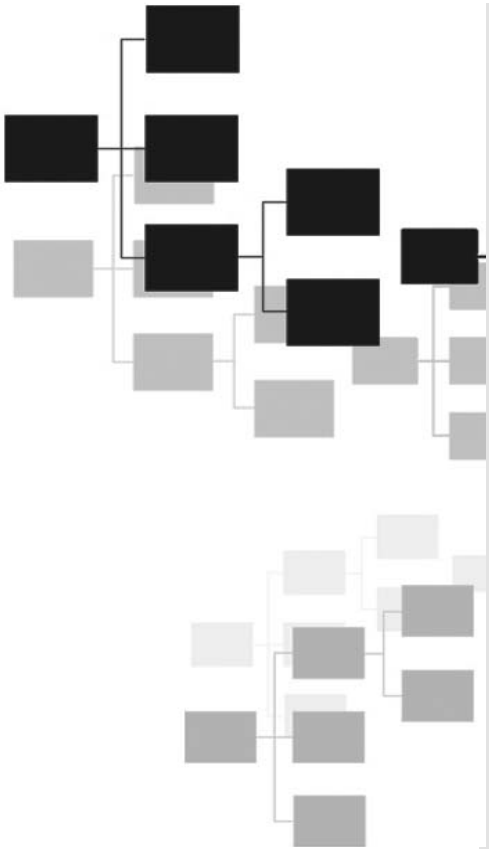
REVISTA Universidad EAFIT
Vol. 43. No. 147. 2007. pp. 68-80

UNC - Diagramador

una herramienta *upper* CASE para la obtención de diagramas UML desde esquemas preconceptuales



Carlos M. Zapata J.
Ph. D. en Ingeniería. Profesor Asociado, Escuela de Sistemas, Facultad de Minas, Universidad Nacional de Colombia, Sede Medellín. Integrante del Grupo en Ingeniería de Software de la misma institución.
cmzapata@unal.edu.co

Luz M. Ruiz C.
Estudiante de Pregrado, Ingeniería de Sistemas e Informática, Universidad Nacional de Colombia, Sede Medellín. Integrante del Grupo en Ingeniería de Software de la misma institución.
lmruiz@unal.edu.co

Fernán A. Villa
Estudiante de Pregrado, Ingeniería de Sistemas e Informática, Universidad Nacional de Colombia, Sede Medellín. Integrante del Grupo en Ingeniería de Software de la misma institución
favilla0@unal.edu.co

Recepción: 06 de junio de 2007 | Aceptación: 09 de septiembre de 2007

Resumen

Las herramientas CASE han tenido tradicionalmente un enfoque hacia actividades relativas a las fases finales del ciclo de vida del software, como la generación de código, por ejemplo. Por ello, este tipo de herramientas, denominadas *Lower* CASE, han podido apoyar muy someramente a los analistas en procesos como la generación de esquemas conceptuales a partir de lenguaje natural. Para esta tarea, han venido surgiendo herramientas CASE enfocadas a las fases iniciales del ciclo de vida del software (conocidas como *Upper* CASE).

Sin embargo, estas herramientas aún presentan inconvenientes: la mayoría de ellas se enfocan en un solo diagrama y las que generan varios diagramas emplean diferentes representaciones intermedias para llegar ellos, lo que puede ocasionar problemas de consistencia en los diagramas resultantes. En este artículo se muestra el desarrollo de UNC-Diagramador, una herramienta del tipo *Upper* CASE para la generación de diagramas de UML 2.0 desde los denominados Esquemas Preconceptuales, con la cual se trata de solucionar las limitaciones presentadas. El uso de UNC-Diagramador se ejemplifica con un caso de estudio.

Palabras Clave

Herramientas *Upper CASE*
Esquemas preconceptuales
Diagramas de UML 2.0

UNC-Layout: an *upper* CASE tool to get UML diagrams out from pre-conceptual schemes

Abstract

CASE tools have been traditionally focused in activities concerned with the final stages in a software's operational life, e.g. code creation. That is why this kind of tools, known as *Lower* CASE, have not been able to lend real support to analysts in processes like the creation of conceptual schemes from a natural language. To accomplish this, CASE tools focused on the starting stages (known as *Upper* CASE) have been developed. However, those tools continue to have some drawbacks, as most of them are focused in a single diagram, and those which create several diagrams use different intermediate representations to achieve them. This might result in consistency problems in the generated diagrams. This paper shows the development of a UNC-layout, an Upper CASE tool for the generation of UML 2.0 diagrams from the so-called Pre-conceptual Schemes in an attempt to overcome its drawbacks. The use of the UNC-Layout is shown in a case study.

