Diagramas de mapeo de atributos para el diseño de almacenes de datos con UML*

Sergio Luján-Mora[1], Juan Trujillo[1], and Panos Vassiliadis[2]

[1] Dept. de Lenguajes y Sistemas Informáticos
Universidad de Alicante (Spain)
{slujan,jtrujillo}@dlsi.ua.es
[2] Dept. de Ciencias de la Computación
Universidad de Ioannina (Grecia)
pvassil@cs.uoi.gr

Resumen En el entorno de los almacenes de datos (ADs), se conoce como ETL (*Extraction, Transformation, Loading*) a los procesos responsables de la extracción de los datos de las fuentes de datos heterogéneas, su transformación (conversión, limpieza, normalización, etc.) y su carga en el AD. Un aspecto crítico por resolver en el modelado de estos procesos ETL es el hecho de que los datos se tratan a unos niveles de granularidad muy bajos que incluyen la definición de reglas de transformación a nivel de atributos, no existiendo hasta el momento ninguna propuesta que nos permita una definición formal de tales transformaciones. En este artículo, extendemos el Lenguaje de Modelado Unificado (*Unified Modeling Language, UML*) con un nuevo diagrama denominado *diagrama de mapeo de datos*, el cual nos permite representar las reglas de transformación entre atributos necesarias para modelar los procesos ETL a nivel conceptual. Finalmente, y para facilitar su manejo, hacemos uso de los diagramas de paquetes de UML, obteniendo una propuesta que nos permite especificar los procesos ETL con diferentes niveles de detalle.

Keywords: mapeo de atributos, ETL, almacenes de datos, UML

1. Introducción

En el entorno de los almacenes de datos (ADs), se conoce como ETL (*Extraction, Transformation, Loading*) a los procesos responsables de la extracción de los datos de las fuentes de datos heterogéneas, su transformación (conversión, limpieza, normalización, etc.) y su carga en el AD. El diseño y mantenimiento de los procesos ETL es un elemento clave en el éxito de un proyecto de AD por diferentes razones, destacando que el desarrollo de los procesos ETL puede suponer hasta el 80 % del tiempo de desarrollo de un proyecto de AD.

Con respecto al diseño de los procesos ETL y, a pesar de la importancia de especificar el mapeo desde las fuentes de datos operacionales a las estructuras

* Realizado con una ayuda concedida por la Secretaría de Estado de Educación y Universidades, Ministerio de Educación, Cultura y Deporte.

de los AD (a través de restricciones y transformaciones), desafortunadamente, existen muy pocos modelos que se puedan utilizar para este objetivo. Hasta el momento, la mayor parte de la investigación en el área del modelado conceptual de los almacenes de datos se ha centrado en los usuarios finales (*front-end*), mientras que pocas propuestas se han centrado en el modelado conceptual de estos procesos ETL (*back stage*) [1,2]. Hasta el momento, podemos afirmar que no existe un modelo que pueda combinar (a) el nivel de detalle deseado para la integración y representación de datos a nivel de atributo y (b) un formalismo de modelado ampliamente aceptado como pueda ser el modelo ER o UML. La principal razón para esta situación es que ambos formalismos no están *pensados* para tal fin; por el contrario, tratan los atributos como entidades débiles (también denominados *atributos o ciudadanos de segunda clase*) con un único y principal papel descriptivo. Un problema principal derivado de esta situación es que en ambos formalismos los atributos no pueden servir como un final en una asociación o cualquier otra relación.

Quizás se podría pensar que la forma actual de modelado es suficiente y que no hay una necesidad real de extenderla para capturar los mapeos y transformaciones a nivel de atributo. Sin embargo, podemos proporcionar varias razones en contra de este argumento: (i) documentar los AD complejos es un problema y, se está demostrando que un formalismo adecuado como pueda ser el ER o UML para la fase del diseño conceptual, facilita enormemente esta tarea, (ii) el diseño conceptual del AD debería permitir el análisis *Qué-pasa-si* de cambios posteriores. Capturar los atributos y sus relaciones como *ciudadanos de primer orden* mejora el diseño significativamente con respecto a estas metas. Además, la forma en que este aspecto se manejaría podría incluir una documentación informal y sencilla a través de notas de UML y, (iii) en líneas de investigación previas [3], se ha mostrado que con el modelado de las relaciones entre atributos podemos tratar los artefactos de diseño como un grafo y medir las metas mencionadas anteriores. De nuevo, esto sería imposible con los actuales formalismos de modelado.

