering/Documentation

Objecteering/UML profile Builder

Objecteering/C++

Objecteering/Java

Objecteering/Design Patterns for C++/Java

Con Objecteering/UML se puede trabajar en la herramienta de modelado todo el tiempo durante el desarrollo. El automáticamente genera los siguientes productos:

documentación

Código en C++, Java u otro lenguaje.

UML Profile Builder

Es una herramienta utilizada para definir y adaptar los modelos a las necesidades particulares, por medio de la definición de valores etiquetados, items de documentos, así como las reglas de chequeo, validación y generación. Se utiliza para diseñar e implementar perfiles UML, los cuales son conjuntos pre-definidos de estereotipos, valores etiquetados, restricciones usados para especializar y ajustar a la medida el metamodelo UML a un dominio específico o proceso. Un perfil UML no extiende UML añadiendo nuevos conceptos básicos, sino que proporciona convenciones para aplicar el estándar UML a un dominio particular.

Por ejemplo, Objecteering/UML profile builder se usa para definir nuevos generadores de código, adaptar generadores existentes, establecer consistencia de las reglas de chequeo, automatizar los patrones de diseño, definir plantillas de documentos o plantillas de generación, entre otros. Además, proporciona un lenguaje dedicado al uso de UML por perfiles UML llamado lenguaje J. La sintaxis del lenguaje J es muy cercana a Java.

Funciones de la herramienta

Objecteering puede ser usada para crear módulos y los siguientes servicios:

- La creación de nuevos servicios (métodos J)
- La definición de parámetros, los cuales permiten que el usuario proporcione opciones al módulo.
- Definición de comandos, los cuales son traducidos a menús en la herramienta de Modelado de Objecteering
- Definición de tipos de notas, tipos de valores etiquetados y estereotipos.
- Definición de plantillas de documentos o de generación de código.

Estructura

Los elementos de parametrización se estructuran en perfiles UML y módulos:

Un perfil UML contiene métodos J, definición de parámetros, tipos de notas y tipos de valores etiquetados, plantillas de documentos y productos de trabajo de generación.

El módulo, el cual es una entidad de más alto nivel, referencia uno o varios perfiles UML. Por ejemplo, el módulo Objecteering/C++ referencia un perfil UML para la generación de código C++ y un perfil UML para la generación de makefiles C++.

2.12 COMPARACIÓN DE LAS HERRAMIENTAS SELECCIONADAS Y OTRAS OFRECIDAS EN EL MERCADO

	ArcStyler	AndroMDA	MDE for Struts	OBJECTEERING UML	OptimalJ
Compañia	IO-Software	Open Source	Metanology	Objecteering	Compuware
Precio	1890-9800 EUR		399$, 799$ (with meta programming)	La edición personal es libre	800 – 10000 $
S.O. Soportados					
Windows	X Modelado requiere Rational	X	X	X	X
Linux	_	X	X	X	X
Mac	_	X	X	_	X
Soporte XMI					
Versión		1.1		1.1	1.1
Exporta	X	X		X	X
Importa	X			X	X
Soporte OCL					
Modelado	X	X	_	_	X
Generación de código	X	_	_	_	X
Soporte a estándares abiertos					
Struts		X	X		X

Ant	X	X	X		X
Generación de código					
Lenguajes soportados	Java, .NET	Cualquiera	Java	Java,C++	Java
Generadores de código	X	X	X	X	X

Tabla 2.1 Comparación de herramientas de MDA

2.13 CONCLUSIONES MDA

1. MDA se postula como una alternativa para los problemas de interoperabilidad entre aplicaciones, vista desde dos perspectivas:
    a. Interoperabilidad Conceptual: ya que las implementaciones partirían del mismo PIM base. Esto implica que sin importar las características de las plataformas, las diversas implementaciones del PIM deben comprender los conceptos en éste definidos. Por lo cual, no sería necesario hacer acuerdos semánticos ni sintácticos.
    b. Interoperabilidad tecnológica: ya que es posible la especificación de enlaces desde una aplicación hasta servicios y facilidades requeridas, y/o otras aplicaciones como parte de los modelos. Trabajando con esta estructura de modelos enlazados, las herramientas de MDA automáticamente construyen puentes (Bridges) permitiendo conectar las implementaciones realizadas en diversas plataformas.

2. MDA ha ido modificando su forma a través de las tres publicaciones que la definen. En cada una de ellas se han definido conceptos y se han refinado las definiciones de los propuestos en las publicaciones anteriores, sin lograr aún una respuesta concreta a la cuestión de su forma de abordaje.

3. MDA es una propuesta que integra diversas especificaciones y estándares. Sin embargo, en sí misma no es una especificación y aún no ha pasado el proceso de adopción tecnológica que la OMG define para la publicación de las especificaciones. No

obstante, la propuesta se lanzó con grandes expectativas y las industrias de software adaptaron las definiciones encontradas en los documentos existentes según su interpretación y se implementaron herramientas e incluso se han propuesto metodologías que integran los modelos de MDA durante el proceso de desarrollo. Pero que es MDA? Un modelo? una referencia? Una metodología de desarrollo? No existe claridad en su definición esencial, solo aproximaciones como la expresada en este trabajo.

4. La propuesta de MDA busca la automatización en la generación de código. Para esto es indispensable contar con herramientas que implementen los perfiles de las diversas plataformas de desarrollo e interoperabilidad, que soporten estándares de intercambio de datos como XMI, repositorios de modelos, y que sus metamodelos sean una instancia de MOF. En este momento, las herramientas no cumplen estrictamente con los estándares de núcleo de MDA y cada cual se encarga de definir sus propios metamodelos impidiendo la reutilización de PIMs independientemente de la herramienta. Por otro lado, las herramientas disponibles en el mercado son costosas y las que son libres no proporcionan un entorno amigable.

5. Aprovechando la experiencia del proyecto Master [MASTER], se puede concluir que el entorno de aplicabilidad de MDA se orienta a desarrollos empresariales donde se haya podido definir una línea de productos. Esto debido a que en ese caso, partiendo de una análisis de dominio, se pueden definir características comunes de la arquitectura que serían descritos en el PIM, que pueden ser complementadas posteriormente con aspectos específicos resultados de la captura de requerimientos de soluciones particulares. De esta forma, y contando con las herramientas adecuadas de generación de código, se puede automatizar prácticamente la implementación de las soluciones generales o específicas.

6. La definición y estandarización de perfiles de dominios podrían ser un punto a favor para la aplicabilidad de MDA en desarrollos relacionados ya que los conceptos necesarios para la implementación de cualquier solución en este campo deberían estar

considerados en el perfil. Las herramientas de generación de código deberían contener dichos perfiles de dominio permitiendo construir un PIM a partir de la instanciación de los conceptos definidos por el perfil que seguidamente serían mapeados por medio de los perfiles tecnológicos a las plataformas de implementación.

CAPITULO 3. INTEGRACIÓN DE APLICACIONES

3.1 APROXIMACIÓN A LA INTEGRACIÓN DE APLICACIONES

La integración de aplicaciones es una asociación estratégica para unir sistemas de información a nivel del servicio y de la información, soportado en su habilidad para intercambiar información y solucionar procesos en tiempo real. La información resultante y el flujo de procesos entre sistemas internos y externos, brindan a las empresas una clara ventaja estratégica de negocios: la habilidad de hacer negocios en tiempo real, en un entorno de eventos dinámico y con tiempos reducidos [DL 03].

La integración de aplicaciones puede tomar muchas formas, incluyendo integración de aplicaciones internas (Enterprise Application Integration EAI) o integración de aplicaciones externas (Business to Business Application Integration B2B). Cada forma tiene su propio conjunto de particularidades, pero cuando se sumerge en la tecnología relacionada con cada una de ellas se puede ver que tanto las soluciones de integraciones internas y externas, comparten muchos patrones comunes. Por ejemplo, casi siempre debe estar presente alguna tecnología de transformación de datos, para sobrellevar los problemas de las diferencias en la semántica de la aplicación, o por otro lado en la tecnología de enrutamiento para asegurar que la información va al destino correcto y en las reglas de procesamiento que definen el comportamiento de integración.

Hay que tener en cuenta que el concepto de integración de aplicaciones no es nuevo. Se ha lidiado con mecanismos para conectar aplicaciones desde que existen más de dos sistemas de negocios y una red entre ellos. Lo nuevo es la necesidad que existe de integrar aplicaciones para soportar iniciativas estratégicas de negocios que evolucionan, tales como participar en mercados electrónicos, visibilidad Web, Manejo de Relaciones con Clientes (CRM), y la necesidad real de tener a todos los sistemas intercambiando información y servicios. De hecho, cada vez más se ve la necesidad de entender la integración de aplicaciones como algo que requiere bastante de definición de negocios y de planeación de arquitecturas [MO 04].

Entonces, ¿cómo se pueden beneficiar las empresas al aplicar la integración de aplicaciones? Para ello primero se necesita entender la necesidad, luego los requerimientos y por último cómo resolver el problema para un dominio específico.

Se necesita pensar más comprensivamente acerca de cómo se captura y se reacciona ante eventos, se necesita reconocer que todos los componentes integrados a una empresa afectan su cadena de productos o servicios.

Simplemente se piensa que la integración de aplicaciones es un problema complejo. La realidad es que la mayoría de proyectos de integración de aplicaciones existen solo en el nivel inicial, todavía no se han visto muchas aplicaciones acopladas en tiempo real. Como con cualquier problema complejo, una vez está dividido en sus componentes, la solución es simplemente la agregación de un grupo de soluciones. En este caso, es la combinación de una variedad de acercamientos compuesto de muchos tipos de tecnologías.

El mundo de la integración de aplicaciones no es muy diferente del gran mundo de la tecnología. Irónicamente a medida que cambia la tecnología, la solución que se debe diseñar para cierto problema también cambia. El problema de integración de aplicaciones está cambiando de lo más simple a lo más complejo, y aún más si se mueve de un problema de un departamento de una empresa a un problema de la empresa completa y luego a un problema de una comunidad de negocios. Pocas empresas han sido capaces de pasar la "curva de integración de aplicaciones", al no sobrepasarla no se tendrá una solución completa sin poder descubrirse el gran potencial y los beneficios de esta.

3.1.1 Clasificación de la Integración de Aplicaciones [DL 03]

La integración de aplicaciones es una combinación de problemas. Cada organización y comunidad de negocios tiene su propio conjunto de problemas. Debido a esto, es imposible encontrar una única solución tecnológica que pueda ser aplicada universalmente. Por consiguiente, cada conjunto de soluciones de integración de aplicaciones generalmente requerirá de diferentes alternativas. Aunque las soluciones a la integración de aplicaciones varían considerablemente, es posible crear algunas categorías generales, las cuales incluyen:

- Soluciones de Integración Orientadas a la Información
- Integración Orientada al Procesamiento de Negocios
- Soluciones de Integración Orientadas al Servicio
- Soluciones de Integración Orientada al Portal

Soluciones Orientadas a la Información

Las tecnologías que promueven un acercamiento orientado a la información para integración de aplicaciones, argumentan que la integración debe ocurrir entre bases de datos (o entre API's propietarias que producen tal información, tales como BAPI[7]) esto es bases de datos o API's que producen información deben ser vistas como puntos primarios de integración.

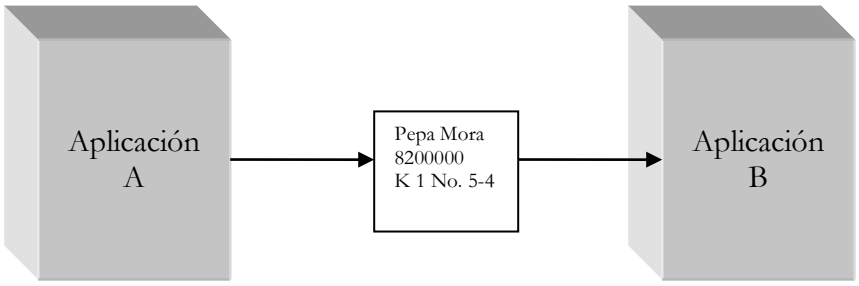

Figura 3.1 Integración de aplicaciones integradas a la información tratan con un simple intercambio de información entre dos o más sistemas

Las soluciones de Integración de Aplicaciones Orientadas a la Información (IOAI Information Orientation Application Integration de sus siglas en inglés) se pueden agrupar en tres categorías: replicación de datos, federación de datos e interfaz de procesamiento.

Replicación de Datos

La replicación de datos es simplemente mover datos entre dos o más bases de datos. Estas bases de datos pueden provenir de diferentes proveedores.

[7] BAPI: Business Application Programming Interfaces. El Sistema SAP R/3 consta, en la vista modular, de áreas empresariales homogéneas, que soportan las operaciones empresariales de una empresa y trabajan integradas en tiempo real. Los Business Objects de SAP (BO) representan las unidades centrales, estos se han formado para todas las aplicaciones según los requerimientos empresariales. SAP gestiona y actualiza los Business Objects en el Business Object Repository (BOR). Un BAPI es un método de un objeto en el BOR y una interfase bien definida para los datos de un sistema de aplicación empresarial [SAP 04].

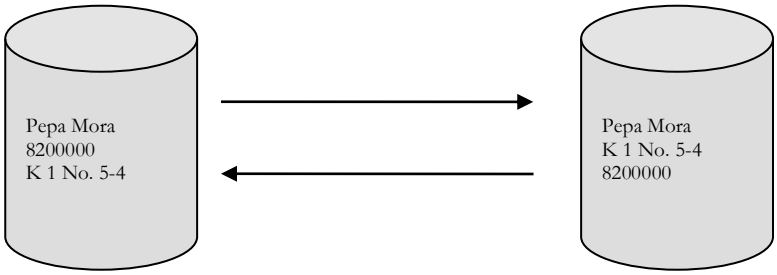

Figura 3.2. La replicación de datos es el simple intercambio de información entre bases de datos

Además las bases de datos involucradas pueden tener diferentes modelos de datos. El requerimiento fundamental de la replicación de las bases de datos es que a pesar de las diferencias entre los modelos y los esquemas de las bases de datos se debe proveer una infraestructura para intercambiar datos. Existen gran cantidad de soluciones a este requerimiento y a bajo costo. Desafortunadamente estas ventajas son rápidamente perdidas si los métodos necesitan estar relacionados con los datos, o si las funcionalidades también necesitan ser compartidas además de los datos. Si existe este requerimiento se deben considerar soluciones basadas en servicios.

La Integración de Aplicaciones Orientadas a la Información se puede hacer de muchas formas, una de ellas es por medio de una capa o Sistema Mediador, este permite la integración de aplicaciones ya que cuenta con módulos que permiten estructurar adecuadamente las consultas de los usuarios y responder independientemente del modelo de datos y/o de la tecnología de implementación de la fuente, además tiene que encargarse de problemas de heterogeneidad como: sintaxis, la cual involucra todo lo referente a los formatos de los datos, semántica, haciendo referencia al significado de los términos del contexto y la estructura, entendida como las diferencias en los modelos de datos.

Los componentes de esta capa o sistema medidador son:

- Componente mediador: Se encarga de recibir las solicitudes de los usuarios del sistema, estructurar las consultas y redireccionarlas a la fuente que contenga la información de respuesta.

- Componente envoltorio o wrapper que se ubica en cada una de las fuentes de información y permite realizar la publicación de la información y el mapeo de la estructura fuente a la del modelo de datos del sistema central.

Cabe anotar que bajo este esquema, no sería necesaria la replicación de los datos a un sistema central, ya que sería el mediador el encargado de realizar las consultas de manera transparente de tal forma que para los usuarios sea como una consulta a un único esquema global.

Federación de Datos

La federación de bases de datos es la integración de múltiples bases de datos y modelos de bases de datos en una vista de las bases de datos única y unificada (ver Figura 3.3). En otras palabras una base de datos federada es una base de datos virtual compuesta de muchas bases de datos reales físicas.

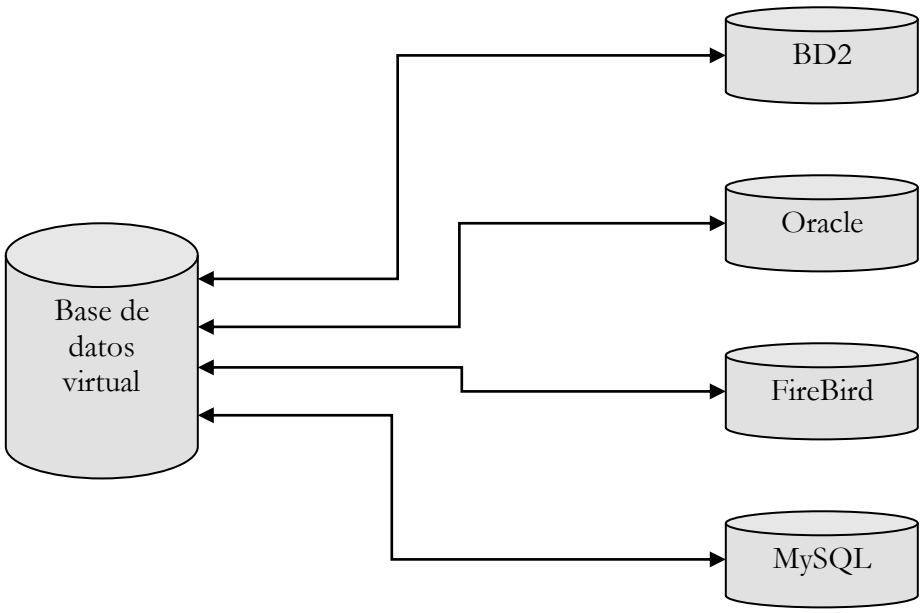

Figura 3.3. La federación de datos permite a muchas bases de datos mostrarse como una sola

El software de federación de datos coloca una capa de software (mediación) entre las bases de datos físicas distribuidas y las aplicaciones que ven los datos. Esta capa se conecta con las bases de datos físicas usando interfaces y mapas disponibles de las bases de datos físicas a un modelo

virtual de bases de datos que solo existe en el software, la aplicación usa esta base de datos virtual para acceder a la información requerida. La federación de datos maneja la colección y distribución de los datos en la forma que se necesite de las bases de datos físicas.

La ventaja de usar este tipo de capa es que puede unir muchos tipos de datos en un modelo unificado que soporte intercambio de información.

