LA TECNOLOGÍA ORIENTADA POR OBJETOS: SUS PRINCIPIOS Y MI EXPECTATIVA

Fernando Ruiz Rojas

"...los programas de computadoras realmente buenos viven para siempre y nunca terminarán de escribirse, al menos mientras dure la máquina, e incluso más".

Charles Simonyi (autor de Multiplan, Word, Excel) [LAMMERS 86]

Prólogo del Dr. Eduardo Aldana Valdés a mi Investigación:

"Algo muy difícil es ser original, y más difícil es que la originalidad tenga valor económico".

1. Resumen

Este artículo pretende dar una visión general de los Sistemas de Bases de Datos Orientados por Objetos (BDOO) y presenta la especificación funcional de Un Sistema Manejador de BDOO (SMBDOO) en fase de desarrollo.

Primero se enuncian los principios del Paradigma Orientado por Objetos, luego se definen las BDOO, se justifica su aparición se describe el estado actual de esta área de investigación. Por ultimo se presentan los resultados de un trabajo de investigación, que pretende establecer la especificación funcional de un SBDOO para aplicaciones de diseño asistido por computador (CAD) de propósito específico.

2. Introducción

Los modelos de datos tradicionales han sido divididos en tres: jerárquicos, en red y relaciónales. Para cada un de estos existe un gran numero de sistemas manejadores de bases de datos (SMBD) comerciales. En ellos los problemas directamente relacionados con el manejo eficiente de los datos como:

persistencia, acceso compartido y seguridad, ya han sido tratados y solucionados.

El diseño y la implementación de estos sistemas se han enfocado a suplir los requerimientos de las aplicaciones de tipo comercial. Por lo tanto, no resultan adecuados para modelar de una manera natural las estructuras complejas de entidades no tradicionales que incluyen las nuevas aplicaciones (CAD[i], CASE[ii], AI[iii] Y OSI[iv]) Estas nuevas aplicaciones difieren de las tradicionales en varios aspectos.

Por una parte cada área de aplicación requiere un conjunto diferente de herramientas de Modelaje, por ejemplo, los tipos de entidades y las relaciones que deben definirse para un sistema de automatización de oficinas.

Por otra parte, cada nueva área de aplicación contiene un conjunto especializado de operaciones que debe ser manejado eficientemente por el SMBD. Es claro que el tratamiento de imágenes en texto dentro de un ambiente OSI es diferente al tratamiento de visualización de sólidos en un ambiente CAD o al tratamiento de inferencias de un ambiente de Inteligencia Artificial.

Estos nuevos requerimientos exigen modelos de datos poderosos, que permitan reducir la brecha semántica entre las bases de datos (BD) y el mundo real que estas modelan. Entre las consecuencias más importantes que traen estos nuevos requerimientos se pueden citar:

- Primero la necesidad de modelos y lenguajes de definición y manipulación de datos expresivos.
- Segundo la definición de nuevos métodos de acceso y recuperación. Los anteriores requerimientos de información, en los que los datos tienen estructura compleja, es decir no pueden ser descritos en forma natural y eficiente con campos de longitud fija y predeterminada; Y cuyas relaciones no se pueden modelar fácilmente en tablas planas, han llevado a la producción de una gran variedad de modelos de datos. Los modelos de datos orientados por objeto (OO) son una muestra representativa dentro de esta nueva tendencia.

---

[i] Computer Aids Design
[ii] Computer Aids Software Engineering
[iii] Artificial Intelligence
[iv] Open Systems Interchange Information

Durante los últimos años, el interés de los investigadores de BD, se han enfocado a solucionar estas necesidades. Los conceptos relacionados de SMBD han comenzado a extenderse, y en algunos casos a cambiarse con el objetivo de cubrir aspectos relacionados con la representación y el manejo de objetos con estructura compleja. De esta manera surgen las bases de datos orientadas por objeto (BDOO).

Sin embargo, los modelos OO aun no tienen madurez, el estado actual de fuerte experimentación, conduce a que en la mayoría de los casos el usuario final seleccione un SMBDOO dentro de los existentes, no porque sea el mejor o el que más se adapte en sus requerimientos, sino porque está disponible al mercado.

Debido a que la especificación del nivel interno, también conocido como nivel físico, se encuentra en proceso de definición, en este articulo solo se presenta la especificación para los niveles de aplicación y conceptual. Sin embargo, el último objetivo de esta investigación es la implementación de un manejador de objetos, que corresponde a la especificación del nivel interno de SMBDOO. Este manejador será probado con una aplicación CAD de propósito específico.

3. El paradigma orientado por objetos

El paradigma OO es un de los más exitosos en muchas áreas de la ciencia computacional. Aunque sus principios datan de un tiempo atrás, es durante la última década cuando han logrado gran aceptación dentro de áreas como Bases de Datos, Inteligencia Artificial y lenguajes de programación.

