UN CONJUNTO DE TRANSFORMACIONES QVT PARA EL MODELADO DE ALMACENES DE DATOS SEGUROS

Emilio Soler[1], Juan Trujillo[2], Eduardo Fernández-Medina[3] y Mario Piattini[3]

1: Departamento de Informática. Facultad de Informática. Universidad de Matanzas
e-mail: emilio.soler@umcc.cu, web: http://www.umcc.cu

2: Departamento de Lenguajes y Sistemas Informáticos. Universidad de Alicante
e-mail: jtrujillo@dlsi.ua.es, web: http://www.dlsi.ua.es

3: Grupo ALARCOS, Departamento de Tecnologías y Sistemas de Información
Centro Mixto de Investigación y Desarrollo de Software UCLM-Soluziona
Universidad de Castilla-La Mancha
Paseo de la Universidad, 4 – 13071 Ciudad Real, España
e-mail: {eduardo.fdzmedina, mario.piattini}@uclm.es, web: http://www.uclm.es

Palabras clave: Transformaciones QVT, seguridad de datos, MDA, modelado seguro multidimensional.

Resumen. *La seguridad es un aspecto crucial para el desarrollo de los Almacenes de Datos (DWs) ya que éstos contienen datos sensibles. Este hecho requiere la necesidad de especificar requisitos de seguridad y de auditoría en el modelado multidimensional, que no pueden ser directamente transformados al modelo relacional del DW. El marco estándar para el desarrollo del software Model Driven Architecture (MDA) permite definir transformaciones entre modelos mediante la propuesta Query/View/Transformations (QVT). Esta propuesta permite definir transformaciones de una manera formal, elegante e inequívoca. En este artículo utilizamos QVT para transformar al esquema lógico relacional del DW toda la información de seguridad representada en el modelo conceptual multidimensional.*

1. INTRODUCCIÓN

El modelado multidimensional (MD) es la base de los Almacenes de Datos (*Data Warehouses*, DWs), las Bases de Datos MDs, y aplicaciones de Procesamiento Analítico En-Línea (*On-Line Analytical Processing*, OLAP). Estas aplicaciones de manera conjunta constituyen un mecanismo muy poderoso para descubrir información de negocio crucial en los procesos de toma de decisiones estratégicas. Los DWs con frecuencia almacenan información histórica y agregada, extraída de múltiples, heterogéneas, autónomas y distribuidas fuentes de información, por ello, la supervivencia de las organizaciones depende de la correcta gestión, seguridad y confidencialidad de la información [1]. La

información de seguridad es un requisito serio que debe ser considerado, no como un aspecto aislado, sino como algo presente en todas las etapas del ciclo de vida de desarrollo, desde el análisis de los requerimientos hasta la implementación y mantenimiento [2]. Lo anterior justifica que es crucial especificar medidas de confidencialidad en el modelado MD y hacerlas cumplir.

En la literatura más relevante sobre el tema, podemos encontrar varias iniciativas para incluir seguridad en el diseño de los DWs [3], [4], [5] y [6], muchas de ellas están enfocadas a aspectos específicos relacionados con el control de acceso, la seguridad multinivel, aplicaciones a las bases de datos federadas, aplicaciones en herramientas comerciales, etc. Estas propuestas no consideran los aspectos de seguridad en todas las etapas del ciclo de desarrollo ni introducen seguridad en el diseño conceptual MD.

Como señalan otros autores [3], [4] y [7], la mayoría de los DWs son implementados en sistemas relacionales, pero las medidas de seguridad y los modelos de control de acceso para bases de datos no son apropiados para los DWs. A pesar de ello, existe un hueco semántico entre modelos de datos conceptuales avanzados e implementaciones relacionales o multidimensionales de cubos de datos [8].