Key Words

Upper CASE tools
Pre-conceptual schemas
UML 2.0 diagrams

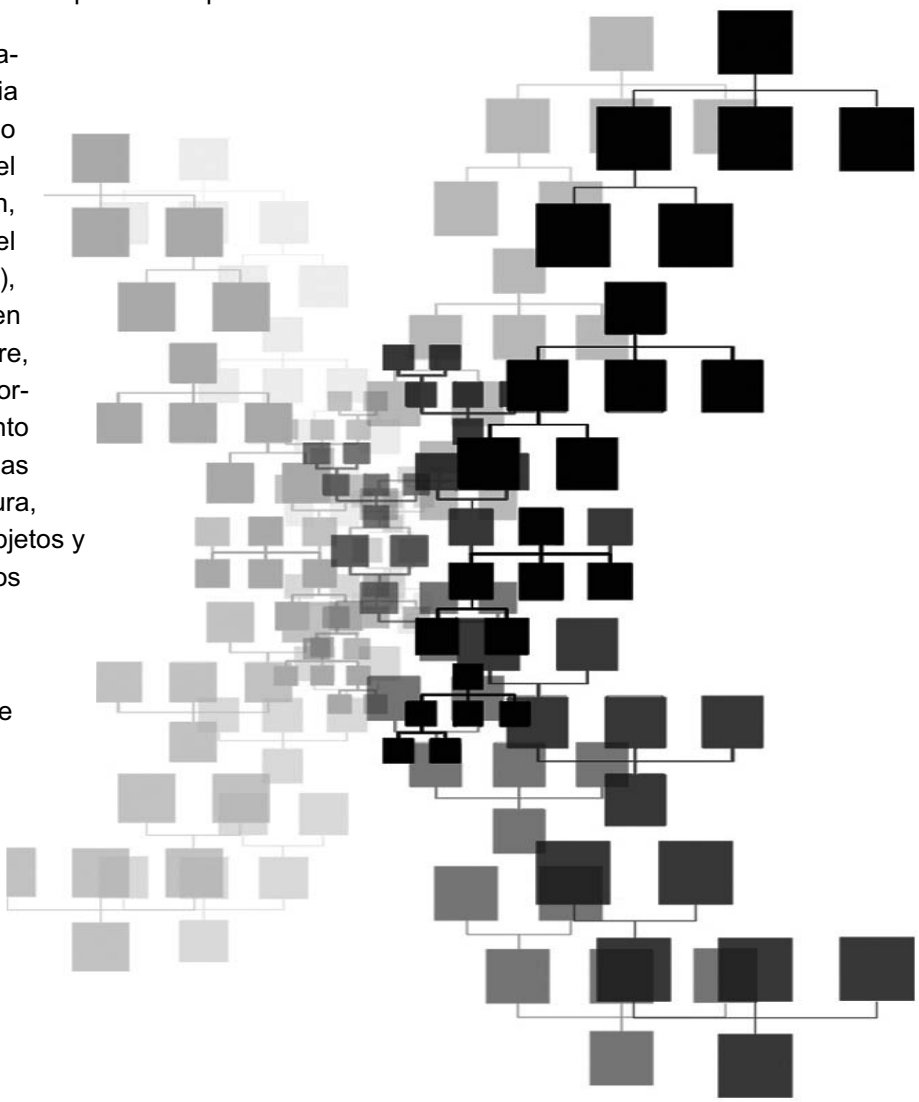
Introducción

La disponibilidad de herramientas que permitan facilitar el trabajo del analista en las diferentes fases del ciclo de vida del software se ha convertido, en la actualidad, en una necesidad, dado que la actividad de captura de la información de los interesados y su posterior conversión a esquemas conceptuales (lo que se suele denominar Elicitación de Requisitos) es uno de los procesos más delicados y que consume más tiempo en dichas fases (Sommerville, 2001). Las herramientas CASE *(Computer Aided Software Engineering)* han procurado apoyar a los analistas en diferentes procesos de la Ingeniería de Software y se han convertido en un "arma eficaz" para esta labor, primordialmente en el trazado de diagramas para su posterior conversión a código y visualización de una posible solución (Pressman, 2001). Una gran cantidad de estas herramientas permite, por ejemplo, el trazado de diagramas UML, el principal lenguaje de modelamiento de aplicaciones de software en la actualidad (Booch, *et al.,* 1998; *Object Management Group*, 2007). La invención de las herramientas CASE se remonta a la década de los años setenta; inicialmente se usaban como editores de gráficos y posteriormente como

generadores de código a partir de diagramas modelados por un analista. Algunas de las más conocidas herramientas CASE son: Rational Rose®, ArgoUML®, Poseidon® y Together®.

A mediados de los años noventa surgió un nuevo tipo de herramientas CASE que permiten generar diagramas automáticamente, en especial los diagramas UML, tomando como punto de partida discursos en lenguajes controlados. La novedad de generar automáticamente los diagramas UML radica en la utilidad que representa para el analista poder obtener rápidamente un primer bosquejo del mundo que pretende modelar. Una revisión crítica de este nuevo tipo de herramientas CASE se puede consultar en Zapata y Arango (2005). En estas herramientas aún existen problemas por solucionar:

- La mayoría de las herramientas se enfoca hacia la generación de un solo diagrama (por ejemplo, el diagrama entidad-relación, el diagrama de clases o el diagrama de secuencias), lo cual es inconveniente en el desarrollo de software, donde se requiere la incorporación de un conjunto de vistas complementarias que muestren la estructura, la interacción entre los objetos y el comportamiento de los mismos.
- Las herramientas que permiten la generación de varios diagramas a partir del mismo discurso, aún presentan problemas de consistencia, pues suelen utilizar diferentes artefactos intermedios para la generación de cada diagrama.



Como una forma de solución a estos problemas, en este artículo se presenta UNC-Diagramador, una herramienta *Upper* CASE elaborada en la Escuela de Sistemas de la Universidad Nacional de Colombia, que permite generar automáticamente el Diagrama de Clases,

Comunicación y Máquina de Estados de UML 2.0 (OMG, 2007). Esta herramienta toma como punto de partida los Esquemas Preconceptuales (Zapata; Gelbukh y Arango, 2006), aquellos esquemas que permiten la representación de un discurso en lenguaje controlado y que contienen la información necesaria para generar estos tres tipos de diagramas UML.

Este artículo tiene la siguiente estructura: en la Sección 1 se exponen las tendencias en herramientas CASE para diagramas UML; en la Sección 2 se presenta UNC-Diagramador, su funcionamiento interno y la descripción de su plataforma; en la Sección 3 se describe un caso de estudio donde se puede apreciar el proceso de generación; en la Sección 4 se presentan algunas conclusiones y finalmente en la sección 5 se presentan los trabajos futuros en relación con esta herramienta.

1. Tendencias actuales en la Generación Automática de Diagramas UML

En la actualidad, las herramientas CASE se pueden utilizar en la aplicación de métodos para el desarrollo del software. La correcta inclusión de una herramienta CASE en uno de esos métodos puede agilizar el desarrollo de la aplicación de software. Las herramientas CASE se suelen clasificar —dependiendo de la fase de desarrollo en que se empleen— en *Upper CASE*, *Lower CASE* e *Integrated CASE* (Gane, 1990). Las herramientas CASE convencionales son de tipo *Lower CASE,* lo cual significa que están dirigidas hacia las últimas fases de desarrollo de software (construcción e implementación). Las herramientas *Lower* CASE tienen como principal objetivo la generación automática de código a partir de determinados diagramas, generalmente de UML, facilitando el desarrollo de prototipos y aplicaciones. Las herramientas *Upper* CASE apoyan los analistas en las fases iniciales del desarrollo de software (definición, análisis y diseño). Finalmente, las herramientas *Integrated* CASE contienen características de los dos tipos.