A pesar de que las diferentes aplicaciones del paradigma han producido diferentes definiciones, los siguientes principios pueden considerarse esenciales.

Objeto:

"Un objeto es la representación formal de un elemento del mundo real, es la abstracción de una entidad autónoma y coherente del mundo de interés" (VILLALOBOS 89).

Encapsulamiento:

Un objeto esta compuesto por una memoria privada y un conjunto de métodos u operaciones, la agrupación de datos y operaciones se conoce comúnmente como Encapsulamiento. De esta forma un objeto tiene una parte de Interfase y una parte implementación. La parte de Interfase

corresponde a la especificación de un conjunto de operaciones que pueden realizarse sobre él. La parte de implementación tiene una parte de datos y una de operación. La parte de datos es la memoria del objeto, es decir la colección de atributos, mientras que la parte de operación describe en algún lenguaje de programación la interpretación de cada operación.

Mensajes:

Los objetos se comunican a través de mensajes. Los mensajes y sus parámetros constituyen la interfase pública del objeto. Para que un Mensaje sea comprendido por el objeto receptor, debe existir un método correspondiente que lo reconozca. El objeto receptor de un mensaje reacciona ante el estimulo ejecutado por el método correspondiente y retorna un objeto como respuesta.

Objetos complejos:

La memoria privada de los objetos tiene una forma de tupla de pares (atributo: objeto). El hecho de que los objetos ligados a los atributos que puedan ser de cualquier tipo, permite la definición recursiva de objetos complejos cuyos atributos pueden tomar como valor otros objetos, incluidos objetos de su mismo tipo.

Clases:

Una clase es un patrón de objetos. En tiempo de ejecución actúa como depósito del conjunto de objetos con las mismas características. En tiempo de compilación actúa como constructor de tipo asociado con cada clase: La noción de clase corresponde con la noción de TAD: Tipo Abstracto de Datos (TAD) (BERT 89).

Instancias:

Los miembros de una clase se llaman instancias. La relación "ES UN" relaciona las instancias con su clase y es análoga la relación de membresía en teoría de conjuntos (JACK89).

Herencia:

La herencia permite construir nuevas clases sobre otras ya existentes menos especializadas (VILLALOBOS 89). Este mecanismo permite que objetos de diferentes clases compartan atributos y operaciones relacionadas con sus partes comunes (BANC89). La herencia permite establecer jerarquías de clases entre los objetos: Al crear una nueva clase, como una especialización de otra ya existente, la subclase es la nueva clase y la

superclase es la clase de la cual se hereda. Los atributos de la subclase son la unión de los atributos de la súper clase y de sus atributos, de la misma manera se tratan los métodos.

La herencia es simple cuando una subclase tiene una y solo una superclase, en caso contrario la herencia es múltiple. La relación "ES CLASE DE" relaciona las subclases con sus súper clases y es análoga a la relación de contenencia en teoría de conjuntos (JACK86).

(BERT89) describe las siguientes características adicionales, obtenidas a partir del concepto de herencia; es importante dentro de este paradigma.

Redefinición explícita:

Permite la redefinición de operaciones dentro de las clases de jerarquía. Una redefinición puede hacerse por necesidad o por eficiencia. Por necesidad: cuando el comportamiento de la subclase es diferente al comportamiento de la superclase; por eficiencia: cuando puede desarrollarse una operación más eficiente o especifica de la subclase.

Polimorfismo:

Polimorfismo es la habilidad de que diferentes clases de objetos respondan a los mismos métodos.

El polimorfismo define el grado de flexibilidad del sistema permitiendo que las variables tomen más de una forma en tiempo de ejecución. El polimorfismo de variables en OO es controlado por el mecanismo de herencia, una variable de tipo C puede referirse a un valor de tipo C^ que es una instancia de cualquier subclase de C^ de C.

Alcance dinámico:

Esta característica determina, en ejecución, a cual de las formas de un elemento polimorfo se refiere en una operación. El alcance dinámico complementa la redefinición explicita. Garantiza que cuando una operación es aplicada a una variable polimorfa y la operación tiene más de una definición, se seleccione la versión de la operación más apropiada. Esto con base en el tipo actual de valor de la variable.

Por ejemplo, la aplicación de método perímetro de una variable polimorfa declarada de tipo POLÍGONO, deberá disparar la ejecución más apropiada dependiendo del valor actual de la instancia de la variable. La instancia se

refiere a un valor de tipo CUADRADO y el perímetro esta redefinido para CUADRADO, este será el método que se invoca.

