UNA HERRAMIENTA CASE PARA EL DISEÑO Y LA GENERACION DE LA ESTRUCTURA ESTATICA DE LA BASE DE DATOS

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

RESUMEN

En este artículo se describe las características principales de una herramienta CASE que automatiza el diseño y la generación de la estructura estática de la base de datos basado en un método para el diseño de la base de datos desarrollado en el Instituto Superior Politécnico "José Antonio Echeverría". Esta herramienta parte de un modelo orientado a objetos y genera hacia una base de datos relacional, explotando las potencialidades de Borland Delphi para el trabajo con estos gestores. Se describen también algunas características internas de su implementación que mejoran la calidad del software, como por ejemplo, el uso de un algoritmo basado en el Escalador del Colinas Estocástico para el trazado automático del Diagrama de clases.

ABSTRACT

In this article it is described the main characteristics of a CASE tool that automates the design and the generation of the static structure of the database based on a method for the database design developed in our university. This tool leaves of a object-oriented model and it generates toward one relational database, exploiting Borland Delphi's potentialities for the work with these management systems. This work are also described some internal characteristics of their implementation that improve the software quality, for example, the use of an algorithm based on the Stochastic Hills Climbing for the automatic classes diagram drawing.

Key words: Borland's Delphi Method, Stochastic Hill Climbing, Object Oriented Programming.

MSC: 68P15.

1. INTRODUCCION

La utilización de una herramienta CASE (Computer-Aided Software Engineering) ha sido uno de los aportes más importante en el campo de la Ingeniería de software. Aunque el concepto fue abordado desde hace tres décadas, son los productos que se encuentran hoy en el mercado los que han provocado una revolución por el aumento que propician en la eficiencia, productividad y calidad del producto final.

Las herramientas CASE son una tecnología para automatizar el desarrollo y mantenimiento del software, combinando herramientas de software y metodologías. Estas herramientas deben constituir un conjunto integrado que automatice todas las partes del ciclo de vida y por tanto ahorren trabajo [Mc Clure (1993)].

Las facilidades que brindan para revisar especificaciones de un sistema, diagnosticar errores cometidos, desarrollar prototipos al generar parte o completamente una aplicación a partir de especificaciones, y el soporte para el mantenimiento como resultado de haber guardado las especificaciones del sistema en un depósito central de datos, Fertuck, (1992) , Hernández (1998), Senn (1998) ; son razones suficientes que justifican la importancia del desarrollo de estas herramientas acompañadas de las metodologías y métodos.

A partir de especificaciones de diseño, las herramientas generadoras de código pueden generar un esquema o un programa completo [Mc Clure (1993)].

En este trabajo se describe la propuesta de CASE (**Gebase**) que permite el diseño y la generación del esquema de la BD, de acuerdo al método de **DI**seño de la **BA**se de datos a partir de un modelo de **O**bjetos (DIBAO), desarrollado en el Instituto Superior Politécnico "José Antonio Echeverría" (ISPJAE).

Email: [1]anaisa@ceis.ispjae.edu.cu,
[2]sofia@ceis.ispjae.edu.cu

Gebase es una herramienta automatizada que genera automáticamente información, documentación, y el diagrama de clases a partir de especificaciones de diseño. Para el trazado automático utiliza una solución flexible y eficiente que se basa en el Escalador de Colinas Estocástico.

2. METODO DIBAO

El método para el DIseño de la BAse de datos a partir del modelo orientado a Objetos (DIBAO), está compuesto por dos partes: un modelo de persistencia y una capa persistente de clases (Figura 1). El modelo de persistencia describe los pasos para el diseño de la BD, las herramientas que se deben utilizar en este proceso, cómo convertir las clases al medio de almacenamiento seleccionado y cómo interpretar la información contenida en los diagramas y especificaciones textuales, en función de este diseño. La capa persistente de clases incluye las subcapas de especificación y de interfaz, que se encargan de definir las clases para describir las nuevas especificaciones asociadas a las relaciones entre los objetos, que se describen en el diccionario de clases; y la relación con el medio de almacenamiento, respectivamente.

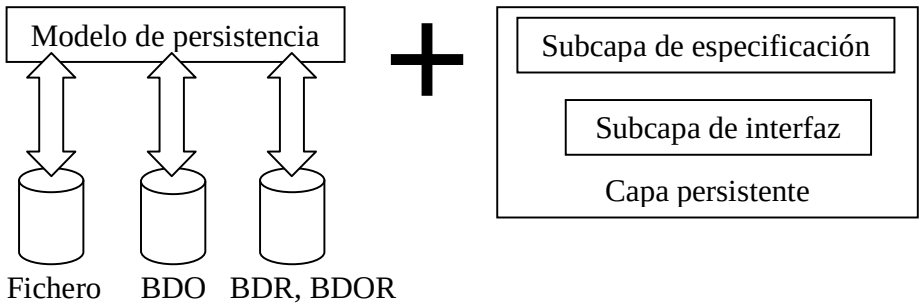


Figura 1. Estructura del método DIBAO.

