UN MODELO DE PERSISTENCIA PARA EL DISEÑO DE LA BASE DE DATOS BASADO EN UN MODELO DE OBJETOS FORMAL

Anaisa Hernández González[1] y Sofía Alvarez Cárdenas[2], Centro de Estudios de Ingeniería y Sistemas (CEIS), ISPJAE

RESUMEN
En este trabajo se describe un modelo de persistencia para el diseño de la base de datos a partir del paradigma de la orientación a objetos, por qué este modelo de persistencia se basa en el modelo formal O^3 (Orientado a objetos y ontológico) y cómo satisface el modelo de objetos de ODMG (Object Database Management Group). Se analizan además varios formalismos del paradigma de la orientación a objetos y las razones que justifican la selección del modelo O^3.

Palabras clave: sistemas de información, técnicas modernas, jerarquía.

ABSTRACT
In this work is described a persistence model for the database design starting from the object-oriented paradigm, why this persistence model is based on the O^3 (object-oriented and ontologic) formal model and how this model satisfies the object model of ODMG (Object Database Management Group). The article also analyzes several formalisms of the object-oriented paradigm and the reasons that justify the selection of the O^3 model.

Keywords: information system, modelling techniques, hierarchy.

MSC: 68P15

1. INTRODUCCION

Un modelo de datos es un formalismo matemático que permite realizar una interpretación de un dominio de aplicación con un determinado nivel de abstracción, de forma que reflejen los elementos u objetos del dominio dado, así como las formas en que los mismos se interrelacionan [Tschritzis-Lochovsky, (1982)].

El modelo orientado a objeto (MOO) es un modelo que permite describir los datos, sus estructuras y las operaciones válidas que se pueden realizar, pero apoyándose en los conceptos empíricos del paradigma de la orientación a objetos (objeto, clase, herencia, encapsulamiento, identificación de objetos, comunicación a través de mensajes y polimorfismo). Los productos comerciales han antecedido a la fundamentación teórica, aunque en los últimos años se ha estado trabajando fuertemente en su fundamentación.

En este trabajo se hace referencia al estado del arte de la formalización MOO, haciendo hincapié en el modelo O^3 (Orientado a Objetos y Ontológico) y en el estándar industrial ODMG (Object Database Management Group). El modelo O^3 es el escogido de base por el modelo de persistencia que se describe, este modelo de persistencia también satisface a ODMG.

2. FORMALIZACION DEL MODELO ORIENTADO A OBJETOS

En [Hofstede-Proper (1997)] se plantea que hay poca literatura dedicada a la formalización del modelo conceptual OO, aunque se reconoce la necesidad de formalizar las técnicas de modelación. Desafortunadamente los fundamentos teóricos del MOO comenzaron después que se desarrollaron los sistemas de gestión bases de datos de objetos (SGBDO). Es por eso que en ocasiones encontramos definiciones diferentes de un mismo concepto [Hubbers-Hofstede (1998)], incluso entre diferentes modelos formales [Fredericks **et al**. (1997)]. La mayoría de los formalismos de este modelo están asociados al desarrollo de un lenguaje de especificación.

E-mail: [1]anaisa@ceis.ispjae.edu.cu
[2]sofia@ceis.ispjae.edu.cu

En [Pastor (1992)] se agrupan los trabajos de formalización en aproximaciones algebraicas y categóricas, y aproximaciones ontológicas. Dentro del primer grupo podrían incluirse los trabajos del grupo IS-CORE (Information Systems-COrrectness and Reusability) del ISTL (Instituto Superior Técnico de Lisboa), que presentan una propuesta formal usando la teoría de categorías. Un objeto es formalizado como una 4-tupla (Conjunto de atributos, Conjunto de eventos, Conjunto de ciclos de vida o secuencias admisibles de eventos, Función de observación).

En [ODMG (2000)] se señala como principal problema que tratan de separar la modelización de la parte estructural (datos) de los objetos, de la modelación de la parte dinámica (procesos). Asocian a los objetos las fórmulas [Sernadas-Fidiero (1991)]: Valuation (indica los efectos de los eventos sobre los atributos), Safety (condiciones que deben satisfacerse para que se permita el evento) y Liveness (condiciones bajo las cuales un evento es obligado que ocurra). Define además los operadores de agregación y herencia. Esta propuesta ha evolucionado, y en la actualidad cuenta con un lenguaje de especificación (Oblog) , que permite la generación de código [Andrade **et al**. (1999)].