Genericidad:

La Genericidad permite que las clases sean parametrizadas con tipos de datos (VILLALOBOS 89). Esta característica permite una extensión "horizontal" para clases (opuestas a la extensión "vertical" que se logra mediante la herencia. Por ejemplo se puede definir un tipo genérico LISTA (T), en el que el parámetro T define el tipo de elementos de la lista.

4. Bases de datos orientadas por objeto

Los SMBOO son el resultado natural de la integración de dos tecnologías: el paradigma OO y las BD. En la actualidad existen muchas investigaciones sobre BDOO y lenguajes relacionados. Los campos de investigación más importantes en esta área incluyen el desarrollo de modelos de datos, manejo de versiones, identidad de objetos, interfaces de alto nivel, mecanismos de protección y acceso concurrente para objetos, manejo de transacciones y herramientas para el diseño y administración de bases de datos orientadas por objeto(PETE87).

Motivación

El interés en la OO como base para el diseño y la implementación de SBD, se basa en el reconocimiento de las siguientes razones:

- La necesidad de que la estructura del sistema sea una representación adecuada y aproximada del modelo del mundo real y del problema. Un sistema debe soportar la definición y manipulación de objetos complejos, en relación uno a uno con los objetos del mundo, sin necesidad de transformarlos en los registros planos y relacionarlos, como lo exige el modelo de datos relacional.
- El hecho de que la parte más cambiante en los sistemas de información son las acciones realizadas sobre los objetos (VILL89). Un cambio en la especificación debería corresponder a un cambio puntual de los programas, no de los desarrollos convencionales guiados por funciones.
- La necesidad de que la Metodología de desarrollo de sistemas de información corresponda con el lenguaje de programación. Ambos aspectos deben manejar los mismos conceptos, una traducción natural facilitaría la correlación e implantación de los sistemas (BANC89).
- Los sistemas relacionales, en general tienen un lenguaje de definición de datos (DDL) y un lenguaje de manipulación de datos (DML), pero no

tienen un lenguaje de definición de aplicaciones. El desarrollo de aplicaciones requiere la comunicación entre el lenguaje de programación tradicional (orientado al registro) y un DDL o DML (orientado al conjunto), hecho que produce la mezcla de dos tratamientos computacionales diferentes "El modelo orientado por objeto elimina esta separación; Hay un modelo único para datos y operaciones que integra la definición de la BD con el lenguaje de manipulación del dato" (BANC89).

- La necesidad de hacer sistemas extensibles: Los sistemas bien estructurados deben permitir su extensión sin necesidad de rehacer excesivamente el software existente (BERT89).
- La necesidad de tener ambientes que faciliten el desarrollo y la implementación de aplicaciones. Las herramientas de 4ª generación son un buen ejemplo de ambientes para el desarrollo eficiente de aplicaciones en SBD. Ellas permiten la generación automática de reportes y formularios de captura de datos. Sin embargo no eliminan el tratamiento diferente entre datos y operaciones, y en algunos casos producen programas de aplicación poco eficientes.

4.1 Direcciones de la investigación: La investigación en el campo SBDOO se ha enfocado en las siguientes tres direcciones:

- Extensión del modelo relacional: En este caso se intenta hacer las adiciones apropiadas al modelo relacional apropiadas al modelo relacional, con el fin de que permitan la definición de objetos complejos. Como ventaja de este enfoque se puede citar el hecho de tener un modelo de base ampliamente conocido y para el que existen lenguajes de soporte eficiente y también conocido.

  Sin embargo la extensión de los lenguajes de soporte, con el consecuente recargo en su sintaxis. Este hecho además de obscurecer la solución planteada puede terminar en la producción de sistemas con ambientes pesados, sintaxis complicadas y difíciles de mantener. Y lo peor del caso, la manipulación de los datos sigue restringida a las operaciones clásicas: formación de conjuntos e inserción, actualización y borrado de sus elementos.

  Como ejemplos de este tipo de solución se pueden citar: POSTGRES, proyecto de investigación de la universidad de California en Berkeley y GENESIS, proyecto Starbust en IBM almadien research center.

- Extensión de los conceptos de POO: En este caso se intentan incluir mecanismos de persistencia, seguridad, consistencia y acceso compartido de datos entre los conceptos de POO.

Dentro de las ventajas de este enfoque se tienen en los lenguajes OO permiten la definición natural de objetos complejos son una herramienta de diseño poderosa.
Sin embargo como desventajas se pueden citar: Primero, no existe un único modelo de datos orientado por objeto coherente, con un fundamento matemático, tal como el cálculo o el álgebra relacional en base de datos relaciónales. Segundo, los conceptos de POO son poco conocidos fuera del ambiente académico. Y por otra parte el ampliar el paradigma OO para incluir aspectos mencionados inicialmente podría obscurecer la riqueza semántica del modelo de base.
Como ejemplos de este enfoque están: GEMSTONE, servidor de BDOO desarrollado por servio Logic Development corp, y JASMINE: prototipo en desarrollo en la Universidad de Washintong Seattle. (PETE87).