La federación de datos permite el acceso a cualquier base de datos conectada en una empresa a través una interfaz bien definida. Esta es la solución más elegante al problema de integración de aplicaciones orientadas a los datos. A diferencia de la replicación, esta solución no requiere cambios en la fuente o en las aplicaciones objetivo. De todas maneras, se deben hacer cambios al nivel orientado a la aplicación para soportar software de bases de datos federadas, debido al hecho de que diferentes interfaces están siendo usadas para acceder a diferentes modelos de bases de datos.

Interfaz de Procesamiento

Las soluciones de interfaz de procesamiento usan interfaces bien definidas para enfocarse en la integración de aplicaciones cerradas y heredadas

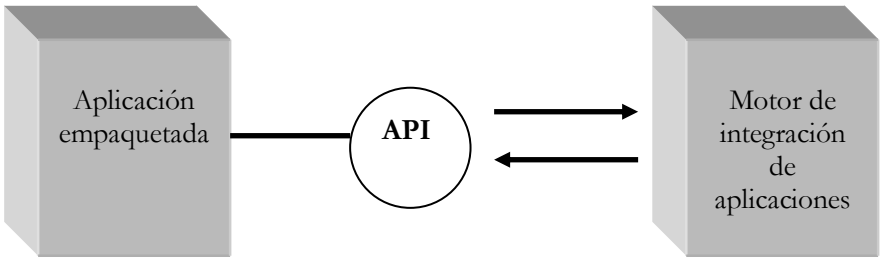

Figura 3.4. El procesamiento de interfaces exterioriza la información fuera de las aplicaciones empaquetadas a través de un API bien definida

Los agentes de integración soportan interfaces de aplicaciones de soluciones de procesamiento proporcionando adaptadores que se conectan a muchas aplicaciones tradicionales o empaquetadas como sea posible, para exteriorizar la información. Estas interfases se pueden

conectar a soluciones tecnológicas que incluyen mediadores y screen scrapers[8] como puntos de integración.

Soluciones Orientadas a los procesos de negocios

La integración de procesos de negocios (BPI Business Process Integration de sus siglas en inglés) es la ciencia y el mecanismo de la manipulación y movimiento de los datos, y de la invocación de procesos en el orden correcto y apropiado para soportar el manejo y ejecución de procesos comunes que existen en y entre las aplicaciones (ver Figura 3.5). La Integración de Aplicaciones Orientada a la Integración de Procesos de Negocios (BPIOAI Business Process Integration Oriented Application Integration de sus siglas en inglés) proporciona una capa de procesamiento manejado centralmente fácilmente definido que se encuentra en la parte superior del sistema de procesos y datos contenidos, existentes dentro de un conjunto de aplicaciones.

El objetivo es condensar procesos relevantes encontrados en una empresa o en una comunidad de negocios para obtener la máxima cantidad de valor mientras se soporta el flujo de información y el control lógico entre estos procesos. Estos productos ven el mediador como una facilidad y proveen interfaces visuales fáciles de utilizar para enlazar los procesos.

La integración de procesos de negocios es una estrategia tanto como una tecnología que consolida las habilidades de una organización para interactuar con aplicaciones disparejas integrando todo el proceso de negocio en y entre las empresas. Ciertamente, la integración de procesos de negocios involucra muchas organizaciones que utilizan diversos metadatos, plataformas y procesos; por esto la tecnología de integración de procesos de negocios debe ser flexible suministrando una capa de interpretación entre la fuente y el sistema objetivo, y el procesador de integración de procesos de negocios.

[8] Un Screen Scraper es una herramienta que se puede utilizar para integrar interfaces de usuario con sistemas heredados existentes. Son muy útiles cuando la interfaz cliente está muy acoplada con el resto de capas. Este traduce una interfaz de usuario existente en un conjunto de objetos, que normalmente funcionan como emuladores de terminal en un extremo y como una interfaz de objetos en el otro. El Sreen Scraper lee datos de campos del terminal de la interfaz heredada y los hace disponible en forma de objetos.

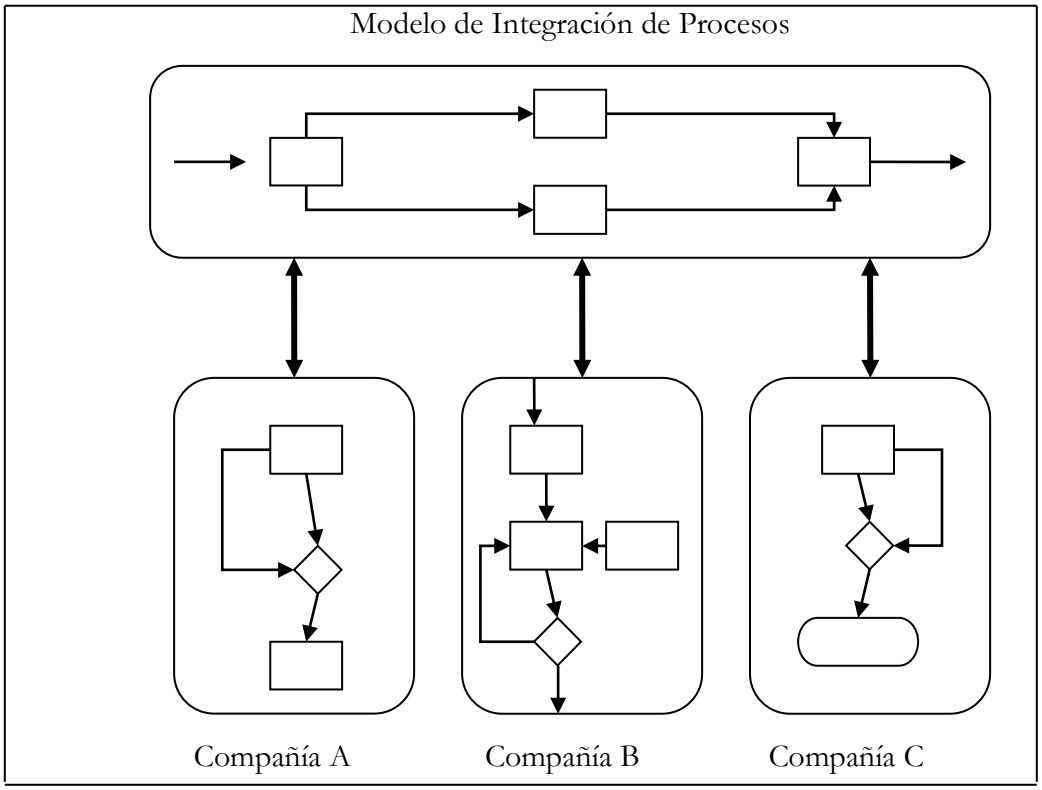

Figura 3.5. La integración de procesos de negocios permite a los arquitectos de integración de aplicaciones dar lugar a procesos de negocio bien definidos como una entidad de control, capaz de acceder a información y a procesos encapsulados en sistemas remotos

Hay muchas diferencias entre la integración de aplicaciones más tradicionales e integración de procesos de negocios.

- Una sola instancia de la integración del proceso de negocios típicamente se extiende a lo largo de muchas instancias de aplicaciones tradicionales.
- Integración de Aplicaciones significa el intercambio de información entre dos o más sistemas sin dar visibilidad a procesos internos.
- El proceso de integración de procesos encabeza un modelo de procesos y mueve información entre aplicaciones para soportar tal modelo.

- La integración de aplicaciones es una solución táctica motivada por el requerimiento de comunicación entre dos o más aplicaciones.
- La integración de procesos de negocios es estratégica, influenciando reglas de negocios para determinar cómo los sistemas deben interactuar y mejorar el movimiento del valor de los negocios para cada sistema, a través de un modelo común de negocios.

El objetivo primordial de BPIOAI y en general de la integración de aplicaciones, es automatizar el movimiento de los datos y el flujo de procesos, para que otra capa de BPIOAI exista sobre los procesos encapsulados en los sistemas existentes. En otras palabras, BPIOAI completa la integración de aplicaciones, permitiendo la integración de sistemas no solo al compartir información sino también manejando el como compartir la información con herramientas fáciles de manejar.

En general, la lógica de integración de procesos de negocios se encamina solo al flujo de procesos y a la información, no es una lógica de programación tradicional, como procesamiento de interfaces de usuario, actualización de bases de datos, o la ejecución de una transacción. De hecho, en la mayoría de escenarios BPIOAI la lógica del proceso está separada de la lógica de la aplicación, funciona sola para coordinar o manejar el flujo de información entre muchas fuentes y aplicaciones objetivo que existen dentro de las organizaciones.

Soluciones Orientadas a los Servicios

La integración de Aplicaciones Orientadas a los Servicios (SOAI) permite a las aplicaciones compartir lógica o métodos de negocios comunes, logrado al definir que métodos son comunes consiguiendo ser compartidos y luego integrados al proveer una infraestructura como Servicios Web. Los métodos pueden ser compartidos almacenándolos en un servidor central y accedidos por una aplicación interna (por ejemplo, objetos distribuidos) o a través de mecanismos estándar de Servicios Web.

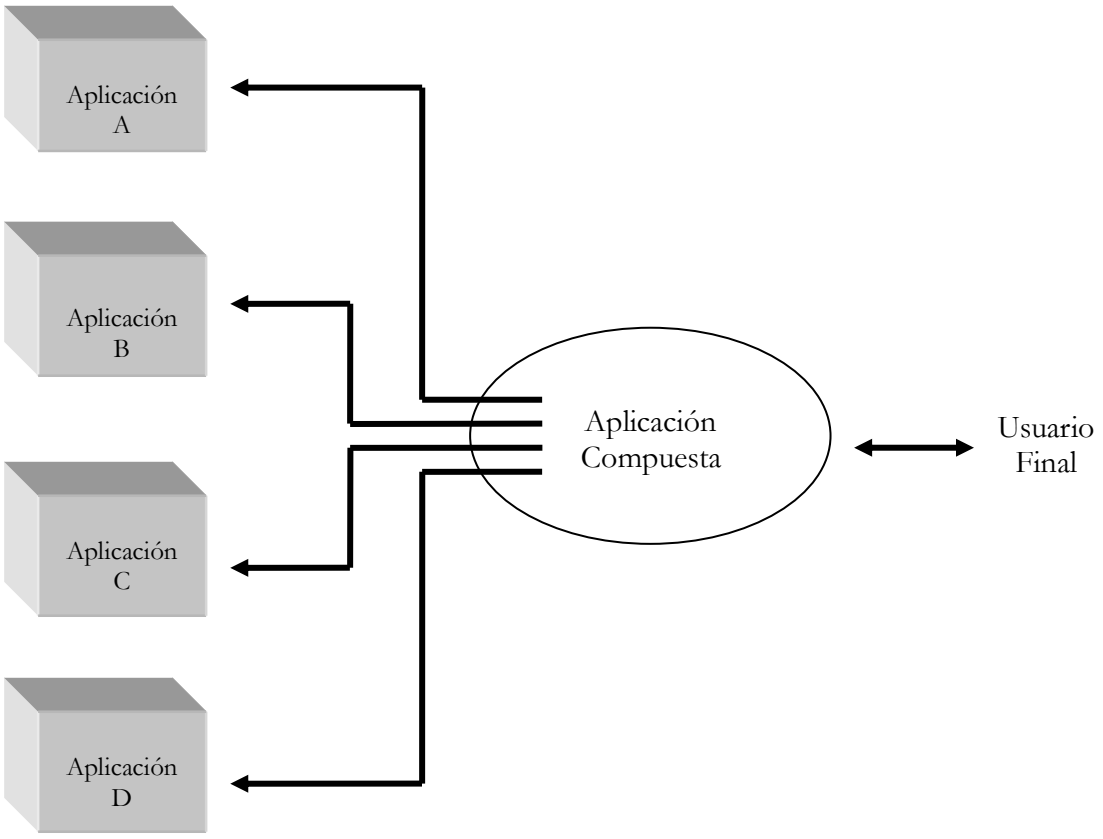

Figura 3.6. La Integración de Aplicaciones Orientadas al Servicio provee mecanismos para crear aplicaciones compuestas, utilizando servicios encontrados en muchos sistemas remotos

Los intentos para compartir procesos comunes tienen una larga historia, empezando hace más de diez años con el sistema multicapa cliente/servidor (un conjunto de servicios compartidos en un servidor común que suministró a la empresa con infraestructura de reuso y ahora de integración) y el movimiento de objetos distribuidos. Un conjunto de métodos comunes entre aplicaciones empresariales son el motivo más propicio para aplicar reusabilidad dando como resultado una reducción significativa en la necesidad de utilizar métodos y/o aplicaciones que terminan siendo redundantes.

Mientras que la mayoría de los métodos existen para la utilización de una sola organización, se está llegando a entender que hay ocasiones en las cuales es necesario compartir entre organizaciones.

Desafortunadamente un absoluto reuso no ha sido todavía llevado a cabo al nivel empresarial. Las razones de esta falla son principalmente políticas, varían de políticas internas a la inhabilidad de seleccionar tecnologías consistentes.

Las herramientas y técnicas de integración de aplicaciones crean la infraestructura necesaria para compartir métodos comunes.

Pero esta gran integración de aplicaciones es confrontada con el hecho de que es un proceso con un alto nivel invasivo y por lo tanto muy costoso.

Mientras que IOAI generalmente no requiere cambios ni en la fuente ni en las aplicaciones objetivo, SOAI requiere que la mayoría, sino todas, las aplicaciones empresariales sean cambiadas para poder aplicar el paradigma.

Antes de incluir un proceso tan invasivo y tan costoso como SOAI, la empresa debe claramente entender tanto las oportunidades como los riesgos, solo así el valor de SOAI puede ser evaluado objetivamente.

Soluciones Orientadas al Portal

La Integración de Aplicaciones Orientadas al Portal (POAI Portal Oriented Applicaction Integration de sus siglas en inglés) permite ver una multitud de sistemas –tanto sistemas empresariales internos, como sistemas de comunidades de negocios- a través de una sola interfaz o aplicación de usuario. El beneficio POAI es evitar el problema de integración, adaptando una interfaz de usuario de cada sistema a una interfaz de usuario común, casi siempre un Web browser (ver Figura 3.7). El resultado es la integración de todos los sistemas participantes a través del browser, aunque las aplicaciones no son directamente integradas en o entre las empresas.

Mientras que otro tipo de integración de aplicaciones se enfoca en el intercambio de información en tiempo real entre sistemas y compañías, POAI se preocupa por exteriorizar la información de muchos sistemas en una sola aplicación e interfaz. Este es claramente un acercamiento que va en contra de nociones de otros tipos de integración de aplicaciones, la cuales son en tiempo real y orientadas al manejo de eventos.

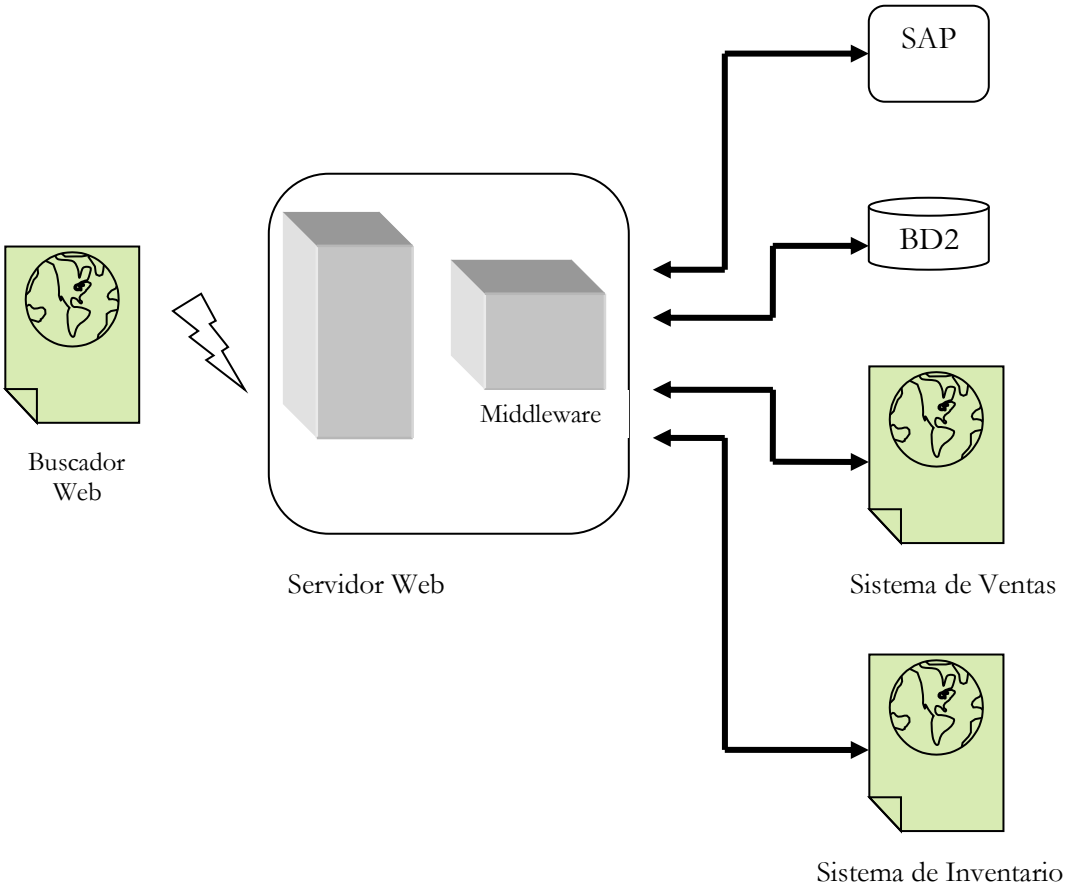

Figura 3.7. Integración de Aplicaciones Orientadas al Portal

3.2 INTEGRACIÓN DE APLICACIONES ORIENTADAS A LA INFORMACIÓN

La mayoría de proyectos de integración de aplicaciones se basan en Integración de Aplicaciones Orientadas a la Información (IOAI). Claro está que IOAI es el piso de la integración de aplicaciones, proporcionando un simple mecanismo para el intercambio de información entre dos o más sistemas. Esto no significa que no se pueda mezclar IOAI con otras soluciones de integración de aplicaciones, incluyendo integración tipo servicio, portal y de procesos de negocios.

IOAI permite que la información sea movida entre los sistemas fuente y objetivo. Los datos pueden venir de una base de datos, una API, o tal vez de una fuente empotrada. Lo importante de entender es que se está lidiando solo con simple información y no con servicios de procesos o de aplicación. La solución orientada a la información como integración de aplicaciones manifiesta que la integración ocurre entre sistemas al intercambiar simple información. Se ha estado haciendo integración de aplicaciones por este camino desde más de 30 años.