Otra formalización del modelo OO desarrolló un lenguaje de especificación semi-formal conocido como Troll [Jungclaus (1995) Posteriormente se realizaron trabajos que unieron la metodología OMT (Object Modeling Technology - la más usada durante gran parte de la década de los ‘90) con Troll [Jungclaus **et al**. (1994)].

Los fundamentos de Oblog y Troll están en la teoría de las categorías.

La tesis para optar por el grado de Doctor de Roef Johannes Wieringa (1990), ofrece una especificación del modelo conceptual, bastante completa en la definición de conceptos estáticos y dinámicos. El mismo autor reconoce como problema de su modelo que tiene muchos detalles que lo hacen muy grande, con más cosas que las que se usan en la modelación. Pero también es cierto que se hacen menos simplificaciones que en otros trabajos porque es más formal.

Siguiendo ésta tendencia se describe en [Tuijn-Gyssens (1996)] un modelo de datos orientado a objetos (OO) basado en grafos, pero relacionado solo con los datos, sus propiedades y sus relaciones con otras entidades, muy cercano al MR. Sin embargo no tiene en cuenta que un objeto es algo más que datos.

En [Frederiks **et al**. (1997)] se describe un modelo que trata de unificar las tendencias de varios modelos existentes. Este modelo se basa en la teoría de las categorías y los tipos de objeto son nodos de un grafo. Solo define conceptos asociados con la estructura.

En [Abiteboul **et al**. (1995)] se describe un modelo de base de datos orientado a objetos genérico en el que formalmente están definidos los conceptos de objeto, jerarquía de clase, clase, método y esquema e instancias; pero faltan conceptos de operadores que permiten ampliar el concepto de clase y otros relativos al comportamiento dinámico.

Dentro de las aproximaciones ontológicas se destacan los trabajos desarrollados por Y. Wand a finales de los ’80 [Pastor (1992)]. La idea es que el modelo orientado a objetos permite tener una visión natural del mundo, siendo los objetos la estructura básica de la modelación. Por lo tanto, el término ontológico se refiere al estudio de los objetos tal como son.

Los trabajos de la UPM (Universidad Politécnica de Madrid) en la formalización del MOO comenzaron tomando como base la propuesta del grupo IS-CORE. En sus primeras versiones eran aproximaciones algebraicas que hacían poco hincapié en la parte dinámica [Ramos **et al**. (1990)], pero en la actualidad es de las más completas.

El modelo O^3 es el resultado de los trabajos de la UPM y debe su nombre a la frase *“orientado a objetos y ontológico”* [Pastor (1992)]. Se caracteriza por basarse en la propuesta de Y. Wand que hace una distinción explícita entre objetos y propiedades y trata la dinámica de un sistema de objetos a través del concepto de Ley (conjunto de estados permitidos o prohibidos, que son propiedad de los objetos). Este modelo:

- añade a la parte estática el concepto de restricciones estáticas que son leyes que fijan las combinaciones de valores de atributos aceptados como válidas para cualquier estado,

- define restricciones de integridad dinámica que son leyes que establecen las relaciones entre los valores de los atributos en diferentes estados,
- incluye precondiciones como leyes que rigen la potencial ejecución de eventos,
- y define los disparadores como un mecanismo adicional de interacción entre objetos, que se implementa como leyes objetuales asociadas a objetos que pueden actuar como agentes, activando el evento correspondiente cuando la condición expresada en la ley se satisfaga.

La última versión de su modelo formal [Letelier (1998)], se basa implícitamente en la transaction frame logic, lo que permite describir con precisión la estructura estática y el comportamiento dinámico de los objetos

Al definir cuál formalismo a utilizar se tomaron en cuenta los principios definidos en [Hofstede-Proper (1997)]. El modelo O^3 los cumple todos, pero en la decisión incidieron fundamentalmente dos: el principio del objetivo principal y el de la ortogonalidad; que se refieren a la correspondencia entre los objetivos del formalismo y el uso que se le dará y que con un conjunto mínimo de conceptos es posible definir las bases de la formalización, respectivamente. Esta propuesta de modelo es la tomada como base en este trabajo debido a que el modelo de persistencia debe permitir obtener tanto características estáticas como dinámicas y las bases escogidas para la formalización del modelo O^3 son más adecuadas para la modelación del comportamiento dinámico ; lo que permite definir, a diferencia de otros formalismos, un grupo más completo de fórmulas dinámicas que expresan las reglas que rigen el comportamiento de los objetos. Además, es un modelo de fácil comprensión, por los términos en que se define, y logra con pocos conceptos una formalización más completa de una clase a través del uso de operadores.