- Creación de nuevos sistemas: En este caso se intenta la integración de las tecnologías mencionadas por el objetivo de definir un nuevo y único sistema.
  Como ventaja de este enfoque puede anotarse la especificación propia del SBDOO. Esto significa, la definición de un modelo de datos orientado por objeto, de su conexión con un lenguaje de programación (o la integración del lenguaje de programación en el modelo) y la determinación de los aspectos dinámicos del sistema.
  En este caso la desventaja se materializa en la necesidad de desarrollar, implementar y optimizar toda la tecnología de soporte. Se deben definir tanto los algoritmos para implementar el sistema, como la teoría para enriquecerle el entendimiento del modelo y del mismo sistema.
  Un ejemplo de este enfoque es O2: prototipo manejador de objetos desarrollado por Gip Altair en Francia (BANC89).

4.2 Estado actual de este campo de investigación: El estado de este campo de investigación esta caracterizado por tres hechos principales:

- La ausencia de un modelo de datos común: No existe aun una definición universal de un modelo de datos y menos de lo que es un SBDOO.

- La ausencia de un modelo de datos coherente: No existe un modelo con la teoría matemática de soporte. Este problema tiene graves implicaciones que incluyen en otras la dificultad de especificar consultas y asegurar consistencia. La necesidad de una teoría es obvia, la semántica de tipos de programas y de objetos está aún pobremente definida.

La fuerte actividad experimental: Los investigadores que están construyendo sistemas prototipos simultáneamente especifican los nuevos sistemas y la tecnología necesaria para soportar su implementación. Se corre el riesgo "que uno de estos prototipos surja y se convierta en modelo, no porque sea el mejor sino porque es el primero". (BANC 89).

4.3 Especificación funcional de un símbolo para ambientes CAD

Uno de los principales intereses del grupo de investigación DFAC (Diseño Fabricación apoyados por Computador de la universidad de Los Andes) es nstruir ambientes de diseño de propósito específico: Para el buen desempeño estos ambientes se requiere incorporar en ellos aspectos relacionados con manejo eficiente de información. Por esta razón el grupo de investigación NBAD (Sistemas de Información y Bases de Datos distribuidas de la misma iversidad) pretende establecer en una primera fase de desarrollo, la pecificación funcional de un SMBDOO para ambientes CAD.

La especificación del SMBDOO tiene como base el esquema triple de la quitectura de los sistemas de bases de datos, propuesto por el grupo ANSI/ ARC/X3 en 1971. La arquitectura propuesta consta de tres niveles uno externo, o conceptual y uno interno (CHEN76) debido a que la especificación del nivel erno, también conocido como nivel físico, se encuentra aun en desarrollo, en a sección solo se presenta la especificación para los dos primeros niveles:

4.3.1 Nivel de aplicación: Según (DATE 86) el nivel extremo o de licación representa el punto de vista del usuario. Es una descripción de s datos visibles a un programa de aplicación en términos de nombres y racterísticas. Esta sección esta organizada de la siguiente manera:

Primero se definen las características relevantes e independientes de la aplicación de diseño asistido por computador (CAD).
Luego se presenta la especificación funcional del nivel externo o de aplicación.
Después se establecen los requerimientos mínimos que debe cumplir este nivel para satisfacer la especificación funcional antes definida.

4.3.2 Características: Las aplicaciones de diseño asistidas por mputador tienen en general las siguientes características:

Los objetos que se manejan tienen estructura compleja. Un objeto típico es representado como una colección de información de diferentes tipos.

Estos objetos constituyen las unidades de recuperación y almacenamiento y son la base para asegurar la integridad.

- Incluye una jerarquía de objetos componentes. Los diseños pueden ser implementados reutilizados o previamente definidos.
- Existe un alto contenido grafico. Un aspecto importante del diseño es la visualización de los objetos comprometidos con el mismo.
- Requieren alta interacción usuario sistema, se desea un ambiente de desarrollo flexible, es decir que facilite crear, actualizar, modificar, recuperar y visualizar los componentes del diseño en todas sus fases de desarrollo.
- Están relacionadas con la diversidad de puestos de trabajo. Resulta deseable que la información producida en cada fase de un diseño sirva como base de retroalimentación de las fases subsiguientes. Un diseño representa la integración de múltiples vistas, cada una de las cuales denota algún aspecto del diseño.
- Se manejan grandes cantidades de información: como los diseños se desarrollan por evolución, la información se enriquece gradualmente a medida que los proyectos alcanzan madurez. La inclusión de detalles de diseño, de composición de partes y el ensamblaje de modelos aumentan el volumen y la complejidad de la información que debe manejar el sistema.
- Contienen transacciones largas y anidadas: las transacciones en ambientes CAD cubren desde la iniciación hasta la terminación (entrega y puesta en marcha) de un proyecto: Una transacción puede irrumpirse en estado inconsistente y continuarse cuando el usuario lo desee.