Para alcanzar todos los objetivos mencionados anteriormente, proporcionamos una aproximación que nos permite modelar las particularidades de los procesos ETL en varios niveles de detalle, a través de un formalismo ampliamente aceptado (UML). En [2] presentamos el modelado de los procesos ETL especificando la serie de transformaciones (ej. generación de claves) que se necesitaban realizar a las fuentes de datos operacionales antes de cargar los datos en el AD.

En este artículo, completamos nuestra anterior propuesta para el modelado de procesos ETL proporcionando una vista adicional del AD que denominamos *diagrama de mapeo de datos*. En este nuevo diagrama extendemos UML formalmente para tratar los atributos como elementos de modelado de primera clase. Una vez realizado esto, los atributos pueden participar en asociaciones que determinan los mapeos entre atributos, junto con cualquier transformación y restricción necesaria. Una de las principales ventajas de la propuesta presentada en este artículo es que está totalmente integrada en una aproximación global

que nos permite realizar el diseño conceptual, lógico y el correspondiente diseño físico de todos los componentes del AD utilizando la misma notación ([4,5,6,7]).

El resto del artículo está estructurado como sigue. En la sección 2, presentamos el estado de la cuestión más relevante. En la sección 3 describimos brevemente nuestro marco general para el diseño de almacenes de datos e introducimos el ejemplo que utilizaremos a lo largo de todo el artículo. En la sección 4, presentamos cómo representar los atributos como elementos de modelado de primera clase en UML. En la sección 5, presentamos nuestro *diagrama de mapeo de datos* como un nuevo diagrama de UML que nos permite modelar mapeos de datos a nivel de atributos en los procesos ETL. Finalmente, en la sección 6 presentamos las principales conclusiones y los trabajos futuros a realizar.

2. Estado de la cuestión

Hasta el momento, el modelado de los procesos ETL y su incorporación dentro de un método de diseño que abarque toda la concepción del AD ha tenido poca repercusión tanto a nivel de investigación como industrial.

En [8], se propone el *model management*, un marco para proporcionar soporte a aplicaciones que trabajen a nivel de metadatos, donde se manipulen modelos y mapeos entre modelos. En [8], se presentan dos estudios de caso de carga en un AD: por un lado, el mapeo entre los orígenes de datos y el propio AD, por otro lado, el mapeo entre el AD y un *data mart*. En esta aproximación, un mapeo es un modelo que relaciona los objetos (atributos) de otros dos modelos; cada objeto en un mapeo se denomina *mapping object* y posee tres propiedades : *domain* y *range*, que refieren a los objetos en el origen y el destino respectivamente, y *expr*, que es una expresión que define la semántica del *mapping object*. Esta aproximación está acompañada de su propia notación gráfica. Desde nuestro punto de vista, se trata de una aproximación aislada que no se integra con el diseño de otros aspectos de un AD.

En [1], los autores presentan la noción *provider mapping* entre atributos. Con el fin de evitar los problemas inherentes a ER y UML, los autores adoptan una aproximación genérica propia. El modelo conceptual estático presentado en [1] se complementa con el diseño lógico de los procesos ETL en [3]. Los procesos ETL se modelan como grafos compuestos de actividades que incluyen a los atributos como ciudadanos de primera clase. Además, diferentes tipos de relaciones capturan los flujos de datos entre los orígenes y destinos de datos.

Respecto el mapeo de datos, en [9] los autores discuten temas relacionados con el mapeo de datos en la integración de datos. Esta aproximación incluye un conjunto de operadores de mapeo y una clasificación de los posibles casos de mapeo. Sin embargo, no se proporciona una representación gráfica de los escenarios de mapeo, por lo que su uso en proyectos reales es difícil.

