TRANSFORMACIÓN DEL MODELO DE CLASES UML A Oracle9i® BAJO LA DIRECTIVA MDA: UN CASO DE ESTUDIO
TRANSFORMATION FROM UML CLASS MODEL TO ORACLE9i® USING THE MDA GUIDELINES : A STUDY CASE

FERNANDO ARANGO
Grupo de Investigación UN-INFO. Universidad Nacional de Colombia, sede Medellín.

MARÍA CLARA GÓMEZ
Grupo de Investigación UN-INFO. Universidad Nacional de Colombia, sede Medellín.

CARLOS M. ZAPATA
Grupo de Investigación UN-INFO. Universidad Nacional de Colombia, sede Medellín.

Recibido para revisar 6 de Abril de 2005, aceptado 29 de Agosto de 2005, versión final 18 de Octubre de 2005

RESUMEN: La Arquitectura Orientada a Modelos (MDA) es la propuesta de refinamiento de la OMG orientada a la generación automática de código a partir de los Modelos UML de Sistemas Independientes de la Plataforma de Implementación. En este trabajo se presenta una metodología para transformar el Modelo de Clases UML a un Modelo UML Dependiente de la Plataforma Oracle9i®, siguiendo los lineamientos básicos presentados por esta arquitectura y utilizando a UML como lenguaje de modelado a través de todos los pasos de dicha transformación.
Inicialmente las reglas de transformación del Modelo de Clases de UML al Modelo Objeto-Relacional soportado por Oracle9i® son recopiladas en Español y adaptadas a nivel de metamodelo, para lo cual fue necesario elaborar un metamodelo simplificado de la plataforma Oracle9i®. Este conjunto de reglas se hace automatizable al expresarlas en un formalismo lógico, que sea fácilmente ejecutable por una herramienta CASE que soporte un lenguaje formal. Finalmente, se aplican las reglas de refinamiento formalizadas al Modelo de Clases de un Caso Práctico de estudio obteniendo como resultado, un Modelo UML instancia del Metamodelo de la Plataforma Oracle9i®. Los aspectos del Modelo de Clases en los que se hace énfasis en la transformación son las invariantes y reglas de derivación de atributos definidas en el lenguaje formal OCL, así como las relaciones de asociación, composición y generalización entre Clases.

PALABRAS CLAVE: Ingeniería del Software, MDA, Refinamiento, UML 2.0, OCL, Oracle9i®, herramientas CASE, formalismo lógico y Metamodelos.

ABSTRACT: Model Driven Architecture (MDA) is the OMG refinement proposal directed to the automatic code generation from UML implementation platform independent models. This work presents a methodology for transforming UML Class Model to UML Platform Dependent Model for Oracle9i®, following the basic ideas proposed by MDA and using the UML language as the modeling language in the transformation process.
Initially, transformation rules from UML class model to the relational-object model supported by Oracle9i® are collected in spanish and adapted to metamodel level; to achieve it, it was necessary to elaborate a simplified Oracle9i® platform metamodel. This set of rules becomes automatizable when is expressed in a logical formalism, that is expected to be executed by a supporting formal language CASE tool. Finally, the formalized refinement rules are applied to UML class model from a practical study case, obtaining as a result an UML Model instance of Oracle9i® platform metamodel. Class Model aspects in which emphasize the transformation are the invariants and derivation rules of attributes defined in the OCL formal language, as well as the association, composition and generalization relationships between classes.

KEY WORDS: Software Engineering, MDA, Refinement, UML 2.0, OCL, Oracle9i®, CASE tools, Logic formalism and Metamodels.

1. INTRODUCCIÓN

La Ingeniería de Software concibe actualmente el proceso de desarrollo de sistemas como una construcción sucesiva de modelos que inicialmente describen el dominio de aplicación hasta llegar a modelos de la solución en los que se incluyen aspectos técnicos de la plataforma de implementación elegida. En este proceso, los modelos obtenidos al final de cada fase del desarrollo alcanzan un mayor nivel de detalle que los construidos en la fase anterior. Es así como el desarrollo de sistemas informáticos se entiende como un proceso de refinamiento sucesivo de modelos, definiendo refinamiento como "una descripción detallada de algo que conforma a otra descripción (su abstracción)" [1].

Los Modelos del Sistema producto de las fases de definición y análisis, en particular, deben ser refinados para incorporarles los elementos propios de la plataforma tecnológica de implantación durante la fase de diseño.

La propuesta de refinamiento del Object Management Group (OMG) [2] es la "Arquitectura Orientada a Modelos", o MDA, que plantea tres aspectos fundamentales:

- Separación de la especificación de los requisitos de un sistema de los aspectos técnicos asociados con su implementación en una plataforma particular. En otras palabras, se pasa de uno o varios Modelos del Sistema "independientes de la plataforma de implementación", obtenidos en la fase de análisis, a un Modelo "dependiente de la plataforma de implementación" elegida, como producto de la fase de diseño.
- Uso de UML como lenguaje único de modelamiento a través de todo el proceso de refinamiento de modelos. Así, tanto los modelos independientes de la plataforma como los dependientes de la plataforma estarán especificados en este lenguaje.
- Formalización de las reglas de transformación del modelo independiente de la plataforma al modelo dependiente de la plataforma para su posterior automatización.

Esta propuesta busca la interoperabilidad entre productos de software, que podrían ejecutarse en cualquier plataforma a la que se pueda efectuar la traducción, a pesar de estar especificados en un lenguaje universal de modelado, diferente del lenguaje de codificación de una plataforma específica.

El Proyecto de Investigación financiado por Colciencias: "Extensiones en Herramientas CASE con énfasis en Formalismos y Reutilización" tiene como fin la incorporación de mecanismos de refinamiento en la herramienta CASE en desarrollo AR_2CA [3] por parte del Grupo de Investigación en Ingeniería de Software de la Universidad EAFIT, así como la identificación de los elementos necesarios para incorporar reglas de refinamiento de modelos en el UN-MetaCASE [4]. En este artículo se presenta un trabajo basado en la propuesta de refinamiento MDA del OMG, que pretende sentar las bases para automatizar la transformación de un Modelo de Clases UML, independiente de la plataforma, a un Modelo UML Dependiente de la Plataforma Oracle9i®.

La estructura del artículo es la siguiente: en la sección 2 se presenta una descripción detallada de la metodología a seguir para llevar a cabo la transformación del Modelo

de Clases UML al Modelo UML Dependiente de la Plataforma Oracle9i®; los aspectos relevantes del Metamodelo del Modelo de Clases de UML2.0 utilizado se presentan en la sección 3; el desarrollo del Metamodelo Simplificado de la Plataforma Oracle9i® se presenta en la sección 4; en la sección 5 se presenta la adaptación de las reglas de transformación a nivel de Metamodelos y la aplicación al Caso de estudio y en la sección 6 se presentan las conclusiones y el trabajo futuro.