Una tendencia en herramientas *Upper* CASE, surgida a mediados de los años noventa, tiene como objetivo la transformación de los requisitos capturados durante los procesos de elicitación y análisis en esquemas conceptuales, algunos de los cuales son diagramas de UML. Dos de los proyectos que siguen esta tendencia y que presentan las características necesarias para analizar los problemas que aún subsisten en ella son CM-Builder (Harmain & Gaizauskas, 2000) y NIBA (Fliedl *et al.*, 2002). Otros trabajos adicionales pueden ser consultados en Zapata y Arango (2005).

El proyecto CM-Builder (Harmain & Gaizauskas, 2000) se enfoca en la construcción de un único diagrama, el de clases de UML, a partir de una forma de lenguaje controlado, empleando para ello redes semánticas como representaciones intermedias. Como algunas de sus desventajas se pueden anotar el hecho de que sólo obtiene el diagrama de clases (y no otros diagramas UML) y que la representación intermedia mediante redes semánticas no permite representar las características dinámicas del modelo del discurso, lo cual permite sólo una vista parcial de la aplicación de software que se piensa desarrollar. Los diagramas de clases representan la estructura estática del sistema, ya que sólo muestran las clases (conceptos importantes del mundo) y sus interrelaciones (herencia, asociación y agregación). Para complementar el modelamiento de un sistema, es necesario modelar también su dinamismo, y para ello se usan los diagramas de comportamiento e interacción (secuencias, casos de uso, comunicación y máquina de estados, entre otros).

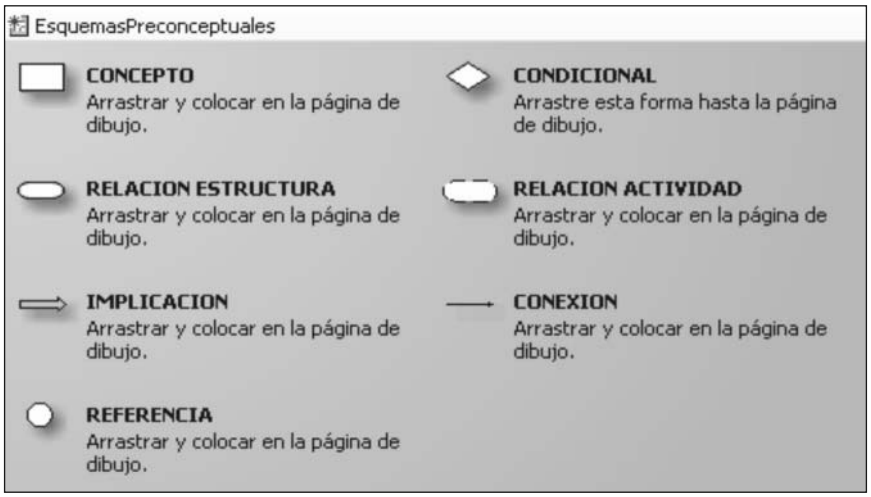
El proyecto NIBA (Fliedl, *et al.*, 2002) busca la generación de los diagramas de clases y actividades de UML; además, plantea que se podrían obtener otros diagramas, como secuencias y comunicación. Para la generación de estos diagramas, NIBA emplea un conjunto de esquemas intermedios que sus autores denominaron KCPM *(Klagenfurt Conceptual Predesign Model),* los cuales poseen formas diferentes para los distintos diagramas de UML, variando desde tablas con información

especial para el diagrama de clases, hasta unos diagramas dinámicos propios de NIBA para el diagrama de actividades; esto puede ocasionar ciertas pérdidas de información entre diagramas y, consecuentemente, fallas de consistencia entre los mismos. Por tanto, en NIBA la información de tipo estático y dinámico no se puede combinar para obtener una representación única con que se puedan generar los diferentes diagramas de UML.

2. UNC-diagramador: Desde Esquemas Preconceptuales hasta Diagramas de UML 2.0

UNC-Diagramador es una herramienta *Upper* CASE actualmente en desarrollo por parte del Grupo en Ingeniería de Software de la Escuela de Sistemas de la Universidad Nacional de Colombia, sede Medellín. UNC-Diagramador emplea la representación de un discurso en un Esquema Preconceptual (cuya simbología básica se puede apreciar en la Figura 1) para generar, de manera automática, tres de los diagramas correspondientes al estándar de UML 2.0: el diagrama de clases, que modela la estructura del dominio, el diagrama de comunicación, que modela la forma como se comunican los objetos del mundo, y el diagrama de máquina de estados, que modela el comportamiento de los objetos. Estos tres diagramas son complementarios y modelan los aspectos fundamentales del dominio de la aplicación de software que se pretende construir. Los esquemas preconceptuales (Zapata *et al.*, 2006) son utilizados por UNC-Diagramador como un esquema unificador que permite la generación de los diagramas mencionados.