4.3.3 Especificación funcional: las aplicaciones CAD intentan ofrecer al diseñador, a quien en adelante se llamara usuario final, un ambiente de desarrollo que cumpla, por lo menos con las siguientes funciones:

- Ofrecer el conjunto de objetos básicos que representan el mundo de interés de la aplicación científica.
- Hacer accesible para cada objeto el conjunto de métodos que describen su comportamiento en relación con el proceso de diseño mismo.
- Ofrecer un conjunto de operaciones genéricas que permitan la manipulación de los objetos definidos: Entre las operaciones genéricas están: crear, borrar, modificar (todas referidas a instancias de objetos de las clases definidas.
- Ofrecer un conjunto de operaciones que permitan la visualización de los objetos. Se requiere entonces un modulo especializado de visualización que realice la representación grafica de objetos.

- Lograr persistencia de los objetos de tal manera que se garantice su contenido semántico, sin atrofiar su complejidad o estructura.

4.3.4 Requerimientos: El nivel externo o de aplicación de una BDOO lo constituye la herramienta de diseño. Esta herramienta es una capa de software sobrepuesta a nivel conceptual y para cumplir la especificación funcional definida en el numeral anterior, este nivel debe satisfacer los siguientes requerimientos:

- Manejar la correspondencia entre los nombres (dados por el usuario final) Todos los objetos definidos por el usuario corresponden a la instalación de los objetos definidos en el sistema, por tanta son manejados de acuerdo con la definición de la clase correspondiente en el nivel conceptual.
- Garantizar el principio de Encapsulamiento de datos, en el sentido que solo es posible conocer el contenido mediante la invocación de los métodos definidos para su clase en el nivel conceptual.

Puede decirse entonces, que una BDOO es una colección de objetos, cada uno de los cuales es una instancia de una clase en particular. Cada instancia definida en el nivel de aplicación debe tener una llave única que se usa como identificador tanto para su recuperación, como para definir las relaciones que este objeto guarda con otros objetos.

Al contrario que en los SBD tradicionales, el concepto de vista parcial no existe. En este caso, el usuario final cuenta con un conjunto de objetos definidos a partir de los cuales diseña su modelo, sin embargo podría encontrarse que en un sistema de diseño y manufactura asistido por computador (CAD/CAM) el modelo de datos de aplicación esta compuesto por varios submodelos. En este caso los objetos constitutivos de cada submodelo definen una vista. El acceso a cada vista depende del estado en que se encuentre el modelo del diseño en un momento determinado.

Aunque a lo largo del proceso de diseño el modelo se enriquece, en cada estado solo se tiene la posibilidad de manipular (crear, borrar, consultar y/o modificar) los objetos directamente relacionados con este estado. Esto significa que las anteriores decisiones de diseño ya han sido tomadas.

Bajo estas condiciones se destaca la importancia del anejo de versiones de objetos. El concepto de Versión permite generar múltiples posibilidades de diseño a partir de diferentes versiones o alternativas de objetos, antes de seleccionar un modelo definido.

4.4 Nivel conceptual: "El nivel conceptual representa el punto de vista de la empresa en términos de sus entidades, atributos y relaciones entre entidades" (DATE86), este nivel contiene la descripción completa del modelo de datos en el sentido que define los requerimientos para las operaciones permitidas, para la integridad semántica y para la seguridad de los datos. En esta sección se tratan dos aspectos fundamentados para la especificación del nivel conceptual: el modelo de datos y el diseño lógico de la base de datos.

En la primera parte de esta sección se presenta la especificación funcional del nivel conceptual. En la segunda parte se enuncian los requerimientos de este nivel. Por ultimo se presenta nuestro modelo de datos

4.4.1 Especificación funcional: El nivel conceptual permite la definición del modelo de datos del mundo de interés: en términos de BD este modelo se conoce como esquema conceptual. Corresponde al diseño lógico de la BD y esta soportado por un modelo de datos. Para que se pueda diseñar completamente el esquema conceptual, el nivel conceptual deberá incluir las siguientes definiciones.