IOAI ofrece ciertas ventajas:

- Primero, solo se está tratando con simple información, entonces no se tiene que cambiar los sistemas fuente y objetivo. La mayoría de aplicaciones y ciertamente bases de datos ya saben como producir y consumir esta información.
- Segundo, no se necesitan manejar complejos problemas como estado, lógica y secuencia, porque no hay noción de comportamiento.
- Finalmente, este acercamiento es fácil de entender y además es muy usado.

En muchos casos, la integración orientada a la información es la solución correcta. Para muchos dominios, utilizar procesos de integración de servicios o de negocios no es la solución adecuada.

Acceder a información que está en las bases de datos y aplicaciones, es relativamente fácil, llevado a cabo con pocos –si hay alguno- cambios significativos a la lógica de la aplicación o a la estructura de la base de datos. Esta es una gran ventaja ya que la modificación de aplicaciones es un proceso complicado y a veces imposible como en el caso de las cadenas de abastecimiento donde se está lidiando con sistemas que están más allá del control directo.

No obstante, la apariencia de IOAI de fácil entendimiento no debe crear la impresión de que es simple. Migrar datos de un sistema a otro suena sencillo pero para que IOAI funcione los arquitectos y desarrolladores necesitan entender todo el sistema integrado en detalle.

La semántica de la aplicación hace este problema aún más complicado. Típicamente la semántica de la aplicación en un sistema no es compatible con la semántica de otros sistemas, esta puede ser tan diferente que los dos sistemas no podrían entenderse. Entonces IOAI no es solo mover información entre almacenes de datos, es también manejar las diferencias en esquema y en contenido.

3.2.1 Acoplamiento versus Cohesión

Al mirar las aplicaciones y las bases de datos que hacen la integración de aplicaciones del dominio de un problema específico, esta es una alternativa que generalmente tiene dos opciones, acoplamiento o cohesión.

Acoplamiento en el contexto de integración de aplicaciones, es la unión de aplicaciones de tal forma que dependa una de la otra, compartiendo los mismos métodos, interfases y tal vez también datos.

A primera impresión el acoplamiento parece la mejor idea, sin embargo no se debe perder de vista lo que esto realmente significa, la unión hermética de un dominio de la aplicación a otro. Una consecuencia de este requerimiento, es que todas las aplicaciones y bases de datos que quieren ser acopladas tendrán que ser extensivamente cambiadas para soportar este acoplamiento.

Como el acoplamiento requiere cambios en el sistema fuente y objetivo, podría no encajar en la mayoría de problemas de dominio de integración de aplicaciones, donde los sistemas participantes no están bajo un control central.

En contraste del acoplamiento, las aplicaciones y las bases de datos integradas cohesivamente son independientes una de la otra. Los cambios a cualquier sistema fuente u objetivo no afectará al otro directamente.

Es necesario considerar sus ventajas. La cohesión provee una gran flexibilidad para cuando la aplicación evolucione, en una solución de integración de aplicaciones cohesivas pueden

adicionarse, cambiarse o removerse sistemas sin requerir cambios a ninguno de los otros sistemas en el dominio del problema.

A pesar de la flexibilidad de la cohesión, si los procesos de negocios comunes necesitan ser reusados entonces el acercamiento por acoplamiento provee mayor valor.

3.2.2 Patrones Comunes de los Productores y Consumidores de Información

En el mundo de IOAI los sistemas fuente y objetivo son siempre entidades que producen y consumen información. Estos tipos se pueden clasificar en estos patrones:

- Bases de Datos
- Aplicación
- Interfaz de usuario
- Fuentes empotradas

Dentro de IOAI es importante entender cada patrón, cómo se comporta cada uno y las ventajas y limitaciones de cada tipo.

- **Bases de Datos**

El más popular consumidor y productor de información es la base de datos. Las bases de datos son puntos naturales de integración porque están diseñados para producir y consumir datos, proporcionando la mejor interfase en las aplicaciones fuente y objetivo para intercambiar información.

Cuando se requiere información de la base de datos fuente se envía una petición usando un lenguaje que la base de datos puede entender como SQL, esta responde con un conjunto de resultados con la información requerida, siendo esta uno o muchos registros. La base de datos fuente acepta la petición y produce los datos o puede trabajar a través de un escenario de procesamiento de excepciones.

Cuando se envía información una base de datos objetivo, se hace una petición de actualización de nuevo utilizando un lenguaje que la base de datos pueda entender y luego se envía la información en el formato apropiado, la base de datos objetivo puede aceptar o rechazar la actualización (ver Figura 3.8).

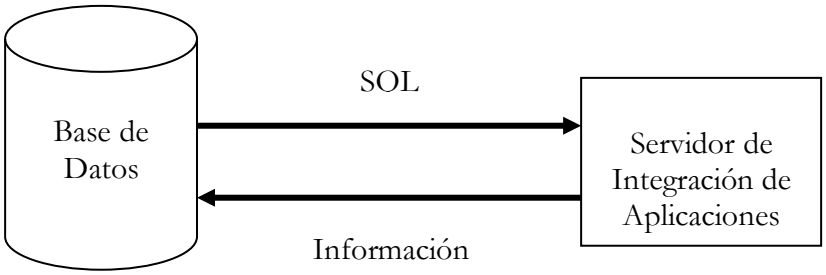

Figura 3.8 Las bases de datos producen y consumen información a través de interfases que ellos proveen

La ventaja de usar bases de datos como punto de integración es que las interfases son casi siempre bien definidas y probadas, y hay diferentes tipos de conjuntos de resultados que se pueden pedir. La desventaja es que la información producida no está ligada a las entidades de negocios.

- **Aplicación**

Las interfases de aplicación son mucho más complejas que las bases de datos por el simple hecho de que todas las aplicaciones toman diferentes acercamientos dependiendo de cómo se va a producir y consumir la información. Por consiguiente, las interfases que existen de aplicación a aplicación no comparten patrones o estándares comunes, por consiguiente se necesita direccionar cada paquete de aplicación individualmente, tal vez por medio de programación o a través de adaptadores provistos por los vendedores de integración de aplicaciones.

Las interfases de aplicación son interfases que los desarrolladores exponen de aplicaciones empaquetadas o a la medida para acceder a varios niveles o servicios.

- **Interfaz de Usuario**

La interfaz de usuario como un punto de integración es un proceso conocido como "screen scraping" o acceder a la información por medio de una interfaz a través de un mecanismo programático. El mediador conduce la interfaz de usuario con el objetivo de acceder a la información. Muchos proyectos de integración de aplicaciones no tendrán otra salida que utilizar las interfases de usuario para acceder a los procesos y los datos de la aplicación.

No hay mucho que decir acerca de interfases de usuario, solo es una de muchas técnicas y tecnologías que pueden ser usadas para acceder o colocar información en una aplicación. La tecnología ha sido utilizada por algunos años, y hay, sin embargo, problemas que necesitan ser superados. Una interfaz de usuario nunca fue diseñada para estar al servicio de datos. Además este tipo de solución es incapaz de escalar, entonces no está capacitada para manejar más de unas pocas interfases en un momento dado. Finalmente, si el arquitecto y el programador no planean estos mecanismos cuidadosamente, la aplicación puede volverse inestable.

Como en otros niveles de integración de aplicaciones, el arquitecto o el desarrollador tienen la responsabilidad de entender la aplicación, la arquitectura y la información de la base de datos en gran detalle. Al nivel de interfaz de usuario esto puede ser complicado. La decisión de construir integración de aplicaciones al nivel de interfases de usuario fue hecha específicamente para evitar las restricciones de sistemas propietarios o porque otros niveles de integración de aplicaciones no eran ajustables.

- **Fuente Empotrada**

Finalmente se puede encontrar que en algunos casos la información viene de fuentes empotradas, como sensores de temperatura, máquinas de conteo de llamadas, o tal vez dispositivos inalámbricos. Tratar con dispositivos embebidos es similar a tratar con aplicaciones, las interfases son típicamente de API's propietarias.

Un patrón común cuando se trata con dispositivos embebidos como sistema fuente u objetivo, es que la información debe fluir libremente desde la fuente porque la mayoría de dispositivos no pueden almacenar información (como lo hacen las aplicaciones). Así cuando se obtiene información, típicamente se obtiene en el momento, no es información que está en cola.

3.2.3 Aproximación a la Integración de Información [DL 03]

Ahora la pregunta es, ¿cómo acercarse al problema del dominio? Aunque hay diferentes caminos para dirigir la integración de información dentro de un conjunto de soluciones de integración de aplicaciones, la mejor manera es siguiendo estos pasos:

- Identificar los datos
- Catalogar los datos
- Construcción del modelo lógico

- Construir el modelo de metadatos de la empresa (el cual será usado como una guía maestra para la integración de varios almacenes de información que existen dentro de la empresa)

En pocas palabras, implementar una solución de integración de aplicaciones demanda más que el movimiento de los datos entre bases de datos y/o aplicaciones.

- **Identificar los Datos**

Desafortunadamente no hay un atajo para identificar datos dentro de una empresa. La información acerca de los datos tanto técnica como de negocios está dispersa a lo largo de toda la empresa.

El primer paso para localizar e identificar la información acerca de los datos es crear una lista de sistemas que contienen información. El siguiente paso es determinar a quién le pertenece la base de datos, donde está localizada físicamente, información de diseño relevante, e información como marca modelo, y revisiones de la tecnología de la base de datos.

Puntos de Integridad

Cuando se analizan bases de datos constantemente aparecen puntos de integridad. Para entender esto, es importante entender las reglas y regulaciones que se han aplicado en la construcción de la base de datos.

La falta de integridad al nivel de los datos puede ser problemático. Los arquitectos y desarrolladores de integración de aplicaciones necesitan tener en cuenta este peligro, asegurando no comprometer la integridad de la base de datos en su afán por mejorar la integración.

Latencia de Datos

La latencia de datos es la característica de los datos que define que tan actual necesita ser la información, es otra propiedad de los datos que necesita sea determinada para propósitos de integración de aplicaciones. Tal información permitirá a los arquitectos de integración de aplicaciones determinar cuando la información debe ser copiada o movida a otro sistema empresarial y que tan rápido.

Para el propósito de la integración de datos dentro de una empresa hay tres categorías de latencia de datos:

i. En tiempo real
ii. En tiempo cercano
iii. Una sola vez

a. En tiempo real: Intercambio de datos en tiempo real, significa es precisamente lo que suena, información que es colocada en la base de datos en el momento en que ocurre, con poca o ninguna latencia. Los datos en tiempo real son actualizados en el momento en que entran en la base de datos y esa información está disponible inmediatamente para cualquier persona, o cualquier aplicación que la requiere para procesamiento.

   Mientras que la latencia cero en tiempo real es claramente el objetivo de la integración de aplicaciones este logro representa grandes desafíos, para esto se necesita que la implementación de la integración de aplicaciones requiera un constante retorno a la base de datos, aplicación u otra fuente para recuperar nueva y/o información actualizada.

b. En tiempo cercano: Se refiere a información que es actualizada en un conjunto de intervalos más que instantáneamente. Datos en tiempo cercano pueden ser considerados como una latencia de datos suficientemente buena. En otras palabras, los datos son tan oportunos como se necesite.

   A pesar de que los datos en tiempo cercano no son actualizados constantemente afronta muchos de los desafíos de los datos en tiempo real, incluyendo mejora de desempeño y asuntos de manejo.

c. Una sola vez: los datos son típicamente actualizados solo una vez. Dentro del contexto de integración de aplicaciones los intervalos de copia o movimiento de datos no requieren la clase de dinámica necesitada para llevar a cabo intercambio de datos en tiempo real o en tiempo cercano.

Estructura de los datos

Otro componente que identifica los datos es la estructura. Como la información es estructurada, incluyendo las propiedades de los elementos existentes dentro de la estructura, estas pueden ser deducidas del conocimiento del formato de los datos. De la misma forma, tamaño, tipo de datos (carácter o numérico), nombre del elemento del dato y tipo de

información guardada (binario, texto, espacial, etc.) son características adicionales que pueden ser determinadas por el formato.

- **Catalogar los Datos**

Una vez que las características lógicas y físicas de la base de datos a integrar sean entendidas es tiempo de catalogar los datos. En el mundo de la integración de las aplicaciones, catalogar datos es el proceso de compartir metadatos y otros datos por todo el dominio del problema. Una vez logrado esto es posible crear un catálogo de la empresa con todos los elementos de los datos que pueden existir dentro de esta. El catálogo resultante se convierte en la base del entendimiento de la necesidad de crear el modelo de metadatos de la empresa, el fundamento de IOAI.

Mientras que no existe un estándar para catalogar datos dentro de un proyecto de integración de aplicaciones, el principio de guía es claro: entre más información mejor. El catálogo se convertirá en un repositorio de la máquina de integración de aplicaciones a ser construida y el fundamento para descubrir nuevos flujos de negocios. También se convertirá en un camino para automatizar flujos de negocios existentes dentro de la empresa.

- **Construcción del Modelo Lógico**

Como en los métodos de diseño de las bases de datos tradicionales, el modelo de metadatos de la empresa usado por IOAI puede ser dividido en dos componentes: el lógico y el físico. Crear el modelo lógico es el proceso de crear una arquitectura para todos los almacenes de datos que son independientes del modelo de bases de datos físico.

Un modelo lógico es un acercamiento a un proyecto de integración de aplicaciones, el cual permitirá a arquitectos y desarrolladores tomar decisiones IOAI objetivas al moverse de requerimientos de alto nivel a detalles de implementación. El modelo de datos lógico es una vista integrada de los datos del negocio a través del dominio de la aplicación, o datos pertinentes a la solución de integración de aplicaciones bajo construcción. La principal diferencia al usar un modelo de datos lógico para integración de aplicaciones versus el desarrollo de una base de datos tradicional, es la fuente de la información. Mientras que el desarrollo tradicional define nuevas bases de datos basadas en requerimientos de negocios, un

modelo de datos lógico que surge de un proyecto de integración de aplicaciones es basado en basas de datos existentes.

Al corazón del modelo lógico está el Diagrama Entidad-Relación (ERD Entity Relationship Diagram de sus siglas en inglés). Un ERD es una representación gráfica de entidades de datos, atributos y relaciones entre entidades para todas las bases de datos existentes en una empresa.

- **Construcción del Modelo de Metadatos**

Cuando toda la información de todos los datos en la empresa está contenida en el catálogo de los datos, es tiempo de enfocarse en el modelo de metadatos de la empresa. La diferencia entre estos dos es muy sutil, es mejor pensar al catálogo de los datos como una lista de soluciones potenciales a un problema de integración de aplicaciones y el modelo de metadatos como la solución IOAI. El modelo de metadatos no solo define la estructura de todos los datos existentes en la empresa sino como esa estructura de datos interactuará dentro del dominio de la solución de integración de aplicaciones. Una vez construido el modelo de metadatos de la empresa se tiene el director maestro de la solución de integración de aplicaciones. En muchos casos este directorio será enganchado al agente de integración y será usado como punto de referencia no solo para localizar los datos sino también las reglas y la lógica que se aplican a los datos. Sin embargo el directorio no es más que un simple almacén de metadatos, es el corazón de la solución de integración de aplicaciones que contiene los datos y la información del modelo del negocio.

3.3 INTEGRACIÓN DE APLICACIONES ORIENTADAS A LOS SERVICIOS

Aunque IOAI y BPIOAI son soluciones funcionales adecuadas para muchos problemas de integración de aplicaciones, la integración de estas es lo que generalmente provee más valor a largo plazo, aunque con un gran costo.

Las organizaciones han estado buscando mecanismos por años para unir aplicaciones al nivel de los servicios. Algunos mecanismos exitosos incluyen frameworks, transacciones, y objetos distribuidos los cuales están en uso. No obstante, la noción de Servicios Web está ganado terreno. El objetivo es identificar un nuevo mecanismo que pueda aprovechar el poder de Internet para proveer acceso a servicios de aplicaciones remotas a través de interfaces bien

definidas y servicios de directorio. El uso apropiado de los Servicios Web en el contexto de integración de aplicaciones es el futuro.

3.3.1 Conceptos Básicos [DL 03]

La Integración de Aplicaciones Orientadas a los Servicios (SOAI Service Oriented Application Integration de sus siglas en inglés) permite a las empresas compartir servicios de aplicaciones comunes e información. Las empresas llevan a cabo esto definiendo servicios de aplicaciones que pueden compartir, y luego integrar o proporcionando una infraestructura para compartir tales servicios de la aplicación.

¿En qué se diferencia un servicio de la aplicación de la integración de información?
Cuando se usa un servicio de la aplicación se hace disponible un método remoto o un comportamiento simplemente extrayendo o publicando información en un sistema remoto. Este servicio remoto típicamente se abstrae en otra aplicación llamada aplicación compuesta (ver Figura 3.9).
Un buen ejemplo de un servicio de aplicación podría ser un proceso de análisis de riesgos el cual corre dentro de una empresa para calcular el riesgo de una transacción financiera, este servicio de aplicación es de poco uso por si mismo, pero cuando es abstraído en una aplicación más grande, por ejemplo una comunidad de negocios, este servicio remoto de la aplicación tiene un valor adicional.
La noción básica de SOAI es utilizar estos servicios remotos utilizando una infraestructura controlada que permite a las aplicaciones invocar servicios remotos como si fueran locales. El resultado (o el objetivo) es una aplicación compuesta hecha de muchos servicios de aplicación locales y remotos.