2. METODOLOGÍA

Las reglas de transformación o refinamiento de modelos se especifican a nivel de Metamodelos, tal como se muestra en la Figura 1, no sólo para hacerlas más genéricas, sino también por los siguientes motivos:

- El UN-MetaCASE, actualmente en desarrollo del Grupo de Informática de la Escuela de Sistemas [4], permite especificar los metamodelos, y expresar las reglas de refinamiento a este nivel haciendo uso de un lenguaje formal.
- El lenguaje de modelamiento que soporta la herramienta CASE AR_2CA [3] es UML y, ya que las reglas de consistencia de Modelos se definen a nivel del Metamodelo de UML, resulta natural optar por definir las reglas de refinamiento a este mismo nivel, para aprovechar la implementación en Java del Metamodelo de Clases UML versión 2.0 en ambos sentidos.

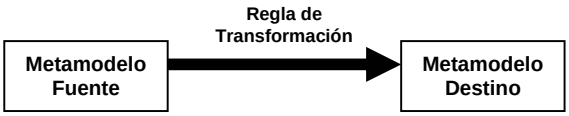

Figura 1. Esquema de Transformación a nivel de Metamodelos.
Figure 1. Transformation Scheme at Metamodel Level

Se tiene entonces como Metamodelo Fuente el Metamodelo del Modelo de Clases de UML 2.0, especificado por la OMG, y como Metamodelo Destino u Objetivo una versión Simplificada del Metamodelo de la Plataforma Oracle9i®, elaborado específicamente para esta propuesta.

La metodología a seguir para la transformación de modelos definiendo las reglas a nivel de metamodelo será la siguiente:

1. Recolección en lenguaje natural de las reglas de transformación del Modelo de Clases UML al Modelo Objeto-Relacional de Oracle9i®, a partir de diferentes fuentes.
2. Estudio y simplificación del Metamodelo Fuente: Metamodelo del Modelo de Clases de UML2.0, extrayendo sus características más recurrentes en los diagramas construidos por los analistas de software como los atributos derivados y las relaciones de asociación, composición y generalización entre Clases.
3. Diseño de un Metamodelo simplificado del Sistema Manejador de Bases de Datos Oracle9i® en el que se especifican las características objeto – relacionales de las que se saca provecho en la transformación de modelos.
4. Selección de un conjunto básico de reglas verbales para transformar el Modelo de Clases UML al Modelo Objeto-Relacional en las que se contemplan aspectos del Modelo de Clases como atributos derivados y relaciones de asociación y generalización para adaptarlas a nivel del Metamodelo.
5. Formalización y aplicación de las reglas seleccionadas al caso práctico de estudio. Para estas actividades se toma como entrada una instancia del Metamodelo de Clases de UML, y como salida, se obtiene una instancia del Metamodelo de Oracle9i® considerablemente cercana a su implementación en dicha plataforma, expresada también en UML.

En la sección siguiente se introducen los aspectos fundamentales del Metamodelo de UML, con el fin de iniciar el proceso de transformación.

3. METAMODELO DE UML

UML es el lenguaje unificado de modelamiento propuesto por el OMG para la especificación de sistemas informáticos [5]. Este lenguaje de propósito general, que puede ser utilizado para la descripción de sistemas de diversos dominios, provee simultáneamente mecanismos de extensión, para que pueda adaptarse de manera óptima a un área de aplicación particular.

Ahora, los metamodelos son modelos que especifican la estructura, semántica y restricciones de una familia de modelos [6]. Por ejemplo, el Metamodelo del Diagrama de Clases UML especifica todos los elementos que puede contener este Diagrama (Clases, Operaciones, Clases de Asociación) y las relaciones que se pueden establecer entre ellos (Asociaciones, Composiciones y Generalizaciones). El Metamodelo de UML está especificado en MOF (Meta Object Facility) [7], que corresponde al lenguaje de metamodelamiento estándar del OMG. Este metamodelo provee mecanismos de extensión que permiten añadir elementos (como clases y relaciones) si se considera necesario; además, esos mecanismos permiten el uso de perfiles UML que brindan la posibilidad de extender un metamodelo existente con construcciones propias de un dominio particular sin modificar el Metamodelo original de UML [8].

La especificación de la versión más reciente de UML (versión 2.0) incluye el Metamodelo correspondiente a varios de los diagramas que se pueden representar con este lenguaje como son: El Diagrama de Clases, de Transición de Estados y de Actividades [5].

Este artículo se centra en el Metamodelo del Diagrama de Clases, siendo éste último el Modelo Independiente de la plataforma que se va a transformar en un modelo Dependiente de la Plataforma Oracle9i®. El Metamodelo del diagrama de Clases gira alrededor de la Metaclase *Class* que describe las características que debe poseer una clase de este Diagrama. En la Figura 2 se presenta la estructura de herencia de la Metaclase UML *Class*, y en la Figura 3 todas las relaciones con otras Metaclases del Metamodelo del Diagrama de Clases.

Dada la complejidad del Metamodelo del diagrama de Clases de UML, para este trabajo se toma como Metamodelo Fuente una versión simplificada que incorpora sólo los elementos necesarios para describir la transformación al Modelo Objeto-Relacional. En particular, se describen las Clases, los Atributos, las Operaciones y las diferentes relaciones que se pueden establecer entre las Clases: generalización, asociación, agregación y composición; además de la definición de invariantes o restricciones sobre los Modelos de Clases [5].

En la Tabla 1 se describen los atributos de la Metaclase Class que se tienen en cuenta para la transformación al Modelo Dependiente de la Plataforma Oracle9i® y en la Figura 4 se presenta el Metamodelo Simplificado para la Clase UML Class que los contiene.

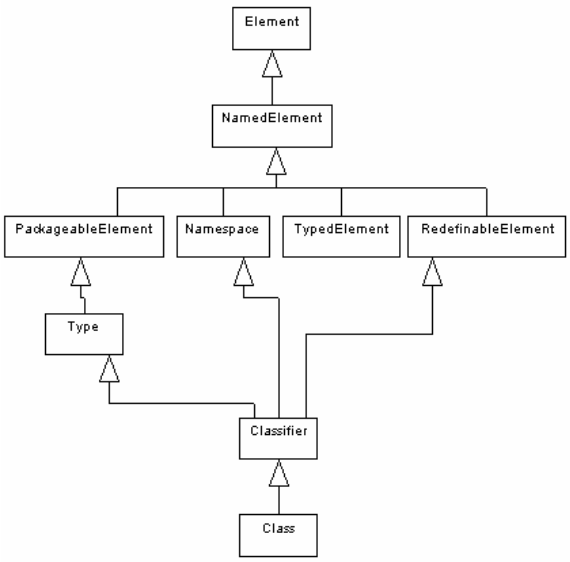

Figura 2. Estructura de Herencia de la Metaclase Class
Figure 2. Inheritance Structure of Metaclass Class.