Respecto a las aproximaciones industriales, el modelo que surge de [10] es un método de documentación informal que comprende el diseño de los mapeos de datos y los procesos ETL. Por otro lado, el CWM (*Common Warehouse Metamodel*) es un estándar industrial iniciado por el OMG (*Object Management*

Group) para la integración de ADs y herramientas de análsis, basado en el uso compartido de metadatos. Este estándar se basa a su vez en tres estándares clave: MOF (*Meta Object Facility*), UML (*Unified Modeling Language*) y XMI (*XML Metadata Interchange*). Estos tres estándares proporcionan al CWM la base tecnológica necesaria para perfectamente representar la semántica de los ADs por medio de metadatos. Sin embargo, desde nuestro punto de vista, el CWM es demasiado general y no permite representar las principales propiedades del modelado multidimensional a nivel conceptual.

El tratamiento de los atributos como elementos de modelado de primera clase ha generado varios debates desde el principio del modelado conceptual [11]. Recientemente, algunas aproximaciones de modelado orientado a objetos como OSM (*Object Oriented System Model*) [12] y ORM (*Object Role Modeling*) [13] rechazan el uso de atributos (*attribute-free models*) básicamente debido a su inherente inestabilidad. En estas aproximaciones, los atributos se representan mediante entidades (objetos) y relaciones.

3. Diseño de un almacén de datos

La arquitectura de un AD se suele representar como varias capas a través de las cuales circulan los datos, de modo que los datos de una capa se obtienen a partir de los datos de la capa previa [14]. A partir de esta arquitectura, consideramos que el desarrollo de un AD se puede estructurar en un marco integrado por cinco etapas y tres niveles que definen los diferentes diagramas empleados para modelar un AD, tal como se resume en la Cuadro 1.

La principal ventaja de nuestra aproximación es que siempre empleamos la misma notación (basada en UML) para el diseño de los diferentes diagramas y las correspondientes transformaciones entre los diagramas de una forma integrada.

En trabajos anteriores, hemos presentado algunos de los diagramas (y los correspondientes perfiles de UML), como el *Multidimensional Profile* [4,5] y el *ETL Profile* [2]. En este artículo, presentamos el *Data Mapping Profile*.

Con el fin de motivar nuestra discusión, a continuación incluimos un ejemplo que se empleará a lo largo de todo el artículo. En este ejemplo, consideramos que se desea construir un AD que refleja las órdenes de compra que recibe una compañía. El origen de datos del AD es un fichero que se genera periódicamente a partir del OLTP de la empresa, conteniendo una información similar a:

```
1234;88844321;Gráficas Martínez;C-12345678-Q;20040512;20040514;
Calle Los Almendros, 13, 03005 Alicante;Calle Játiva s/n, 03008
Alicante;75222;2;23,55;0,230;Cartucho de tinta Epson 440;
```

Sin embargo, la propuesta de este artículo se refiere al modelado conceptual, por lo que no tenemos en cuenta aspectos como la estructura o el formato de los datos. En la Figura 1, mostramos una vista general del AD compuesta de tres paquetes estereotipados que representan el SCS (*Source Conceptual Schema*), el DWCS (*Data Warehouse Conceptual Schema*) y el DM (*Data Mapping*) que define el mapeo entre el SCS y el DWCS y que es el núcleo de este artículo.

- **Etapas**: distinguimos cinco etapas en la definición de un AD:
  - Origen: define los orígenes de datos del AD, como los sistemas OLTP (OnLine Transaction Processing), las fuentes de datos externas (datos sindicados, datos censales), etc.
  - Integración: define el mapeo entre los orígenes de datos y el propio AD.
  - Almacén de datos: define la estructura del AD.
  - Adaptación: define el mapeo entre el AD y las estructuras empleadas por el cliente.
  - Cliente: define las estructuras concretas que son empleadas por los clientes para acceder al AD, como *data marts* o aplicaciones OLAP.
- **Niveles**: cada etapa se analiza desde tres niveles o perspectivas que se crean en el siguiente orden:
  - Conceptual: define el AD desde un punto de vista conceptual, es decir, desde el mayor nivel de abstracción y contiene únicamente los objetos y relaciones más importantes.
  - Lógico: abarca aspectos lógicos del diseño del AD, como la definición de las tablas y claves, la definición de los procesos ETL, etc.
  - Físico: define los aspectos físicos del AD, como el almacenamiento de las estructuras lógicas en diferentes discos o la configuración de los servidores de bases de datos que mantienen el AD.
- **Diagramas**: cada etapa o nivel necesita formalismos de modelado diferentes. Por lo tanto, nuestra aproximación se compone de 15 diagramas (5 etapas y 3 niveles), pero el diseñador del AD no necesita definir todos los diagramas en cada proyecto de AD. En nuestra aproximación, usamos UML [15]como lenguaje de modelado, porque su potencia expresiva es la suficiente para el modelado de todos los diagramas de nuestra aproximación. Pero como UML es un lenguaje de modelado general, necesitamos usar los mecanismos de extensión de UML para adaptarlo al dominio específico de los ADs.