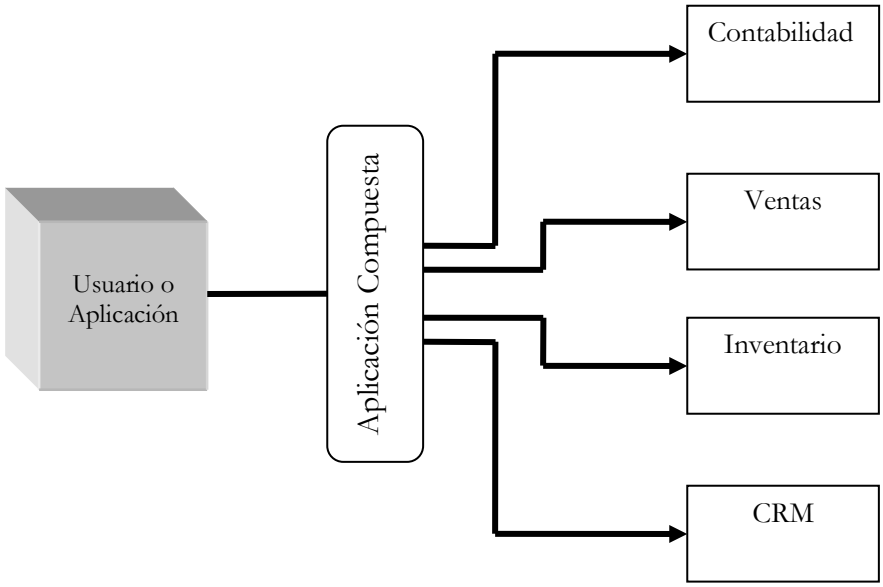

Figura 3.9 Una aplicación compuesta es la integración de muchas aplicaciones remotas en una sola aplicación

La integración de aplicaciones da las herramientas y las técnicas para aprender cómo compartir servicios de aplicación, y más que eso, cómo estas herramientas y técnicas crean la infraestructura necesaria para que los servicios puedan ser compartidos; al tomar ventaja de esta oportunidad se pueden integrar aplicaciones para compartir información y además se crea la infraestructura para el reuso de la lógica del negocio.

A pesar de que IOAI no requiere cambios en la fuente o en la aplicación objetivo, SOAI si requiere cambios en casi toda (si no en toda), la empresa para poder tomar ventaja de este paradigma. Esta desventaja hace la solución orientado al servicio algo complicado en las empresas. Los Servicios Web prometen cambiar eso, colocando todo en los mismos estándares tecnológicos, pero hay algunos cambios inevitables que se deben hacer en la fuente y en el sistema objetivo para soportar Servicios Web. En otras palabras, la mayoría de los sistemas que deciden utilizar SOAI son sistemas existentes construidos antes de la llegada de los Servicios Web (u otra tecnología SOAI hecha para lo mismo) deben ser cambiados o reconstruidos.

¿Cuándo usar SOAI? Aunque muchos negocios están buscando intercambiar información con empresas y además participar en procesos compartidos, pocas están buscando crear aplicaciones que compartan servicios de aplicación con sistemas que no están bajo su control. Sin embargo, en algunas instancias SOAI es una buena opción:

- Cuando dos o más compañías necesitan compartir la lógica de un programa común, como el cálculo de los costos de envío de un proveedor común, el cual cambia constantemente.
- Cuando dos o más compañías quieren compartir el costo de desarrollo y el valor de una aplicación común.
- Cuando el problema del dominio es pequeño y especializado y puede llegar a ser parte de una aplicación que todas las compañías comparten.

3.3.2 Introduciéndose en los Servicios Web

Un Servicio Web es una interfaz que contiene un conjunto de operaciones que son accesibles a través de la red por medio de mensajes en un formato estándar. La interfaz esconde los detalles de implementación, permitiendo que este sea usado independientemente de la plataforma hardware o software en la que este implementado y del lenguaje de programación utilizado. Esto permite que las aplicaciones basadas en servicios Web tengan un alto grado de interoperabilidad entre las distintas tecnologías utilizadas para las implementaciones. Un Servicio Web puede llevar a cabo desde simples funciones hasta complejos procesos trabajando en conjunto con otros servicios web de manera remota.

A pesar de los inconvenientes usuales de la tecnología emergente los Servicios Web –al menos la noción de Servicios Web- es una tecnología interesante para el mundo de la integración de aplicaciones en y entre compañías. Los Servicios Web mantienen la promesa de ir más allá del simple intercambio de información (el mecanismo dominante para la integración de aplicaciones hoy en día), al concepto de acceder a servicios de aplicación que se encapsulan dentro de viejas y nuevas aplicaciones. Esto significa que las organizaciones no solo pueden mover información de aplicación a aplicación, sino que también pueden crear aplicaciones compuestas aprovechando cualquier número de servicios de aplicación encontrados en cualquier número de aplicaciones locales o remotas. Esta es la idea detrás de SOAI.

3.3.3 Posibilidades de los Servicios Web

Los usos de los servicios de aplicación utilizados para la integración son infinitos, incluyendo la creación de aplicaciones compuestas o aplicaciones que agregan procesos o servicios e

información a muchas aplicaciones. Por ejemplo, usando este paradigma los desarrolladores de aplicaciones solamente necesitan crear la interfase, para luego adicionar los servicios de la aplicación uniendo la interfase a tantos servicios de aplicación conectados a Internet como sea requerido.

Claro está que los Servicios Web prometen entregar a la integración de aplicaciones un valor adicional incluyendo servicios de aplicación estándar para publicar y suscribirse a servicios software locales y remotos. Las aplicaciones, localizan estos servicios utilizando UDDI (Servicio Universal de Descripción, Descubrimiento e Integración) y determinan la definición de la interfase el Lenguaje de Descripción de Servicios Web (WSDL).

WSDL. El lenguaje de descripción de servicios Web, WSDL es un lenguaje XML que contiene información acerca de la interfaz, semántica, y administración de una llamada a un Servicio Web.

Una vez que se ha desarrollado un servicio Web, se publica su descripción y se construye un enlace o apuntador en un depósito UDDI (Universal Description, Discovery and Integration) para que los usuarios potenciales lo puedan utilizar. Cuando alguien piensa en utilizar este Servicio Web, solicitan el archivo WSDL para conocer la ubicación del servicio, llamado de funciones, y cómo acceder al Servicio Web. Luego utilizan la información en el archivo WSDL para construir una petición SOAP (Simple Object Access Protocol) y enviarla hacia el proovedor del servicio.

3.3.4 ¿Dónde Realizar Ajustes?

Entonces si los Servicios Web están apareciendo y la integración de aplicaciones necesita estos mecanismos para adicionar aún más valor a la empresa o a la comunidad de negocios, ¿cómo encajan los Servicios Web en la integración de aplicaciones, de las aplicaciones "tradicionales"?

El punto importante es que muchos de los dominios del problema en y entre las empresas no necesitan acceso a aplicaciones al nivel de los servicios; el intercambio de información es suficientemente bueno, más no conveniente. Además la mayoría de las organizaciones no tienen estrategias de integración de aplicaciones, estas organizaciones necesitan primero colocar su casa en orden luego determinar sus requerimientos de integración, crear un plan, y luego seleccionar el acercamiento correcto a la integración de aplicaciones y la tecnología

necesaria. Al saltar simplemente a una tecnología como Servicios Web sin entender los requerimientos de los servicios puede ser desastroso, o peor, puede causar que la organización pierda valiosas oportunidades estratégicas.

Las organizaciones que requieren Servicios Web tienen la oportunidad de acceder tanto a la información como a los servicios de la aplicación que existen en sistemas de información locales y remotos.

Típicamente el dominio del problema que requiere una solución orientada a los servicios tiene las siguientes características:

- Hay servicios de aplicación redundantes que existen en dos o más sistemas.
- Hay la necesidad de crear una nueva aplicación que satisface una necesidad del negocio, pero que también es capaz de soportar servicios de aplicación agregados de sistemas remotos.
- La información que reside en la fuente o el sistema objetivo no tiene casi ningún valor cuando no está relacionada con los servicios.

Cuando se aplica la tecnología de los Servicios Web para resolver problemas de integración de aplicaciones hay patrones a considerar, estas son:

- Manejado por eventos
- Compuesto
- Distribución autónoma

- **Manejado por eventos**

La solución de Servicios Web manejado por eventos, se refiere a esas arquitecturas que se relacionan más con movimiento de información, que con servicios de agregación de la aplicación (ver Figura 3.10). Los datos se mueven de sistema a sistema soportando una transacción de negocios particular, pero también hay un requerimiento de acceder a servicios de aplicaciones. Por ejemplo mover una orden de información de sistema a sistema y de compañía a compañía para soportar la compra de un carro, o al emplear un Servicio Web para calcular la información de logística que es compartida por todos los sistemas fuente y objetivo. Esta es una arquitectura hibrida que mezcla Servicios Web y la tecnología de integración de aplicaciones tradicional, como la integración de servidores.

- **Aplicación compuesta**

Se refiere a aplicaciones que requieren agregar muchos servicios en una simple instancia de una aplicación (ver Figura 3.11). Las organizaciones han estado lidiando con este paradigma por años como la programación orientada a componentes, donde muchos componentes de aplicaciones predefinidas, se combinan para crear una sola aplicación. No obstante, dentro de la noción de Servicios Web, los componentes de la aplicación residen en un computador remoto y los Servicios Web son accedidos como piezas de una aplicación. Por ejemplo, una aplicación que monitorea envíos, invoca una serie de Servicios Web (que corren en computadores remotos) que proveen servicios de aplicación para procesos logísticos. Mirando a futuro esta será la arquitectura más popular de Servicios Web porque es la más cercana al concepto.

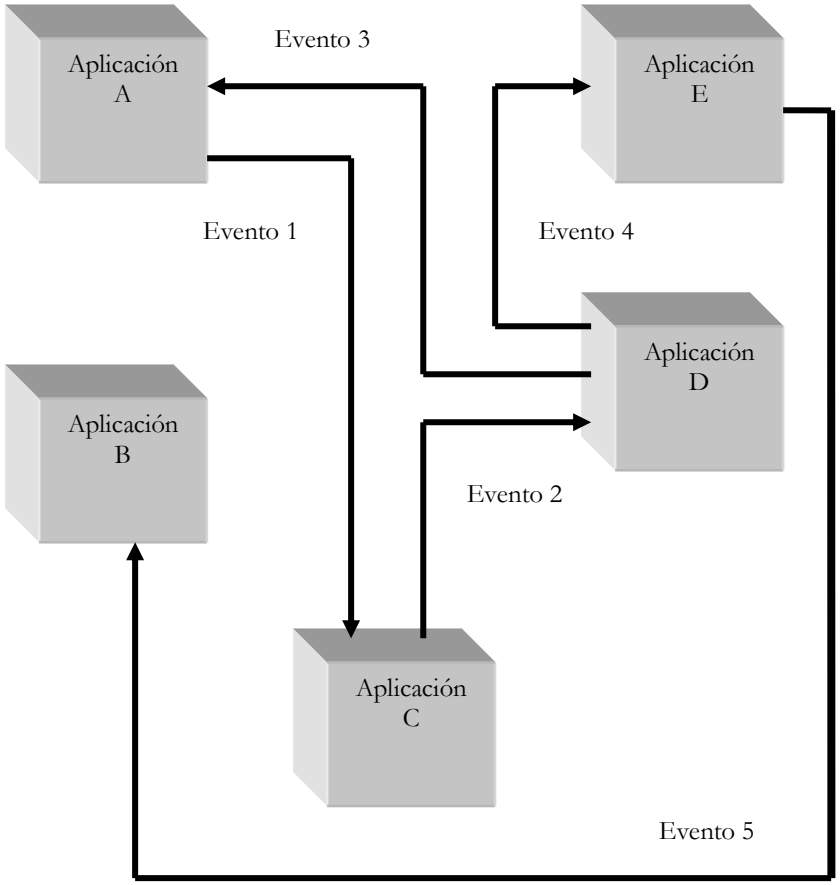

Figura 3.10 Solución manejada por eventos

- **Distribución autónoma**

Se refiere a aquellas arquitecturas donde los Servicios Web están tan relacionados que aparecen como una sola aplicación (ver Figura 3.12). Este es el destino final de los Servicios Web, relacionar muchas aplicaciones en y entre compañías en un todo simple y unificado. Sin embargo la proliferación de esta arquitectura está aún muy lejana.

Finalmente la tecnología disponible está cambiando para acomodarse a los Servicios Web, los productos de integración de aplicaciones tradicionales se están moviendo a los Servicios Web a través de la adición de muchas nuevas características. Esto incluye la creación de nuevos adaptadores que provee acceso a sistemas remotos a nivel de servicios (también a nivel de información), mecanismos de enlace de servicios de aplicación innatos al servidor, e interfases de agregación que soportan la combinación de muchos Servicios Web en uno solo.

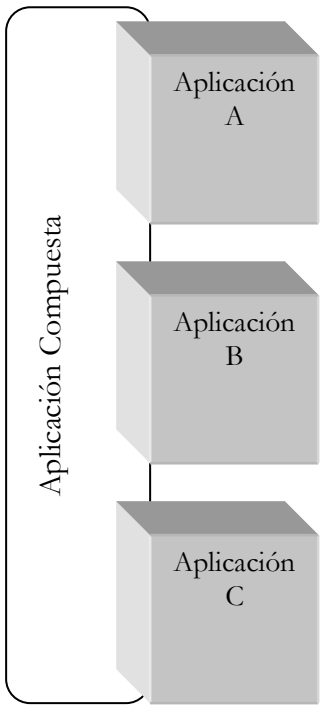

Figura 3.11 Solución composición de aplicaciones

UDDI. La especificación del Servicio Universal de Descripción, Descubrimiento e Integración define un mecanismo común para publicar y descubrir información sobre Servicios Web. UDDI es realmente un conjunto de bases de datos donde se pueden registrar y localizar Servicios Web. Por ejemplo una compañía puede registrar un programa único para predecir la fractura encontrada en un envió de productos hechos en vidrio, esto depende del servicio de la

aplicación del envío, punto de origen y destino. La compañía puede publicar esta información en la base de datos del UDDI permitiendo a otras organizaciones encontrar este Servicio Web, entender cómo acceder a este y las interfases empleadas [DW 04].

3.3.5 Escenarios [DL 03]

Antes de implementar SOAI, se debe entender todo el proceso, los servicios de la aplicación y los programas que existen dentro de la empresa. Confrontado con esta tarea monumental, se debe preguntar inicialmente, ¿Cuál es el mejor procedimiento?

El primer paso es desmembrar el proceso en escenarios o tipos. Para una integración de aplicaciones estos tipos son reglas, lógica, datos y objetos.

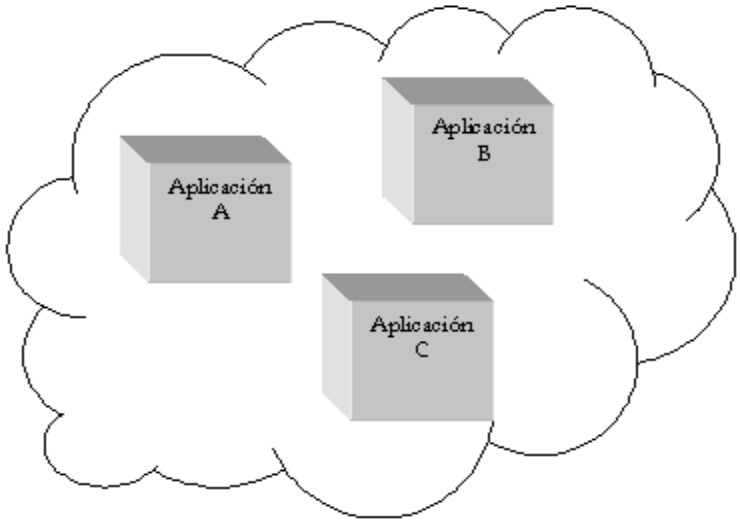

Figura 3.12 Solución distribuida autónoma

Reglas

Es un conjunto de condiciones. Por ejemplo, una regla puede establecer que los empleados no pueden volar en primera clase en vuelos de menos de 5.000 millas o que todas las transacciones de más de $100 millones deben ser reportadas al gobierno. Estas reglas que existen dentro de una empresa están dentro de las aplicaciones para controlar el flujo de información. Las reglas también pueden ser colocadas para controlar el flujo de información entre empresas.

Normalmente las reglas existen en una sola aplicación y son accesibles por un solo departamento. El reto de la integración de aplicaciones es proveer la infraestructura que permitirá el compartir estas reglas entre organizaciones, así las reglas son accesibles a muchas aplicaciones.

Las reglas necesitan ser bien entendidas ya que ellas afectan cada aspecto de los datos que se mueven dentro de las empresas incluyendo identificación, procesamiento y transformación de ellos. De hecho, las reglas que procesan al nivel de un sistema mediador a través de mensaje o de herramientas de automatización de procesos se convertirán en la primera generación de integración de aplicaciones.

Lógica

La lógica es una simple secuencia de instrucciones en un programa. Por ejemplo si un botón es presionado, entonces el menú se visualizará. La gran dificultad al lidiar con la lógica es la consecuencia de una realidad básica, si hay diez programadores y se les da la misma especificación cada uno de ellos desarrollará una versión diferente de la lógica del programa funcionando perfectamente. En muchas maneras la lógica es más un arte que una ciencia.

Hay tres clases de lógica: el proceso secuencial, selección e iteración.

- Proceso secuencial: es una serie de pasos en el procesamiento de datos existente. Entrada, salida, calcular y copiar, son ejemplos de instrucciones usadas en un procesamiento secuencial.
- Selección: es una decisión tomada dinámicamente dentro del programa. Es hecha al comparar dos conjuntos de datos y dependiendo del resultado se va hacia diferentes partes del programa.
- Iteración: es la repetición de una serie de pasos. Está soportada por bucles DO y FOR en lenguajes de programación de alto nivel.

Datos

En este contexto los datos no son más que la información que es compartida entre aplicaciones, computadores o humanos. Sistemas de reportes, contabilidad, recursos humanos, todos estos sistemas comparten datos. Porque los servicios de aplicaciones actúan sobre datos,

se necesita entender el camino por el cual la información es compartida al nivel del servicio de la aplicación para que SOAI sea exitosa.

Objetos

Los objetos son simplemente datos y servicios de negocios unidos como objetos. Ellos son un paquete de datos encapsulados dentro de un objeto y rodeados por servicios de aplicación que actúan sobre esos datos.

Los objetos en los sistemas generalmente usan tecnología orientada a objetos, como C++ o Java. A pesar del hecho de que la mayoría de objetos son orientados a servicios (estos son objetos que hacen algo cuando una interfase es invocada), la mezcla de objetos puede también incluir objetos distribuidos.

Identificar objetos dentro del dominio del problema es más complicado que identificar procesos de negocio. Requiere un entendimiento del modelo del objeto usado para construir las aplicaciones, si existe o si no, o si alguna vez ha sido creado. En la ausencia de un modelo de objetos la aplicación debe ser reingeniada del código base existente. (Afortunadamente un buen conjunto de herramientas y tecnologías existen para ayudar en este proceso de reingeniería).