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:

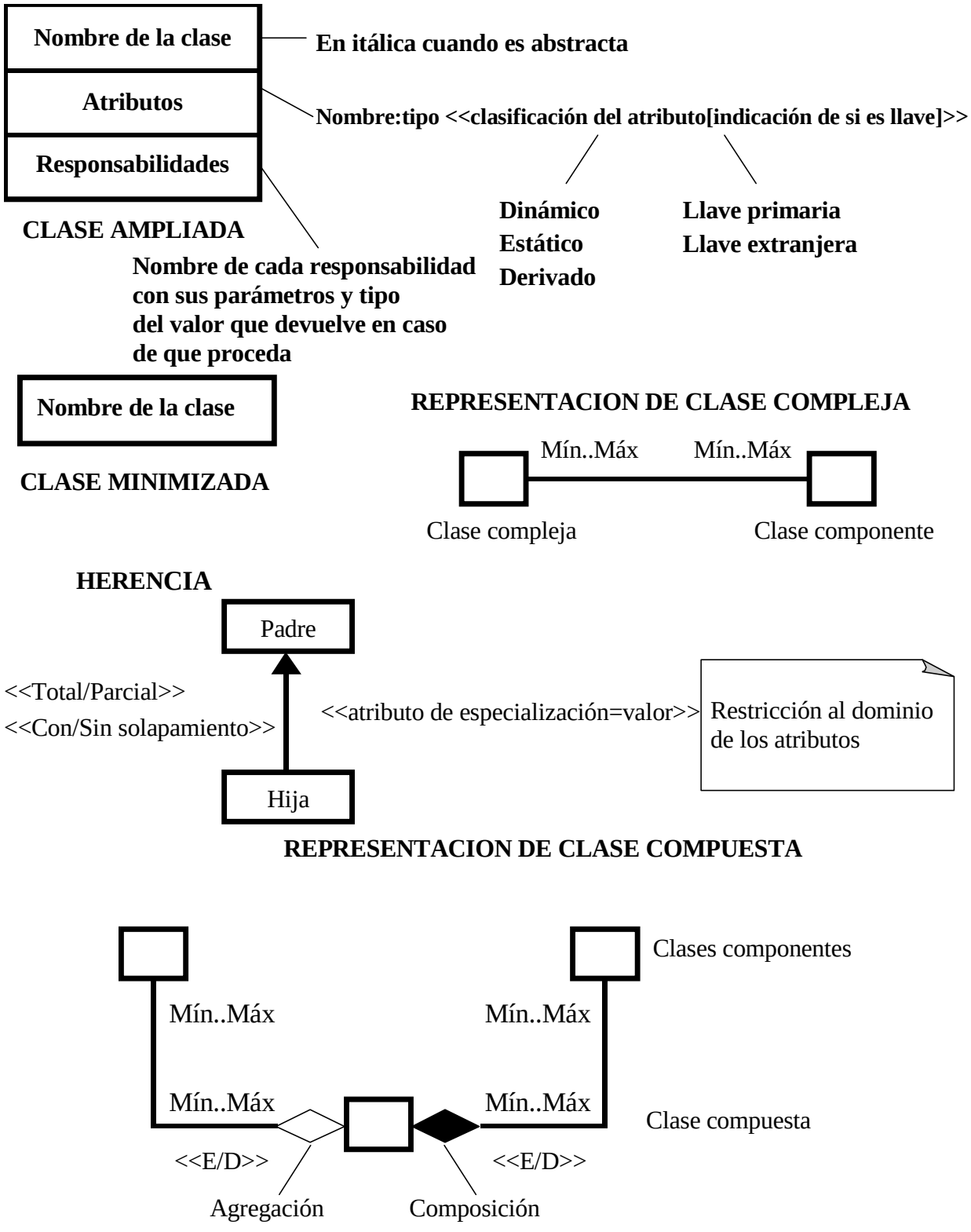
1. Definir las clases persistentes.
2. Refinar las clases.
3. Clasificar las clases y los atributos.
4. Realizar el diagrama de clases.
5. Realizar el diagrama de transición de estado.
6. Obtener las restricciones estáticas y las fórmulas dinámicas.
7. Convertir las clases al medio de almacenamiento.

En el paso relacionado con la conversión de las clases al medio de almacenamiento es que se tiene en cuenta hacía qué lugar se guardarán los objetos, por lo que se particulariza qué hacer en cada caso.

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-Jacobson-Booch (1999) y Ambler (2000)], aunque se añaden nuevas características usando estereotipos y notas, como se muestra en la Figura 2. Un ejemplo de un DC se representa en la Figura 3.

Asociada a las relaciones todo/parte, se especifica la dimensión estática/dinámica (E/D), que se representa gráficamente usando estereotipos. Esta dimensión se define como [Letelier, P. **et al**. (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.



Notas: << >>es el símbolo de estereotipo.
Mín..Máx simboliza las cardinalidades mínima y máxima respectivamente
<<E/D>>representa la dimensión

Figura 2. Notación del DC.