3. MODELO O^3

Las definiciones, de los conceptos del modelo O^3 que se presentan, fueron tomadas de [Letelier (1998)], [Pastor (1992)], [Sánchez **et al**. (2000)] y se basan en un marco formal definido sobre una variable de la lógica dinámica que permite representar los operadores de obligación, prohibición y permiso usados en la lógica deóntica [Sánchez **et al**. (2000)]. La principal definición sintáctica es el concepto de clase.

- Clase: Una clase incluye un conjunto de atributos (A) que toman valor en el dominio asociado a cada uno de ellos. Dentro de estos atributos hay algunos que no cambian su valor durante el ciclo de vida del objeto (constantes) y otros que sí como consecuencia de la ocurrencia de eventos o por la modificación de otros atributos de los cuales dependen (variables).

  Posee un conjunto de eventos (E) que condicionan el estado del objeto, distinguiéndose los eventos *new* y *destroy***,** que representan los eventos de creación y destrucción del objeto. Para cada evento se identifica una función rango, que caracteriza el dominio de sus argumentos.

  Cada clase posee además, un conjunto de trazas o ciclos de vida (T) que, comenzando por el evento de creación, definen la secuencia de eventos que conforman la vida del objeto.

  Por último, una función de observación (α) que permite expresar el valor de los atributos que conforman la estructura de un objeto en un instante dado, en función del comportamiento (su vida). Es decir, relaciona trazas con atributos obteniéndose pares (atributo,valor) como una función de la secuencia de eventos que le han sucedido.

  Formalmente: $C = \{E,new,destroy,r,A,T,\alpha\}$

- Agregación: La agregación es por lo general la forma de representar que un objeto tiene como componentes a otros objetos. Desde el punto de vista formal se puede expresar como:

  Dadas dos clases C1, C2; con $C1 = \{E1,new1,destroy1,r1,A1,T1,\alpha1\}$ *y* $C2 = \{E2,new2,destroy2,r2,A2,T2,\alpha2\}$*, se define su agregación como una clase* $C = \{E,new,destroy,r,A,T,\alpha\}$*.*

  Donde:

E - Conjunto de eventos compartidos entre ambas clases y que son propias de la clase compleja agregada, más nuevos eventos definidos para la agregación.

r - Al definirse sobre E, se toman en cuenta los r1 y r2 de C1 y C2 relacionados con los eventos compartidos.

A - Conjunto de atributos formado por los atributos propios de la agregación y los atributos constantes que identifican a las clases C1 y C2.

α - la función de observación no guarda relación con la de las clases componentes, es una propiedad emergente de la clase agregada.

T - conjunto de trazas que están relacionadas con los eventos de la clase agregada.

Las formas en que se puede dar ésta agregación están relacionadas con la dimensión Estática/Dinámica, que se retoma más adelante.

- Especialización/generalización: Estos operadores permiten definir el concepto de herencia, que es una de las características más importantes del paradigma de la orientación a objetos. Formalmente:

  Dadas dos clases C1, C2; con C1 = {E1,new1,destroy1,r1,A1,T1,α1} y C2 = {E2,new2,destroy2,r2,A2,T2,α2}, se define su generalización como la clase C = {E,new,destroy,r,A,T,α}. En la clase C se definen los elementos de intersección de los atributos, eventos, trazas, rangos y función de observación de las clases que se generalizan.

  Dada una clase C1, con C1={E1,new1,destroy1,r1,A1,T1,α1} se define la especialización de C1 como la clase C, C = {E,new,destroy,r,A,T,α}. En la clase C1 se definen todos los elementos de C y otros propios de ella.

La semántica se define en términos de una estructura de Kripke (W, τ, ρ) [Letelier (1998)], donde W es el conjunto de todos los mundos que un objeto puede alcanzar (los mundos son estructuras sobre las que se interpretan las fórmulas dinámicas), la función τ asigna a una fórmula en la lógica de estado (la lógica de predicados de primer orden) el conjunto de mundos en los cuales se satisface, y la función ρ asigna a cada paso una relación binaria entre mundos.

Cada objeto de una clase va a encapsular su estado y las reglas que rigen su comportamiento, por lo que puede ser visto estáticamente a través de sus atributos y, desde el punto de vista dinámico, a través de las fórmulas de: evaluación, derivación, precondiciones, restricciones de integridad y disparadores (las dos primeras cambian los valores de los atributos y el resto son reglas a cumplirse por los objetos). Se definen como:

- Evaluaciones: Caracteriza explícitamente parte del estado de un objeto antes y después de la ocurrencia de una determinada acción.