3.3.6 Migración a Aplicaciones de Servicios

El camino hacia SOAI no es fácil, pero puede ser la mejor ruta a la solución o a la integración de todas las aplicaciones a través del reuso de servicios de aplicaciones comunes. La mayoría de los dominios del problema que requieren integración de aplicaciones optarán por los otros acercamientos de integración de aplicaciones antes de irse por SOAI. Los primeros tipos son más fáciles de desplegar y tienen menos riesgo, sin embargo, si no hay esfuerzo no hay gloria. Dependiendo de los requerimientos del dominio del problema, el acercamiento al nivel de los servicios, para todas sus dificultades, podría tornarse en ser la mejor solución para el problema de integración de aplicaciones.

CAPITULO 4. SISTEMA DE INFORMACIÓN TECNO-ECONÓMICO PARA EL DEPARTAMENTO DEL CAUCA

4.1 DESCRIPCIÓN DEL PROBLEMA DE SITEC

Descripción del problema

El adecuado desarrollo de una región depende en gran medida de la forma en que se gestionen, distribuyan y administren sus recursos, y del control efectivo que sobre éstos se realice. Esta gestión, que implica la toma de decisiones relevantes para las entidades, debe estar adecuadamente fundamentada.

Es así como la toma de decisiones en una organización requiere de una adecuada valoración de la información que permita y facilite el proceso de análisis de las situaciones. Para esto, no es suficiente con el almacenamiento y recuperación de datos. Un dato no dice nada sobre el porqué de las cosas, y por sí mismo tiene poca o ninguna relevancia o propósito. Por sí solos no pueden orientar la ejecución de acciones.

La información es resultado de la interpretación y la transformación de los datos, entendida como la contextualización, categorización, corrección, cálculo y condensación de los mismos, que integrada con experiencia, valores y saber hacer dan paso a la generación de conocimiento útil para la ejecución de acciones [DP 98].

La información económica requerida por el departamento del Cauca para poder generar planes y políticas de desarrollo está limitada a una gran cantidad de datos dispersos que, al ser generados por diferentes fuentes sin tener una base estructural común, son incompatibles y redundantes lo cual imposibilita la generación de un contenido informativo confiable para llevar a cabo dichas acciones. En este sentido solo se conoce hasta el momento en el país, la experiencia departamental del sistema de Información para la Competitividad del Tolima (SITCOL) con miras a incrementar la productividad y la competitividad, pero basado en un sistema de cuentas departamental y no bajo un análisis de técnicas económicas [ST 03].

Es por esto que surge la necesidad de generar un sistema de información económico que permita condensar y estructurar los datos que existen en diferentes fuentes, adaptándolos, para consolidar una herramienta que permita procesar y generar información relevante que sea la base para el análisis económico que diversas instituciones del orden regional, nacional o local (Cámara de Comercio del Cauca, Gobernación del Cauca, Grupos de Investigación, etc.) puedan hacer con fines de proyección.

4.2 PROYECTO SITEC

El desarrollo del Sistema de Información e Integración Tecno Económico para Análisis Econométrico y Soporte a Gestión Local, es un proyecto que tiene como objetivo Constituir un sistema de información tecno-económica para el Departamento del Cauca, con un alto valor estratégico, que permita hacer seguimiento a la evolución económica de la región y que contribuya a sustentar las decisiones por parte de las instituciones gubernamentales, empresariales y la comunidad caucana en general. En busca de este objetivo se han reunido inicialmente instituciones como la Universidad del Cauca, dando soporte en los campos de la economía y econometría a través del Grupo de Investigación en Ciencias Contables Económicas y Administrativas (GICEA) y en el aspecto técnico y tecnológico por parte del Grupo de Ingeniería Telemática y la Cámara de Comercio del Cauca facilitando las gestiones administrativas con los propietarios de la información. Para lograr una integración real de la información del departamento, se requiere conformar una estructura de coordinación con todas las instituciones que por su dinámica generen y almacenen información relevante para las proyecciones, que facilite los acuerdos administrativos para contar permanentemente con la disponibilidad de tal información.

Objetivos

- Constituir un sistema de información tecno-económica para el Departamento del Cauca, con un alto valor estratégico, que permita hacer seguimiento a la evolución económica de la región y que contribuya a sustentar las decisiones por parte de las instituciones gubernamentales, empresariales y la comunidad caucana en general.

- Construcción de una plataforma informática y tecnológica en Internet para el procesamiento de información económica procedente de diferentes fuentes, que permitan mejorar el análisis de los datos existentes en la región.

  Producir de manera sistemática Indicadores de coyuntura y predicción de la actividad económica caucana, con el propósito de contribuir al análisis y la previsión del comportamiento de la actividad económica.

4.3 ¿POR QUÉ SITEC COMO ESCENARIO DE VALIDACIÓN?

Una exigencia para la concepción de un sistema económico para el departamento del Cauca, es que debe fundamentarse en la definición adecuada del dominio en el que se enmarca, de tal forma que permita representar acertadamente los conceptos y las relaciones que entre ellos existen.

Partiendo de la premisa de que las características de la información de SITEC no pueden diferenciarse de las características de cualquier sistema que gestione información económica y teniendo en cuenta que las instituciones del departamento, y en general, del país se han visto en la necesidad de construir sistemas informáticos que faciliten la gestión de la información que manejan, se vio la necesidad de plantear una primera aproximación de un modelo arquitectónico de referencia que permitiera lograr un acuerdo en la forma de describir los conceptos y relaciones involucrados en la realización de cualquier sistema que manejara información económica y que sirviera de base para las implementaciones independientemente de la tecnología utilizada.

Teniendo en cuenta entonces, el requerimiento de manejar la conceptualización y caracterización del dominio separadamente de los detalles de la implementación, se propuso tomar como referencia para el desarrollo el enfoque de MDA (Model Driven Architecture) de la OMG, con la cual se pueden distinguir dos fases fundamentales, la primera es la concepción lógica del sistema, enfocada a los aspectos funcionales y conceptuales del dominio del sistema y la segunda, la cual permite aterrizar la concepción lógica a una plataforma real de implementación.

Después de realizar una revisión documental y verificar el estado del arte de MDA, se concluyó que el desarrollo no podía abordarse completamente con esta alternativa debido a la falta de madurez en su definición y de las herramientas de apoyo. (Ver conclusiones Capitulo 3).

Por esta razón, se decidió plantear una aproximación al Modelo Independiente de la Computación y por medio de su refinamiento, a Modelos Independientes de la plataforma en este dominio que sirviera de base para desarrollos posteriores en lo que respecta a MDA, y orientar el desarrollo con una metodología adecuadamente definida, como lo es el modelo de construcción de soluciones [SC 02].

4.4 DESCRIPCIÓN DE LAS CARACTERÍSTICAS DEL ENTORNO

Elementos del contexto

- **Instituciones o Fuentes:** Se define una institución o fuente como aquellas entidades que por su dinámica, generan y gestionan información relevante para el departamento, que con acuerdos previos entre las partes se pone a disposición de otras entidades o grupos de proyección económica. Cada institución o fuente tiene su propio mecanismo para el almacenamiento de la información.

- **Instituciones de proyección:** Son grupos o entidades que se encargan de recolectar la información de las diferentes instituciones o fuentes para ponerla a disposición de los usuarios interesados. En este rol encontramos por ejemplo a la cámara de comercio y su esfuerzo por publicar el Anuario estadístico del departamento del Cauca.

- **Información económica:** En su forma más básica, la información económica son datos. Esto quiere decir que son valores asociados a una unidad de tiempo.
  Dato = valor + unidad de tiempo
  Una colección de datos económicos debe contar con las características siguientes:
  Nombre: nomenclatura o forma de nombrar o conocer la colección de datos
  Unidades: si los datos están dados en porcentajes, decenas, centenas, miles de unidades, millones de unidades etc.

Periodo o referencia de tiempo: Referencia del tiempo en el que se encuentran consignados los datos.

Periodicidad: la diferencia en el tiempo de un dato y el siguiente.

4.5 CASOS DE USO DEL NEGOCIO

4.5.1 Diagrama de Casos de uso del negocio

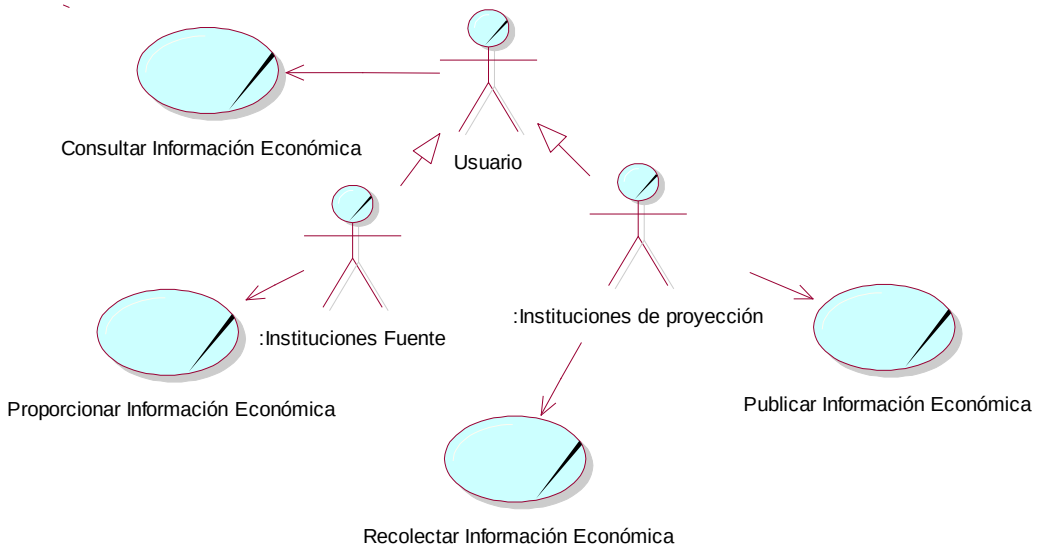

Figura 4.1. Diagrama de casos de uso del negocio

4.5.2 Descripción de los casos de uso del negocio

Proceso del negocio: Proporcionar información Económica

Responsable: Instituciones/Fuentes

Objetivo: Poner a disposición de los grupos económicos, los datos recolectados debido a su interés.

Descripción: A medida que las instituciones o fuentes de información económica recogen datos de las variables económicas que les competen, ponen esta información a disposición de las instituciones de proyección para que puedan realizarse análisis de la situación del departamento.

Prioridad: Básico

Proceso del negocio: Recolectar información Económica

Responsable: Instituciones de Proyección

Objetivo: Recolectar e integrar la información que proporcionan las diversas instituciones en un solo formato.

Descripción: Los grupos de investigación de economía, entidades gubernamentales, entre otros, tienen la función de proyectar la situación económica del departamento. Por esta razón, recogen información de diversas instituciones que puede soportar la generación de indicadores, y la estructuran en un formato común para permitir su posterior publicación.

Prioridad: Básico

Proceso del negocio: Publicar información Económica

Responsable: Instituciones de Proyección

Objetivo: Dar a conocer la información recolectada.

Descripción: Una vez se ha recogido la información económica, las entidades de proyección publican los resultados. Estos informes se ven materializados en el anuario estadístico del Cauca, estudios como el ICER, la encuesta empresarial y la información del registro mercantil.

Prioridad: Básico

Proceso del negocio: Consultar información Económica

Responsable: Usuarios en general

Objetivo: Consultar la información publicada por las instituciones de proyección.

Descripción: Gracias a las publicaciones de las instituciones de proyección, las personas y entidades interesadas en conocer los resultados pueden tener acceso a la información recolectada a través de las diferentes publicaciones de análisis y estudios económicos.

Prioridad: Básico

4.6 DIAGRAMA DE SECUENCIA

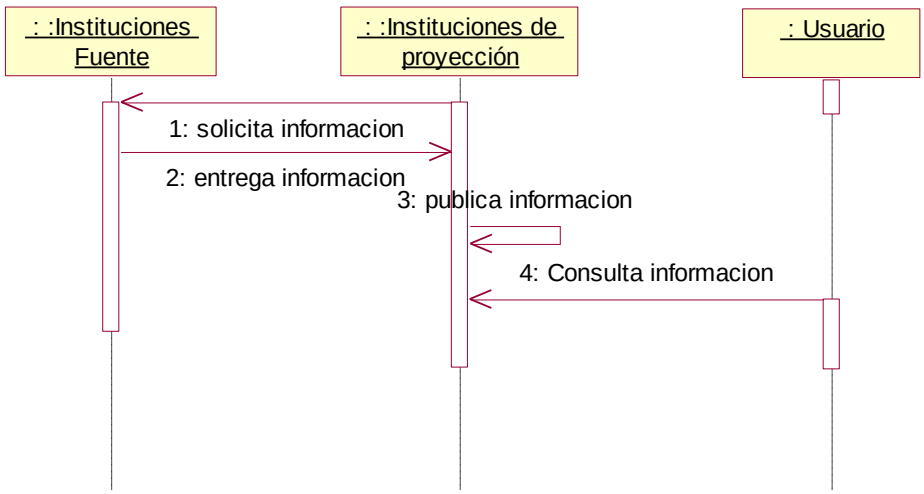

Figura 4.2. Diagrama de secuencia del negocio

4.7 MODELO DE REFERENCIA

El modelo de Referencia describe la división de la funcionalidad y el flujo de datos entre las partes. Partiendo del modelo del dominio, se deduce la funcionalidad que debe tener el sistema a desarrollar y se divide en partes o subsistemas.

Para la descomposición en partes funcionales, se tomaron tres estructuras base:

1. El sistema central el cual contiene el repositorio de los datos recolectados de las diferentes fuentes y que brinda las interfaces de consulta de información, de ingreso de datos y de administración y gestión.
2. Las fuentes: Las cuales publican la información requerida de su base de datos propia para ser replicada en el sistema central. Este sistema proporciona interfaces de administración, gestión y configuración.
3. Información económica: Entendido como el flujo de datos que circula entre los sistemas.

4.7.1 Sistema Central SITEC

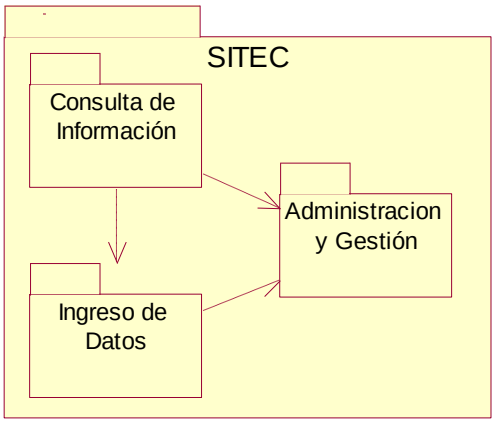

Figura 4.3. Subsistemas Funcionales SITEC

4.7.2 Sistema Fuente

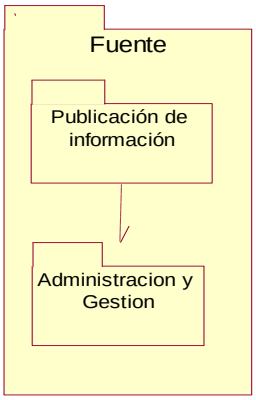

Figura 4.4. Subsistemas Funcionales Fuente

4.7.3 Modelo de información Económica

Esta estructura de datos, se definió en conjunto con los integrantes de GICEA, de acuerdo al análisis planteado acerca de la Integración de Aplicaciones, más específicamente al de la Integración de aplicaciones orientada a la información que se explicará en detalle en la sección 4.11. Las entidades y las relaciones que contiene la estructura representan la conceptualización general del contexto económico. Este es el punto de partida para el desarrollo del sistema

central, el lenguaje común para la comunicación con las fuentes y el punto de referencia de cualquier desarrollo que se enmarque en este dominio.

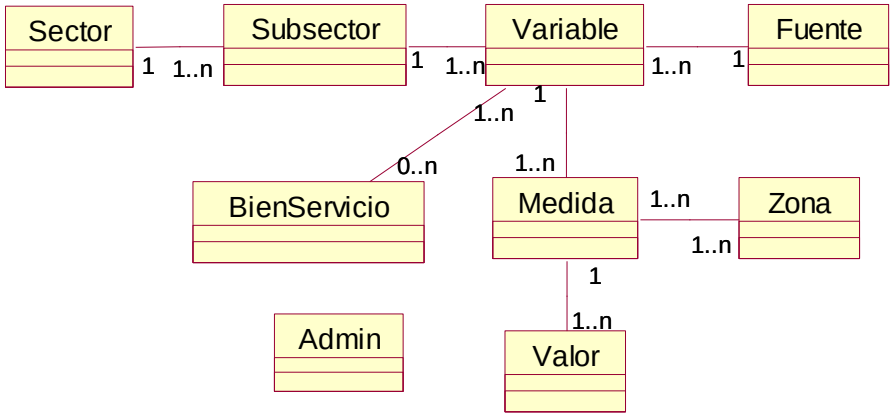

Figura 4.5. Modelo de información Económica.

Este modelo de referencia se aproxima a la definición del modelo independiente de la computación (CIM), definido para MDA, ya que expresa la funcionalidad del sistema sin ser afectado por aspectos tecnológicos ni de implementación y permite visualizar lo que se esperará que realice. Debe anotarse que este modelo está en un nivel muy alto de abstracción, por lo cual no define estructuras funcionales de más bajo nivel. Este refinamiento se presentará cuando se exploten las funcionalidades planteadas.

4.8 LOS ESTILOS ARQUITECTÓNICOS SELECCIONADOS

Buscando soportar los procesos del negocio, se propuso la creación de un sistema que permitiera almacenar los datos de todas las fuentes de información que se acordaran, en un único repositorio central y que proporcionara una interfaz web que permitiera la realización de diverso tipo de consultas por parte de los usuarios y de gestión y configuración.

Teniendo en cuenta que el sistema central debe alimentarse de los datos ingresados en las instituciones fuente, se implementaron dos alternativas de solución para soportar el ingreso y actualización de datos:

- **Sistema distribuido:** Que permita la actualización semiautomática de los datos. La responsabilidad de la actualización del sistema central recae sobre sí mismo, a través de funciones de administración.
- **Sistema Web:** Que permita el ingreso manual de los datos. Este mecanismo sirve como la interfaz entre las fuentes y el sistema central, y la responsabilidad de la actualización de los datos recae sobre las instituciones fuente.