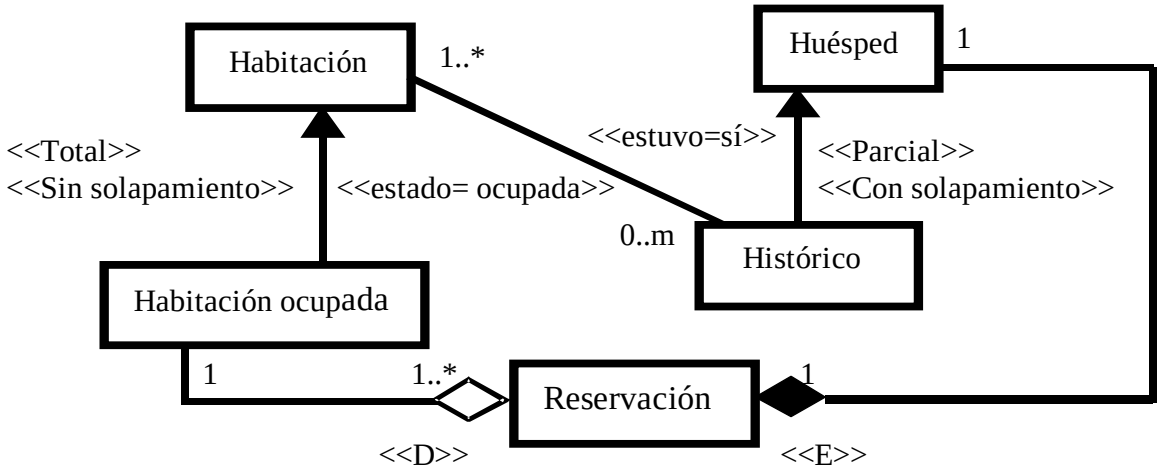


Figura 3. Fragmento de un DC de un sistema de reservación de habitaciones.

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	Unico	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.

Por ejemplo, No. Pasaporte=NOT EXIST IN Huésped
(Huésped.No pasaporte=Nuevo No pasaporte) AND
(Huésped.País=Nuevo.País)

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. Por ejemplo, Tipo de habitación =’Suite’ OR ’Simple’ OR ’Doble’.

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.

3. CONCEPCION GENERAL DEL CASE

El CASE que se presenta esta insertado dentro de la concepción de un CASE integrado más abarcador que se desarrolla para la metodología ADOOSI. El método DIBAO está definido dentro de la metodología ADOOSI, siendo la forma en que esta metodología propone realizar el diseño de la base de datos. La metodología es usada ampliamente en el país cuando se desarrollan proyectos utilizando el paradigma de la orientación a objetos. La versión actual de la metodología utiliza la notación UML [Alvarez- Hernández (2000)].

Podría pensarse que, como la propuesta presentada usa la notación UML y existen CASE como Rational Rose (de amplia divulgación) que se basan en esta notación, no se justifica la realización de este CASE. Hay dos razones importantes que dan respuesta a esta interrogante. En primer lugar, Rational Rose genera la estructura de la BD a partir del DC y en el modelo de persistencia propuesto se han incluido nuevas especificaciones que no serían tomadas en cuenta en la generación. En segundo lugar Rational Rose está preparado para un transición suave hacía gestores de objetos, pero no así relacionales que es el campo de **Gebase**.

Por otra parte, la estructura de los ficheros que genera esta herramienta no se conoce, lo que limita las posibilidades reales de utilizar, como fuente de entrada del CASE, los resultados obtenidos con el uso de este software. No obstante, es una variante acertada tratar de descifrar las salidas de Rational Rose para utilizarlas como fuente de entrada del CASE que se propone.

Aunque muchas personas usan las herramientas CASE como herramientas de documentación, **Gebase** permite además generar información automáticamente, ayuda en la realización de diagramas, valida inconsistencias, trabaja con rapidez, soporta la forma en que el diseñador piensa y trabaja y presenta una interfaz agradable.

4. INTEGRACION CON OTRAS HERRAMIENTAS

Antes de que un diseñador proceda a diseñar la BD, es necesario que un analista haya estudiado el dominio del análisis para interpretar los requerimientos del usuario y traducirlo en las clases, con los atributos y comportamiento, que son de interés para esa aplicación. El método de diseño propuesto parte de este conocimiento, aunque no descarta la posibilidad de adicionar o eliminar algún comportamiento.

El CASE integrado que se propone permite al diseñador introducir toda esta información manualmente o utilizar la información generada por otra herramienta automatizada. En particular, identifica los ficheros generados por las herramientas CASE: Analyze, Designer y GenProo; ampliamente utilizadas por los usuarios de la metodología ADOOSI.