El mencionado hueco semántico está relacionado con la propuesta MDA [9]. MDA se ha convertido en el actual estándar para el desarrollo del software, en ella todo el proceso está guiado por la transformación de modelos. Diversas propuestas han aparecido en los últimos para establecer las transformaciones MDA [10], por ello, en abril de 2002 el consorcio OMG anunció el MOF 2.0 *Query/Views/Transformations Request For Proposals* (QVT RFP), un intento para establecer un estándar que define el modo en que han de llevarse a cabo las transformaciones entre modelos cuyos lenguajes han sido definidos usando MOF.

Hay varias propuestas que tratan de integrar seguridad con la tecnología MDA [11], [12], [13] y [14], pero todas ellas están relacionados con sistemas de información, control de acceso, servicios de seguridad y aplicaciones distribuidas seguras, es decir, ninguno está relacionado con el diseño de DWs seguros. En [15] los autores proponen un marco MDA orientado al desarrollo de los DWs, pero sin considerar requisitos de seguridad.

En este trabajo proponemos un conjunto de transformaciones QVT para el modelado de los DWs seguros. Nuestro objetivo es establecer una transformación segura entre los niveles conceptual y lógico. Centraremos la atención únicamente en la transformación PIM – PSM.

El resto de este artículo se estructura como sigue. La sección 2 presenta los principales aspectos relacionados con la tecnología MDA y las transformaciones QVT. Una arquitectura MDA multidimensional segura (*Secure Multidimensional* MDA) se introduce en la sección 3, en ella definimos un *Secure Multidimensional Platform Independent Model* (SMD PIM), un *Secure Multidimensional Platform Specifict Model* (SMD PSM) y un conjunto de transformaciones QVT entre ellos. Finalmente la sección 4 presenta las principales conclusiones y el trabajo futuro inmediato.

2. MDA Y TRANSFORMACIONES QVT

En esta sección vamos a resumir las principales características de MDA y de las transformaciones QVT. Para más detalles vea [9] y [16].

Model Driven Architecture (MDA) es un nuevo paradigma para el desarrollo del software que considera como artefacto primario a los modelos. MDA parte de la bien establecida y tradicional idea de separar la especificación de la operación de un sistema de los detalles de su plataforma [9]. De este modo, MDA promueve la especificación de un Modelo Independiente de la Plataforma (PIM) que no contiene información específica para la plataforma o la tecnología que será usada para desarrollarlo. Este PIM puede ser transformado en uno o varios Modelos Específicos de Plataforma (PSM), para incluir información acerca de la tecnología específica que será usada en el desarrollo sobre una plataforma específica. Después, cada PSM es transformado en código para ser ejecutado en una plataforma y obtener el producto software final.

Para realizar las transformaciones del PIM al PSM consideramos la aproximación declarativa de QVT [16]. QVT ofrece dos notaciones (diagramática y textual) para definir la transformación. Una transformación se caracteriza por los siguientes elementos:

• Dos o más dominios: Cada dominio identifica un modelo candidato (es decir los metamodelos PIM o PSM), y un conjunto correspondiente de elementos en ese modelo por medio de patrones. Un patrón de dominio puede ser considerado una plantilla para objetos. Sus propiedades y sus asociaciones deben estar localizadas, modificados, o creadas en un modelo candidato para satisfacer la relación.

• Un dominio de relación: Especifica el tipo de relación entre dominios, este puede ser marcado como *checkonly* (etiquetado como C) o como *enforced* (etiquetado como E). Un dominio *checkonly* es verificado para ver si existe una correspondencia válida en el modelo que satisfaga la relación. Mientras que para un dominio que es *enforced*, cuando el patrón del dominio no corresponde, los elementos del modelo pueden ser creados, suprimidos o modificados en el modelo destino para satisfacer la relación. Además, para cada dominio el nombre de su metamodelo subyacente es especificado.

• La cláusula *when*: especifica las condiciones que deben ser satisfechas para ejecutar la transformación (es decir, pre-condición).

• La cláusula *where*: especifica las condiciones que deben ser satisfechas por todos los elementos del modelo participantes en la relación (es decir, post-condición).