Para cada uno de estos sistemas se escogió un estilo arquitectónico que es, respectivamente:

4.8.1 Arquitectura multinivel: Para los subsistemas de consulta, administración e ingreso web. El primer nivel consiste en la capa de presentación que incluye no sólo el navegador, sino también el servidor web que es el responsable de dar a los datos un formato adecuado. El segundo nivel está referido a la lógica de la implementación y finalmente, el tercer nivel proporciona al segundo los datos necesarios para su ejecución.

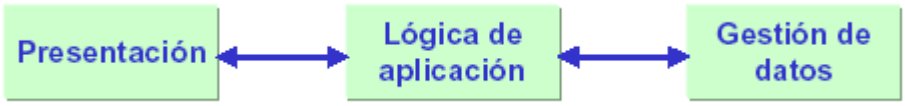

Figura 4.6 Arquitectura multinivel

4.8.2 Arquitectura de mediación: Para el sistema distribuido, que comprende los subsistemas Wrapper y mediador. La arquitectura de mediación está compuesta de un mediador, el cual localiza la información relevante a través de los sistemas de información, resolviendo los conflictos de estructura y semántica de los datos, y, los envoltorios, que son las interfaces de cooperación con los sistemas de información. Bajo este contexto los datos permanecen en las fuentes y es el mediador el responsable de proporcionar a los usuarios la apariencia de estar realizando consultas sobre un único esquema global. [CJ 03]

La arquitectura del sistema mediador adoptado en el proyecto se muestra en la Figura 4.6. Se realizó una adaptación en las capas y las posiciones de éstas, propuestas por la arquitectura original, debido al requerimiento de la construcción y mantenimiento del sistema central.

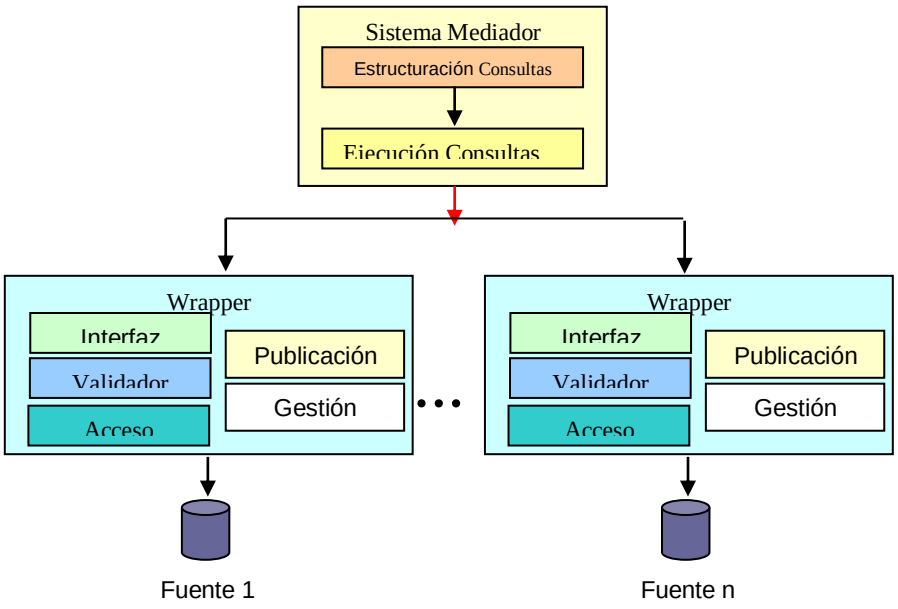

Figura 4.7. Arquitectura de sistema mediador adaptada para SITEC

Capas del sistema mediador

Estructuración de Consultas: Esta capa se encarga de recibir las consultas del cliente y estructurarlas acorde con los formatos establecidos en los servicios web de las fuentes.

Ejecución de Consultas: esta capa realiza las consultas propiamente dichas al sistema Wrapper de las fuentes, por medio de la invocación de los servicios web publicados en cada una de las instituciones.

Capas del Sistema Wrapper

En el sistema Wrapper se pueden distinguir dos frentes:

Gestión Wrapper: Este frente es accedido por el administrador de la fuente. Permite publicar la información que se desee poner a disposición de los clientes de los servicios y realizar labores de gestión del sistema como la gestión de usuarios y clientes de los servicios.

Wrapper: Se encarga de la implementación de la lógica de los servicios web por medio de los cuales da respuesta a las solicitudes hechas desde el sistema mediador. Consta de una interfaz con el sistema mediador que es el servicio web propiamente dicho, una parte de validación de los clientes y las consultas y el acceso a los datos.

4.9 ARQUITECTURAS DE REFERENCIA

La arquitectura de referencia permite mapear las funcionalidades encontradas en el modelo de referencia a un estilo arquitectónico determinado, generando una descomposición del sistema que de respuesta a las necesidades funcionales de la aplicación.

La arquitectura de referencia se relaciona con la definición de modelo independiente de la plataforma, porque mantiene la característica de comprender únicamente aspectos funcionales y de comportamiento del sistema, pero empieza a involucrar aspectos tecnológicos que no dependan de ninguna plataforma particular los cuales son propios del estilo arquitectónico seleccionado.

Las funcionalidades ofrecidas por el sistema central, tales como consulta, gestión, administración sugieren la adopción de la arquitectura multinivel. La carga y actualización automática de los datos, se basa en el estilo arquitectónico del sistema mediador.

Partiendo de la premisa que las instituciones cuentan con sus propios sistemas de información, y sabiendo que en el dominio económico no existe ningún estándar para el manejo de la información en este contexto, se deduce que los sistemas no tienen el mismo modelo de datos y se implementan en sistemas gestores de bases de datos diferentes. Sabiendo que no es posible solicitar la implementación de sistemas nuevos, la replicación de los datos de las instituciones en el sistema central tiene que solucionar los problemas de heterogeneidad, por lo que fue fundamental la elección del estilo arquitectónico del sistema mediador. Además, para sobrellevar el problema de la distribución geográfica de las fuentes, en la capa de interfaz de las fuentes se implementaron servicios web.

4.9.1 Inserción manual de datos vía Web

Sistema Central SITEC: Para la inserción manual de consultas, el sistema central se compone de los siguientes subsistemas:

a. **Subsistema de Consulta:** El cual permite que los usuarios de SITEC tengan acceso a la información que haya sido almacenada en el sistema central, por medio de la selección de diferentes tipos de consultas y de diversos parámetros.
b. **Subsistema de Administración SITEC:** Permite gestionar toda la información de los parámetros (configuración) y actores (usuarios) que se relacionan con el sistema.

c. **Subsistema de Ingreso Web:** Este subsistema es una solución alternativa al subsistema Wrapper, ya que cuenta con una interfaz web que puede ser accedida por los administradores de las fuentes o los responsables de la información permitiendo el ingreso de datos, de forma manual, al sistema central.

Todos los subsistemas nombrados son aplicaciones web que aplican el patron Modelo, Vista, Control.

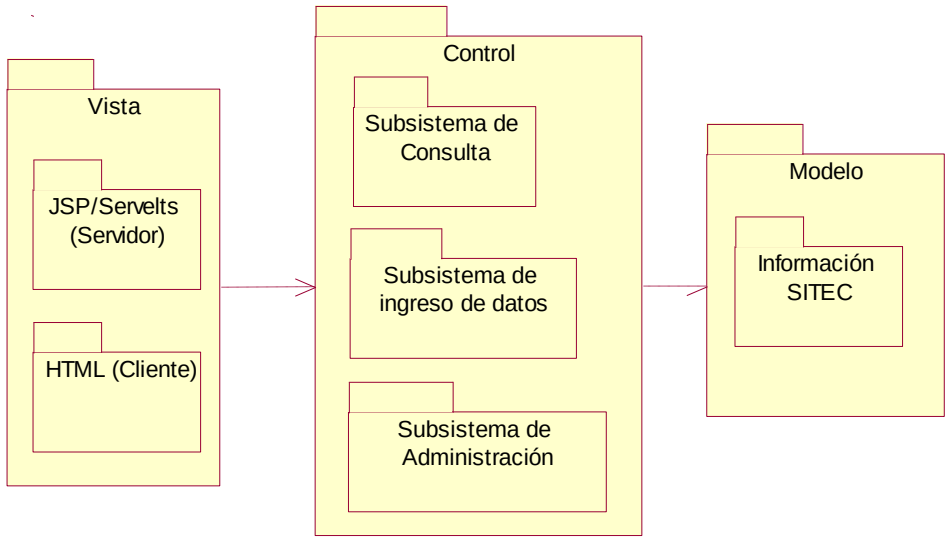

Figura 4.8. Arquitectura de referencia del Sistema Central para el ingreso manual de datos vía Web

Para esta solución, la fuente es un cliente sin ningún procesamiento ya que la funcionalidad de la publicación e ingreso de datos es soportado por una aplicación web en el sistema central.

4.9.2 Inserción automática de datos

Para la inserción automática de datos se pueden distinguir dos ambientes de la aplicación, el sistema central y el sistema fuente.

- El sistema central consta de los mismos subsistemas descritos en la inserción manual de datos excepto por el subsistema de inserción de datos que en este escenario es reemplazado por el subsistema mediador el cual cumple las funciones de inserción de los datos en el entorno distribuido. El cliente del sistema mediador es el administrador de SITEC y es el responsable de hacer la actualización de los datos y registrar las diferentes fuentes.

- El sistema fuente incluye un subsistema Wrapper el cual es el encargado de responder las solicitudes del mediador. Cada subsistema Wrapper tiene una interfaz de gestión que permite que los administradores de las fuentes publiquen la información que quieren poner a disposición de SITEC y realicen labores de administración del sistema. Como mecanismo de acceso y transporte de información, cada fuente implementa un servicio web que es de conocimiento del sistema central para poder realizar las actualizaciones de los datos

4.9.2.1 Vista de Implantación:

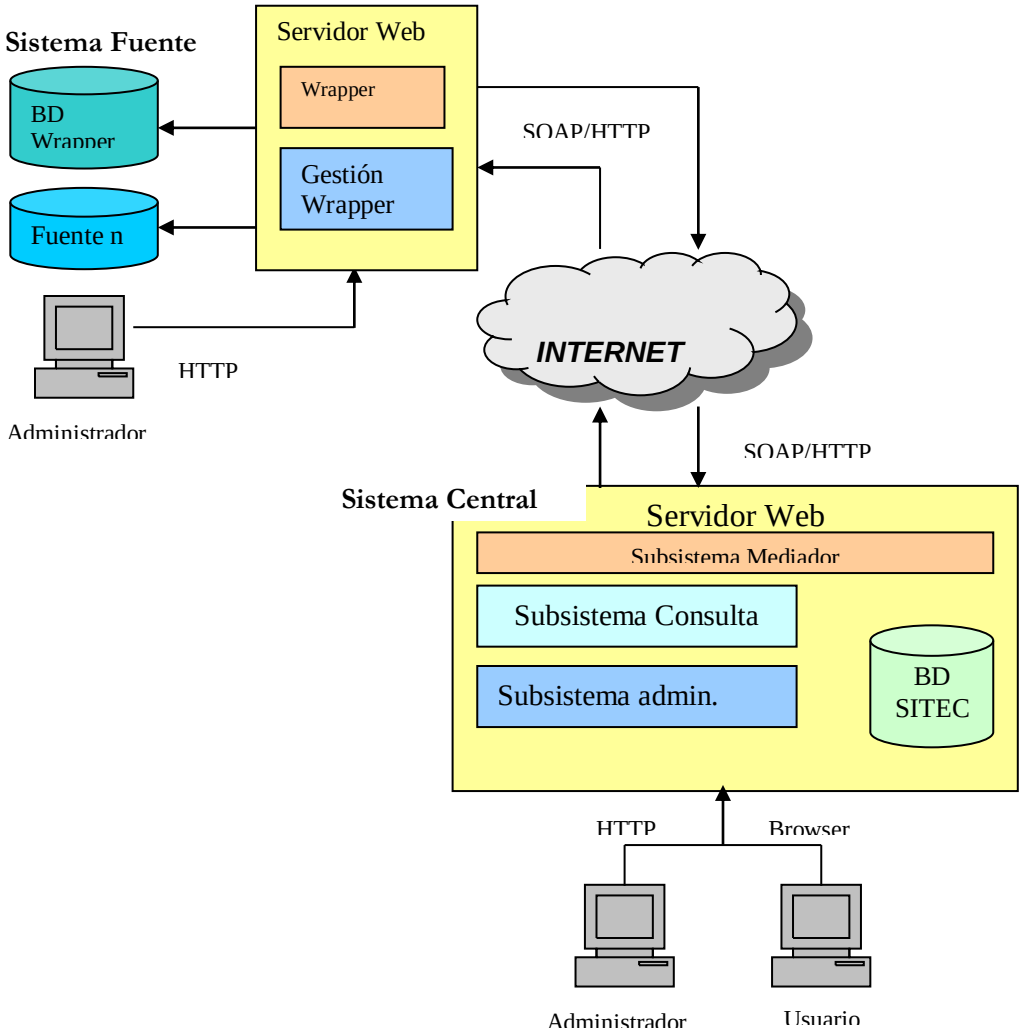

Figura 4.9. Arquitectura de Referencia para la inserción automática de datos. Sistema Distribuido

4.9.2.2 Vista lógica

En esta parte se hará énfasis en la arquitectura de referencia para la aplicación distribuida propiamente dicha. No se nombrarán los subsistemas de consulta y de administración del sistema central ya que éstos son comunes y no difieren en los dos escenarios.

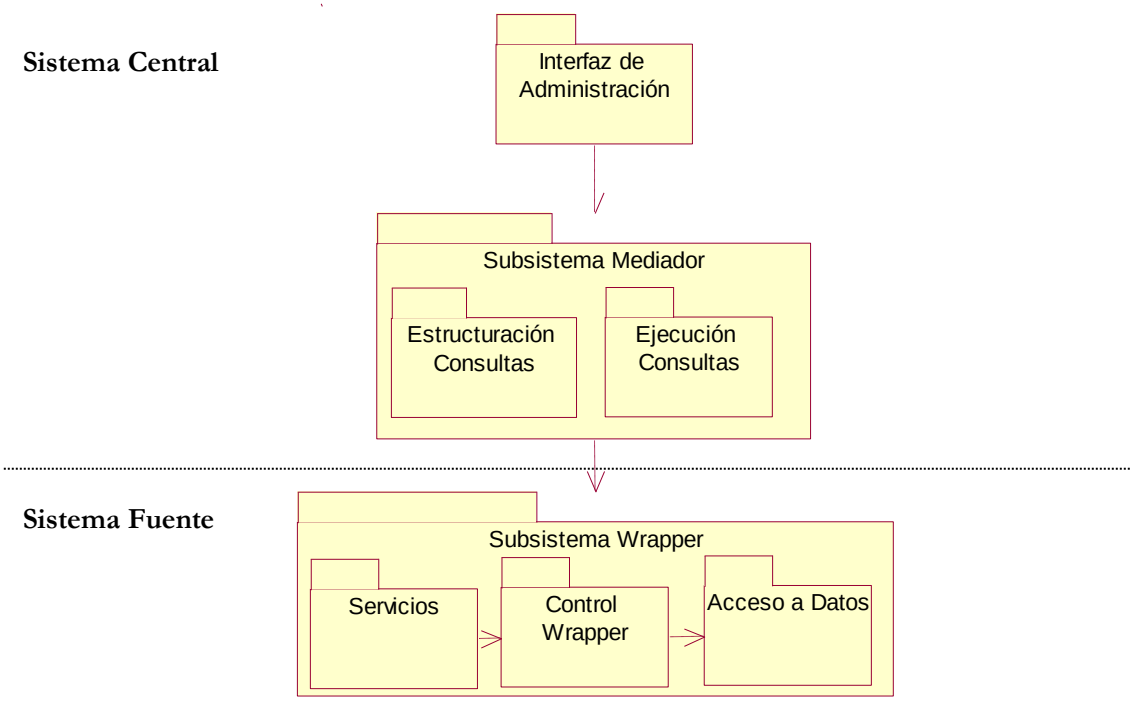

Figura 4.10 Vista lógica del sistema distribuido

Sistema Central: Para el escenario distribuido, el sistema central cuenta de la interfaz de administrador y el Subsistema mediador:

- **Interfaz de Administración:** Acorde con el modelo MVC este paquete involucra los componentes de presentación al usuario. En esta interfaz el Administrador tendrá la posibilidad de escoger cuales son los datos a obtener de cada una de las fuentes adscritas a SITEC.
- **Subsistema Mediador:** Ejecuta las labores de control que le permiten solicitar peticiones al sistema Wrapper. Como se describió en el estilo arquitectónico, este subsistema cuenta con una capa de estructuración de consultas y una de ejecución de consultas.

- **Sistema Fuente:** Sin mencionar la interfaz de gestión del wrapper, el sistema fuente se comprende del Subsistema Wrapper el cual a su vez consta de las siguientes capas:
    - **Servicios:** Es la interfaz del wrapper con el sistema mediador realizada a través de servicios web.
    - **Control del Wrapper:** En este paquete se incluye la lógica de los servicios web (interfaz de invocación de los métodos de consulta), incluyendo la validación de las consultas y de los clientes solicitantes, de la publicación de datos y de la gestión del sistema.
    - **Modelo:** Este paquete contiene las clases que se encargan del acceso a los datos almacenados en el Wrapper y a los datos originales de la fuente.

4.10 SOLUCIÓN TECNOLÓGICA DE INTEGRACIÓN DE LAS APLICACIONES FUENTE Y EL SISTEMA CENTRAL

Debido a la diferencia en las instituciones que contienen las fuentes originales de los datos económicos y la falta de estándares en relación con los términos y notaciones del contexto económico, es de suponer que cada una de dichas instituciones posea un mecanismo de almacenamiento que utilice una estructura de datos específica, y en caso de contar con sistemas de información, posean sistemas gestores de bases de datos de diferentes proveedores que se ajusten a sus requerimientos particulares, además de estar separadas geográficamente.

El sistema central, SITEC, requiere tener la información de las diferentes fuentes, es decir, debe haber una réplica de datos en SITEC de lo que se encuentra en cada una de las fuentes. Según este contexto se puede diferenciar dos procesos: uno, la integración de información y el otro, la trasmisión y distribución de dicha información. De acuerdo a esto se han identificado dos tipos de integración de aplicaciones, una es orientada a la información y el otro es orientado a los servicios.

4.10.1 Integración de aplicaciones orientado a la información.

De acuerdo con los patrones de clasificación de los consumidores y productores de información vistos en el Capítulo 3, la Integración de Información para el dominio particular de SITEC, se encuentra dentro del patrón de Bases de Datos, porque en términos generales lo que se hace es una petición SQL a las fuentes para poder nutrir la Base de Datos de SITEC.