Cuadro 1. Marco de desarrollo de almacenes de datos

Figura 1. Vista general del almacén de datos

Evidentemente, esta vista de alto nivel del AD se puede explorar con más detalle. En la parte izquierda de la Figura 2, mostramos la definición detallada del SCS, que representa las fuentes de datos que alimentan el AD con datos. En este ejemplo, la fuente de datos se compone de cuatro entidades representadas mediante clases de UML: Producto, Orden, Cliente y ConPago. Por otro lado, en la parte derecha de la Figura 2 mostramos el DWCS de nuestro ejemplo. El AD se compone de un hecho (OrdCompra) y tres dimensiones (Destino, FechaEnvio y Ordenante).

Por último, en la Cuadro 2, mostramos un ejemplo parcial del formulario para documentar procesos ETL propuesto en [16], que se compone de tres columnas principales: *origen* (identifica el sistema y el campo del cual provienen los datos), *transformación* (describe la transformación que se debe de realizar sobre los datos) y *destino* (describe el destino de los datos en el AD). Sin embargo, documentar un proceso de transformación de esta forma presenta algunos problemas importantes, como la falta de coherencia entre los diagramas de los modelos y el formulario de transformación, o la dificultad para entender y ges-

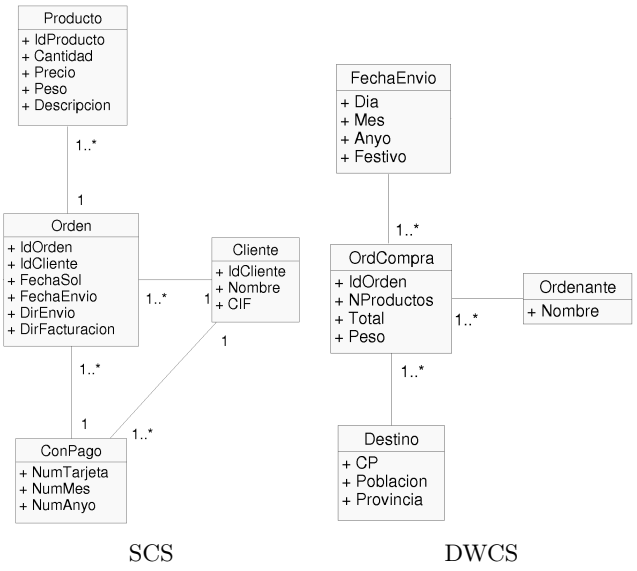

Figura 2. SCS y DWCS

tionar un formulario de transformación en el caso de ADs enormes. Por lo tanto, creemos que los procesos de transformación se deberían definir en los diagramas de los modelos como otras partes del diseño del AD.

Origen			**Transformación**	**Destino**	
Sistema	**Tabla o fichero**	**Campo o columna**		**Tabla**	**Campo o columna**
...	...	...	...	...	...
OLTP	Orden	DirEnvio	Extraer el CP	Destino	CP
			Extraer la población	Destino	Poblacion
			Extraer la provincia	Destino	Provincia
...	...	...	...	...	...
OLTP	Orden	FechaEnvio	Extraer el dia	FechaEnvio	Dia
			Extraer el mes	FechaEnvio	Mes
			Extraer el anyo	FechaEnvio	Anyo
			Calcular si es festivo	FechaEnvio	Festivo
...	...	...	...	...	...

Cuadro 2. Formulario de transformación

En este artículo, presentamos un diagrama adicional a nuestra propuesta de desarrollo de ADs [6], que denominamos *Data Mapping* y que muestra las relaciones que existen entre las fuentes de datos y el AD y entre el AD y las estructuras empleadas por las herramientas cliente de explotación del AD. En este nuevo diagrama, necesitamos tratar los atributos de una clase como elemen-

tos de modelado de primer nivel, ya que necesitamos representar las relaciones existentes a nivel de atributo. Por lo tanto, en este artículo también proponemos una extensión de UML para lograr dicho objetivo. Hasta donde nosotros sabemos, ésta es la primera propuesta de representar atributos como elementos de modelado de primer nivel en un diagrama de UML.