• Una transformación contiene dos tipos de relaciones: *top-level* y *non-top-level*. La ejecución de una transformación requiere el cumplimiento de todas las relaciones *top-level*, mientras que las relaciones *non-top-level* son requeridas para su cumplimiento solo cuando son invocadas directamente o transitivamente desde la cláusula *where* de otra relación.

3. SMD MDA: MDA MULTIDIMENSIONAL SEGURO

Las reglas de seguridad y auditoria especificadas a nivel conceptual en el modelado multidimensional de DW no pueden ser representadas directamente en el modelo

relacional [17], por consiguiente tenemos, un hueco semántico entre los esquemas conceptual y lógico [8]. En este apartado usamos la aproximación MDA para cubrir el mencionado hueco semántico mediante las transformaciones QVT. En las subsecciones 3.1 y 3.2 introduciremos los términos: *Secure Multidimensional* PIM (SMD PIM) y *Secure Multidimensional* PSM (SMD PSM) respectivamente.
En la Figura 1 ilustramos mediante un diagrama la arquitectura MDA Multidimensional Segura. A la izquierda aparece el *Secure Multidimensional* (SMD) esquema conceptual, es decir, el SMD PIM. Mediante la transformación T_1 obtenemos el esquema relacional lógico, es decir, el SMD PSM, representado en el centro de la Figura 1. Si elegimos un SGBD que implemente aspectos de seguridad, entonces el SMD PSM se transforma según T2 en código para la plataforma destino. Este código lo llamamos Código Multidimensional Seguro (*Secure Multidimensional Code*, SMD *Code*). Observe como la restricción de seguridad (representada mediante una nota de UML) es transformada del nivel conceptual al nivel lógico mediante T_1, y más tarde transformada en código mediante la transformación T_2.

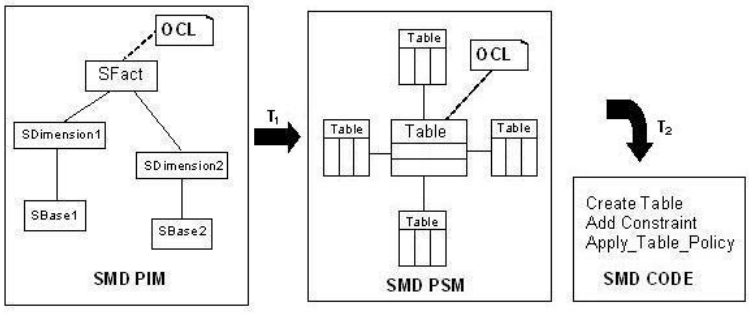

Figura 1. Obteniendo el SMD PSM y el SMD Code a partir del SMD PIM

3.1. Definición del SMD PIM

El *profile* UML presentado en [18] llamado *Secure Data Warehouse* (SECDW), nos permite representar los principales requisitos de seguridad en el modelado conceptual de DWs. La Figura 2 representa al metamodelo SECDW, hemos omitido algunos atributos para hacer más comprensible el metamodelo.
Como modelamos requisitos de seguridad en este PIM, lo hemos llamado SMD PIM (*Secure Multidimensional* PIM). Las principales características de este metamodelo son las relaciones *many-to-many* entre un hecho y una dimensión específica. Las dimensiones degeneradas, la clasificación múltiple y los caminos alternativos de jerarquía, así como las jerarquías no estrictas y completas. La clase llamada *UserProfile* contendrá información de todos los usuarios con derecho de acceso al modelo multidimensional.
El metamodelo también permite representar los principales aspectos de seguridad para el modelado multidimensional de DWs. Para cada elemento del metamodelo (*SecureFact*,