  $\varphi \rightarrow [a]\phi$ Produce cambios de estado.

- Derivaciones: Son fórmulas de la forma $\phi \rightarrow \phi'$ que permiten definir atributos derivados (ϕ) en términos de una condición de derivación declarada en ϕ.

  [a] ($\phi \rightarrow \phi'$) Expresa como se obtiene el valor de un atributo derivado. Se debe satisfacer en cada estado del objeto. No produce cambio de estado.

- Precondiciones: Prohibiciones para la ocurrencia de acciones.

  $\neg\phi \rightarrow$[a]false Si $\neg\phi$ se satisface, entonces la ocurrencia de a está prohibida, es decir, ϕ es una fórmula que tiene que ser válida para que pueda ejecutarse la acción.

- Disparadores: Se disparan automáticamente, no como consecuencia de acciones de los usuarios.

  φ [$\neg\phi$]false La acción se activa cuando la condición que plantea la fórmula ϕ se satisface.

- Restricciones de integridad: Son fórmulas que deben ser satisfechas por lo que se evalúan en el modelo cuando se produce un evento que provoca un cambio de estado. Se clasifican en estáticas y dinámicas. Las estáticas siempre son aplicables, en cambio las dinámicas dependen del estado actual por lo que requieren de una lógica temporal con sus correspondientes operadores.

Donde ϕ, ϕ' y φ son fórmulas bien formadas de la lógica de predicado de primer orden y $a \in A$, A es el conjunto de acciones.

4. ODMG

El grupo ODMG surgió en 1991 con el objetivo de definir estándares para los SGBDO. Este grupo considera que el éxito de los Sistemas de Gestión de Base de Datos Relacionales (SGBDR) no solo está en las potencialidades del modelo relacional (MR), sino además en poseer un lenguaje de consulta estándar que permite un alto grado de portabilidad y simplifica el aprendizaje de un nuevo SGBDR [Cattel **et al**. (1997)]. Partiendo de esto, ODMG se propuso dotar a los clientes de SGBDO de un conjunto de normas que le permitieran escribir aplicaciones portables [Cattel **et al**. (1997)], lo que ha dado un impulso en su desarrollo. En 1993 se divulgó ODMG-93 como el estándar industrial para el almacenamiento persistente de objetos.

En 1997 aparece la versión 2.0 de ODMG que es usada como referencia por algunos gestores, por ejemplo Objectivity/DB 5.2, uno de los gestores de objetos más utilizados últimamente [11]. ODMG 2.0 está compuesto por: un modelo de objetos, un lenguaje de definición de objetos (Object Definition Language - ODL), un lenguaje de consulta (Object Query Language - OQL) y un lenguaje de manipulación de objetos (Object Manipulation Language – OML) para los lenguajes Java, Smalltalk y C++.

En enero del 2000 se publicó la versión 3.0 [ODMG (2000)]. Esta nueva versión hace hincapié en el diseño, desarrollo e implementación en una base de datos de objetos (BDO) y en productos donde se convierten los objetos al MR. Sobre esté último aspecto es importante definir estándares porque ya los SGBDO comienzan a incluir conversiones de clases a tablas, por ejemplo, Poet [Duhl-Barry (2000)].

ANSI (American National Standards Institute) lleva varios años trabajando en la versión SQL3 (Structured Query Language), que ahora se conoce como SQL:1999 [7]. ODMG 3.0 posee un lenguaje que genera un esquema que puede ser trasladado al lenguaje de definición de datos de SQL:1999 [ODMG (2000)].

5. MODELO DE PERSISTENCIA

Un modelo es un conjunto de reglas, conceptos y convenciones que nos permiten describir “algo”. Cuando se le añade el término de “persistente”, se refiere a los procesos definidos para modelar los aspectos persistentes de una aplicación orientada a objetos [Ambler (2000)].

Independientemente del medio de persistencia seleccionado para almacenar los objetos persistentes, se deben realizar un grupo de pasos que permiten completar la semántica de los objetos en aspectos que son importantes referentes a su estructura estática y comportamiento dinámico.

Los pasos que propone el modelo de persistencia para el diseño de la BD son:

1. Definir las clases persistentes.

   La persistencia es la capacidad de un objeto de mantener su valor en el espacio y en el tiempo. Es responsabilidad del diseñador definir cuáles clases son las que deben ser persistentes.