4. METAMODELO DE Oracle9i®

La especificación de las reglas de transformación de un Modelo de Clases Independiente de la plataforma, a un Modelo Dependiente de esta plataforma, exige la utilización de un Metamodelo que especifique un conjunto de características básicas soportadas por la plataforma; en este artículo se emplea Oracle9i® [9] como Metamodelo de llegada para la aplicación de las reglas de refinamiento. Este Metamodelo representa adecuadamente las características de la plataforma de las que se desea tomar ventaja en el proceso de traducción. Las características objeto-relacionales más relevantes que se describen en el metamodelo simplificado de Oracle9i® son:

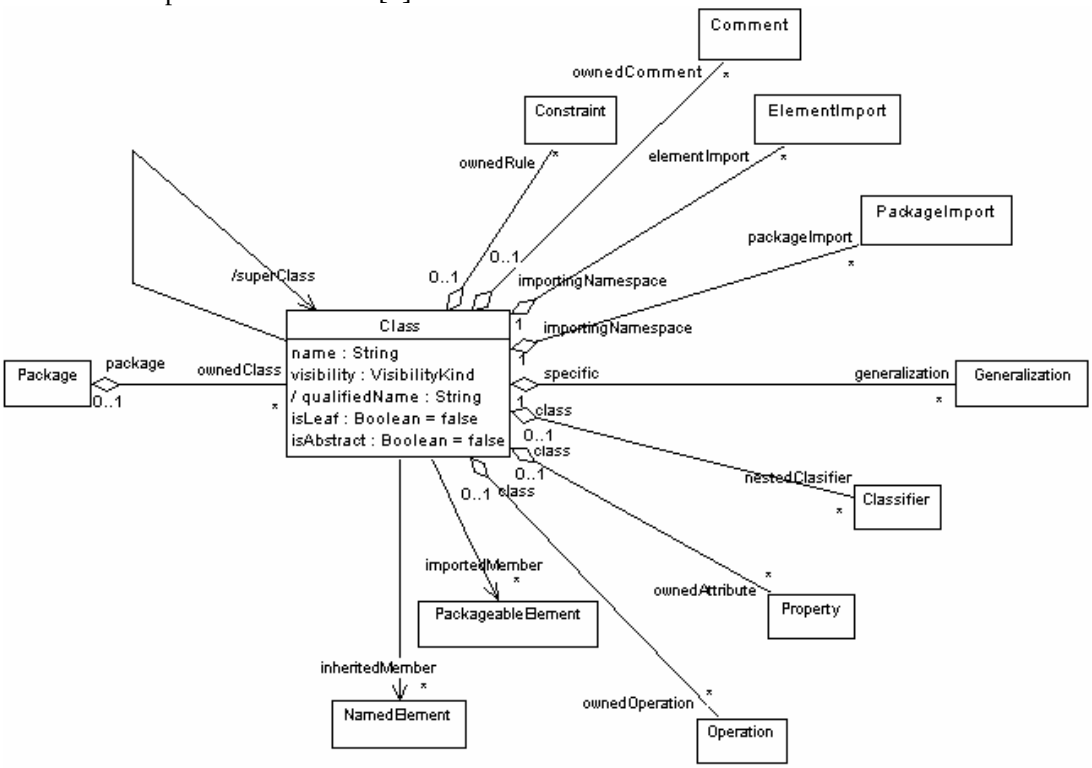

Figura 3. Relaciones de la Metaclase UML Class
Figure 3. Relationships of UML Metaclass Class

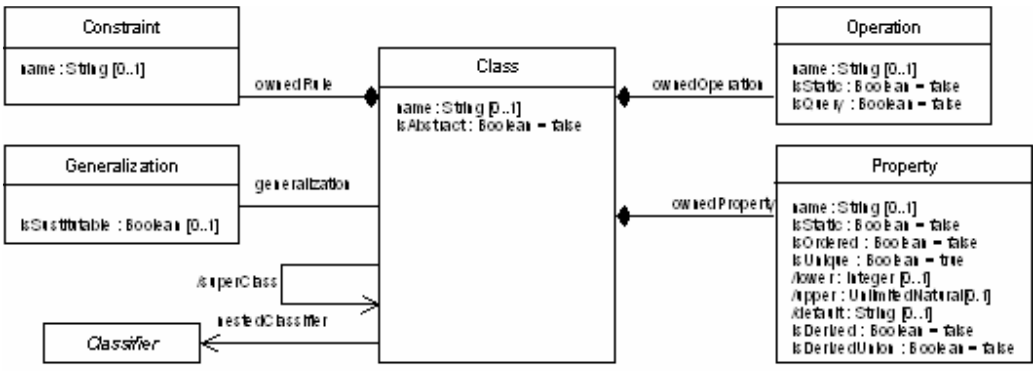

Figura 4. Metamodelo Simplificado de la Metaclase UML Class
Figure 4. Simplified Metamodel of UML Metaclass Class

- Tipos definidos por el usuario: El usuario puede definir sus tipos de datos complejos y asignarlos a columnas o tablas dentro del esquema de la base de datos.
- Generalización entre Tipos: Definición de relaciones de generalización entre tipos creados por el usuario, de tal manera que el tipo específico hereda todos los atributos y funciones definidas para el tipo general, tal como lo establece la teoría de orientación a objetos.
- Tipos Colección: Se tiene nuevos tipos de datos que permiten almacenar varios elementos de un tipo dado, bien sea predefinido dentro de Oracle o creado por el usuario. Estos tipos de datos son los arreglos o Varray y las Tablas Anidadas. Esta característica en particular, hace posible la definición de atributos multivaluados y se convierte en una opción a tener en cuenta al momento de decidir la implementación de asociaciones entre Clases con cardinalidad muchos.

Tabla 1. Atributos seleccionados de la Metaclase UML Class
Table 1. Selected Attributes of UML Metaclass Class

Atributo	Descripción
name:String [0..1]	Nombre del elemento nombrable.
isAbstract : Boolean[1]	Si es verdadero, indica que el clasificador no corresponde a una declaración completa y no se instancia. Valor por defecto *falso.*
Atributo que representa la	**Descripción**

En el Diagrama de la Figura 5 se describe una tabla como una asociación lógica de atributos o columnas, que posee un nombre y es de tipo abstracto.

Ahora, esta metaclase Tabla se especializa en las metaclases Vista y Tabla Persistente. La primera hace referencia a un elemento de Oracle9i® que agrupa un conjunto de atributos de una o varias tablas, mientras que una Tabla Persistente implica una agrupación

asociación	
ownedRule : Constraint[*]	Especifica el conjunto de restricciones poseídas por el espacio de nombres.
Generalization : Generalization[*]	Especifica las relaciones de generalización para el Clasificador.
nestedClassifier:Classifier[*]	Referencia todos los clasificadores que son definidos (anidados) dentro de la Clase.
ownedAttribute:Property[*]	Atributos poseídos por la clase. La asociación es ordenada.
ownedOperation:Operation[*]	Operaciones poseídas por la clase. La asociación es ordenada.
/ superClass: Class[*]	Especifica las superclases de la Clase. Redefine *Classifier::general*

- Referenciación de Tipos: El tipo de una columna de una tabla puede ser una referencia a un tipo creado por el usuario asociado a otra tabla de la misma base de Datos.