- Un modelo de datos. En el caso de SMDBOO, el esquema conceptual debe estar soportado por un modelo de datos OO. Es decir por un modelo que incluya por lo menos los principios fundamentales del paradigma OO enunciados en 1 (objetos, clases y mensajes). El hecho anterior requiere la definición de un usuario especializado que defina el esquema conceptual de la aplicación. Este usuario en adelante se llamara Diseñador del Esquema Conceptual (DEC) y además de conocer el paradigma de OO debe tener la relación directa con los expertos en el dominio de la aplicación.
- Un esquema de representación o esquema conceptual, que soporte el modelo de datos OO propuesto y permita al DEC representar las abstracciones del mundo de interés (representadas en el modelo de datos).
- Un conjunto de operaciones permisibles sobre el esquema y el estudio de sus implicaciones sobre la consistencia del mismo. Estas implicaciones definen un conjunto de reglas que gobiernan el esquema conceptual propuesto: Las reglas así definidas se pueden considerar como los invariantes del esquema conceptual.

4.4.2 Requerimientos: Los conceptos de diseño, principal componente de las aplicaciones CAD; incluyen un conjunto de conceptos abstractos, identificados como requerimientos conceptuales de diseño independientes del dominio. Estos conceptos permiten identificar las siguientes dimensiones fundamentales dentro del esquema conceptual de una aplicación CAD:

- Nivel de diseño de entidad, A este nivel debe ser posible diseñar un objeto individual, lo que equivale a crear una clase de diseño.
- Nivel de diseño estático. A este nivel debe ser posible definir la estructura estática de los objetos de diseño, su interfase y su implementación. Lo que se pretende con la definición de este nivel es permitir que el DEC defina el esquema conceptual de diseño.
- Nivel de diseño dinámico. A este nivel debe ser posible definir la estructura dinámica y evulocionaria de los objetos del diseño. Lo que pretende con la definición de este nivel es permitir al usuario final el manejo de versiones de objetos. Para lograrlo se requiere incluir en el modelo de datos la descripción de una estructura evolucionaria, es decir, el predecesor y el sucesor de cada clase definida, así como su relación con las demás clases del diseño.

4.4.3 Modelo de datos: Como se menciono no existe la definición universal de un modelo de datos OO. Los prototipos de SMBDOO existentes muestran diferentes enfoques y funcionalidades en este aspecto. Nosotros tomamos como base el modelo de datos implícito en INDUCON "un sistema de apoyo a la industria de la construcción" aplicación en fase de desarrollo por el grupo DFAC.

En INDUCON un objeto define su nombre para denotar una clase de objetos y un conjunto de atributos que describen su estado interno. Los objetos se encuentran estructurados en tres niveles:

1. Objetos Básicos: Son aquellos que contienen atributos de tipo elemental (int., real, float, char, bolean, long, double).
2. Objetos compuestos: Son aquellos que contienen dentro de sus atributos por lo menos un objeto básico, en general contienen conjuntos de objetos básicos.
3. Objetos encapsuladores: Son aquellos que incluyen objetos tanto básicos como compuestos. Tienen como característica adicional que sirven de cierre de contorno entre los conjuntos de todos los objetos del sistema.

Adicionalmente cada objeto tiene un conjunto de métodos que permiten la simulación de su comportamiento, tal como el OO los método definidos constituyen la interfase del objeto. Tanto las estructuras de representación abstracta de una clase, como el código de los métodos asociados a ella se encuentran en un mismo archivo (de esta manera se simula la creación de clases y el principio de Encapsulamiento).

A continuación se presenta el MODELO PROPUESTO, que se basa en INDUCON pero incluye algunas abstracciones de las que muestran otros modelos. Vale la pena notar que el modelo no es completo, solo muestra los aspectos de interés para el almacenamiento y acceso de los datos.

Tipos: La definición de clases en el modelo propuesto este soportada por la definición de tipos:

- Como base para la construcción de tipos se define el conjunto de tipos básicos: int, char, double, signed y unsigned (note que estos tipos corresponden con los tipos básicos que ofrecen el lenguaje C –lenguaje de implantación de INDUCON–.
- A partir de los tipos básicos es posible definir tipos compuestos. Un tipo compuesto es un conjunto de parejas (atributo: tipo) donde cada atributo tiene asociado un tipo Básico es compuesto.
- Todo tipo Básico es compuesto.
- Un atributo puede ser multivaluado, es decir puede tener asociado un conjunto de valores. Por esta razón se introduce un tipo especial llamado tipo lista que permite modelar conjunto de valores asociados a un atributo.

Clases: Dentro del modelo una clase es un patrón de objetos: su definición comprende:

- La asignación de un nombre para denotarla.
- La definición de un tipo asociado a la clase que determina la estructura interna de los objetos de la clase, es decir que determina las variables de instancia que caracterizan los objetos de la clase.
- La definición de un conjunto de métodos que determinan el comportamiento de los objetos de la clase. En este caso el conjunto de métodos corresponde a procedimientos implantados en lenguaje C.
- Un objeto es una instancia de una clase. Para que un objeto sea conocido dentro de un ambiente debe tener un nombre que lo identifique.