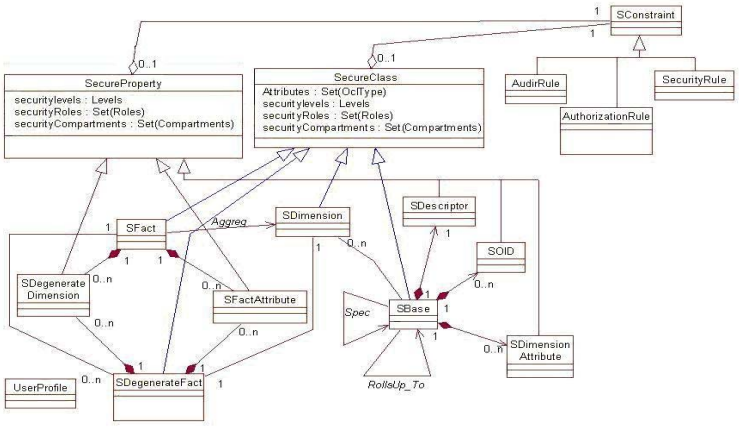

Figura 2. Metamodelo SECDW usado para diseñar el SMD PIM

SecureDegenerateFact, SecureDimension y *SecureBase, SecureDegenerateDimension, SecureFactAttribute, SecureDescriptor, SecureOID y SecureDimensionAttribute*) se define su información de seguridad, mediante la especificación de una secuencia de niveles de seguridad (*SecurityLevels*), un conjunto de categorías de usuario (*SecurityCompartment*) y un conjunto de roles de usuario (*SecurityRoles*). Adicionalmente estas clases tienen restricciones de seguridad (*SConstraint*) para indicar el nivel de seguridad que tienen los usuarios para acceder a cierta información. Por ejemplo, con una regla de autorización (*AuthorizationRule*) podemos representar como denegamos el acceso de los usuarios a cierta información. Este acceso puede depender del valor de ciertos atributos contenidos en diversas clases. Este hecho lo podemos representar en el modelo mediante una regla de seguridad (*SecurityRule*). Si un usuario trata de acceder a una información para la que tiene denegado el acceso, entonces con una regla de auditoría (*AuditRule*) podemos modelar este hecho. Estas restricciones son representadas a nivel de modelo por medio de notas de UML.
Más detalles sobre este *profile* se puede encontrar en [18] y [19].

3.2. Definición del SMD PSM

Un modelo específico de plataforma (PSM) es una vista de un sistema desde el punto de vista de la plataforma. En el diseño de bases de datos y almacenes de datos, el modelado conceptual da lugar al PIM, y el modelado lógico al PSM. En el modelado multidimensional el nivel lógico es diseñado según las propiedades específicas del SGBD (ROLAP, MOLAP o HOLAP). Aunque Kimball [7] asegura que la representación más común es sobre plataformas relacionales, es decir, sobre sistemas ROLAP.
En [17] presentamos nuestro PSM, a partir de una extensión del metamodelo relacional del *Commom Warehouse Metamodel* (CWM) que nos permite modelar los aspectos de seguridad y auditoria representados a nivel lógico en el modelado de los DWs. Este metamodelo es llamado *Secure Relational* DW (SECRDW).

En la Figura 3 mostramos el metamodelo SECRDW que será designado como PSM. Para distinguir que modelamos aspectos de seguridad será llamado *Secure Multidimensional* PSM (SMD PSM). En este metamodelo podemos representar tablas, columnas, claves primarias y ajenas, etc. El contenedor *Schema*, permite la seguridad a nivel de modelo. Las metaclasses *SecurityProperty* y *SecurityConstraint* tienen asociación con las metaclasses *Table* y *Column* respectivamente, de este modo se establece la seguridad en atributos y tablas. Además *SecurityConstraint* nos permite expresar las restricciones (*AuditRule*, *AuthorizationRule* y *SecurityRule*) modeladas mediante notas de UML en el metamodelo *Secure* DW (SECDW), es decir, en el PIM. La metaclase *userProfile* especifica las restricciones sobre una información particular de un usuario o un grupo de usuarios.