El Metamodelo simplificado de Oracle9i® que se elaboró para este trabajo está conformado por un Diagrama de Clases que especifica sus componentes fundamentales, tales como Tablas, Columnas, Tipos y Vistas, entre otros. El Diagrama, que se muestra en la Figura 5, va acompañado por una explicación relativa a sus Metaclases y sus atributos siguiendo el mismo esquema de definición del Metamodelo de UML 2.0. Las reglas de consistencia de la especificación Oracle9i® se pudieron expresar por medio del (Meta) diagrama de clases, definiéndolas en el lenguaje OCL [10] como restricciones.

no sólo lógica sino también física del conjunto de atributos comunes de una entidad del dominio, es decir, exige almacenamiento permanente dentro de la base de datos. Una instancia de la Metaclase Tabla Persistente posee un conjunto de atributos o columnas (*atributo_tabla*) y también puede tener restricciones asociadas (*restricción_tabla*) a uno o varios de sus atributos, que corresponden a sus reglas de cálculo o derivación.

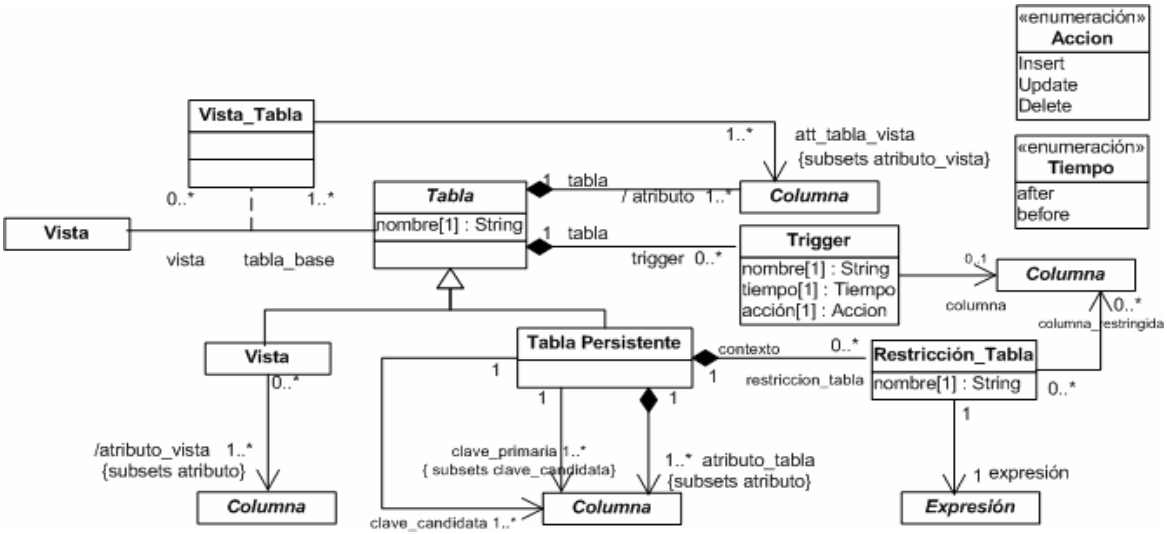

Figura 5. Diagrama de Tablas Oracle9i®
Figure 5. Oracle9i® Tables Diagram

Restricción
Las columnas que corresponden al valor del atributo *clave_candidata* de la Metaclase Tabla Persistente deben ser únicas y obligatorias.
context Tabla Persistente **inv:**
self.clave_candidata->forAll(c I c.obligatorio and c.unico)

5. ADAPTACIÓN Y APLICACIÓN DE LAS REGLAS DE TRANSFORMACIÓN

Las reglas de transformación del Modelo de Clases UML a una especificación en Oracle9i® inicialmente fueron planteadas verbalmente con base en las planteadas en otros trabajos [11], [12], [13], [14], [15] y [16]. Una vez planteadas, estas reglas se deben expresar a nivel meta de tal forma que, al aplicarlas sobre cualquier Modelo de Clases, instancia del Metamodelo de Clases de UML, dicho modelo se transforme en una instancia del Metamodelo de Oracle9i®. En la Figura 6 se presenta un esquema general de la transformación.

La estrategia seguida para especificar en el nivel meta las reglas de transformación, consistió en seleccionar un pequeño subconjunto de ellas y aplicarlas primero manualmente al Modelo de Clases del Caso de estudio, visto como una instancia del metamodelo UML, para transformarlo en una especificación Oracle9i®, vista como una instancia de su metamodelo.

Este ejercicio permite identificar:
(1) Correspondencias entre las Clases del Metamodelo Fuente y Destino.
(2) Atributos de las Metaclases Fuente cuyos valores determinan los valores de los atributos de las clases destino correspondientes.

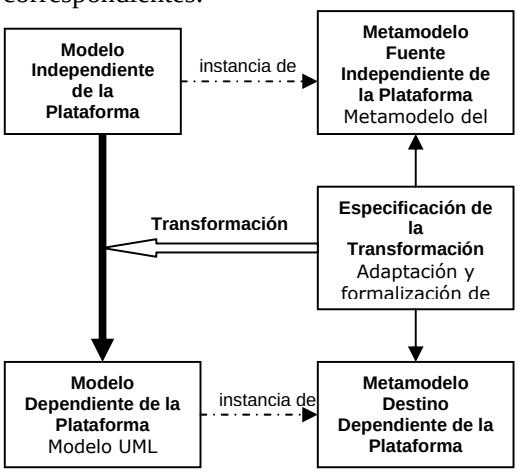

Figura 6. Esquema de la Transformación de Modelos para el Caso de Estudio
Figure 6. Model Transformation Scheme for the Case Study

A partir de estos dos criterios se pueden adaptar las reglas de transformación llevándolas al nivel Meta y finalmente expresarlas en un formalismo lógico.
Las reglas seleccionadas se orientan a la transformación de cinco elementos básicos del Modelo de Clases de UML:

- Clases
- Invariantes
- Atributos Derivados
- Asociaciones, incluyendo la relación de composición
- Generalización

En la Figura 7 se ilustra la Clase UML "Materia" que hace parte del Modelo de Clases del Caso de Estudio que corresponde al Sistema de Registro de asignaturas por parte de los estudiantes de una Universidad.

La correspondencia entre cada elemento de esta porción del Modelo de Clases y las clases del Metamodelo de UML2.0 de la que son instancias se presentan gráficamente en el Diagrama de Objetos de la Clase UML Materia (véase Figura 8). Por ejemplo, de acuerdo con la Figura 7 la Clase UML Materia posee cuatro atributos: código, nivel, clasificable y reemplazada; cada uno de ellos corresponde a una instancia de la Metaclase UML *Property* tal como se observa en el Diagrama de Objetos correspondiente.