Analyze, Reyes (1997) automatiza parte de la etapa de estudio preliminar y el análisis de esta metodología. Designer, [Hernández-Anache-Alvarez (1996), Hernández (1997)] toma estas especificaciones y facilita el desarrollo de casi todas las fases de la etapa de diseño, a excepción del diseño de la BD. Estas dos herramientas se corresponden con versiones anteriores de la metodología y no incluyen ningún tipo de generación de código. Aquí es donde entra el CASE GenProo, [Mato (2000)], que permite obtener prototipos del sistema generando automáticamente parte de la interfaz y el código asociado a las clases del dominio. En todos los casos se genera la documentación asociada a las fases que automatizan y se almacena la información en un fichero sin estructura, pero con un formato definido, una extensión y marcas de chequeo que los identifican como resultados de estos softwares (todos usan la persistencia a fichero).

En la versión actual de la metodología se propone usar Rational Rose porque es una herramienta automatizada que brinda altas prestaciones de servicio e incluye los diagramas que usa ADOOSI, aunque en lo relativo a la persistencia, las modificaciones que se hacen en el diagrama de clases y la interpretación que se da de este diagrama y en las restricciones en función del diseño de la BD, no son tomadas en cuenta por este CASE y aquí es donde se insertaría **Gebase**. Rational Rose no soporta una metodología en particular por lo que es pobre en le chequeo lógico del desarrollo del software.

El CASE que se propone toma la información contenida en los diccionarios de datos y automatiza el diseño y la generación estática de la BD de acuerdo al método DIBAO. La forma de integración utilizada en este caso es la denominada por Roger Pressman (1997) como intercambio dinámico de datos. La aplicación cuenta con un traductor que convierte la información del diccionario de datos ya que requiere añadir nuevas especificaciones imprescindibles como parte del diseño (Figura 4). Para aquellos que trabajen con Rational Rose o no utilizan una herramienta CASE en el desarrollo de las etapas predecesoras al diseño, permite la entrada manual de los datos.

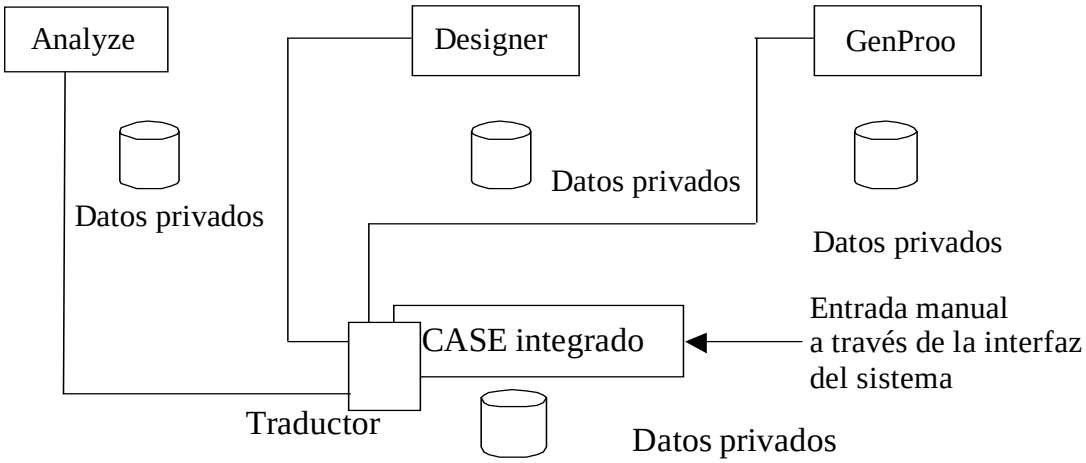


Figura 4. Intercambio de información entre herramientas.

Como el CASE está diseñado para un método de diseño en particular (DIBAO), soporta las tres filosofías descritas en Vessey (1992). Es restrictivo pues hay un conjunto de acciones que están restringidas (por ejemplo, no permite que clases controladores sean persistentes); guía al usuario ya que hay acciones que, aunque no se restringen, se le alerta de las posibles consecuencias (por ejemplo, cuando se intenta eliminar una clase, dejando al usuario la libertad de decidir continuar el proceso, implementándose las acciones en cascada que genera), y es flexible porque el usuario puede realizar algunas acciones con completa libertad sin seguir estrictamente el orden de los pasos definidos por el método (por ejemplo, el refinamiento de las clases y la clasificación de clases y atributos, puede ser hecho en cualquier orden.

En la generación del esquema se trabaja a partir de las potencialidades de Borland Delphi para el trabajo con gestores relacionales. Esto no obliga a desarrollar la aplicación en este lenguaje.

4.1. Módulos

El CASE propone desarrollar los pasos del diseño dividiendo el proceso en 5 módulos (Figura 5). Estos módulos se ejecutan secuencialmente, aunque se permiten retrocesos. La ejecución secuencial implica que cada módulo utiliza la información obtenida por el módulo predecesor y genera automáticamente información. Un cambio en la información precedente, es interpretada como un cambio en la especificación por lo que se vuelve a generar.