4. Atributos como elementos de modelado de primer nivel en UML

Tanto en el modelo ER como en UML, los atributos se incluyen en la definición del elemento al que pertenecen (clase, entidad, relación, etc.), y no es posible crear una relación directa entre dos atributos. Tal y como hemos comentado en la introducción, en algunas situaciones (ej. integración de datos, restricciones sobre atributos, etc.) sería importante representar los atributos como elementos de modelado de primera clase. Por lo tanto, en esta sección resumiremos nuestra extensión de UML para definir atributos como ciudadanos de primera clase.

A lo largo del artículo, frecuentemente mencionamos el uso del término *ciudadanos de primera clase* para los elementos de nuestros lenguajes de modelado. Conceptualmente, con ciudadanos de primera clase nos referimos a conceptos fundamentales de modelado, sobre los cuales nuestros modelos se construyen. Técnicamente, ciudadanos de primera clase incluyen una identidad por sí solos, y posiblemente soporten restricciones de integridad (ej. las relaciones deben tener al menos dos *finales* en sus extremos). En un diagrama de clases de UML, sólo las clases y las asociaciones son tratados como ciudadanos de primera clase.

De forma natural, para permitir que los atributos tengan un papel en ciertos casos, proponemos la representación de atributos como elementos de modelado de primera clase en UML. En nuestra propuesta, las clases y los atributos se definen de forma usual en UML. Sin embargo, en los casos en los que es necesario tratar los atributos como elementos de modelado de primera clase, las clases se importan al *diagrama de atributos/clases*, donde los atributos se representan de forma automática como clases; y de esta forma, el usuario sólo tiene que definir las clases y los atributos una vez.

De forma resumida, los estereotipos que hemos definido para poder representar los *diagramas de atributos/clases* son:

- ≪Attribute≫: se aplica a una clase y representa el atributo de una clase. Posee una serie de valores etiquetados (changeability, initialValue, multiplicty, etc.) para representar las propiedades que poseen los atributos.
- ≪Contain≫: es la relación de agregación existente entre la clase contenedora (la clase que posee los atributos) y las clases que representan sus atributos.

Volviendo a nuestro ejemplo (Figura 2), en la parte izquierda de la Figura 3, se muestra la representación tradicional de la clase Orden, que contiene seis atributos. En la parte derecha de la misma figura, aparece representada la misma clase en nuestro diagrama de atributos/clases: la clase ha sido importada desde otro paquete y sus atributos no se representan dentro de la clase, sino que aparecen como clases independientes con el estereotipo ≪Attribute≫.

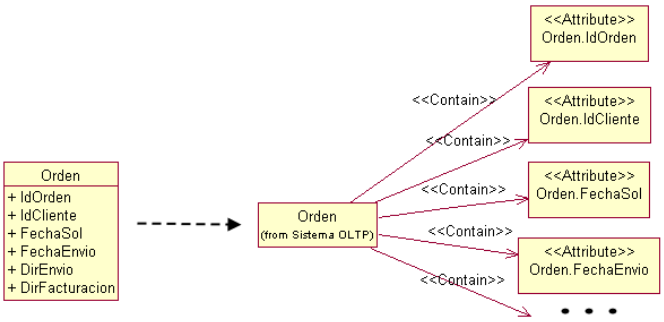

Figura 3. Atributos representados mediante elementos de modelado de primera clase

5. El diagrama de mapeo de datos

Una vez introducido el mecanismo de extensión mediante el cual podemos tratar los atributos como ciudadanos de primera clase en UML, podemos definir los diagramas donde los empleamos. En esta sección, presentamos el *diagrama de mapeo de datos*, que es un nuevo tipo de diagrama adaptado para representar el flujo de datos, con varios niveles de detalle, en un AD.

Para capturar las interconexiones entre distintos elementos de diseño, en términos de los datos, empleamos la noción de *mapeo*. Un mapeo se define mediante tres elementos lógicos:

- El proveedor: una entidad (esquema, tabla o atributo) responsable de generar los datos que posteriormente se propagan.
- El consumidor: que recibe los datos del proveedor.
- El emparejamiento: que define la forma en la cual el mapeo se realiza, incluyendo cualquier tipo de transformación o filtrado.