Relaciones de Interés: Dentro del modelo de datos la s relaciones entre objetos representan las siguientes abstracciones.

- Instanciación: Un objeto puede ser considerado como un tipo especial de relación que agrupa objetos con las mismas características, esta relación es la más importante y existe inherentemente en el modelo.

Recordemos que la definición de un objeto denota un nombre para identificar una clase de objetos con las mismas características.

- Generalización y particularización: Esta abstracción permite relacionar los objetos mediante una jerarquía de especialización, donde cada subclase es una particularización de su superclase, o en sentido contrario, cada súper clase es una generalización de sus correspondientes subclases. La definición de esta jerarquía de especialización para efectos de eficiencia en implantación tiene como requerimiento inmediato la inclusión del mecanismo de herencia dentro del modelo de datos.
- Agregación: Un objeto puede considerarse (generalmente más simples. Esta abstracción permite relacionar los objetos mediante una jerarquía de composición en la que un objeto agregado incluye referencias o sus objetos componentes. En el modelo los objetos constructores (conjunto lista) permiten la composición de objetos.
- Esquema Conceptual: El esquema conceptual de la BDOO propuesta lo constituye la raíz de la jerarquía de composición. (Note que es el equivalente a un objeto encapsulador del modelo INDUCON). Un objeto de este tipo define la estructura completa de agregación y especificación determinada por el diseñador del modelo conceptual de la aplicación.

  Vale la pena notar que cada instancia nombrada de la raíz de la jerarquía de composición denota una BD independiente. En el estado actual del prototipo el ambiente de desarrollo es monousuario, por lo tanto no hay acceso compartido a los datos. Sin embargo, sé esta considerando la posibilidad de crear instancias exportables. Una instancia de un OBJETO definida como exportable, podrá ser importada (accesada) desde cualquier instancia de cualquier otra aplicación o proyecto (de INDUCON claro está).
- Evolución del esquema conceptual. Debido a que nuestro SMBDOO el usuario final es decir el diseñador no tiene la posibilidad de definir nuevos objetos la evolución del esquema se realiza únicamente al nivel de diseño del esquema conceptual.
- En un modelo no se cuenta con ningún lenguaje OO, el esquema conceptual que representa el modelo de datos propuesto por INDUCON esta interpretado en C. Sin embargo las siguientes reglas son aplicables para garantizar la consecuencia semántica del esquema:
- Todos los acuerdos (instancias de cualquier clase definida por el diseñador de la aplicación) son independientes es decir, la existencia de un objeto no esta condicionada por la existencia de otro objeto.