Según las consideraciones previas, para la integración de aplicaciones orientadas a la información se han identificado ciertos pasos, se tiene: identificación de datos, catalogación de datos y por último construcción del modelo de metadatos.

Identificación de datos

Los datos que van a ser integrados en SITEC provienen de diferentes fuentes, cada una de estas fuentes se encuentra en sitios geográficos diferentes. A manera de ejemplo, en la tabla que se muestra a continuación se listan las fuentes que actualmente se encuentran adscritas a SITEC y las variables que comparten.

VARIABLE	MEDIDA	PERIODICIDAD
DANE (DEPARTAMENTO ADMINISTRATIVO NACIONAL DE ESTADÍSTICA)		
Consumo de Energía Eléctrica	KW por hora	Anual
Consumo Intermedio	Pesos	Anual
Evolución	Número de Personas	Anual
Inversión Neta	Pesos	Anual
Licencias de Construcción	Número de Licencias	Mensual
Número de Establecimientos	Pesos	Anual
Personal Ocupado	Pesos	Anual
Sacrificio de Ganado	Cabezas, Peso (Toneladas)	Anual
Terrestre Urbano	Parque Automotor, Número de Pasajeros Trasportados	Mensual
Valor Agregado	Pesos	Anual
Valor de la Producción	Pesos	Anual
Valor de las Ventas	Pesos	Anual
Valor en Libros de Activos	Pesos	Anual
BANCO DE LA REPÚBLICA		
Bancos Comerciales	Pesos	Anual
Corporaciones de Ahorro y Vivienda	Pesos	Anual
Gastos del Gobierno Central Departamental	Pesos	Anual

Ingresos del Gobierno Central Departamental	Pesos	Anual
Otros Intermediarios	Pesos	Anual
CEDELCA (CENTRALES ELÉCTRICAS DEL CAUCA S.A.)		
Alumbrado Público	MWH	Anual
Consumo Comercial	MWH	Anual
Consumo Industrial	MWH	Anual
Consumo Oficial	MWH	Anual
Consumo Recuperado	MWH	Anual
Consumo Residencial	MWH	Anual
Número de Suscriptores	Número de Personal	Anual
Otros Servicios Extras	MWH	Anual
AERONÁUTICA CIVIL		
Aéreo	Toneladas	Anual
Aéreo	Número de Personas	Anual

Tabla 3.1 Identificación de datos de SITEC

Además de los datos que se pueden obtener de las diferentes fuentes, la Cámara de Comercio del Cauca produce anualmente un Anuario Estadístico en el que se encuentran condensadas todas las variables relevantes para SITEC, los datos más actualizados van desde el año 1990 hasta el año 2002.

Latencia de datos

Según las necesidades del proyecto SITEC y según la periodicidad de los datos de la tabla presentada anteriormente, la latencia de SITEC debe ser en tiempo cercano, ya que los datos pueden ser tan oportunos como se necesiten, y esta oportunidad se ha identificado mensual.

Estructura de los datos

Se sabe que todos y cada uno de los datos económicos que van a ser parte de SITEC deben tener la estructura que para este se ha establecido, esto es: cada uno de los datos (Dato = valor + unidad de tiempo) debe estar comprendido dentro de la clasificación de Sector, Subsector, Variable, Bien y/o Servicio, Zona y Medida.

Catalogar los Datos

Ya conociendo las fuentes y las variables que cada una de estas van a compartir y sabiendo que cada uno de los datos tiene que pertenecer a la clasificación de Sector, Subsector, Variable, Bien y/o Servicio, Zona y Medida, el catálogo de datos que se dedujo para SITEC se encuentra en la figura 4.5 representando el modelo de información económica.

En él encontramos los siguientes conceptos:

- **Sector:** Donde se enmarcan los sectores de la economía como primario, secundario, terciario y sector público.
- **Subsector:** Son actividades económicas que se enmarcan dentro de cualquiera de los sectores mencionados. Tales como Ganadería, Agricultura, Finanzas, Transporte, etc.
- **Variable:** Una variable económica es una magnitud de interés que puede definirse y medirse (por ejemplo: ventas, producción, gastos, etc.), que influye en las decisiones relacionadas con lo que se ocupa la economía, o describe los resultados de esas decisiones.
- **Bienes/Servicios:** Los bienes o servicios se relacionan con variables para dar información más precisa. Por ejemplo, la variable producción está relacionada con los bienes azúcar, caña molida, arroz riego, etc. No todas las variables tienen asociados bienes o servicios.
- **Medida:** Las unidades en las que se realizan las mediciones de las variables se denominan medidas. Por ejemplo, Toneladas es la medida para la variable producción del Bien Arroz.
- **Zona:** Indica la zona para la cual se consignan los valores de la medida. Una medida es única por su periodicidad, unidad y la zona a la que pertenezca.
- **Valor:** Esta entidad representa el concepto de dato económico descrito en el modelo de información económica.
- **Fuente:** Representa el repositorio de datos de la institución.

La cardinalidad significa, un Sector puede estar relacionado con muchos Subsectores y un Subsector solo puede tener relación con un Sector, un Subsector puede tener relación con

muchas Variables pero una Variable con un solo Subsector, una Variable puede tener ninguno o muchos Bienes y/o Servicios, y un Bien y/o Servicio puede estar relacionado con una o muchas Variables, y una Variable puede estar relacionada con solo una Fuente y una Fuente puede tener una o muchas Variables; una Variable puede estar relacionada con muchas Medidas pero una Medida con una sola Variable; una Medida puede estar relacionada con muchas Zonas y una Zona con muchas Medidas; y una Medida puede tener relación con uno o mas valores.

Modelo de Metadatos: El modelo de metadatos utilizado para SITEC se enmarca dentro del catalogo de datos identificado, pero por las condiciones de los datos encontrados en las fuentes, se suprimen los conceptos de Sector y Subsector. La razón fundamental es que las fuentes no establecen una relación explícita en su modelo de datos entre las variables que manejan y los conceptos Sector y Subsector definidos en el catálogo.

4.10.2 Integración de Aplicaciones Orientada a los Servicios

Dentro de SITEC se identifica una Integración Orientada al Servicio, debido a que la información que se obtiene de las fuentes para ser luego mapeada a la Base de Datos en SITEC, es obtenida a través de la invocación a un Servicio Web. Dentro de este contexto, se encuentra que el patrón al que se ajusta esta integración es Manejado por Eventos, ya que este patrón involucra el movimiento de información y la invocación a una aplicación, que en el caso particular se encarga de recibir la petición con los datos que se quieren obtener de la fuente, ejecutar la consulta solicitada por el sistema central, construir y retornar un archivo XML con los resultados de la consulta enmarcados en una estructura que corresponde al modelo de metadatos definido. En la figura 4.11 se muestra el resultado de una consulta resuelta por la aplicación del wrapper con la estructura de los metadatos definidos para la integración de información.

```
<?XML VERSION="1.0" ENCODING="ISO_8859_1"?>
<VARIABLE>SACRIFICIO DE GANADO
  <BIENSERVICIO>GANADO PORCINO
    <ZONA>POPAYÁN
      <MEDIDA PERIODICIDAD="ANUAL"
        UNIDADES="UNIDADES">CABEZAS
        <VALOR>3738</VALOR>
```

```
        <TIEMPO>2000/01/01</TIEMPO>
        <VALOR>2324</VALOR>
        <TIEMPO>2001/01/01</TIEMPO>
        <VALOR>2425</VALOR>
        <TIEMPO>2002/01/01</TIEMPO>
      </MEDIDA>
    </ZONA>
  </BIENSERVICIO>
</VARIABLE>
```

Figura 4.11 Resultados de un Consulta estructurados según el modelo de metadatos

4.11 REFINAMIENTO DEL MODELO DEL SUBSISTEMA DE CONSULTA

Para ilustrar la forma en que se llevó a cabo el refinamiento de los modelos presentados, se presenta a continuación en proceso seguido con un único subsistema, el subsistema de consulta, ya que el proceso con los demás es similar y está documentado en los anexos de análisis y diseño.

4.11.1 Descripción de Paquetes de Análisis

El subsistema de consulta es una aplicación web que utilizará un estilo arquitectónico multinivel, aplicando el patrón de diseño MVC (Modelo, Vista, Control).

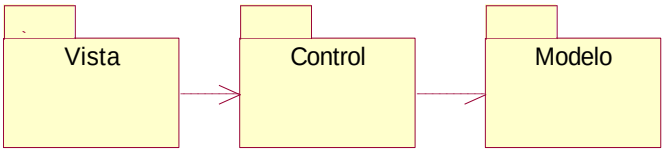

Figura 4.12 Paquetes de análisis del subsistema de consulta

4.11.2 Diagrama de paquetes de diseño

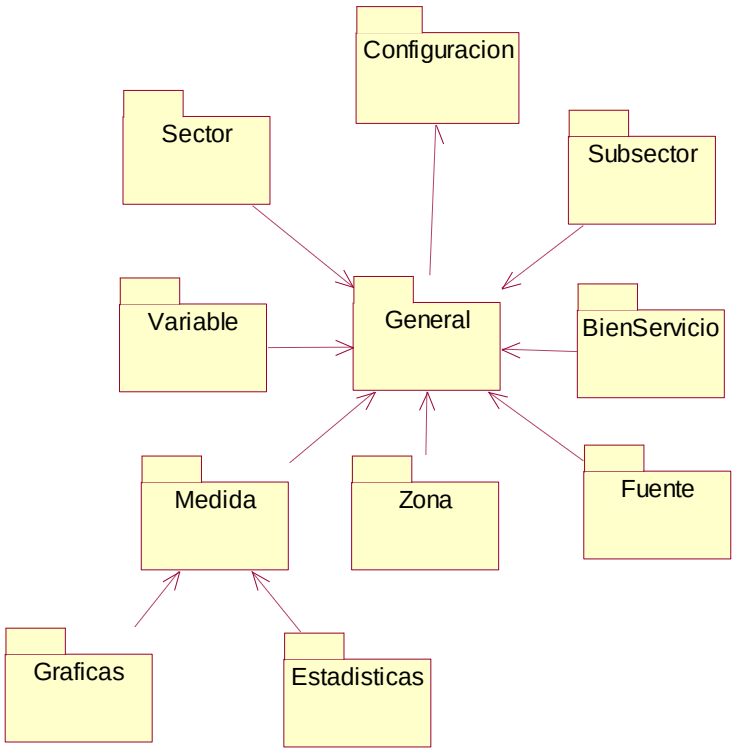

Figura 4.13 Diagrama de paquetes de diseño

- **General:** Este paquete contiene las clases que prestan funcionalidades comunes a los demás paquetes. Por ejemplo, en él se enmarcan las funcionalidades relacionadas con la conexión a la base de datos, la factoría abstracta, entre otras utilidades como los lectores de XML , etc.
- **Configuración:** Como su nombre lo indica, contiene la funcionalidad de configurar los parámetros iniciales de la aplicación.
- **Gráficas y Estadísticas:** Se encargan de presentar los resultados de las consultas en formato gráfico y generar estadísticos básicos.

Cada uno de los demás paquetes se encarga ofrecer las funcionalidades relacionadas con un concepto del modelo de información de la aplicación. De esta forma se puede ver que el modelo de información se refleja en la arquitectura lógica del sistema.

4.11.3 Clases de análisis

Como se puede ver en el diagrama de paquetes, cada uno de ellos encapsula la funcionalidad de un concepto particular del dominio. De esta forma, la estructura interna de los paquetes es equivalente. Cambia únicamente por las propiedades particulares del concepto con el cual se relaciona cada uno. Tomando como ejemplo el paquete sector, y sabiendo que la aplicación es web y por lo tanto obedece al patrón Vista, Control, Modelo se puede deducir la estructura básica de clases es como se muestra en la figura 4.14.

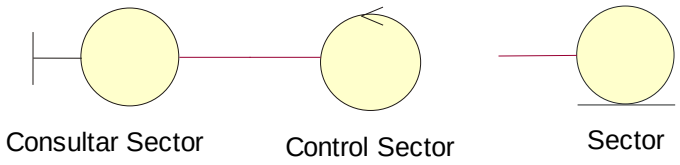

Figura 4.14 Estructura de Clases de análisis para el concepto Sector

Donde Consultar sector es la interfaz, control sector se encarga de implementar la lógica de la consulta relacionada con este concepto y Sector representa la entidad propiamente dicha, con sus propiedades y métodos de acceso a los datos.

4.11.4 Clases de diseño

Buscando desacoplar la lógica de negocio de la lógica de acceso a datos, de manera que se lograra cierta independencia de fuente de datos, se utilizó una aproximación al patrón de diseño Data Access Object de la plataforma J2EE. Debido a que el almacenamiento subyacente no está sujeto a cambios de una implementación a otra, la estrategia DAO se puede implementar utilizando el patrón Factory Method para producir el número de DAOs que necesita la aplicación. En la siguiente figura se muestra el diagrama de clases para este caso:

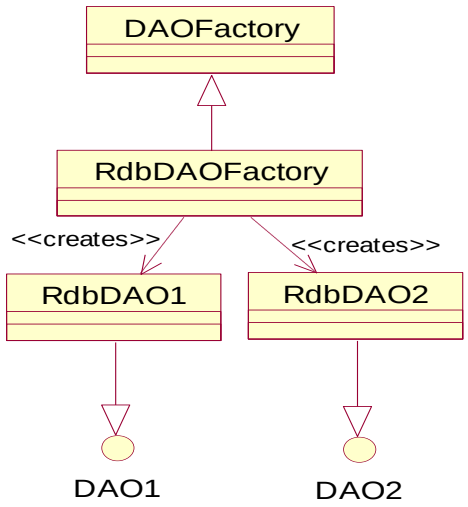

Figura 4.13. Patrón de Diseño DAO de la arquitectura J2EE

DAOFactory es una clase abstracta que extienden e implementan las diferentes factorías concretas de DAOs () para soportar el acceso específico a la implementación del almacenamiento. El cliente puede obtener una implementación de la factoría concreta del DAO como una RdbDAOFactory, la cual definirá la forma de acceso a un repositorio particular y utilizarla para obtener los DAOs concretos que funcionan en la implementación del almacenamiento, como por ejemplo RdbClienteDAO, RdbCuentaDAO, etc. Los DAOs pueden extender e implementar una clase base genérica (mostradas como DAO1 y DAO2) que describa específicamente los requerimientos del DAO para el objeto de negocio que soporta. Cada DAO concreto es responsable de obtener y manipular los datos para el objeto de negocio que soporta.

A Pesar de que el patrón de diseño es de una plataforma particular, se considera que los patrones son independientes de la plataforma ya que su utilización y paradigma puede utilizarse en casos disímiles en cuanto a implementación.

Entonces, al involucrar el patrón en el diseño del sistema, se estableció una estructura de clases similar para cada entidad del sistema, que corresponden a un PIM de diseño del sistema. Siguiendo con el paquete Sector, después de aplicar el patrón se tiene la estructura de clases mostrada por la figura. 4.14

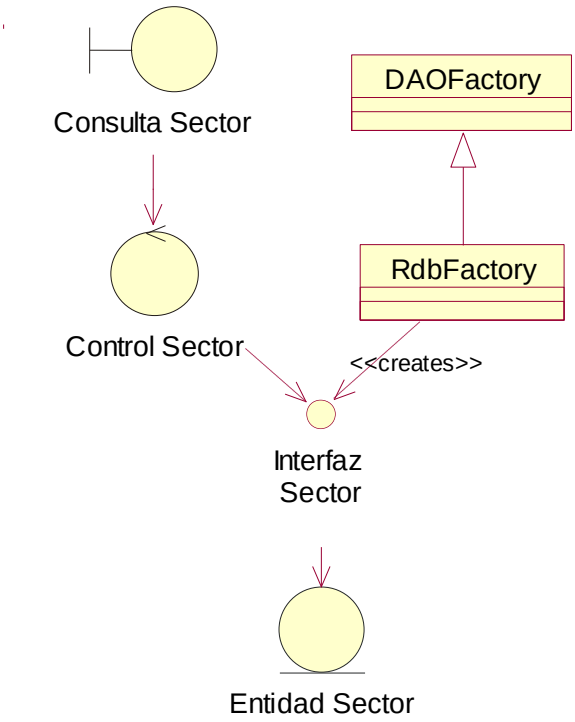

Figura 4.14 Estructura de clases con la aplicación del patrón DAO

Teniendo en cuenta que la idea del patrón es hacer a la lógica de control lo más independiente posible del almacenamiento se optó por separar las propiedades de la entidad particular, de la funcionalidad de acceso al modelo de datos. Por esto, la clase que representaba el objeto particular se dividió en una clase que contuviera únicamente las propiedades de los conceptos y otra que se encargue de la manipulación de los datos dicho objeto. De esta forma, siguiendo con el ejemplo, la Entidad Sector se divide en Sector Bean Sector.

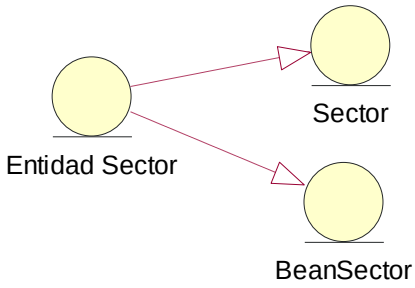

Figura 4.15 División funcional de la Entidad Sector

Así, teniendo claros todos los elementos que están relacionados con un objeto particular de la aplicación, se define la estructura interna de cada paquete y se propone un PIM parcial de diseño de la aplicación.

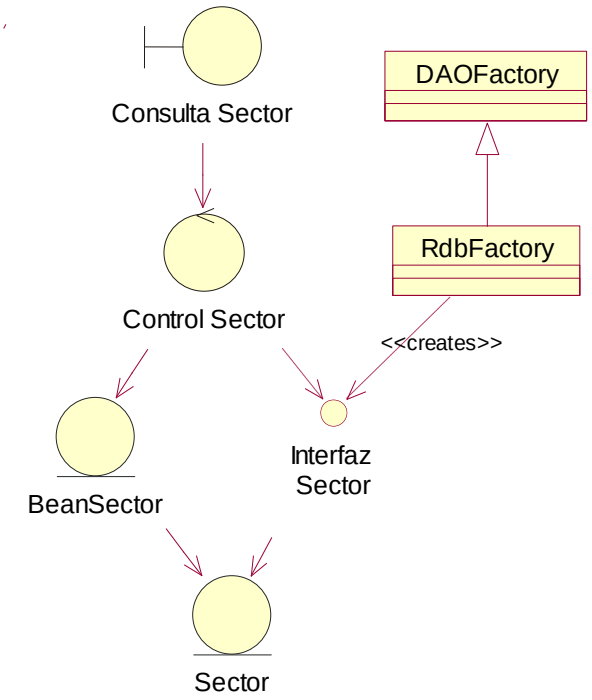

Figura 4.16 Estructura interna de clases de los paquetes de la aplicación. PIM parcial de diseño del subsistema de consulta.