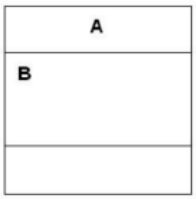
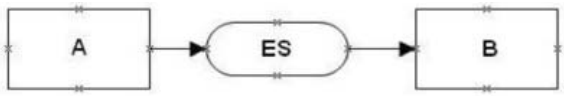
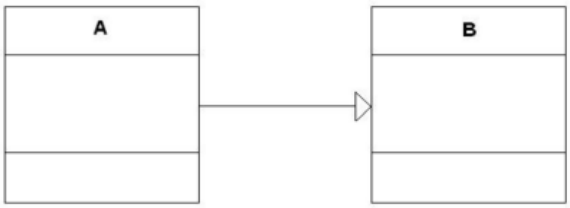
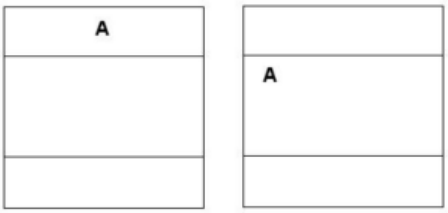
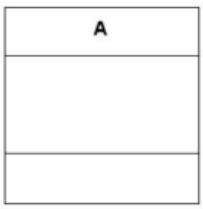
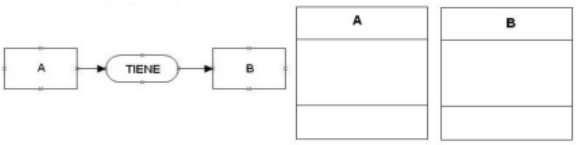
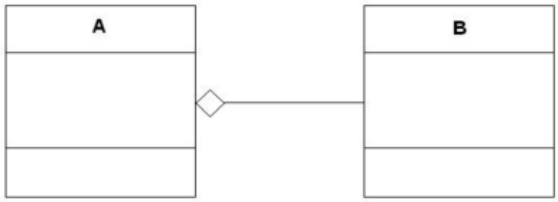
Figura 1. Plantilla de Dibujo Visio® de Los Esquemas Preconceptuales



Fuente: Los autores (2007)

Para el proceso de generación de los diagramas, UNC-Diagramador cuenta con un conjunto de "Reglas de Conversión" (Zapata y Arango, 2007) que permite realizar la transformación de un Esquema Preconceptual al subconjunto de UML 2.0 ya mencionado. La utilización de estas reglas, en un proceso automático, garantiza la consistencia entre los tres diagramas. Algunas de las reglas que se emplean en la conversión se presentan en la Tabla 1, que incluye la sintaxis de los esquemas preconceptuales combinada, para las reglas presentadas, con la sintaxis del diagrama de clases. Nótese que las reglas 1 y 4, que emplean relaciones estructurales del tipo "tiene", generan dos tipos de elementos diferentes en el diagrama de clases, pues sus precondiciones son diferentes.

Tabla 1. Algunas de las reglas empleadas para la obtención del diagrama de clases a partir de los esquemas preconceptuales

No.	Precondición	Resultado
1	En una relación estructural con el verbo “tiene” que liga dos conceptos A y B, el primer concepto A es una clase candidata y el concepto B es un atributo candidato de la clase A.	
		
2	En una relación estructural con el verbo “es” que liga dos conceptos A y B, ambos conceptos son clases candidatas y existe una relación de generalización en la que la clase B es la clase padre de la clase A.	
		
3	Un concepto A que simultáneamente se haya identificado como clase y como atributo por diferentes reglas será una clase.	
		
4	Si en la regla 1 ambos conceptos han sido identificados como clases candidatas, se presenta una relación de agregación entre ellas, siendo A el agregado y B la parte.	
		

Fuente: Los autores (2007)

UNC-Diagramador se implementó bajo la plataforma Microsoft Visual Studio .NET®, versión 2005, utilizando el lenguaje de programación C# y un paquete especial Microsoft Visio®, versión 2003, denominado *Software Development Kit* (SDK); C# permite utilizar todo el conjunto de clases contenidas en el SDK de Visio. El SDK, a su vez, contiene todas las clases necesarias para manipular y utilizar todos los elementos incluidos en

Microsoft Office Visio®, cada documento, página, estilo, forma, grupo, forma u objeto de un grupo, maestro, objeto de otro programa, guía y punto de guía (*Microsoft Developer Network,* 2003). Estos elementos cuentan con una hoja de cálculo *ShapeSheet* en la que se almacena la información acerca de cada objeto, la cual contiene datos como el alto, ancho, ángulo, color y otros atributos que determinan el aspecto y el comportamiento de cada elemento gráfico de un diagrama. El SDK evita programar desde cero toda la interfaz de dibujo y edición de diagramas, razón por la cual fue seleccionado por el grupo de desarrollo para la programación de las clases e interfaces necesarias en la implementación del UNC-Diagramador. Para ejecutar UNC-Diagramador es necesario tener instalado sistema operativo Microsoft Windows®, versión 2000 en adelante, .NET® Framework 2.0 y realizar una Instalación Completa Microsoft Visio®, versión 2003.

Para generar los diagramas UML en UNC-Diagramador, en primer lugar hay que crear o cargar un dibujo en formato VDX (XML *Visio Drawing*), que contenga el Esquema Preconceptual que se va a convertir (MSDN, 2003); el archivo en formato VDX se puede crear tanto con Microsoft Visio® como con UNC-Diagramador y puede ser abierto y editado por el UNC-Diagramador debido a que es un formato basado en XML (Lenguaje extensible de etiquetado), que es un estándar de comunicación entre aplicaciones. Una vez que se haya cargado el esquema preconceptual en la aplicación, se puede iniciar la generación de diagramas UML, empleando las reglas de transformación descritas en Zapata y Arango (2007); este proceso muestra y entrega los tres Esquemas Conceptuales resultantes (clases, comunicación y máquina de estados), los cuales se almacenan en un solo archivo VDX. UNC-Diagramador construye correctamente el dibujo a partir del archivo XML en Visio®, es decir, crea código XML correctamente formado con el Esquema XML para Visio® y sus reglas internas. Este archivo VDX puede ser abierto con Microsoft Visio®.

La Clase fundamental que pertenece al SDK de Visio® y se utiliza en la implementación de UNC-Diagramador es *AxDrawingControl* (Control de Dibujo); esta clase provee los principales métodos para manipular Diagramas Visio®, tales como guardar, abrir, copiar, pegar, deshacer, entre otros; además, contiene el Área de Dibujo, que es donde se grafican los diferentes diagramas, utilizando los elementos de las Plantillas de Dibujo. Estas plantillas configuran el entorno de dibujo para ajustarlo a un tipo de gráfico posible (MSDN, 2003), es decir, cada Plantilla de Dibujo delimita los objetos con los cuales se puede graficar un tipo de diagrama.