Los mapeos se pueden definir con distintos niveles de granularidad: al nivel de esquema, tabla o atributo. En nuestra propuesta, el mapeo se establece a nivel de tabla/atributo entre las fuentes de datos (el SCS) y el AD (el DWCS), y entre el AD (el DWCS) y las estructuras empleadas por los clientes (el CCS).

Como un diagrama de mapeo de datos puede ser muy complejo, nuestra propuesta permite organizarlo en diferentes niveles gracias al uso de los paquetes de UML. Nuestra propuesta consiste de cuatro niveles (ver la Figura 4):

Nivel de base de datos (o **Nivel 0**). En este nivel, cada esquema del AD (por ejemplo, esquema de las fuentes de datos a nivel conceptual en el SCS, esquema conceptual del AD en el DWCS, etc.) se representa mediante un paquete [6]. Los mapeos entre los diferentes esquemas se modelan en un único paquete de mapeo, que encapsula todos los detalles.

Nivel de flujo de datos (o **Nivel 1**). Este nivel describe las relaciones de datos a nivel individual entre las fuentes de datos hacia los respectivos destinos en el AD.

Nivel de tabla (o **Nivel 2**). Mientras que el diagrama de mapeo en el nivel 1 describe las relaciones entre las fuentes y los destinos de datos mediante un único paquete, el diagrama de mapeo de datos en el nivel de tabla detalla todas las transformaciones intermedias que tienen lugar durante ese flujo.

Nivel de atributo (o **Nivel 3**). En este nivel, el diagrama de mapeo de datos captura los mapeos existentes a nivel de atributo.

En la parte más izquierda de la Figura 4, una única relación entre el DWCS y el SCS (representada mediante un único paquete llamado Data Mapping) y estos tres elementos de diseño constituyen el diagrama de mapeo de datos a nivel de base de datos (o Nivel 0). Suponiendo que existan tres tablas en el AD que se quieren poblar con datos, el paquete Data Mapping abstrae el hecho de que existen tres escenarios, uno para cada una de las tablas. En el nivel de flujo de datos (o Nivel 1), se modelan mediante un paquete las relaciones de flujo de datos existentes entre las fuentes y los destinos de datos en el contexto de cada escenario. Si se explora con más detalle uno de estos escenarios, por ejemplo el llamado Mapeo 1, podemos observar las particularidades del mapeo: los datos de Fuente 1 se transforman en dos pasos (sufren dos transformaciones), como se muestra en la Figura 4. Se puede ver que existe un almacenamiento temporal denominado Intermedio, que almacena los datos generados por la primera transformación (Paso 1), antes de dirigirse a la segunda transformación (Paso 2). Por último, en la parte inferior derecha de la figura, se muestra como se realiza el mapeo a nivel de atributo entre Fuente 1 e Intermedio. De este modo, en el caso de que se esté modelando un AD grande y complejo, nuestra propuesta permite ocultar los detalles de la transformación de los atributos en el nivel 3.

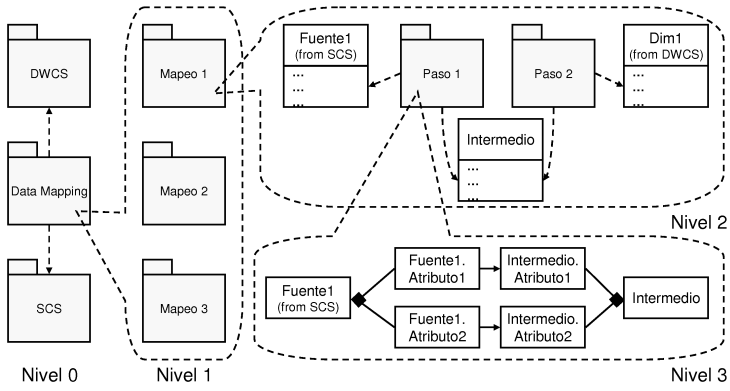

Figura 4. Niveles de mapeo de datos

Para representar los mapeos que proponemos, hemos desarrollado una extensión de UML mediante un perfil. Brevemente, los elementos de modelado que empleamos para realizar los mapeos en cada nivel son:

- Los diagramas de base de datos y de flujo de datos (niveles 0 y 1) emplean una notación estándar de UML. Más concretamente, en estos diagramas empleamos (a) los paquetes para modelar las relaciones de datos y (b) dependencias entre los elementos involucrados. Las dependencias indican que los paquetes de mapeo son sensibles a los cambios en las fuentes y los destinos de datos.
- El diagrama de nivel de tabla (nivel 2) extiende UML con tres estereotipos: (a) ≪Mapping≫, empleado con paquetes para encapsular las relaciones entre los distintos almacenamientos de datos y (b) ≪Input≫ y ≪Output≫ para definir los distintos roles de proveedores y consumidores en un mapeo.
- El diagrama a nivel de atributo (nivel 3) emplea los estereotipos: ≪Map≫, ≪MapObj≫, ≪Domain≫, ≪Range≫, ≪Input≫, ≪Output≫ e ≪Intermediate≫.

5.1. El ejemplo de nuevo

A partir del ejemplo de AD mostrado en la Figuras 1 y 2, hemos definido el correspondiente nivel 1 del diagrama de mapeo de datos mostrado en la Figura 5. En este nivel, cada paquete de UML representa un mapeo de datos que posee un destino único en el AD.

Figura 5. Nivel 1 del ejemplo de mapeo de datos

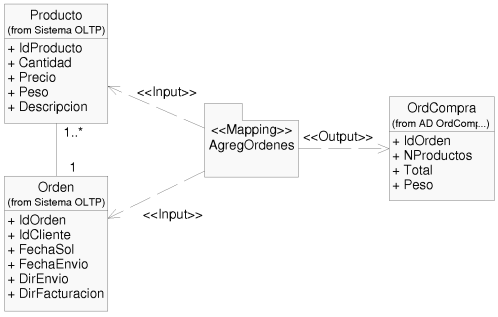

Figura 6. Nivel 2 del ejemplo de mapeo de datos

En la Figura 6 mostramos el contenido del paquete Mapeo OrdCompra de la Figura 5. El resultado de este mapeo se almacena OrdCompra y tiene como objetivo agrupar las órdenes de compra que recibe la compañía, de modo que se reduzca la granularidad de los datos almacenados: no se almacenan uno a

uno los productos que componen una orden de compra, sino que se agregan y se almacena el número de productos que componen la orden (NProductos), el importe total (Total), el peso total del envío (Peso) y el identificador de la orden de compra (IdOrden). Este mapeo se compone de un solo proceso AgregOrdenes y que se representa mediante un paquete con el estereotipo ≪Mapping≫ y emplea dos orígenes de datos (Producto y Orden).

Por último, en la Figura 7 mostramos el contenido del paquete AgregOrdenes de la Figura 6, que agrega las ordenes de compra. Usamos la cardinalidad muchos-a-uno para indicar que muchos datos de entrada son necesarios para calcular un único dato de salida. Además, mediante notas indicamos como calcular los atributos OrdCompra.NProductos, OrdCompra.Total y OrdCompra.Peso.

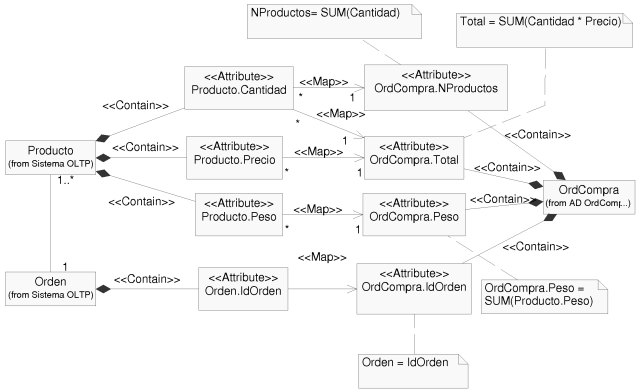

Figura 7. Nivel 3 del ejemplo de mapeo de datos

6. Conclusiones y trabajos futuros

En este artículo, hemos presentado un marco para el diseño del *back stage* de un almacén de datos (AD), basado en la observación clave de que las tareas que en él se desarrollan implican normalmente trabajar con los datos con niveles de granularidad muy bajos. Más concretamente, hemos presentado un marco para modelar las relaciones existentes entre las fuentes y los destinos de datos mediante distintos niveles de detalle. Desgraciadamente, los lenguajes de modelado estándar como ER o UML presentan un grave inconveniente para este tipo de modelado, ya que no tratan los atributos como elementos de modelado de primera clase. Por ello, con el fin de conseguir el objetivo marcado, hemos extendido UML para modelar los atributos como elementos de modelado de primera clase.