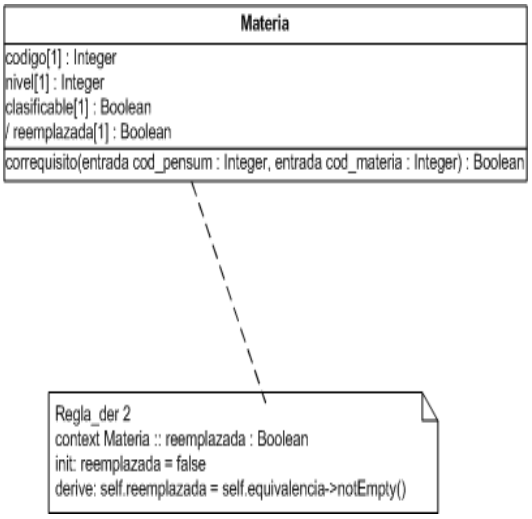

Figura 7. Clase UML Materia
Figure 7. UML Class Subject

A continuación se describen algunas de las reglas que fueron aplicadas a esta porción del Modelo de Clases para transformarlo en una instancia del metamodelo de la Plataforma Oracle9i®, junto con el planteamiento detallado de su proceso de adaptación a nivel de Metamodelo y posterior formalización.

Transformación de Clases a Tablas

Regla Verbal: *Todas las clases no abstractas de un Modelo de Clases se convierten en tablas de un Modelo Objeto-Relacional.*

Estudiando los atributos de la Metaclase UML Class (véase Tabla 1) se encuentra que, para transformar una instancia de la Metaclase UML Class a una instancia de la Metaclase Oracle Tabla Persistente (véase Figura 5), la regla de transformación debe verificar que:

1. El atributo *isAbstract* de la instancia de la clase a transformar debe ser falso.
2. La instancia de la Metaclase UML *Class* que se quiere transformar no participa en ninguna relación de generalización, ni como Clase general ni como especialización de otra clase, porque la herencia entre Clases en Oracle9i®, en el marco de este trabajo, se maneja tomando ventaja de la generalización entre tipos creados por el usuario. Por esta razón el atributo *generalization* de la Metaclase UML Class no puede tener asociado ningún valor.

A partir del Metamodelo Simplificado de la Figura 4 para la instancia de la Metaclase UML Class con nombre Materia se obtiene la Tabla 2.

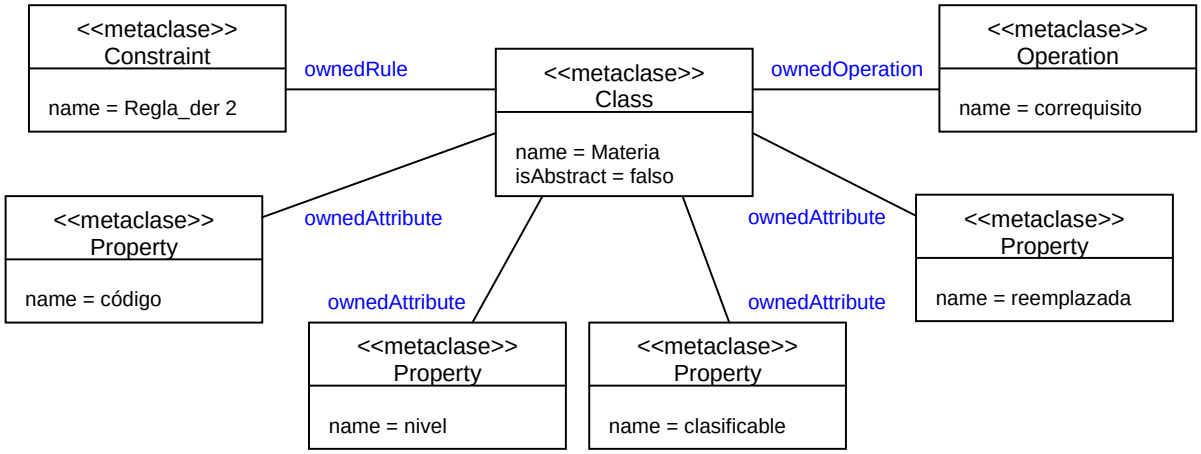

Figura 8. Diagrama de Objetos de la Clase UML Materia
Figure 8. Object Diagram for UML Class Subject

Tabla 2. Evaluación de la Clase UML *Class*
Table 2. Evaluation of UML Class *Class*

Atributos de la Metaclase UML Class	Instancia de la Metaclase UML Class
Name	Materia
isAbstract	Falso
Generalization	

La Clase UML "Materia" cumple con las condiciones 1 y 2, transformándose en una instancia de la Metaclase Oracle Tabla Persistente (véase Figura 5), conservando el mismo nombre, como se indica en la Tabla 3.

Tabla 3. Evaluación de la Clase Tabla Persistente.
Table 3. Evaluation of Class Persistent Table.

Atributos de la Metaclase Oracle Tabla Persistente	Instancia de la Metaclase Oracle Tabla Persistente
Nombre	Materia

La aplicación manual de esta regla de transformación permite identificar qué atributos de la instancia de la Metaclase UML Class son relevantes para la transformación y llegar a la siguiente expresión de esta regla a nivel de Metamodelo:

Toda instancia de la Metaclase UML Class, que no participe en ninguna relación de generalización se transforma en una instancia de la Metaclase Oracle Tabla Persistente si su atributo isAbstract es falso, conservando el mismo nombre.

En términos de Diagramas de Objetos la aplicación de esta regla a la Clase UML Materia arroja como resultado una instancia de la Metaclase Oracle Tabla Persistente llamada "Materia" tal como se observa en la Figura 9.

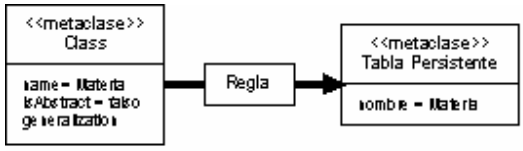

Figura 9. Correspondencia entre las Metaclases y las Instancias.
Figure 9. Correspondence between Metaclasses and Instances.

Finalmente, una vez expresada la regla a nivel de Metamodelo se lleva a un formalismo lógico equivalente haciendo uso de la teoría de conjuntos:

$$\text{Re}\,gla:\forall(c\in MetaclaseUML\_Class)$$
$$[(\neg c.isAbstract)\wedge(c.generalization=\varnothing)\Rightarrow$$
$$\exists(t\in MetaclaseORACLE\_Tabla\ Persistente$$
$$[t.nombre=c.name])$$

Transformación de Atributos a Columnas

Regla Verbal: *Los atributos de una Clase se transforman en columnas de la Tabla Correspondiente en el Modelo Objeto-Relacional.*

Es importante mencionar que en esta nueva especificación de UML las asociaciones entre clases se establecen a partir de atributos que pertenecen a las Clases o a las Asociaciones según la navegabilidad de las mismas, es decir, el concepto de rol en los extremos de las asociaciones se modela igual que un atributo. En consecuencia, para transformar una propiedad o atributo de una Clase en una columna de una Tabla en el Modelo Objeto – Relacional es necesario verificar que la propiedad pertenezca a una Clase y no a una asociación entre clases.