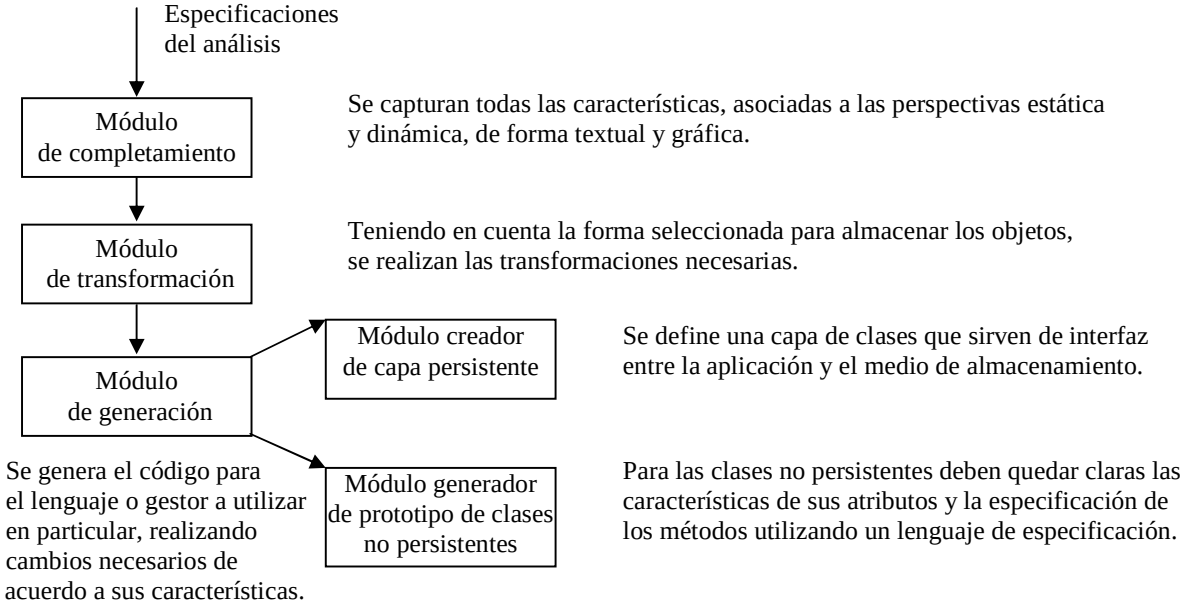


Figura 5. Módulo del proceso de diseño y generación.

4.2. Persistencia a gestores relacionales

Gebase es una herramienta que, a partir de algunas especificaciones correspondientes al análisis de un sistema informático, permite automatizar parte de la etapa de diseño través de la adición de información y la definición de los parámetros necesarios para generar la base de datos, [Hernández-Alvarez-Pastor (1998)]. Se encarga de:

- Establecer la relación con el fichero resultante del Analyze [Reyes (1997)] y MetProo [Mato (2000)].
- Capturar las especificaciones necesarias visualmente y por pantallas de edición.
- Convertir las clases persistentes a tablas a partir de las reglas definidas en el método DIBAO para esta alternativa de almacenamiento.
- Generar el DC a partir de las características de la clases, utilizando un algoritmo eficiente de trazado automático [Rosete (2000)].
- Chequear la calidad y consistencia de la información.
- Generar la documentación asociada a los procesos que automatiza.

Las reglas de conversión a tablas que se aplican son:

1. Cada clase con dominios simples se convierte en una tabla.
2. En el caso de la jerarquía de clase, se pueden aplicar tres posibilidades. Estas posibilidades son:
   a. Definir una única tabla con los atributos del padre y los atributos de cada hija.
   b. Definir una tabla para cada clase hija hoja que incluya los atributos de la clase padre.
   c. Definir una tabla para cada clase hija y para cada clase padre. Para recuperar la información de una clase hay que consultar la clase hija y sus ancestros.

   Gebase adopta la última variante porque está más preparada para la evolución del esquema ya que en las variantes a y b las tablas creadas se modifican si ocurre algún cambio al no implementar el concepto de herencia. Además las tablas tendrían atributos que tienen sentido solo para algunos hijos, lo que implica permitir el valor nulo y por lo tanto los programadores tendrían que definir los procedimientos para validar estas restricciones de dominio. Las propuestas a y b son superiores en la rapidez de la recuperación de la información.
3. Para clases complejas la conversión depende de las cardinalidades máximas en cada extremo. Las cardinalidades mínimas introducen restricciones de integridad estática. Las reglas que se aplican en estos casos son:

m:n	Se convierte en una tabla que tiene como la llave las llaves de las clases que conforman la relación. Otra variante es transformarla en dos relaciones bidireccionales en cuyo caso se añade un atributo de referencia a cada tabla. Esta variante hace más complejas las búsquedas y actualizaciones.
1:m ó m:1	Se puede convertir en una nueva tabla o puede ser añadida una llave extranjera al extremo m, que se corresponde con la llave de la clase del extremo 1.
1:1	Se puede convertir en una nueva relación o se adiciona un nuevo campo a alguna de las tablas que es la llave de la otra.
c:m ó m:c	En el caso de que en uno de los extremos exista en una cantidad mayor que 1 de elementos con los que se relaciona la clase del extremo m, se puede definir una nueva tabla que tenga como atributos llave, la llave del extremo m y tantos atributos del tipo de la llave del extremo constante como indique le valor de la constante. También se pueden adicionar a la clase del extremo m, tantos campos del tipo de la llave del extremo constante como indique el valor de la constante.
c:c1	Se puede definir una nueva tabla que tendrá c+c1 llaves, en la que habrá c campos del tipo de la llave del primer extremo, y c1 campos del tipo de la llave del extremo. También se puede adicionar a la clase de uno de los extremos, tantos campos del tipo de la llave del otro extremo como indique el valor de la constante.