4.11.5 Diagramas de Realización

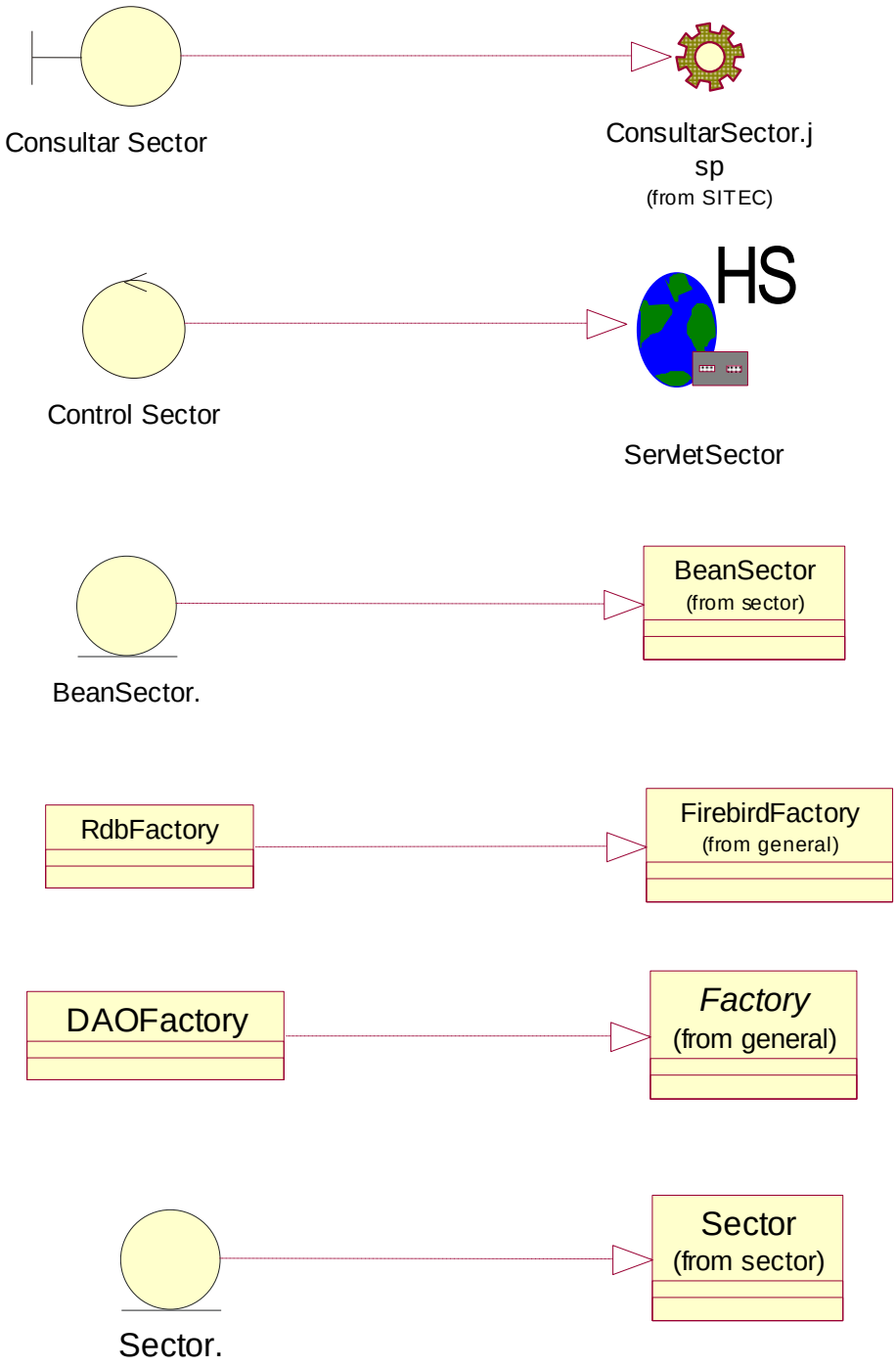

Figura 4.17 Diagramas de Realización

4.11.6 Diagrama detallado de clases de diseño

Siguiendo con el ejemplo de Sector, en el siguiente diagrama se muestra la correspondencia del PIM con la plataforma J2EE. Este diagrama representa el Modelo dependiente de la plataforma (PSM) de este subsistema.

A pesar de que se nota una transformación de modelos y una relación de refinamiento entre ellos, que parten de la descripción de la funcionalidad hasta conseguir un nivel de detalle que refleje la implementación tal y como lo plantea la iniciativa de MDA, es claro que el proceso seguido no tiene la formalidad requerida en lo que respecta a las transformaciones de modelos ya que no se siguieron reglas estrictas y adecuadamente formalizadas con lenguajes idóneos como QVT, ni los modelos han sido acompañados de mayor especificación brindada por lenguajes como OCL, al carecer de herramientas acordes a estos estándares

Además, cabe mencionar que la generación de modelos estuvo acompañada y guiada por artefactos propios del modelo de construcción de soluciones que MDA no considera formalmente, tales como diagramas de casos de uso y de secuencia. Es por esto que se pone en cuestión la madurez de esta iniciativa para permitir abordar de una manera estricta un desarrollo desde sus fases de identificación de contexto hasta la implementación.

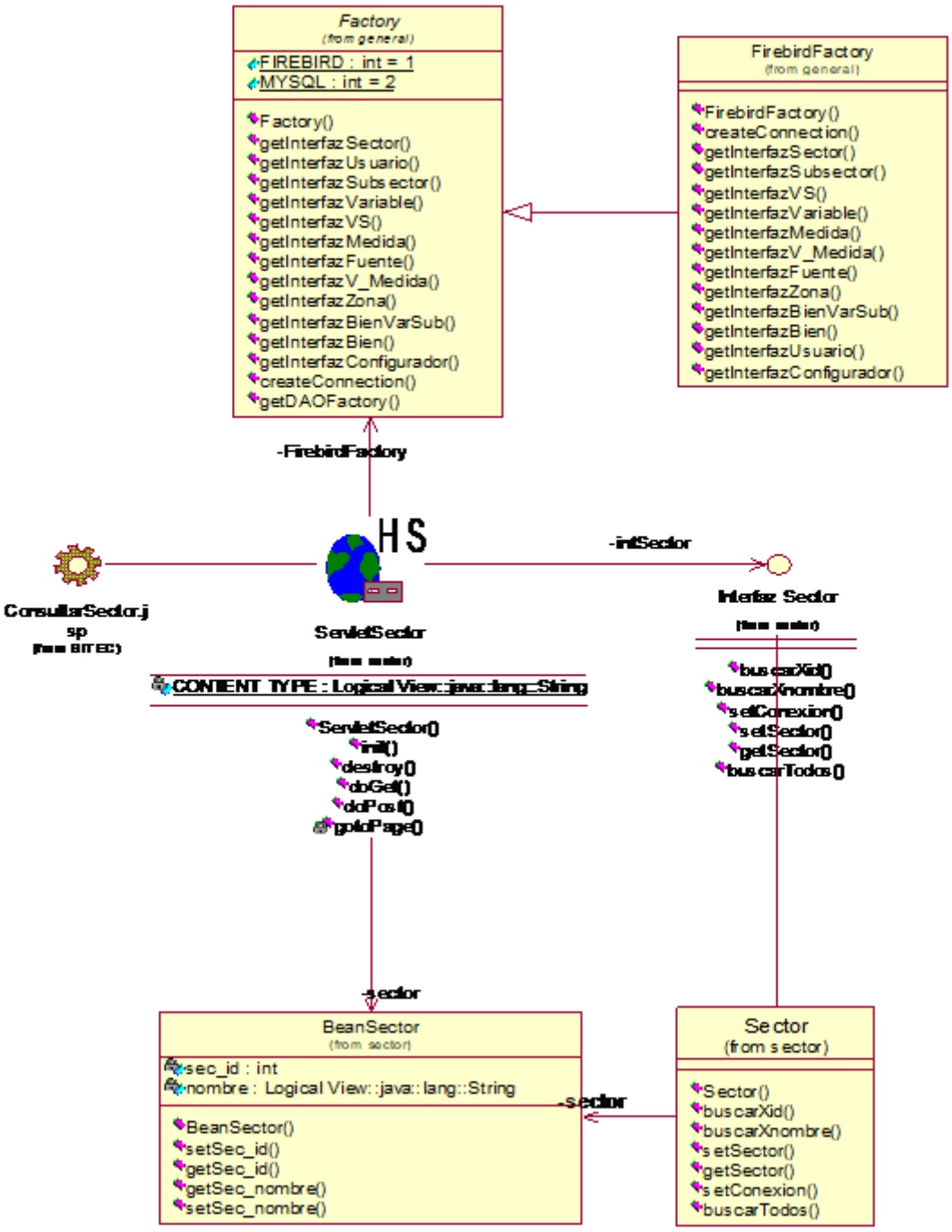

Figura 4.15 Diagrama detallado de clases de diseño para el paquete Sector. PSM parcial del subsistema de consulta

CAPITULO 5. CONCLUSIONES

1. MDA se postula como una alternativa para los problemas de interoperabilidad entre aplicaciones, vista desde dos perspectivas:

a. Interoperabilidad Conceptual: ya que las implementaciones partirían del mismo PIM base. Esto implica que sin importar las características de las plataformas, las diversas implementaciones del PIM deben comprender los conceptos en éste definidos. Por lo cual, no sería necesario hacer acuerdos semánticos ni sintácticos.
b. Interoperabilidad tecnológica: ya que es posible la especificación de enlaces desde una aplicación hasta servicios y facilidades requeridas, y/o otras aplicaciones como parte de los modelos. Trabajando con esta estructura de modelos enlazados, las herramientas de MDA automáticamente construyen puentes (Bridges) permitiendo conectar las implementaciones realizadas en diversas plataformas.

2. MDA ha ido modificando su forma a través de las tres publicaciones que la definen. En cada una de ellas se han definido conceptos y se han refinado las definiciones de los propuestos en las publicaciones anteriores, sin lograr aún una respuesta concreta a la cuestión de su forma de abordaje.

3. MDA es una propuesta que integra diversas especificaciones y estándares. Sin embargo, en sí misma no es una especificación y aún no ha pasado el proceso de adopción tecnológica que la OMG define para la publicación de las especificaciones. No obstante, la propuesta se lanzó con grandes expectativas y las industrias de software adaptaron las definiciones encontradas en los documentos existentes según su interpretación, adicionalmente se implementaron herramientas e incluso se han propuesto metodologías que integran los modelos de MDA durante el proceso de desarrollo. Pero que es MDA? Un modelo? una referencia? Una metodología de desarrollo? No existe claridad en su definición esencial, solo aproximaciones como la expresada en este trabajo.

4. La propuesta de MDA busca la automatización en la generación de código. Para esto es indispensable contar, con herramientas que implementen los perfiles UML de las diversas plataformas de desarrollo e interoperabilidad, que soporten estándares de intercambio de datos como XMI, repositorios de modelos, y que sus metamodelos sean una instancia de MOF. En este momento, las herramientas no cumplen estrictamente con los estándares de núcleo de MDA, y cada cual se encarga de definir sus propios metamodelos impidiendo la reutilización de PIMs independientemente de la herramienta. Por otro lado, las herramientas disponibles en el mercado son costosas y las que son libres, no proporcionan un entorno amigable.

5. Aprovechando la experiencia del proyecto Master [MASTER], se puede concluir que el entorno de aplicabilidad de MDA se orienta a desarrollos empresariales donde se haya podido definir una línea de productos. Esto debido a que en ese caso, partiendo de una análisis de dominio, se pueden definir características comunes de la arquitectura que serían descritos en el PIM, que pueden ser complementadas posteriormente con aspectos específicos resultados de la captura de requerimientos de soluciones particulares. De esta forma, y contando con las herramientas adecuadas de generación de código, se puede automatizar prácticamente la implementación de las soluciones generales o específicas.

6. La definición y estandarización de perfiles UML de dominios, podrían ser un punto a favor para la aplicabilidad de MDA en desarrollos relacionados, ya que los conceptos necesarios para la implementación de cualquier solución en este campo deberían estar considerados en el perfil. Las herramientas de generación de código deberían contener dichos perfiles de dominio, permitiendo construir un PIM a partir de la instanciación de los conceptos definidos por el perfil, que seguidamente serían mapeados por medio de los perfiles tecnológicos a las plataformas de implementación.

7. La integración de aplicaciones es una asociación estratégica para unir sistemas según su necesidad, va desde integración orientada a la información hasta la integración al nivel

del servicio; al integrar aplicaciones los sistemas obtienen la habilidad para intercambiar información y solucionar procesos en tiempo real.

8. La mayoría de las organizaciones no tienen estrategias de integración de aplicaciones, pero es necesario que estas organizaciones primero coloquen su casa en orden, luego determinen sus requerimientos de integración, creen un plan, y luego seleccionen el correcto acercamiento a la integración de aplicaciones y a la tecnología necesaria. Al saltar simplemente a una tecnología como Servicios Web, sin entender los requerimientos de los servicios puede ser desastroso, o peor, puede causar que la organización pierda valiosas oportunidades estratégicas.

9. La mayoría de los dominios del problema que requieren integración de aplicaciones, optarán por los acercamientos de integración de aplicaciones diferentes a los de integración orientada a los servicios. El primer tipo es más fácil de desplegar y tienen menos riesgo, sin embargo si no hay esfuerzo no hay gloria. Dependiendo de los requerimientos del dominio del problema, el acercamiento al nivel de los servicios, incluyendo todas sus dificultades, podría convertirse en la mejor solución para el problema de integración de aplicaciones.

10. La implantación de un sistema como el sistema de información Tecno Económico, depende enteramente de la coordinación y colaboración que se logre entre las instituciones objetivo, y entre los grupos interdisciplinarios involucrados. La validación de la fiabilidad de la información que cada institución posea, es un procedimiento que requiere mucha cooperación de las partes, y que es fundamental para lograr el objetivo de proyección de la situación económica de una zona determinada como en este caso, la de un departamento.

11. Se debe comenzar por introducir una cultura de información compartida, que permita que las instituciones abran su conocimiento a los interesados y que sean actores protagónicos de los procesos de observación económica del regiones.

12. El trabajo interdisciplinario es una tendencia que está tomando mucho auge en los entornos académicos y organizacionales. Sin embargo, éste debe cumplir con características fundamentales de colaboración y disponibilidad entre las partes involucradas asegurando un trabajo en conjunto acordado de una manera constante y permanente que evite la ocurrencia de errores y desacuerdos de concepciones.

13. Las soluciones tecnológicas son herramientas que facilitan los procesos operativos entre las personas. Sin embargo, los mayores limitantes que se encuentra un desarrollo con proyección social no es la aplicabilidad de la tecnología sino la adecuación del entorno que permita la aceptación del cambio.

14. Aunque la tecnología evoluciona a pasos agigantados y cada vez son más frecuentes los avances significativos, se debe entender que los ritmos de adopción tecnológica en las regiones son diferentes y dependen de las condiciones de cada una. Los ingenieros debemos pensar en dar soluciones a los problemas del entorno y acercarnos a la realidad para darnos cuenta que la tecnología de punta no es siempre pertinente, y que muchas veces lo complejo de las soluciones no se encuentra en la tecnología de los desarrollos sino en los procesos de implantación.

REFERENCIAS

[AJW 03] Anneke Kleppe, Jos Warmer, Wim Bast. MDA Explained, The Model Driven Architecture: Practice and Promise. Abril 2003

[AB 01] Architecture Board MDA Drafting Team , Model Driven Architecture. A Technical Perspective. Febrero 2001

[RS 00] Richard Soley and the OMG Staff Strategy Group, Model Driven Architecture. Noviembre 2000

[OMG 03] OMG, MDA Guide Version 1.0.1. 2003

[LPR 98] Len Bass, Paul Clements, Rock Kazman. Software Architecture in Practice. 1998

[TP 03] Tom Pender, UML Bible. 2003

[MDA 03] Object Management Group. Model Driven Architecture [en línea]. 2003 [citado Agosto de 2003]. Disponible en World Wide Web: http://www.omg.org/mda

[JS 01] Jon Siegel and the OMG Staff Strategy Group, Developing in OMG's Model-Driven Architecture, 2001.

[UML] Object Management Group Unified Modeling Language (UML). Version 1.5. International Standard

[CWM] Object Management Group. Common Warehouse Metamodel (CWM) Specification. OMG document, ad/2001-02-01. 2001.

[MOF] Object Management Group. Meta Object Facility (MOF) Specification. OMG document: formal/2002-04-03. 2003.

[QVT] MOF Query / Views / Transformations, First Revised Submission

[XMI] Object Management Group. XML Metadata Interchange (XMI) Specification. OMG document: formal/03-05-02.2003

[MTR 01] Model driven Architecture inSTrumentation, Enhancement and Refinement IST-2001-34600.

[AJD 01] Ana Belen Garcia, Jason Mansell, David Sellier .From Customer Requirements to PIM: necessity and reality. 2001

[SA 04] El sistema SAP R/3, http://www.udlap.mx/~sapudla/r3/

[DL 03] David. S. Linthicum. Next Generation Application Integration: From Simple Information to Web Services. 1st Ed., Addison Wesley, Boston 2003

[MO 04] Integración de Aplicaciones, www.es.morse.com

[DW 04] Artículo UDDI (Universal Description Discovery and Integration), www.desarrolloweb.com

[DP 98] Davenport, T.; Prusak, L. (1998), "Working Knowledge: How Organizations Manage What They Know", Harvard Business School Press.

[ST 03] Sistema de información para la competitividad del Tolima. SITCOL. 2003

[SC 02] Serrano Castaño, Carlos Enrique. "Un modelo para construcción de soluciones" Universidad del Cauca. 2002

[CJ 03] J. C. Corrales, "Framework de interoperabilidad". Informe técnico, Universidad del Cauca, Colombia. 2003.