2. Refinar las clases.

   El objetivo de este paso es la obtención de la jerarquía de clases y la definición de nuevas clases. No es obligatorio que aparezcan nuevas clases, se recomienda revisar si existe algún comportamiento de interés que no haya sido tomado en cuenta y sea trascendental en la solución del problema, para incluirlo.

3. Clasificar las clases y los atributos.

   Clasificar las clases en dependencia de los atributos que la integran y los atributos teniendo en cuenta si son o no afectados por los eventos.

4. Realizar el diagrama de clases.

5. Realizar el diagrama de transición de estado.

6. Obtener las restricciones estáticas y las fórmulas dinámicas.

   Especificando textualmente o derivando a partir de los diagramas de clases y transición de estado.

7. Convertir las clases al medio de almacenamiento.
   Es en este paso en el que se tiene en cuenta hacía qué lugar se guardarán los objetos. El factor decisivo en la selección depende de las aplicaciones y los datos que maneja, pudiendo utilizarse alguna de las siguientes variantes:

   - Si se tiene un Sistema de Gestión de Base de Datos de Objetos, hay que garantizar que las definiciones obtenidas se correspondan con el modelo ODMG.

   - Si lo que se tiene es un lenguaje de programación orientado a objetos, que permite relacionarse con una base de datos relacional o un Sistema de Gestión de Base de Datos que mezcle conceptos relacionales y de la orientación a objetos, la solución es llevar las clases definidas a tablas.

   - Si lo que se tiene es un lenguaje de programación orientado a objetos, que no permite relacionarse con una base de datos relacional, pero que permite salvar registros, entonces se deben utilizar las clases que brinda para almacenar los atributos de las clases persistentes en ficheros estructurados. Puede que el lenguaje permita la relación con una base de datos relacional, pero se escoge como forma de almacenamiento la organización en ficheros.

La mayoría de las características de la perspectiva estática son capturadas gráficamente en el diagrama de clases (DC) cuya notación toma como base la del diagrama de estructura de UML [Rumbaugh **et al**. (1999)], aunque se añaden nuevas características usando estereotipos y notas.

El DC incluye la definición de restricciones de integridad estática a través de las cardinalidades de las relaciones, la dimensión E/D y el tipo de datos asociado a cada atributo. Para el resto de estas restricciones, el método DIBAO sugiere que se especifiquen textualmente completando para cada clase lo indicado en la Tabla 1.

Tabla 1. Especificación de atributos de una clase.

Atributo	Único	Nulo	Rango	Valor por defecto	Fórmula de derivación

/ /

Para el caso de numérico o enumerativo Para los atributos derivados

Las restricciones a columnas descritas en la Tabla 1, se corresponden con diferentes categorías que se referencian en la literatura. Existen otras condiciones que deben satisfacerse y que no es posible categorizarlas, por lo que se pueden definir como restricciones. Para poder determinar la necesidad de estas restricciones al dominio, es necesario especificar entonces, para todos aquellos atributos que lo requieran, reglas que indiquen qué valores pueden tomar, o lo que es igual, qué condiciones deben cumplir los atributos. El formato general que se propone en este método para expresar estas restricciones es: <atributo> = <condición>. En la condición hay que especificar las tablas involucradas y se puede usar NOT, EXIST, funciones de agregación, AND, OR, IN, BETWEEN.

Otra característica de un atributo que puede ser implementada de esta forma, es cuando el dominio restringe a un conjunto de valores predefinidos, que se pueden especificar usando el tipo de dato enumerativo. Este tipo de dato aparece por primera vez en el estándar SQL3, por lo que aún varios gestores no lo incluyen.

Las restricciones se validan cada vez que se intente cambiar el valor de los atributos a los que se asocia cada una. Su implementación depende de los predicados que brinde el lenguaje de definición de datos del gestor seleccionado. Se asocian en el DC junto a las clases usando la notación de notas.

El modelo de persistencia toma la definición de OO-Method [Pastor **et al**. (1998)] de dimensión estática/ dinámica (E/D), para las relaciones entre la clase compuesta y cada una de sus clases componentes, que se representa gráficamente usando estereotipos. Esta dimensión se define como [Letelier (1998)]:

Estática	Cuando se crea un objeto de la clase compuesta, el objeto de la clase componente se mantiene sin cambiar su valor, independientemente de los eventos que afecten la compuesta.
Dinámica	Producto de eventos que afectan a la compuesta, puede cambiar el objeto componente asociado a la compuesta.