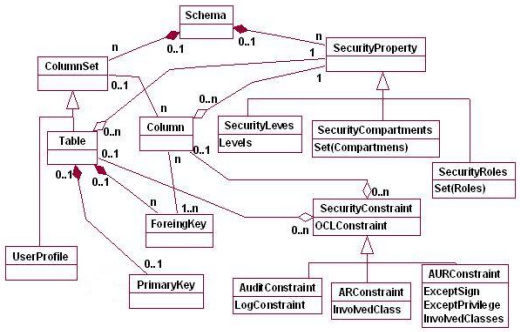

Figura 3. Metamodelo SECDW usado para diseñar el SMD PSM

```
Transformation SMD To SREL(SMD: SECDW,
 SREL: SECRDW)
 {
key Table{name, Schema};
key Column {name, owner};
key UserProfile{name, Schema};
key PrimaryKey{name, owner};
key ForeingKey{name, owner};
key SecurityProperty{name, owner};
key SecurityConstraint(name, owner);
top relation SStarPackage2Schema{}
top relation SFact2Table {}
top relation SDegenerateFact2Table{}
top relation SDimension2Table{}
  //Association SFact with SDimension
top relation AssocSF_D2FKey{}
  //Association SDegenerateFact with SDimension
top relation Assoc SDF_SD2FKeyFKey
 // Association SDegenerateFact with SFact
top relation AssocSDF_SF2FKey{}}
```

Figura 4. Notación textual para la transformación SMDPIM To SMDPSM

3.3. Transformaciones QVT de SMD PIM a SMD PSM

Comenzamos con la transformación principal para representar algunas relaciones del tipo *top-level* y *non-top-level*. Una relación *top-level* tiene la palabra clave *top* para distinguirla sintácticamente. En las subsecciones 3.3.1, 3.3.2 y 3.3.3 vamos a explicar en detalle cómo se definen relaciones para establecer la transformación entre el SMD PIM y el SMD PSM. Para ello emplearemos la notación diagramática y textual que ofrece QVT. Centraremos la atención en los requisitos de seguridad y auditoría para hacer más comprensible la transformación.

En la Figura 4 hemos empleado la notación textual para establecer la transformación principal, es decir, la transformación SMD PIM To SMD PSM. Las relaciones están precedidas de la palabra clave *top* para especificar que nunca serán invocadas por otras relaciones en la transformación. Cada una de estas relaciones tiene sus propias cláusulas *when* y *where* para satisfacer pre y post-condiciones.