A nivel de Metamodelo esto implica que el atributo *association* de la Metaclase Property no toma ningún valor, es decir, la propiedad debe pertenecer a la Clase y no participar en ninguna relación de asociación.
En el ámbito de Oracle9i® los atributos de la Metaclase Oracle "Atributo Estructural" a la que se llevan las propiedades que no participan en ninguna asociación se presentan en la Tabla.

Partiendo ahora del Metamodelo de UML y de Oracle9i®, en la Tabla 5 se presentan los valores de los atributos relevantes de la Metaclase UML Property para las instancias *código* y *reemplazada* (véase Figura 4) así como los valores de los atributos de las instancias de la Metaclase Oracle *Atributo Estructural* a la que se llega por la aplicación de esta regla.

Tabla 4. Atributos de la Metaclase Oracle "Tabla Persistente"
Table 4. Attributes of the Oracle Metaclass "Persistent Table"

Atributo	Descripción
nombre: String [0..1]	Especifica el nombre del Atributo Estructural.
obligatorio : Boolean[1]	Si es verdadero, indica que el atributo estructural es obligatorio.
único : Boolean[1]	Si es verdadero, indica que el atributo estructural toma un único valor.
Atributo que representa la asociación	**Descripción**
tabla : Tabla[1]	Referencia la tabla a la que pertenece el atributo estructural.
Regla_der : Expresión[0..1]	Especifica la regla de calculo del valor del atributo estructural en caso de que sea derivado

Tabla 5. Atributos de las Metaclases UML "Property" y Oracle "Atributo Estructural"
Table 5. Attributes of Metaclasses "Property" and "Structural Attribute"

Atributos de la Metaclase UML *Property*	**Instancias de la Metaclase UML *Property***	
Name	código	reemplazada
Association		
Class	→ Instancia de la Metaclase UML Class Materia	→ Instancia de la Metaclase UML Class Materia
Atributos de la Metaclase Oracle Atributo Estructural	**Instancias de la Metaclase Oracle *Atributo Estructural***	
Nombre	código	reemplazada
Tabla	→ Instancia de la Metaclase Oracle Tabla Persistente Materia	→ Instancia de la Metaclase Oracle Tabla Persistente Materia

A nivel de Metamodelo la regla se expresaría así:

Todas las instancias de la Metaclase UML Property que pertenecen a una Clase y no participan en ninguna relación de asociación se convierten en instancias de la Metaclase Oracle Atributo Estructural pertenecientes a la instancia de la Metaclase Oracle Tabla Persistente, que mapea la clase a la que pertenecen, y conservan el mismo nombre.

La especificación formal de esta regla exige el uso de pre y postcondiciones como restricciones que se deben cumplir antes y después de la aplicación de regla respectivamente.

$$\begin{aligned} &Pre: \exists (t \in MetaclaseORACLE\_Tabla\ Persistente) \\ &\quad \wedge\ \exists (c \in MetaclaseUML\_Class) \\ &\quad t = f(c) \end{aligned}$$

$$\begin{aligned} &Regla: \forall (p \in MetaclaseUML\_Property)[((p \in c.ownedAttribute) \\ &\wedge (p.association = \varnothing)) \Rightarrow \\ &\exists (att\_est \in MetaclaseORACLE\_Atributo\ Estructural \\ &[(att\_est \in t.atributo\_tabla) \wedge (att\_est.nombre = p.name)]) \end{aligned}$$

En la precondición de esta regla se utiliza la notación *t=f(c)* para indicar que *t* es una instancia de una Clase del Metamodelo de Oracle que se obtiene a partir de la aplicación

de la función de mapeo f a una instancia de una Metaclase UML c. Esta función de mapeo estará conformada por una o varias reglas de transformación.

Una vez se conoce el procedimiento de adaptación de las reglas a nivel de Metamodelo la siguiente regla se enunciará a nivel de Metamodelo junto con su equivalente en lógica:

Toda restricción asociada a una Clase UML corresponde a una instancia de la Metaclase UML Constraint (véase Figura 4) y se transforma en una instancia de la Metaclase Oracle Restricción_Tabla (véase Figura 5) asociada a la instancia de Tabla Persistente correspondiente, conservando el mismo nombre.

$$\mathrm{Pr}\,e: \exists\,(t \in MetaclaseORACLE\_Tabla\ Persistente) \wedge \exists\,(c \in MetaclaseUML\_Class\,) / t = f(c)$$

$$\mathrm{Re}\,gla: \forall(\,r \in MetaclaseUML\_Constra\,\mathrm{int}) [(r \in c.ownedRule) \Rightarrow \exists\,(rt \in MetaclaseORACLE\_\mathrm{Re}\,striccion\ Tabla [(rt \in t.restriccion\_tabla) \wedge (rt.nombre = r.name)\,)]]$$

Como resultado de la aplicación de estas reglas a la porción de Modelo de Clases presentada en la Figura 4 se obtienen: el diagrama de objetos (Figura 10), donde se observa la correspondencia de cada elemento de este Diagrama con las Clases del Metamodelo de Oracle9i® que instancian, y su correspondiente Diagrama UML Dependiente de la Plataforma Oracle9i® (Figura 11).