En las asociaciones entre clases se especifica en ambos extremos de la relación las cardinalidades máximas y mínimas (<máximas, mínimas>).

Al representar la herencia se debe indicar el atributo del padre y el valor que toma, cuando el padre se especializa en cada hija. Si éste atributo no existe en la definición de la clase padre, se añade. Además se especifica si las herencias que se derivan del padre son total/parcial y con o sin solapamiento

Total: todos los objetos del tipo de la clase padre se especializan en al menos un objeto de alguna clase padre.

Solapamiento: al menos un objeto del tipo de la clase padre se especializa en dos o más objetos de clases hijas diferentes.

Para mostrar la dinámica del comportamiento de un sistema, se ha hecho uso durante años de los diagramas de transición de estado (DTE). Este modelo recomienda utilizar los DTE en función del diseño de la BD, construyéndolos para aquellas clases que posean atributos dinámicos pues en función de estos es que están los posibles estados por los que transita un objeto. Un DTE está compuesto por: estados regulares, estados agregados, estados finales, estados iniciales y transiciones.

Del DTE se pueden obtener las fórmulas dinámicas: precondiciones, disparadores y fórmulas de evaluación de los atributos que cambian su valor en el tiempo cuando ocurren determinados eventos (atributos dinámicos).

Un disparador es una sentencia que el sistema ejecuta automáticamente como efecto secundario de una modificación de la BD. No siempre un cambio de estado tiene como antecedente o evento que lo provoque, un valor específico de uno o varios atributos, por lo que todos los disparadores no se pueden obtener del DTE. El modelo de persistencia propone que la especificación de los mecanismos de disparo debe seguir el siguiente formato:

<causa que lo provoca>	<tabla>: condición]	/<acción>	{ANTES/DESPUES}
↓	↓	↓	
INSERT - adición	Nombre de la tabla	Acciones	Momento en que se
DELETE - borrado	Nombre de la tabla	que se ejecutan	producen las acciones
UPDATE - modificación	Nombre de la tabla, atributo y valor	como efecto del disparo.	con relación al cambio que provoca el disparo.

6. ¿POR QUE SE BASA EL MODELO DE PERSISTENCIA EN EL MODELO O^3?

En la implementación de los comportamientos estático y dinámico de los objetos, a través de los diagramas de clase y transición de estado y especificaciones textuales, se ha tomado como referencia las definiciones formales del modelo O^3 ya que:

- Las clases representadas en el diagrama de clases tienen perfectamente definidos sus atributos, a los cuales, este modelo de persistencia, define que se le asocie un estereotipo que indique su tipo (<<estático>>, <<dinámico>> o <<derivado>>).

- Asociadas a las transiciones se indican los eventos, que provocan los cambios de estado de los objetos vinculados a un tipo de clase, y qué clases generan esos eventos. La suma de todos los eventos definidos en las transiciones del DTE de una clase, constituyen el conjunto de eventos (E) que condicionan el estado de sus objetos.

- En la descripción de las actividades del DTE, que se ejecutan cuando se está en un estado, están definidos los pasos que se efectúan para dar un nuevo valor a el o los atributos dinámicos que se afectan y también se pueden incluir las acciones que se están haciendo mientras el objeto está en ese estado. En esta descripción se puede identificar el o los atributos que se modifican pues, directamente o como parte de un método de la clase, se utiliza una asignación en la que en la parte izquierda está un atributo de la clase, y en la derecha el nuevo valor.

  Las fórmulas de evaluación describen los posibles valores que puede tomar un atributo dinámico, a partir de un estado dado, y cómo se determina ese nuevo valor. Su especificación se obtiene siguiendo la traza del atributo en el DTE a través de los estados del objeto y las actividades que en ellos se ejecutan.

- Los eventos de creación y destrucción están contenidos dentro de los estados inicial y final, respectivamente, del DTE correspondiente a una clase

- Los operadores de agregación y especialización/generalización, se representan directamente en el DC. Las restricciones asociadas a estos operadores (dimensión, cardinalidad, relación entre padre/hijo en el árbol de jerarquía) constituyen restricciones de integridad dinámica.

- En el DC se especifica para cada atributo el tipo de datos asociado. Otras restricciones al dominio se obtienen especificando textualmente la restricción y asociándola al DC a través del símbolo de notas.