3.3.1. Transformación de SFact a Tabla

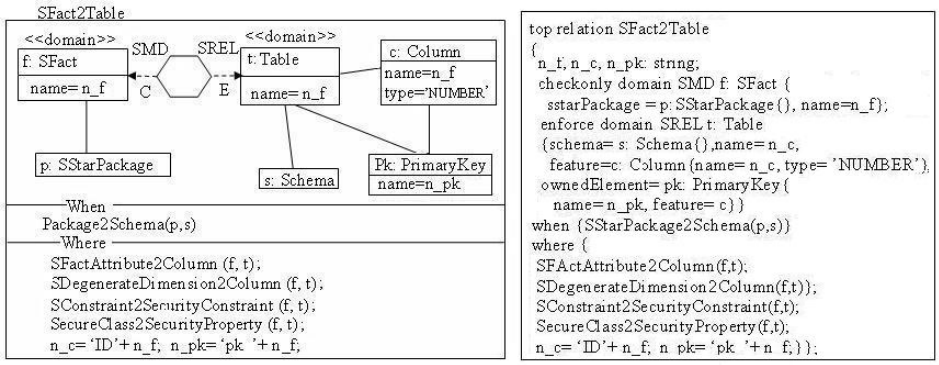

Figura 5. Transformando SFact en Table

La Figura 5 muestra la relación *SFact2Table*. A la izquierda aparece la notación diagramática, mientras que a la derecha la textual. Aquí a *SFact* le corresponde una tabla con el mismo nombre de *SFact*. Esta tabla tendrá una columna con un nombre (especificado en la cláusula *where*), el cual también será la clave primaria de la tabla. La relación *SFAct2Table* será satisfecha solo cuando se satisfaga la pre-condición *SStarPackage2Schema*, con ello aseguramos que la tabla estará contenida en *Schema*.
Los atributos de *SFact* conjuntamente con su información de seguridad y restricciones son transformados según las relaciones *SFactAttribute2Column* (vea la Figura 6) o *SDegenerateDimension2Column*, según corresponda. Observemos que esta última relación nunca será invocada porque solo consideramos a nivel lógico el esquema estrella [7]. *SFact* hereda de *SecureClass* restricciones de seguridad que son transformadas por *SConstraint2SecurityConstraint* en restricciones de seguridad de la tabla (vea la Figura 7). Estas restricciones son modeladas mediante notas de UML. De igual manera, *SFact* hereda información de seguridad que es representada a nivel lógico cuando es llamada la relación *SecureClass2SecurityProperty*. Esta información de seguridad es modelada a nivel lógico en el encabezamiento de la tabla que representa a la *SFact*.
En las siguientes subsecciones definimos en detalle las transformaciones *SFactAttribute2Column* y *SConstraint2SecurityConstraint*.

3.3.2. Transformación de SFactAttribute a Columna

En la Figura 6 mostramos la notación diagramática y textual para la relación *SFacAttribute2Column*. *SFactAttribute* hereda información y restricciones de seguridad de la clase *SecureProperty*, pues esta contiene a la clase *SConstraint*. Por ello la relación *SFacAttribute2Column* transforma no solo atributos en columnas, sino también todas las informaciones y restricciones de seguridad que tiene *SFActAttribute* a nivel conceptual. Esto lo garantizan las relaciones que aparecen como post-condiciones en la relación *SFacAttribute2Column*. La información de seguridad de cada *SFactAttribute* se modela a nivel lógico junto a cada columna de la tabla. Las restricciones se modelan como notas de

UML asociadas a la columna. Observe que la function *SMDType2SRELType* convierte un tipo de dato del metamodelo origen (es decir, *Secure* DW, SECDW) en un cierto tipo de dato del metamodelo destino (es decir, *Secure Relational* DW, SECRDW).