Para el UNC-Diagramador se creó una Plantilla de Dibujo Visio® de cada diagrama requerido (Esquema Preconceptual, Clases, Máquina de Estados y Comunicación). Cada elemento de la Plantilla de Dibujo Visio® tiene su correspondiente *ShapeSheet* y Representación Gráfica o dibujo del Elemento. Las Plantillas de Dibujo garantizan que todos los diagramas sean graficados con los mismos tipos de elementos. UNC-Diagramador sólo puede procesar Esquemas Preconceptuales que hayan sido elaborados con la mencionada Plantilla Visio®.

La Plantilla de Dibujo Visio® de Los Esquemas Preconceptuales contiene los siguientes elementos: Relación Estructural, Relación Dinámica, Concepto, Condicional, Implicación, Conexión y Referencia; la representación gráfica de estos elementos se puede apreciar en la Figura 1.

Utilizando la librería de clases *System.Xml* de .NET, UNC-Diagramador "lee" el archivo VDX del esquema preconceptual, y sólo reconoce aquellas etiquetas como: nombre, tipo, contenido y conexiones de la forma, que contengan información relevante dentro del *ShapeSheet*; luego, busca y aplica las "Reglas de Conversión" que se ajusten a los datos reconocidos. Las "Reglas de Conversión" han sido programadas en C#, aprovechando toda la potencia y facilidad que brinda la programación orientada a objetos. Por cada "Regla de Conversión" aplicada, UNC-Diagramador genera el *ShapeSheet* del (de los) elemento(s) mapeado(s) con la regla y los inserta en la página correspondiente, ya sea,

en la del Diagrama de Clases, Comunicación y/o Máquina de estados, para posteriormente agregar las páginas en el dibujo de Visio (que se incluye en un archivo VDX).

Para generar los Tres diagramas UML, UNC-Diagramador utiliza las siguientes Plantillas Visio®: para el diagrama de Clases, la plantilla "Clases" que contiene Clase, Conexión, Herencia y Agregación, como se muestra en la Figura 2; para el de Comunicación, la plantilla "Comunicación" que contiene Objeto y Comunicación, que se presenta en la Figura 3; para la de Máquina de Estados, la plantilla "Transición" que contiene Estado, Transición e Inicio, como se indica en la Figura 4.

Figura 2. Plantilla de Dibujo Visio® para el Diagrama de Clases

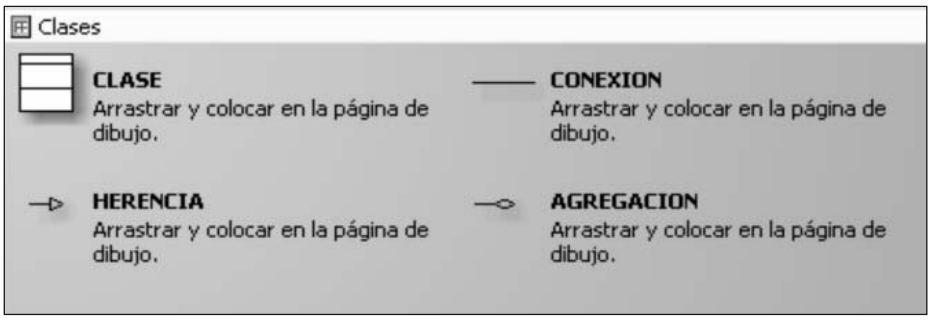


Figura 3. Plantilla de Dibujo Visio® para el Diagrama de Comunicación

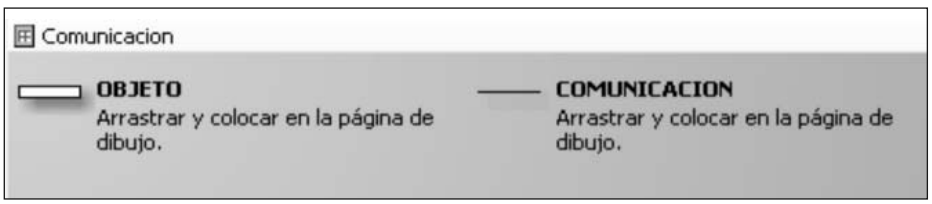
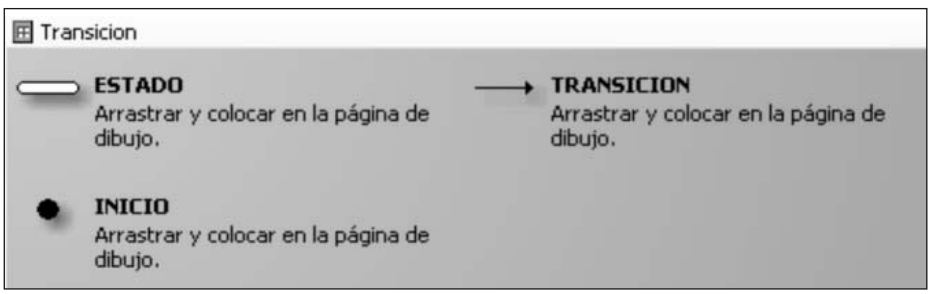


Figura 4. Plantilla de Dibujo Visio® para el Diagrama de Máquina de Estados



Fuente: Los autores (2007)

Luego del Proceso de Conversión, UNC-Diagramador permite visualizar y editar cualquiera de los cuatro Diagramas (el Esquema Preconceptual y los tres de UML). Después de la edición del esquema preconceptual, es posible realizar nuevamente el proceso de generación de los tres diagramas de UML.