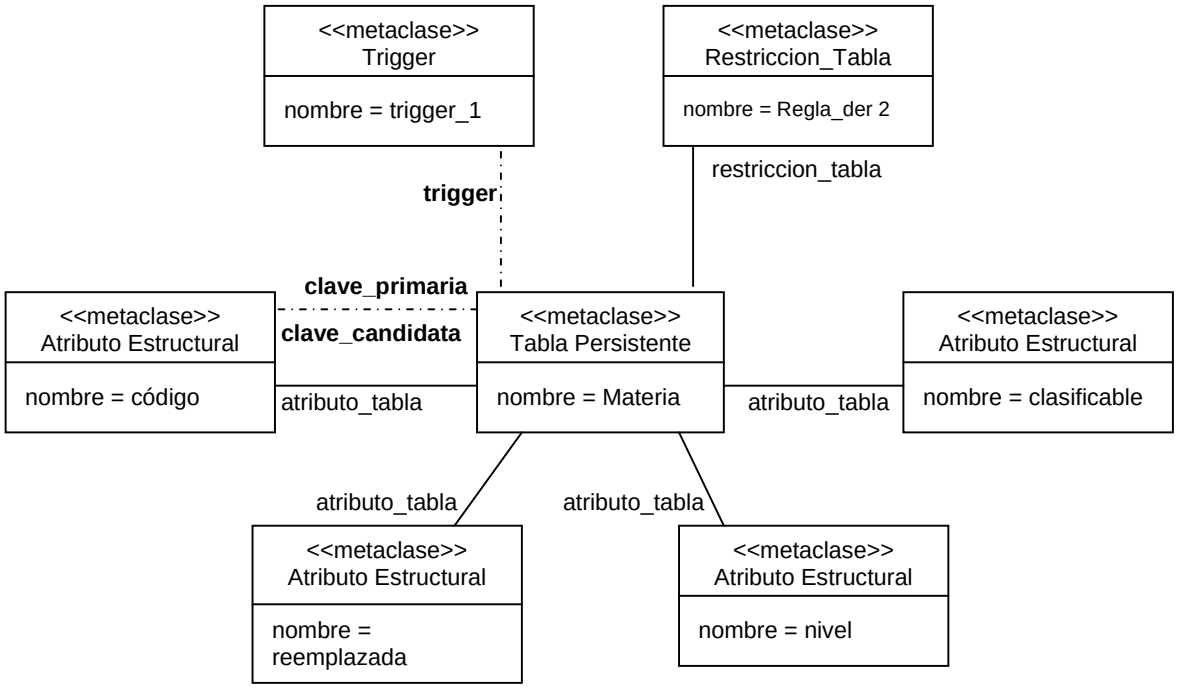

Figura 10. Diagrama de Objetos de la Metaclase Oracle Tabla Persistente
Figure 10. Object Diagram of Oracle Metaclass Persistent Table

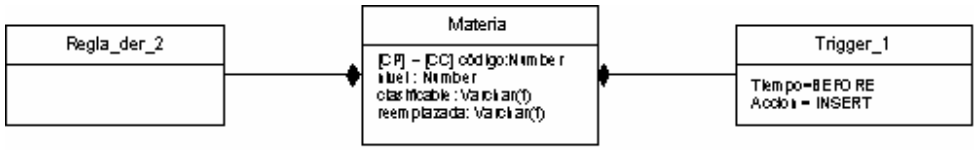

Figura 11. Diagrama UML Dependiente de la Plataforma Oracle9i para la Tabla Materia
Figure 11. UML Oracle9i Platform Dependient Diagram of the Table Subject

En el Diagrama UML Dependiente de la Plataforma Oracle9i® (Figura 11) Los símbolos [CP] y [CC] denotan a una columna de una tabla en Oracle como clave primaria y clave candidata respectivamente y no fueron especificados dentro del Metamodelo como notación sino elegidos como convención en el contexto del trabajo. Posteriormente en el Metamodelo de Oracle9i® pueden añadirse aspectos de notación además de los estructurales presentados en la Sección 4.

En la Figura 10 los atributos de la Metaclase Tabla Persistente cuyo valor corresponde a instancias de otras metaclases, (vinculados en asociaciones), que van acompañados por líneas punteadas son producto de las decisiones de diseño y las reglas para obtenerlas se describen de manera detallada a continuación:

Elección de la clave primaria de la Tabla Persistente

Todas las instancias de la Metaclase UML Property que pertenecen a una Clase y cumplen con ser: únicas, obligatorias, no derivadas y no participar en ninguna asociación se constituyen en claves candidatas para la instancia de Tabla Persistente que mapea la Clase correspondiente. Entre estas propiedades, que se transforman en Columnas Oracle, para cada Tabla Persistente es necesario seleccionar como mínimo una para que se constituya en su identificador o clave primaria.

Dicha decisión se deja al criterio del modelador del sistema, ya que la regla automatizable determina únicamente cuáles instancias de la Metaclase *Atributo Estructural* pueden desempeñar ese rol.

La instancia o instancias de Atributo Estructural elegidas son los valores del atributo *clave_primaria* de la Metaclase Oracle Tabla Persistente que las posee (véase Figura 5).

$$\begin{aligned}\mathrm{Pre}: 1.\ &\exists\,(t \in \mathit{MetaclaseORACLE\_Tabla\ Persistente})\\ &\wedge\ \exists\,(c \in \mathit{MetaclaseUML\_Class}\,)\ /\\ &\ t = f(c)\\ &\ 2.\ \exists\,(\mathit{att\_est} \in \mathit{MetaclaseORACLE\_Atributo\ Estructural}\ /\\ &\ \mathit{att\_est} \in t.\mathit{atributo\_tabla})\\ &\wedge\ \exists\,(p \in \mathit{MetaclaseUML\_Property}\ /\\ &\ p \in c.\mathit{ownedAttribute}\,)\ /\\ &\ \mathit{att\_est} = f(p)\end{aligned}$$

$$\begin{aligned}\mathrm{Regla}: \forall(\ p\ [\ (\ &(p.\mathit{isUnique} = \mathit{verdadero}) \wedge\\ &(p.\mathit{lower} = 1) \wedge (p.\mathit{isDerived} = \mathit{falso})\ ) \Rightarrow\\ &(\ \forall\,[\ (\mathit{att\_est}\ /\ (\mathit{att\_est} = f(p)) \Rightarrow\\ &\mathit{att\_est} \in t.\mathit{clave\_candidata}\,]\ )\\ &\wedge (t.\mathit{clave\_primaria} \subseteq t.\mathit{clave\_candidata})]\end{aligned}$$

Implementación de columnas derivadas de Oracle por medio de Triggers

La regla de derivación de una columna se implementa en Oracle por medio de un Trigger que verifica o calcula su expresión asociada y pertenece a la instancia de la Metaclase Oracle *Tabla Persistente* poseedora del atributo o columna derivada.

En la Tabla 6 se presentan los atributos de la Metaclase Oracle Trigger observados en la Figura 5.

Para este caso se asume el disparo del trigger después de insertar un registro en la tabla, con la cual se asocia a través de su atributo *tabla*.

Tabla 6. Atributos de la Metaclase Oracle Trigger
Table 6. Attributes of Oracle Metaclass Trigger

Atributo	Descripción
nombre: String [0..1]	Indica el nombre del trigger.
tiempo : Tiempo[1]	Determina el instante del tiempo en que se debe verificar la regla de integridad de la que el trigger es responsable.
acción : Acción[1]	Especifica frente a cual acción realizada sobre la tabla a la que pertenece se dispara el trigger (insert, update, delete).
Atributo que representa la asociación	**Descripción**
tabla : Tabla[1]	Referencia la tabla a la que pertenece el trigger.
columna: Columna[0..1]	Si la acción asociada con el trigger es una actualización (UPDATE), este atributo indica frente a la manipulación de cual columna se dispara el trigger.
expresión : Expresión[1]	Determina la expresión que se verifica en la ejecución del trigger

El tiempo y la acción que dispara el trigger, por tratarse de un atributo derivado, se asume que es antes de insertar un registro o actualizar el valor de dicho atributo en la instancia de Tabla Persistente; sin embargo, esta decisión exige un análisis más detallado de la regla de derivación que está por fuera del alcance del trabajo.

$$\begin{aligned}&\text{Re}\,gla:\exists(att\_est\in\\&MetaclaseORACLE\_Atributo\,Estructural/\\&att\_est.derivado=verdadero)\Rightarrow\\&\exists(tr\in MetaclaseORACLE\_Trigger/\\&((tr.tabla=att\_est.tabla)\wedge\\&(tr.\text{exp}resion=att\_estr.regla\_der)\wedge\\&(tr.tiempo=BEFORE)\wedge(tr.accion=INSERT)))\\&\wedge(att\_est.tabla.trigger=tr)\end{aligned}$$

6. CONCLUSIONES Y TRABAJO FUTURO

El estudio y aplicación de los conceptos fundamentales que soportan el refinamiento en la Arquitectura Orientada a Modelos, propuesta por el OMG, permitió determinar una estrategia adecuada para incorporar el refinamiento de modelos a las herramientas CASE UN-MetaCASE y AR_2CA que consiste en definir las reglas de mapeo a nivel de los Metamodelos Fuente y Objetivo, facilitando su formalización y automatización posterior.