- Del DTE se puede obtener directamente las precondiciones que deben cumplirse para que ocurra un cambio de estado. Estas precondiciones quedan reflejadas, en la descripción de las transiciones, como la unión del evento que ocurre y las condiciones que deben evaluarse para provocar el cambio. Las restricciones que tienen los estados se deben cumplir para que el objeto se mantenga en ese estado, por lo tanto lo contrario de una restricción tiene que ser una condición de alguna de las transiciones de salida que se deriven de ese estado.

- Los disparadores se obtienen directamente del DTE o especificando los mecanismos de disparo.

- Las fórmulas de derivación de los atributos derivados, se definen textualmente utilizando los operadores matemáticos necesarios, valores constantes (por ejemplo, si es el doble de valor de un atributo sería dos * <valor del atributo>) y atributos (pueden ser de la clase a la que pertenece el atributo derivado o de otra clase, en cuyo caso hay que especificar de cuál <nombre de la clase>.<nombre del atributo>).

MODELO DE OBJETOS DE ODMG

El modelo de objetos es la base del estándar ODMG y el esquema de la BD [ODMG (1997)]. Este modelo plantea que un objeto está formado por propiedades y comportamiento; de manera que el estado de un objeto está definido por los valores de sus propiedades y la conducta se define por las operaciones (pueden tener parámetros de entrada y salida, y retornar o no un valor). Establece las relaciones que se pueden dar entre los objetos de herencia, asociación y composición; cómo los objetos pueden ser nombrados e identificados; y que tienen un identificador único no asociado al valor de sus atributos, sino que lo genera automáticamente el gestor objeto. Como conceptos importantes incluye, además, los de listas, colecciones, conjuntos y herencia múltiple (esto implica que no es necesario darle solución en el paso del refinamiento de las clases). Estas definiciones básicas se mantienen en la versión 3.0 [ODMG (2000)].

En el modelo de persistencia las definiciones de clases, con sus comportamientos estático y dinámico, incluyen totalmente los conceptos de este modelo y van más allá. Esto último sugiere que con el gestor y lenguaje de programación con que se trabaje tendrán que implementarse algunas características (por ejemplo, las fórmulas de evaluación, las restricciones asociadas a las relaciones entre clases).

En el análisis de cómo el modelo de persistencia se basa en el modelo O^3, se describe cómo se obtienen los conceptos asociados a las propiedades y comportamiento de los objetos.

Este modelo de persistencia se inserta dentro de la etapa de diseño de la metodología ADOOSI-UML (Análisis y Diseño Orientado a Objetos de Objetos utilizando la notación UML) [2], desarrollada en nuestra universidad y de gran uso en el país. Con anterioridad al diseño de la base de datos, se han identificado las clases del dominio del problema con sus atributos y responsabilidades, usando como herramientas los diagramas de casos de uso y de interacción, y definiendo el modelo conceptual de acuerdo al objeto de automatización.

CONCLUSIONES

En estos últimos años se ha avanzado en la fundamentación del modelo orientado a objetos. Dada la complejidad del proceso de formalización [Hofstede-Proper (1997)] y no ser objetivo de este trabajo la elaboración de un marco formal para el MOO, se considera que el modelo O^3, desarrollado en la UPM, se adecua a los requerimientos del modelo de persistencia que se propone.

Estática y dinámica son dos perspectivas importantes que han de tenerse en cuenta cuando se diseña la BD. Por un lado se busca definir estructura y por el otro cambios permitidos sobre esa estructura. Obviar alguna de estas vistas en el proceso de diseño nos puede llevar a una BD inconsistente. El modelo de persistencia describe cómo representar estas características utilizando los diagramas de clase y transición de estado y formatos de especificación de algunas restricciones de integridad y fórmulas dinámicas.

El modelo de objetos que se obtiene como resultado, cumple las características definidas por ODMG y, además, si se le añade el comportamiento dinámico, entonces podemos decir que también cumple lo referente al marco formal definido en O^3.

REFERENCIAS

ABITEBOUL, S.; R. HULL and V. VIANU (1995): “Foundation of Database”. Addison-Wesley Publishing Company.

ALVAREZ, S. and A. HERNANDEZ (2000): “Metodología ADOOSI-UML, versión 5.0”, Centro de Estudios de Ingeniería y Sistemas, Instituto Superior Politécnico “José Antonio Echeverría”, Cuba.

AMBLER, S. (2000): “Mapping objects to relational databases”. October 21. http://www.Ambysoft.com/mappingObjects.pdf.