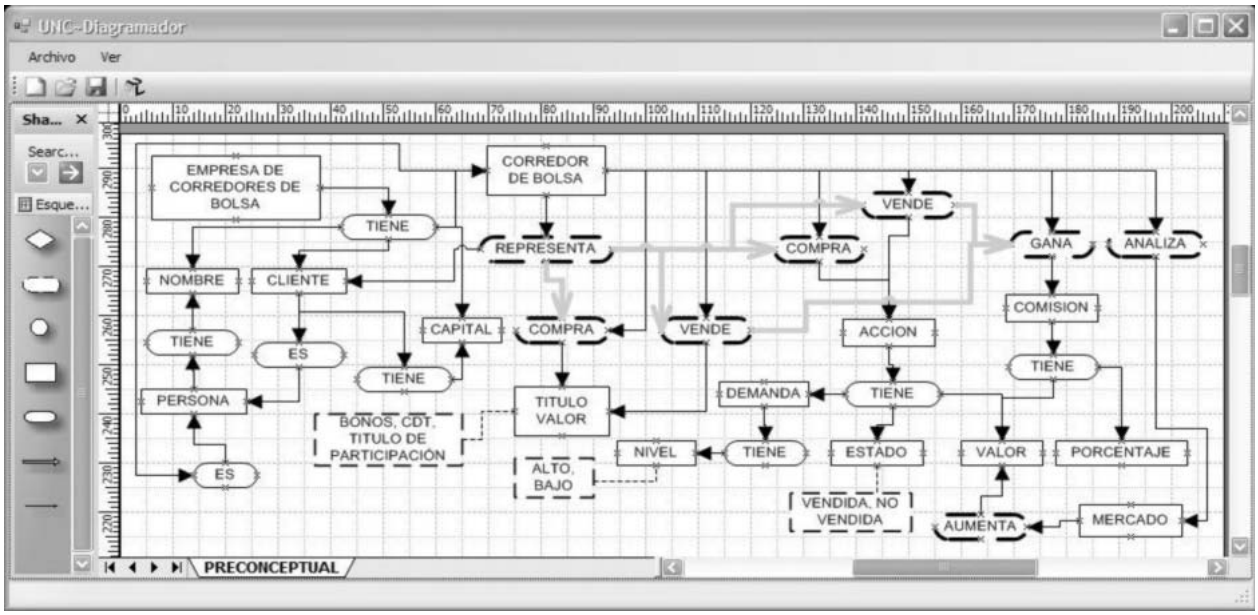
3. Caso de Estudio

En esta sección se presenta un ejemplo de la conversión de un esquema preconceptual a tres diagramas conceptuales: Clases, Comunicación y Máquina de Estados, utilizando UNC-Diagramador. El siguiente discurso representa el dominio de una bolsa de valores:

> Existen dos tipos de personas; los clientes y los corredores de bolsa. El cliente tiene un capital, y es representado por un corredor de bolsa en la bolsa de valores. El corredor de bolsa pertenece a una empresa de corredores y puede vender o comprar títulos valores (tales como bonos CDT's y títulos de participación) y acciones, después de analizar el mercado. Cuando el corredor vende, gana una comisión. La empresa de corredores de bolsa y las personas tienen nombre. Las comisiones se caracterizan porque tienen un valor y un porcentaje. El mercado se encarga de aumentar el valor de las comisiones. Las acciones tienen una demanda, que a su vez tienen un nivel (que puede ser alto o bajo); además, las acciones tienen un estado (vendidas o no vendidas).

El esquema preconceptual que representa cierta problemática, puede ser tan completo y complejo como se desee; el discurso que se presenta podría ser más extenso, pero sólo se presenta una porción con fines ilustrativos. Empleando el editor del UNC-Diagramador, se puede elaborar el esquema preconceptual de la Figura 5.

Figura 5. Esquema Preconceptual de una Bolsa de Valores



Para generar los diagramas de UML a partir del Esquema Preconceptual de la Bolsa de Valores, se debe presionar el botón "UML" ubicado en la barra de herramientas. Los diagramas obtenidos después del proceso automático de conversión son el Diagrama de Clases, Comunicación y Máquina de estados, como se puede apreciar en las figuras 6, 7, 8 respectivamente.

Figura 6. Diagrama de Clases obtenido a partir del Esquema Preconceptual de la Bolsa de Valores

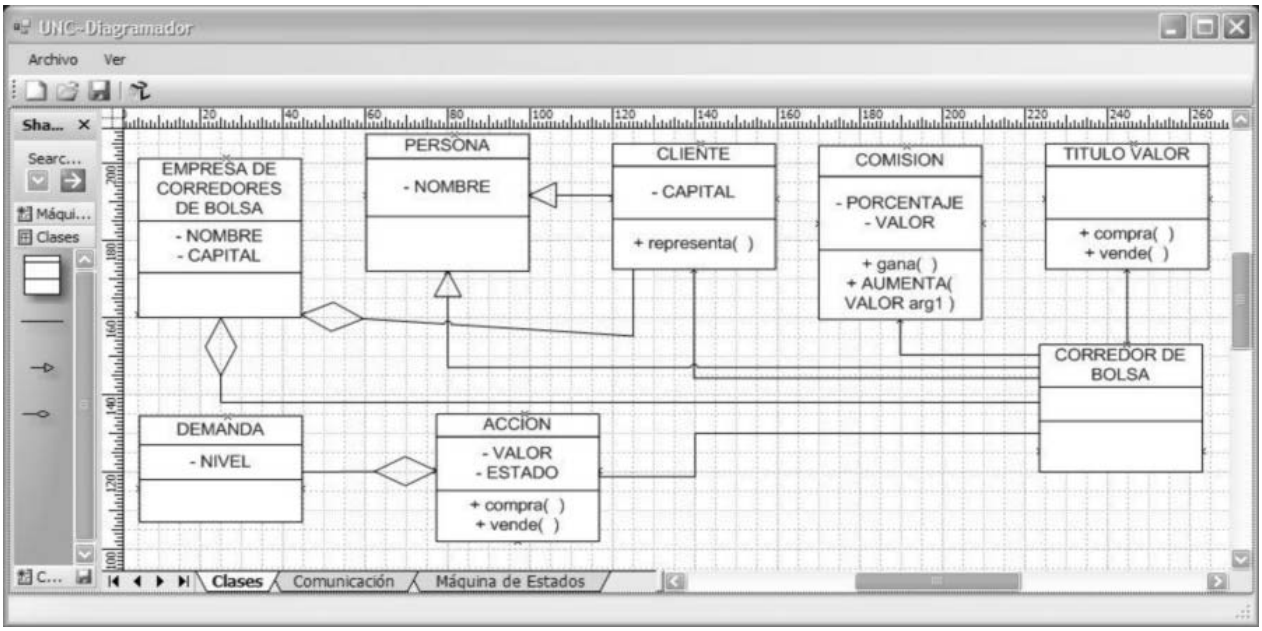


Figura 7. Diagrama de Comunicación obtenido a partir del Esquema Preconceptual de la Bolsa de Valores