Ahora, la aplicación manual de un conjunto básico de reglas al Modelo de Clases del Caso de Estudio mostró que su especificación en un formalismo lógico constituye una alternativa directa para su posterior automatización en el marco de una herramienta CASE, independientemente de la plataforma objetivo. La formalización de las reglas bien podría realizarse en OCL [10], que es un estándar más cercano al OMG y se constituye en un trabajo futuro que podría derivarse de este artículo.

La apropiación de la estrategia de refinamiento de modelos propuesta por el OMG, para llevar Modelos Generales de un dominio de aplicación a modelos del mismo dominio orientados a una plataforma dada, sugiere la posibilidad de generar código en una plataforma a partir de su definición formal, empleando para ello los metamodelos correspondientes. Así, se podría pensar a futuro en desarrollar herramientas para la generación automática de código que reciba como entradas el Metamodelo del Modelo Independiente de la plataforma y la especificación formal de la plataforma en la que se desea obtener una codificación parcial del Sistema modelado.

Por otro lado, se verificó experimentalmente en el Modelo de Clases UML del Caso de estudio, que la utilización de OCL para la definición de invariantes y reglas de derivación de atributos, además de añadir semántica al modelo, es una alternativa para facilitar su automatización o semi-automatización en una plataforma dada. En el caso de Oracle9i®, se plantea la construcción de un traductor de OCL a PL/SQL para lograr que la transformación comprenda no sólo aspectos estructurales sino también aspectos dinámicos del Modelo de Clases UML. En lo que tiene que ver con los métodos de las Clases, resultaría interesante expresarlos en OCL en términos de pre y postcondiciones para la generación parcial de código PL/SQL en Oracle9i® o en cualquier otra plataforma,

aspecto que no fue contemplado en esta etapa del trabajo.

En general, la transformación de un Modelo de Clases a un modelo dependiente de la plataforma requiere la elección de la manera como se van a implementar aspectos semánticos de modelado de UML, tales como relaciones de agregación y generalización entre clases, atributos derivados e invariantes, entre otros. Naturalmente, existen varias alternativas de implementación posible para estos aspectos y la elección de la opción más adecuada constituye un problema de optimización que será objeto de un trabajo posterior.

Como trabajos futuros a partir de esta exploración inicial de la Arquitectura Orientada a Modelos se tiene:

- Afinamiento de la formalización de las reglas de transformación para que puedan ser aplicadas automáticamente en la herramienta UN-MetaCASE.
- Desarrollo de un traductor de OCL a PL/SQL como alternativa para incorporar aspectos dinámicos del Modelo de Clases a la generación de código, mas allá de la definición del "esqueleto de los métodos", como ocurre actualmente en las herramientas CASE tradicionales.
- Aplicación de los lineamientos básicos propuestos por MDA para transformar Modelos Independientes de la Plataforma a Modelos dependientes de otras plataformas (.NET, J2EE) garantizando interoperabilidad a través de la existencia de un lenguaje único de modelado que se puede extender con terminología propia de cualquier plataforma orientada a objetos.
- Ampliación del enfoque propuesto para el refinamiento de modelos a otros modelos UML de entrada como el Diagrama de Transición de Estados y el Diagrama de Colaboración.
- Construcción de un Perfil UML como "puente" entre el Metamodelo Origen y Destino que facilite la especificación de las reglas de transformación entre modelos [8], [17].

AGRADECIMIENTOS

Este artículo se realizó dentro del marco del proyecto 479 de COLCIENCIAS denominado "Extensiones en herramientas CASE con énfasis en formalismos y reutilización"

REFERENCIAS BIBLIOGRAFICAS

[1] D'SOUZA D., WILLIS C. *"Objects, Components and Frameworks with UML: The Catalysis Approach"* . Addison Wesley, 1999

[2] Object Management Group. Disponible en: http://www.omg.org [Citado 12 de Noviembre de 2004]

[3] ANAYA, RAQUEL. *AR_2CA: Una Herramienta para la Construcción de componentes reutilizables a través de niveles de refinamiento*. Memorias de 3das Jornadas Iberoamericanas de Ingeniería de Requisitos y Ambientes de Software IDEAS2000, 2000.

[4] ARANGO, F. *Áreas y Líneas de Investigación*, Grupo de Investigación en Informática Universidad Nacional de Colombia Sede Medellín.2003

[5] *Unified Modeling Language: Superestructure versión 2.0*. OMG Final Adopted Specification,2002.

[6] MELLOR J., SCOTT K., UHL A., Y WEISE D. *"MDA Distilled: Principles of Model-Driven Architecture"*.Addison Wesley. 2002

[7] *Meta Object Facility (MOF) Specification*. OMG Document: formal/2002-04-03, 2003.

[8] FUENTES L., VALLECILLO A . , *Using UML Profiles:A case study*. Workshop on Automating Object-Oriented Software Development Methods. 2002

[9] Oracle Technology Network. Disponible en: http://www.otn.oracle.com [Citado 12 de Noviembre de 2004]

[10] *UML2.0 OCL Specification*. OMG Final Adopted Specification.200*2*

[11] DUITAMA, J. FREDDY. *"Transformación Automática de Especificaciones expresadas bajo un*

Paradigma Objetual al Esquema Relacional". Tesis de Maestría Universidad Nacional, Sede Medellín, 1998.

[12] DORSEY P. , HUDICKA R. *"Oracle 8 Diseño de Bases de Datos con UML"* . Mc Graw-Hill. México. 1999

[13] DIETRICH S. , URBAN S. *Using UML Class Diagram for a Comparative Analysis of Relational, Object-Oriented, and Object-Relational Database Mappings.* Arizona State University. 2002

[14] MOK W. Y. , PAPER D. , *On Transformations from UML Models to Object – Relational Databases.* 34th Hawaii International Conference on System Sciences.2001

[15] CHENNAMANENI R., GRANT E.S. *Comparison and Evaluation of Methodologies for Transforming UML Models to Object-Relational Databases.* University of North Dakota. 2002

[16] CAVERO J., MARCOS E. Y VELA B. *A methodological Approach for Object-Relational Database Design using UML* . Informatik- Forschung und Entwicklung. 2004

[17] BORDELEAU F., ZAMORA J. P., *Defining UML Profiles and Model Mappings in the context of MDA.* Worshop on Model-Driven Embedded Systems. Washington. 2003