En las relaciones de 1:1 y 1:m, si no hay ciclos, se puede tener en una sola tabla los objetos que participan en la relación, pero esto viola la segunda FN porque introduce redundancia.

En el caso de las relaciones n-arias (n>2), se crea una tabla con las llaves de las clases que intervienen y las cardinalidades se analizan como restricciones de integridad estáticas.

A pesar de que no es necesario crear nuevas tablas que reflejen la relación en casi ningún caso, el método recomienda hacerlo porque el esquema puede evolucionar y no sería en estos casos necesario hacer cambios a la estructura de la BD, solo a las restricciones de integridad estática asociadas a la cardinalidad. Además hay menos complejidad en las búsquedas y en la actualización de la información, y se conserva la propiedad de encapsulamiento, clave en el enfoque orientado a objetos, ya que un objeto no debe conocer cosas de otro objeto si no es necesario.

4. Para las clases compuestas se crea una nueva tabla con el nombre de la clase compuesta y que tiene como llave las llaves de las clases componentes. Las cardinalidades se definen como restricciones de integridad estática. En esta conversión no se tienen en cuenta las cardinalidades, pues se trata de llevar al esquema de la BD el objeto del mundo real tal como se aprecia, aunque esto introduce validaciones relativas a su cardinalidad. Además hay que tener en cuenta su dimensión con respecto a cada componente.

En el caso de que la implementación sea hacía un SGBDOR, que permita tener campos que referencien a otro objeto, no es necesario que se cambie el tipo de este campo para el tipo de la llave de la clase a la que pertenece el objeto referenciado.

La implementación de las características asociadas al comportamiento estático dependen de las potencialidades del gestor que se utilice. Por ejemplo, **SQL Server** incluye el predicado **check** que permite especificar restricciones de dominio de los atributos [Dalton (1997)].

Una herramienta CASE que genere la base de datos debe ser capaz de obtener y almacenar las especificaciones necesarias y crear un modelo que contenga las características de la información generada. Para este tipo de productos, en este trabajo se sigue un conjunto de criterios que ayudan a evaluar su calidad, entre los que se encuentran:

- *Tiempo de generación:* **Gebase** la realiza instantáneamente después de activarse esta opción.
- *Complejidad de la presentación:* las propiedades y los campos de las tablas resultantes son mostradas de forma similar al formato utilizado por los gestores relacionales, en forma de tabla.
- *Número de tablas mostradas a la vez:* los sistemas de gestión, por la extensión que puede alcanzar una tabla de este tipo, solo muestran una cada vez y así está implementado en **Gebase**.
- *Posibilidad de modificación de la estructura de la tabla antes de generar:* el sistema parte del principio de que si se utiliza un generador, lo más eficiente es cambiar la especificación y volver a generar.

Es muy común que las metodologías incluyan herramientas gráficas. Debido a esto se han definido un conjunto de criterios que ayudan a evaluar su automatización [Waterson (1993)], entre ellos se encuentran:

- cuán rápido pueden los dibujos ser cambiados y redibujados,
- cuán rápido pueden los dibujos ser salvados y recuperados,
- cuántos diagramas diferentes pueden ser vistos en la pantalla al mismo tiempo,
- si el documento soporta niveles de diagramas, ¿el usuario puede moverse de un nivel a otro?, ¿es fácil mostrar dos niveles al mismo tiempo?, y
- control del usuario sobre el tamaño, posición y color de los elementos que aparecen.

Estos aspectos son muy importantes para que el dibujo cumpla su cometido de describir con más facilidad los detalles. Para ello pueden utilizarse algoritmos que decidan, por ejemplo, cuántos niveles de representación se requieren (que se traduce en cantidad de vistas para un gráfico), cómo se organiza la información en dependencia del tipo de gráfico que sea, entre otros aspectos [Hernández (1997)].

Algunas veces la forma de representar, lejos de ayudar a la comprensión, la hace muy difícil debido a la gran cantidad de líneas y puntos que la conforman. Aunque se han desarrollado algoritmos para el trazado de diagramas para algunos problemas como son los DER (Diagrama Entidad Relación), en otros diagramas no se ha logrado esto [Rosete-Ochoa (1998)].

Gebase, para el trazado automático del DC, usa el método del Escalador de Colinas Estocástico para la ubicación espacial de los elementos. El algoritmo utilizado fue un resultado de la tesis de doctorado que se describe en [Rosete (2000)]. Como resultado de la investigación descrita en esta tesis se obtuvieron DLL's,

programadas en C++, que automatizan el algoritmo. Las DLL's se usa en dos momentos, para generar el diagrama a partir de las clases contenidas en el diccionario de datos, y después para redibujar el diagrama ya que el diseñador puede modificarlo.