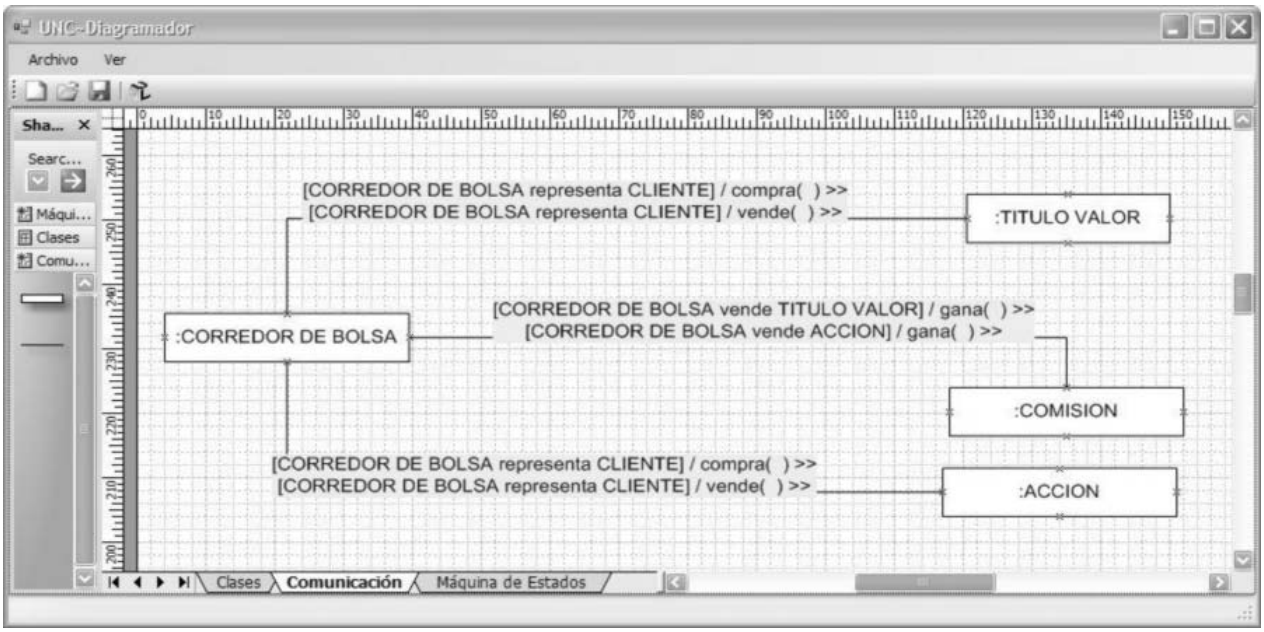
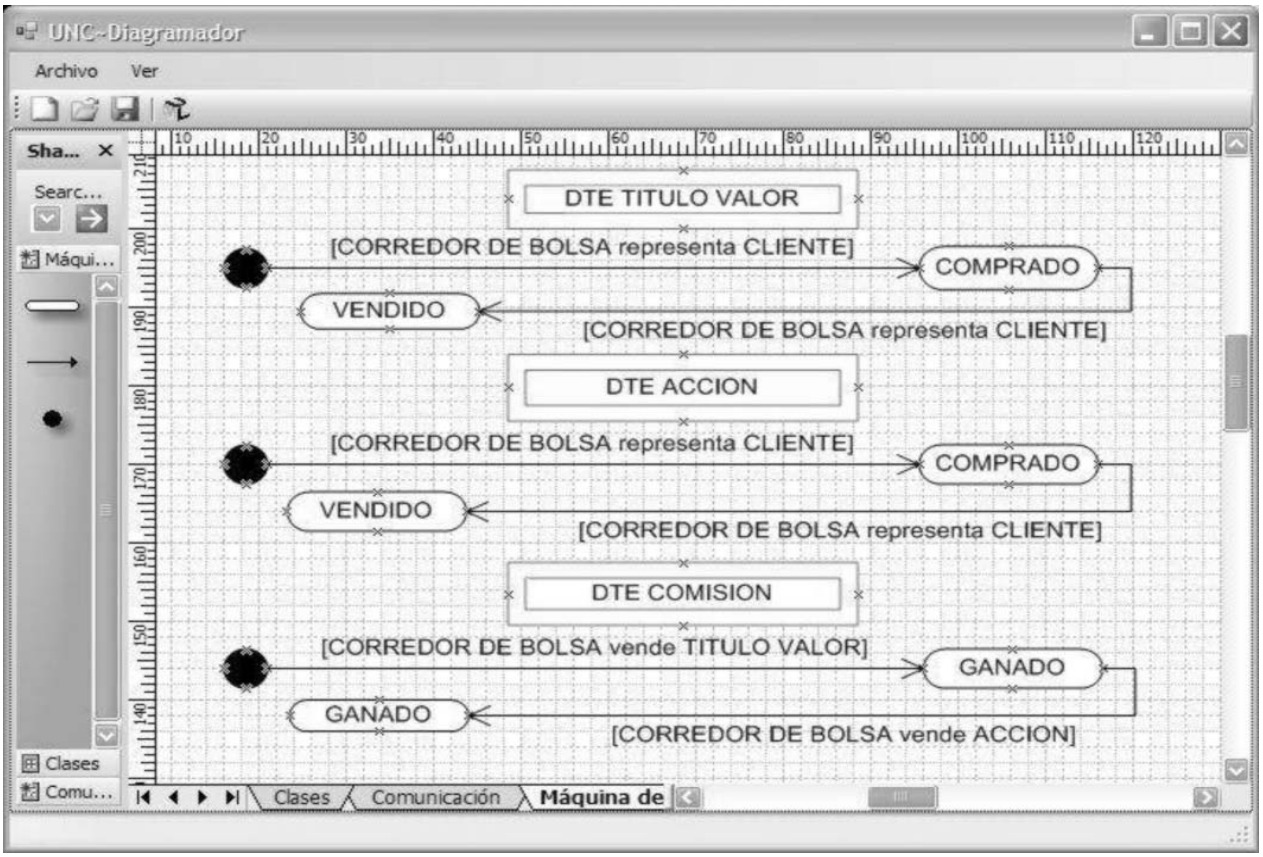