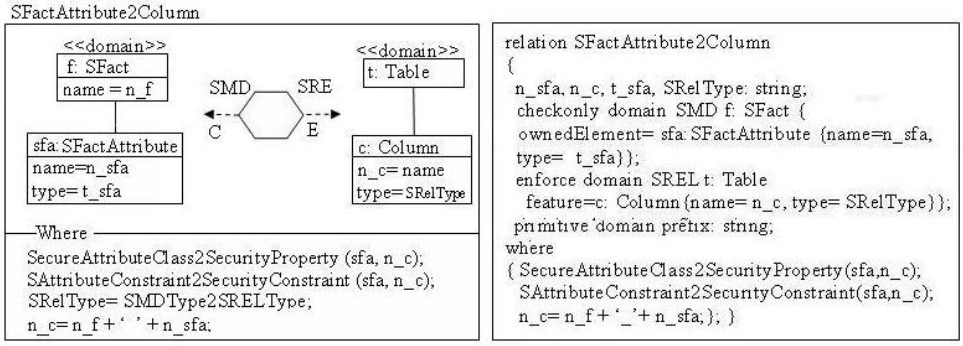

Figura 6. Transformando SFactAttribute en Column

3.3.3. Transformación de SConstraint a SecurityConstraint

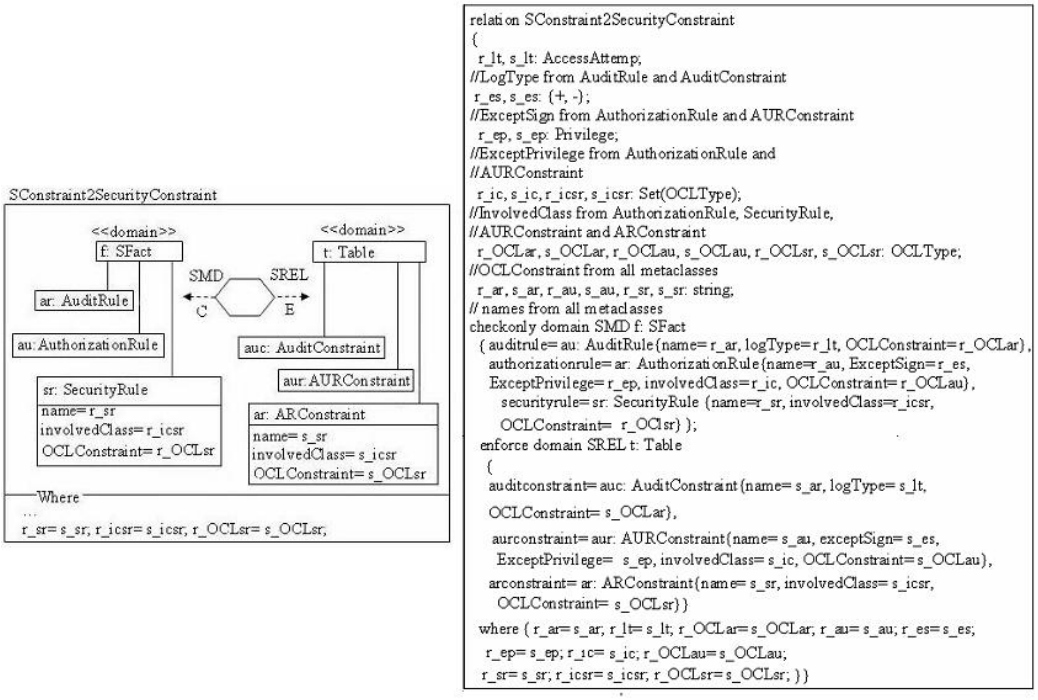

Figura 7. Transformando SConstraint en SecurityConstraint

Vamos a asumir que en la relación *SFactAttribute2Column*, todas las relaciones de la

cláusula *where* han sido definidas. Regresemos al contenido de las post-condiciones que aparecen en la relación *SFact2table*, corresponde ahora establecer la transformación *SConstraint2SecurityConstraint*. A la izquierda de la Figura 7 aparece la notación diagramática que ofrece QVT, omitimos algunos atributos para hacerla más comprensible. Sin embargo, la notación textual (que aparece a la derecha de la Figura 7) para esta transformación incluye todos los atributos de las metaclases. Cuando se aplica la relación a cada *SConstraint* (es decir, *AuditRule*, *AuthorizationRule* o *SecurityRule*) asociado a la *SFact* le corresponde un *SecurityConstraint* (es decir, *AuditConstraint*, *ARConstraint* o *AURConstraint*) de la tabla que representa a nivel lógico a la *SFact*.

3.4. Transformación de SMD PSM a MSD Code

Elegimos Oracle 9i como SGBD para obtener código, este sistema nos permite implementar bases de datos multiniveles, tiene además un componente llamado *Oracle Label Security* (OLS) que permite establecer seguridad mediante predicados y funciones etiquetadas. Estas funciones y predicados son accionados cuando una operación es ejecutada, de este modo definen el valor de seguridad para la etiqueta según el cumplimiento de cierta condición. El código generado con esta transformación lo hemos llamado *Secure Multidimensional Code* (SMD *Code*), ya que nos referimos a un código que implementa requisitos de seguridad.

4. CONCLUSIONES Y TRABAJO FUTURO

En este artículo hemos presentado un conjunto de transformaciones MDA mediante el estándar QVT (*Query*/*Views*/*Transformations*) para transformar un modelo conceptual multidimensional seguro en un esquema lógico relacional seguro. Estas transformaciones junto a las definiciones de un PIM y un PSM multidimensional seguro (SMD PIM, SMD PSM) nos permiten definir una arquitectura MDA Multidimensional Segura. La principal ventaja de esta arquitectura se manifiesta en un ahorro de tiempo y esfuerzo para los diseñadores, pues solo tienen que crear un PIM seguro y de manera automática se obtiene el modelo relacional con su correspondiente código para una plataforma relacional. De este modo proponemos una elegante solución al hueco semántico existente entre los modelos de datos conceptuales avanzados e implementaciones relacionales.
Nuestro trabajo futuro inmediato consiste en estudiar la posibilidad de representar mediante casos de uso los requerimientos de seguridad de un DW, construir un CIM seguro y establecer transformaciones CIM seguro-PIM seguro. Un trabajo a más largo plazo es extender esta propuesta para considerar los procesos ETL.