El algoritmo se adapta a los requerimientos del CASE integrado porque permite cualquier tipo de nodos, las líneas, para unir los elementos, pueden ser rectas o quebradas, y tiene en cuentas los siguientes criterios estéticos: minimizar la cantidad de cruces entre arcos, minimizar la longitud total entre arcos, balancear la ubicación de los nodos respecto a determinados ejes, minimizar el solapamiento entre nodos y minimizar el área total ocupada. Este último criterio se utiliza además para introducir la semántica relacionada con la necesidad de que las clases compuestas puedan estar ubicadas espacialmente cerca de sus componentes.

A pesar de que no siempre se obtiene un dibujo óptimo con el trazado automático, se logra una solución factible si se considera que la complejidad de los diagramas que se automatizan puede ser alta. En los ejemplos de pantallas de **Gebase**, que se incluyen en la Figura 6, se muestra la calidad visual que se obtiene con la aplicación de este algoritmo. La legibilidad en la representación gráfica forma parte de los criterios de calidad a tener en cuenta para este tipo de software.

Gebase se estima que ahorra entre un 20%-25% del tiempo de desarrollo de un proyecto informático, de acuerdo al volumen de clases persistentes (mientras mayor es la cantidad con respecto al total de clases, se obtiene un ahorro mayor). Constituye además una ayuda para el diseñador pues ya no tendrá que cambiar el código cada vez que desee cambiar la estructura de la BD, solo tiene que cambiar la especificación y volver a generar.

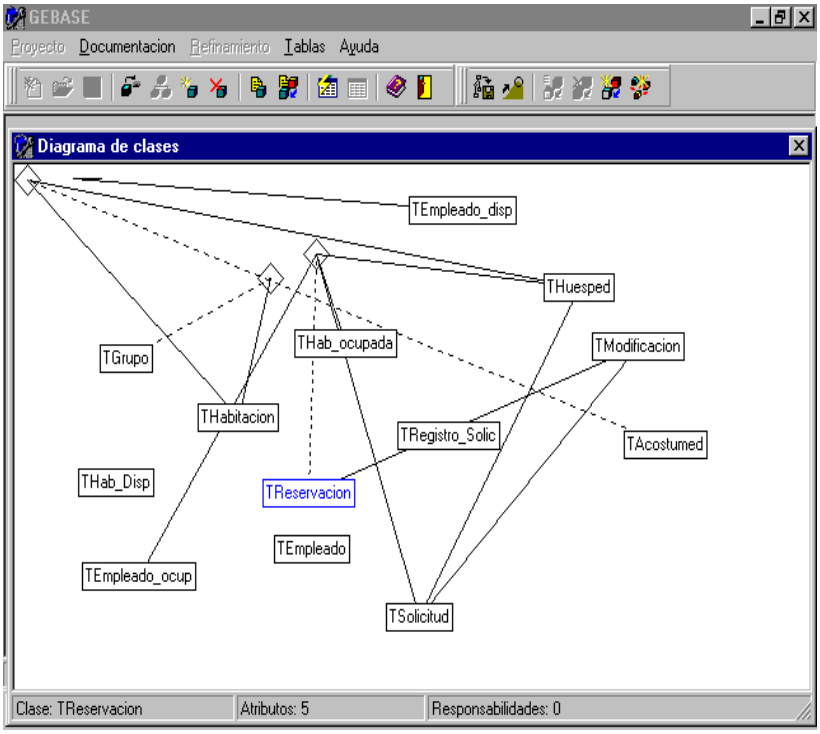


Diagrama de clases antes de aplicar el algoritmo.