Figura 8. Diagrama de Máquina de Estados obtenido a partir del Esquema Preconceptual de la Bolsa de Valores



Fuente: Los autores (2007)

Conclusiones

- UNC-Diagramador es una herramienta *Upper* CASE que permite obtener automáticamente tres diagramas UML (Clases, Comunicación y Máquina de Estados) a partir de un esquema unificador: los Esquemas preconceptuales.
- UNC-Diagramador emplea reglas de conversión que garantizan la consistencia entre los diagramas resultantes; además, el analista no tiene que preocuparse por el uso correcto de los símbolos de UML, porque la generación de los diagramas es automática a partir de los esquemas preconceptuales.
- Debido a que la generación de los diagramas de UML toma sólo unos minutos, el analista puede realizar un proceso iterativo para el mejoramiento de los diagramas; en otras palabras, el analista puede incorporar elementos en el esquema preconceptual y examinar, de manera casi inmediata, el resultado en los tres diagramas mencionados de UML.
- En el desarrollo de UNC-Diagramador se empleó tecnología .NET combinada con Microsoft Visio®. El uso del *Software Development Kit* de Visio®, permitió al grupo de desarrollo reducir el tiempo en la implementación de UNC-Diagramador, ya que no fue necesario elaborar un editor de diagramas desde cero, sino que se aprovecharon las capacidades gráficas de Visio® para el manejo de los diagramas.

Trabajo Futuro

Existen algunas líneas de trabajo que pueden dar continuidad al desarrollo de UNC-Diagramador, tales como:

- La realización de un conjunto de experimentos que permita medir la efectividad de los esquemas preconceptuales en la construcción de diagramas UML *versus* la construcción directa de dichos diagramas.
- La generación de otros Diagramas UML a partir del esquema preconceptual, tales como secuencias, actividades o casos de uso, o incluso de diagramas diferentes a UML, como objetivos, procesos o causa-efecto.
- La generación de mecanismos de comunicación con herramientas *Lower* CASE, con el fin de permitir la generación de código ejecutable a partir de los esquemas preconceptuales.
- La elaboración de un sistema de reconocimiento de reglas (compilador de reglas) que facilite la inserción de nuevas reglas en UNC-Diagramador, sin necesidad de modificar el código fuente de la aplicación.
- La conversión de UNC-Diagramador en un sistema multiplataforma, que no sólo funcione en el entorno Windows® sino también en Linux®.
- El desarrollo de una versión web de UNC-Diagramador, empleando, por ejemplo, ASP.NET® y otras tecnologías disponibles para ello. De esta manera, se podría ensayar la obtención de los diferentes diagramas como un trabajo conjunto de diferentes analistas ubicados geográficamente distantes.

Bibliografía

Booch, G.; Rumbaugh, J. y Jacobson, I. 1998. *Unified Modeling Language User Guide*. Reading: Addison-Wesley.

Fliedl, G. *et al.* 2002. "The NIBA workflow: From textual requirements specifications to UML-schemata". In: *Proceedings of the ICSSEA '2002—International Conference "Software & Systems Engineering and their Applications"*, Paris: Centre d'étude pourla Maîtrise des Systèmes et du Logiciel.

Gane, C. 1990. *Computer-Aided Software Engineering—The Methodologies, the Products, and the Future*. Londres: Prentice-Hall.

Harmain, H. & R. Gaizauskas. 2000. "CM-Builder: An Automated NL-based CASE Tool". In: *Proceedings of the fifteenth IEEE International Conference on Automated Software Engineering ASE'00*. Grenoble: IEEE Computer Society.

Microsoft Developer Network MSDN 2003. *Visio 2003 SDK Documentation.* [on line]: http://msdn2.microsoft.com/en-us/library/aa221218office.11.aspx 6 de Junio de 2007.

Object Management Group OMG. 2007. *OMG Unified Modeling Language Specification*. [on line]: http://www.omg.org/UML/ 2 de junio de 2007.

Pressman, R. 2001. *Software Engineering: A Practitioners' Approach 5th ed.* New York: McGraw-Hill.

Sommerville, I. 2001. *Software Engineering*. Massachussetts: Addison-Wesley.

Zapata, C. M. y F. Arango. 2007. "Elicitación de Requisitos empleando UN-Lencep y Esquemas Preconceptuales". En: *Memorias de las VI Jornadas de Ingeniería del Software e Ingeniería del Conocimiento*. Lima: Pontificia Universidad Católica del Perú. pp. 69–78.

_____. 2005. "Los Modelos Verbales en Lenguaje Natural y su utilización en la elaboración de esquemas conceptuales para el desarrollo de software: Una revisión crítica". En: *Revista Universidad EAFIT.* Vol. 41. No. 137, pp. 77–95.

Zapata, C. M.; Gelbukh, A. & Arango, F. 2006. "Pre-conceptual Schema: A Conceptual-Graph-Like Knowledge Representation for Requirements Elicitation". En: *Lecture Notes in Computer Science*. Vol. 4293. pp. 17–27.