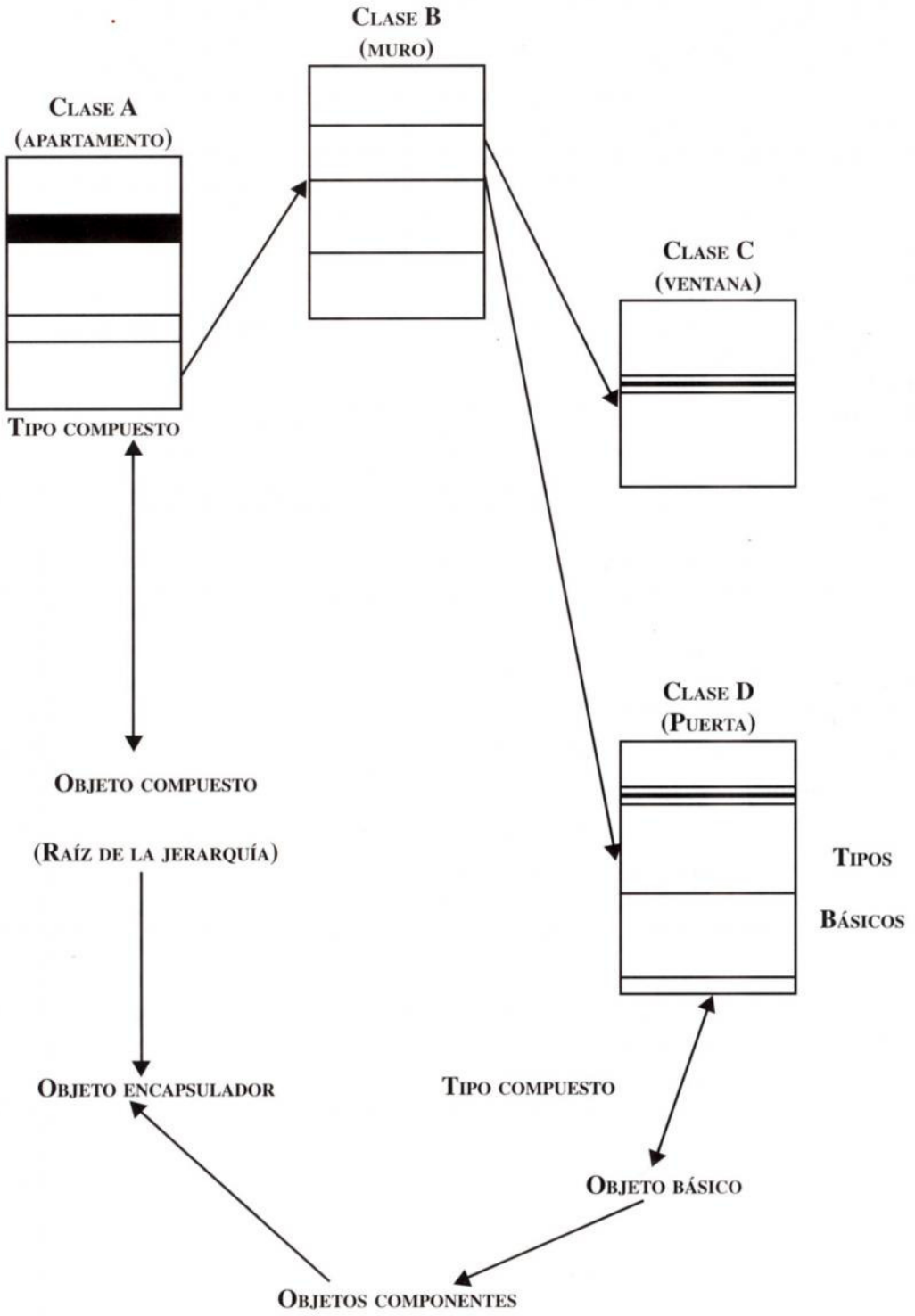
- Las referencias entre objetos son unidireccionales y se establecen desde objetos compuestos hacia objetos componentes.
- La definición de una clase define un tipo de datos asociado para sus instancias pero no las mantiene. Para este fin, existen objetos de tipo conjunto que guardan sus instancias. Como consecuencia de las características antes enunciadas una misma instancia puede pertenecer a más de un objeto de tipo conjunto.

5. Estado actual de la investigación

Mi actual preocupación radica en la definición del nivel interno o físico del SMBDOO. Este nivel representa la organización física de los datos en los dispositivos de almacenamiento y contiene detalles de recuperación y eficiencia de los procesos de obtención y actualización de los mismos vale la pena notar que es para el caso SMBDOO los métodos hacen parte de los datos del modelo, por lo tanto deben existir mecanismos adicionales para el almacenamiento, la recuperación y el correcto encadenamiento de los métodos definidos para cada clase.

La especificación de este nivel comprende dos grandes etapas. La primera en estado inicial, comprende la definición de la interfaz entre el esquema conceptual y el nivel interno (conocida como interfaz conceptual/interna) que garantiza tal como los sistemas tradicionales la independencia de los datos. La segunda consiste en la determinación de las estructuras de almacenamiento físico, que optimicen espacio en disco y faciliten la recuperación integral de los objetos.

Diagrama de Jerarquía de Composición (modelo propuesto)



6. Conclusiones

Este artículo enuncia bondades del paradigma OO como medio para solucionar necesidades emergentes de nuevos requerimientos de información de las aplicaciones "no tradicionales". Sin embargo los aspectos directamente relacionados con las BDOO continúan en estado de investigación, por lo tanto cualquier contribución por pequeña que sea, es vital para la madurez de los conceptos.

En el artículo se presenta el resumen del trabajo de investigación que se ha desarrollado con el objetivo de definir una estructura de almacenamiento adecuada para objetos típicos de ambientes de diseño arquitectónico asistido por computador, como en el caso de INDUCON. El hecho de hacer un estudio exhaustivo de los requerimientos, características y fundamentos de las aplicaciones CAD esta encaminado a suplir las necesidades de almacenamiento en memoria secundaria para ambientes CAD en diferentes dominios.

Aunque no esta directamente relacionado con el tema del artículo vale la pena mencionar que la aplicación del paradigma OO contribuye con la producción eficiente de software de buena calidad. Es decir, de su aplicación se derivan sistemas reutilizables, modulares, robustos y fáciles de depurar y mantener.

"Los buenos programas de computación hacen algo más que satisfacer simplemente sus requisitos funcionales. Los programas de computación que siguen unas reglas de diseño correctas tienen más probabilidad de ser aceptables, reciclables, extensibles, expandibles y fáciles de mantener."

Fernando Ruiz Rojas