Un tema de investigación interesante es la definición de un conjunto de métricas de complejidad que permitan al diseñador comparar diferentes opciones

de diseño y seleccionar la mejor en base a unas medidas objetivas. En otra línea de investigación diferente, otro tema prometedor es la definición de una sintaxis para especificar las semánticas de los mapeos de datos.

Referencias

1. Vassiliadis, P., Simitsis, A., Skiadopoulos, S.: Conceptual Modeling for ETL Processes. In: Proc. of 5th Intl. Workshop on Data Warehousing and OLAP (DOLAP 2002), McLean, USA (2002) 14–21
2. Trujillo, J., Luján-Mora, S.: A UML Based Approach for Modeling ETL Processes in Data Warehouses. In: Proc. of the 22nd Intl. Conf. on Conceptual Modeling (ER'03). Volume 2813 of LNCS., Chicago, USA (2003) 307–320
3. Vassiliadis, P., Simitsis, A., Skiadopoulos, S.: Modeling ETL Activities as Graphs. In: Proc. of 4th Intl. Workshop on the Design and Management of Data Warehouses (DMDW'02), Toronto, Canada (2002) 52–61
4. Luján-Mora, S., Trujillo, J., Song, I.: Extending UML for Multidimensional Modeling. In: Proc. of the 5th Intl. Conf. on the Unified Modeling Language (UML'02). Volume 2460 of LNCS., Dresden, Germany (2002) 290–304
5. Luján-Mora, S., Trujillo, J., Song, I.: Multidimensional Modeling with UML Package Diagrams. In: Proc. of the 21st Intl. Conf. on Conceptual Modeling (ER'02). Volume 2503 of LNCS., Tampere, Finland (2002) 199–213
6. Luján-Mora, S., Trujillo, J.: A Comprehensive Method for Data Warehouse Design. In: Proc. of the 5th Intl. Workshop on Design and Management of Data Warehouses (DMDW'03), Berlin, Germany (2003) 1.1–1.14
7. Trujillo, J., Palomar, M., Gómez, J., Song, I.: Designing Data Warehouses with OO Conceptual Models. IEEE Computer, special issue on Data Warehouses **34** (2001) 66–75
8. Bernstein, P., Rahm, E.: Data Warehouse Scenarios for Model Management. In: Proc. of the 19th Intl. Conf. on Conceptual Modeling (ER'00). Volume 1920 of LNCS., Salt Lake City, USA (2000) 1–15
9. Dobre, A., Hakimpour, F., Dittrich, K.R.: Operators and Classification for Data Mapping in Semantic Integration. In: Proc. of the 22nd Intl. Conf. on Conceptual Modeling (ER'03). Volume 2813 of LNCS., Chicago, USA (2003) 534–547
10. Kimball, R., Reeves, L., Ross, M., Thornthwaite, W.: The data warehouse lifecycle toolkit. John Wiley & Sons (1998)
11. Falkenberg, E.: Concepts for modelling information. In: Proc. of the IFIP Conference on Modelling in Data Base Management Systems, Amsterdam, Holland (1976) 95–109
12. Embley, D., Kurtz, B., Woodfield, S.: Object-oriented Systems Analysis: A Model-Driven Approach. Prentice-Hall (1992)
13. Halpin, T., Bloesch, A.: Data modeling in UML and ORM: a comparison. Journal of Database Management **10** (1999) 4–13
14. Jarke, M., Lenzerini, M., Vassiliou, Y., Vassiliadis, P.: Fundamentals of Data Warehouses. 2 edn. Springer-Verlag (2003)
15. Object Management Group (OMG): Unified Modeling Language Specification 1.4. Internet: http://www.omg.org/cgi-bin/doc?formal/01-09-67 (2001)
16. Giovinazzo, W.: Object-Oriented Data Warehouse Design. Building a star schema. Prentice-Hall, New Jersey, USA (2000)