5. AGRADECIMIENTOS

Este proyecto ha sido parcialmente financiado por los proyectos METASIGN (TIN2004-00779) del Ministerio Español de Educación y las Ciencias, DADASMECA (GV05/220) del gobierno regional de Valencia y DIMENSIONS (PBC-05-012-1 y DADS (PBC-05-012-2) FEDER y por la Consejería de Ciencia y Tecnología de la Junta de Comunidades de Castilla-La Mancha.

6. REFERENCIAS

[1] G. Dhillon and J. Backhouse, "Information systems security management in the new millenium.," Communications of the ACM, vol. 43 (7), 2000.

[2] P. Devanbu and S. Stubblebine, "Software engineering for security: a roadmap," actas del The future of Software engineering, Limerick, Ireland, 2000.

[3] R. Kirkgöze, N. Katic, M. Stolda, and A. M. Tjoa, "A Security Concept for OLAP," actas de la 8th (DEXA), Toulouse, France, 1997.

[4] N. Katic, G. Quirchmayr, J. Schiefer, M. Stolba, and A. M. Tjoa, "A Prototype Model for DW Security Based on Metadata," actas del 9th (DEXA), Vienna, Austria, 1998.

[5] T. Priebe and G. Pernul, "Towards OLAP Security Design - Survey and Research Issues.," actas del 2nd Workshop on (DMDW), Sweden, 2000.

[6] A. Rosenthal and E. Sciore, "View Security as the Basic for Data Warehouse Security," actas del 2nd Workshop on (DMDW), Sweden, 2000.

[7] R. Kimball and M. Ross, The Data Warehousing Toolkit, 2 edition ed: J. Wiley, 2002.

[8] J. Hammer, M. Schneider, and T. Sellis, "Data Warehousing at the Crossroads," actas del Dagstuhl Perspectives Workshop, Dagstuhl, 2004.

[9] J. Miller and J. Mukerji, "MDA Guide Version 1.0.1," 2003.

[10] K. Czarnecki and S. Helsen, "Classification of Model Transformation Approaches," actas del 2nd OOPSLA Workshop, USA, 2003.

[11] D. Basin, J. Doser, and T. Lodderstedt, "Model Driven Security: from UML models to access control infrastructures," ETH, Zürich 4 - September 2003.

[12] C. C. Burt, B. R. Bryant, R. R. Raje, A. M. Olson, and M. Auguston, "Model Driven Security: Unification of Authorization Models for Fine-Grain Access Control," actas del 7th IEEE International. EDOC, Brisbane, Australia, 2003.

[13] S. N. Sivanandam and G.R.Karpagam, "A Novel approach for Implementing Security services," Academic Open Internet Journal, vol. 13, 2004.

[14] U. Lang and R. Schreiner, "OpenPMF: a Model-Driven Security Framework for Distributed Systems," actas de (ISSE), Berlin, Germany, 2004.

[15] J. N. Mazon, J. Trujillo, M. Serrano, and M. Piattini, "Applying MDA to the development of data warehouses," actas del DOLAP, 2005.

[16] QVT, "2nd Revised Submission: MOF 2.0 Query/Views/Transformations.," 2006.

[17] E. Soler, R. Villarroel, J. Trujillo, E. Fernández-Medina, and M. Piattini, "Representing security and audit rules for Data Warehouses at the logical level by using the Common Warehouse Metamodel," actas del ARES, Vienna, Austria, 2006.

[18] E. Fernández-Medina, J. Trujillo, R. Villarroel, and M. Piattini, "Extending UML for designing secure Data Warehouses," actas del 23rd (ER), Shangai, China, 2004.

[19] E. Fernández-Medina, J. Trujillo, R. Villarroel, and M. Piattini, "Access Control and Audit Model for the Multidimensional Modeling of DW," DSS, vol. in press, 2006.