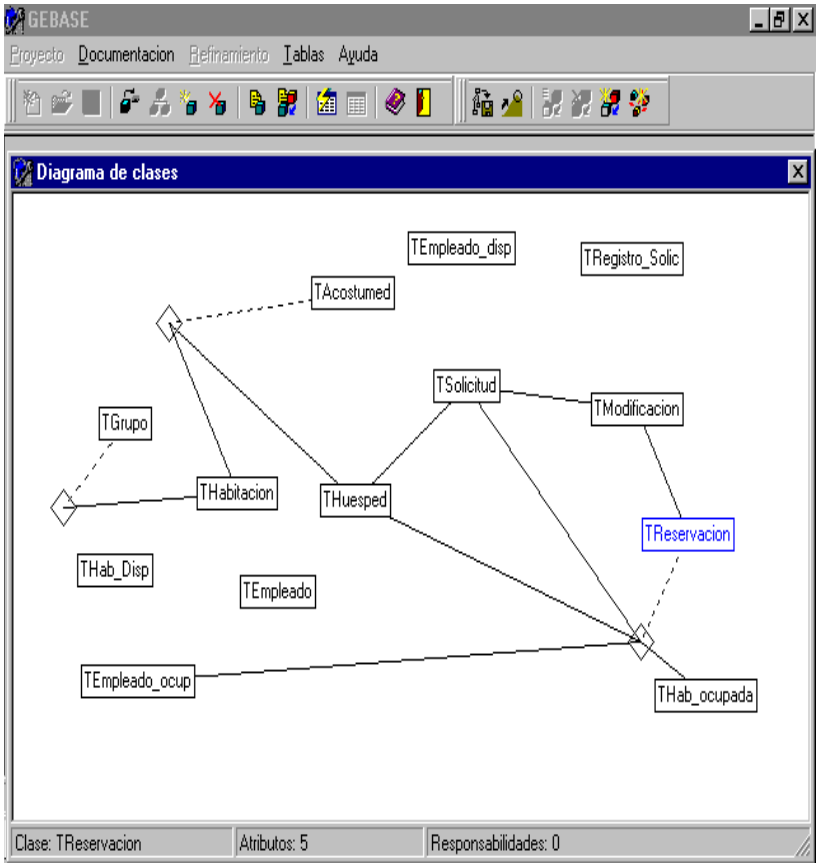


Diagrama de clases después de aplicado el algoritmo.

Figura 6. Ejemplo de un diagrama de clases antes y después de aplicado el algoritmo del Escalador de Colinas Estocástico.

5. CONCLUSIONES

La herramienta **Gebase** está basada en un método de diseño en particular (DIBAO), lo que añade valor ya que brinda recomendaciones, chequea contradicciones y genera automáticamente información. Todo esto mejora la productividad.

No se descartó el uso de herramientas comerciales que utilicen el estándar UML, como fuente de entrada de información para el CASE integrado, aunque en la actualidad todavía no es posible por los mecanismos de seguridad que ha implantado Rational Rose alrededor de sus productos. Se trabaja en la actualidad en la interpretación de un fichero texto al que es posible exportar la información de un proyecto creado con este CASE. De esta forma **Gebase** se convertirá en una extensión que cubre un espacio en el que Rational Rose no es fuerte.

La utilización del algoritmo del Escalador de Colinas Estocástico mejora la calidad visual del diagrama de clases y por lo tanto permite que se entienda mejor la información representada.

REFERENCIAS

ALVAREZ, S. y A. HERNANDEZ (2000): "Metodología ADOOSI-UML, versión 5.0". CEIS, ISPJAE, Habana.

AMBLER, S. (2000): "How the UML models fit together", **Software Development Magazine**, March. http://www.sdmagazine.com/articles/2000/003z/ focus.ambler.htm.

DALTON, P. (1997): "Microsoft SQL Server black book", The Corilis group, Inc. International Thomson Publishing Company, New York.

FERTUCK, L. (1992): "Systems Analysis and Design with CASE tools". **Wm. Brown publishers**, 460-504.

HERNANDEZ, A; I. ANACHE y S. ALVAREZ (1996): "Diseño orientado a objetos y bases de datos relacionales". Memorias del evento internacional Compac'96, (ISBN-959237-024-9.), Cuba.

HERNANDEZ, A. (1997): "Automatización del diseño orientado a objetos". **Revista Ingeniería Industrial,** XVIII(3).

HERNANDEZ, A. "CASE (1998): Una solución a la baja productividad en el software", **Revista Ingeniería Industria**l, XIX(1), 35-39.

HERNANDEZ, A.; S. ALVAREZ y O. PASTOR (1998): "Generación de código para implementar la persistencia a partir de especificación de diseño orientado a objetos". XXV Convención de la UPADI. 1er Congreso de Informática, Perú.

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

MATO, R.M. (2000): "MetProo and GenProo as learning tools", Proceedings of International Conference on Engineering and Computer Education (ICECE 2000). Brasil.

Mc CLURE, C. (1993): "CASE, la automatización del software".

PRESSMAM, R. (1997):"Software engineering practitioner's approach". Fourth edition. McGraw-Hill, New York.

REYES, J. (1997): "Analyze: Herramienta CASE para el análisis orientado a objetos". **Revista Ingeniería Industrial**, XVIII(3). 27-30.

ROSETE, A. and A. OCHOA (1998): "Genetic Graph Drawing". Proceedings of 13th International Conference of Applications of Artificial Intelligence in Engineering, AIENG'98, Adey, R. A.; Rzevski, G. and Nolan, P. (Eds). Galway, Computational Mechanics Publishing, 37-40.

ROSETE, A. (2000): Tesis para optar por el grado científico de Doctor en Ciencias técnicas: “Una solución flexible y eficiente para el trazado de grafos basada en el Escalador de Colinas Estocástico”. ISPJAE, Habana, Cuba.

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

SENN, J.A. (1998): “Análisis y diseño de sistemas de información”. 2da. edición. McGraw-Hill Interamericana de México, México DF.

VESSEY, I. (1992): “Evolution of vendors products: CASE tools as methodology companions”. Communications of ACM.

WATERSON, K. (1993): “Easy your database development woes”. Database Advisor, VII(4).

.