ANDRADE, L.F.; J.C. GOUVEIA; P.J. XARDONE and J.A. CÂMARA (1999): “Architectural concerns in automating code generation”. OBLOG Software S.A. http://www.oblog.com/news/concerns.html.

CATTEL, R. (1997): “ODMG 2.0”. Morgan Kaufman. http://www.odmg.org.

DUHL, J. and D.K. BARRY (2000): “Object storage Fact Book 4.1, Object-relational mapping”. Mapping object to external DBMSs, 132. Barry & Associates, Inc. . http://www.objecr-relational.com/ormapping2.pdf.

EISENBERG, A. and J. MELTON (1999): “SQL:1999, formerly knows as SQL3”. **SIGMOD Record**. 28(1). 131-138. March. http://www.cssinfo.com/ncits.html.

FREDERIKS, P.J..M.; A.H.M. ter HOFSTEDE and E. LIPPE (1997): “A Unifying Framework for Conceptual Data Modelling Concepts “. **Information and Software Technology**. 39(1).15-25. January. http://www.icis.qut.au/~authur/articles/ HowTo.ps.Z.

HOFSTEDE, ter A.H.M. and H.A. PROPER (1997): “How to Formalize It? Formalization Principles for Information Systems Development Methods”. **Technical report #4/97**, Faculty of Information Technology, Queensland University of Technology, Brisbane, Australia. June. http://www.icisqut.edu.au/~authur/ Work.ps.Z.

HUBBERS. W.G.M. and A.H.M. ter HOFSTEDE (1998): "Exploring the Jungle of Object-Oriented Conceptual Data Modeling", In Chris McDonald, editor, Proceedings of the 9th Australasian Database Conference, ADC'98, volume 20(2) of **Australian Computer Science Communications**, 65-76. Perth, Australia, February, Springer. Berlin.

JAVA (2000, Junio): http://java.sun.com/features/2000/06/javapro.html.

JUNGCLAUS, R.; G. SAAKE; T. HARTMAN and C. SERNADAS, C. (1995): "Troll- a language for object-oriented specification of information systems", **ACM Transactions on Information systems** 14(2). 175-2111. 1995

JUNGCLAUS, R.; R.J. WIERINGA; P. HARTEL; G. SAAKE and T. HARTMAN (1994): "Combining Troll with OMT". B. Wolfinger (ed.). Innovation bei Rehen- und Komminikationssystems, Springer-Verlag.35-42.1994.

LETELIER, P. y otros (1998): "Oasis versión 3.0: un enfoque formal para el modelado conceptual orientado a objetos:. Servicio de publicaciones, DSIC-UPV, España.

ODMG (1997): "Standard overview. ODMG 2.9-Book extract". March. http://www.ddj.com/articles/1997/971/917ª/9717a.htm.

ODMG (2000): "Standard overview". 2000. http://www.odmg-org/standard/standardoverview.htm.

PASTOR, O. (1992): "Diseño y desarrollo de un entorno de ejecución automática de software en el modelo orientado a objetos". Tesis doctoral. DSIC-UPV, España. Abril.

PASTOR, O.; V. PELECHANO; B. BONET e I. RAMOS (1996): "OO-Method 2.0: una metodología de análisis y diseño orientado a objetos". Reporte de investigación DSIC,UPV. España.

RAMOS, I.; J.H. CANOS; J. FORRADELLAS and J. OLIVER (1990): "A conceptual schema specification system for Rapid Prototying". **Proceedings of the XI ASTED Conference on Applied Informatics**, Insbruck. February.

RUMBAUGH, J.; I. JACOBSON, I. and G. BOOCH (1999): "The unified modeling language: reference manual". Addison-Wesley Longman, Inc. Canadá.

SANCHEZ, P.; P. LETELIER; I. RAMOS y O. PASTOR (2000): "Modelo conceptual con un lenguaje formal y orientado a objeto". DSIC-UPV. España.

SERNADAS, C. and J. FIDIERO (1991): "Toward object-oriented conceptual modeling". **Data& Knowledge Engineering**. 7, 479-508. 1991.

TSCHRITZIS, D. and F. LOCHOVSKY (1982): "Data models". Prentice Hall, Inc. New York.

TUIJN,CH. and M. GYSSENS (1996): "CGOOD, a categorial graph-oriented object data model", **Theoretical Computer Science** 160. 217-239.

WIERINGA, R,J. (1990): "Algebraic Foundations for Dynamic Conceptual Model", Centrale Huisdrukkerij Vrije Universiteit.