UN AMBIENTE PARA LA OBTENCION AUTOMATICA DE DIAGRAMAS UML A PARTIR DE UN LENGUAJE CONTROLADO

AN ENVIRONMENT FOR AUTOMATED UML DIAGRAMS OBTAINING FROM A CONTROLLED LANGUAGE

CARLOS M. ZAPATA

Escuela de Sistemas, Universidad Nacional de Colombia, sede Medellín. cmzapata@unal.edu.co

FERNANDO ARANGO I

Escuela de Sistemas, Universidad Nacional de Colombia, sede Medellín

Recibido para revisar Septiembre 25 de 2006, aceptado Diciembre 20 de 2006, versión final Febrero 09 de 2007

RESUMEN: El apoyo suministrado por las herramientas convencionales de la Ingeniería de Software a los analistas se ha basado en ayudas para el trazado y edición de modelos, siendo exiguo el apoyo ofrecido a la concepción misma del modelo. Existe actualmente una tendencia hacia la generación automática de esquemas conceptuales y, si bien se han realizado grandes avances, se ha trabajado con pocos diagramas y subsisten algunos inconvenientes relacionados especialmente con la consistencia de los diagramas obtenidos. En este artículo se propone un ambiente para la obtención automática de algunos de los diagramas de UML 2.0, conformado por un lenguaje controlado (UN-Lencep), un mecanismo para la representación del conocimiento (los denominados Esquemas Preconceptuales) y un sistema de reglas para la traducción del lenguaje controlado a un conjunto de diagramas equivalentes de UML; este ambiente se implementó en la herramienta CASE UNC-Diagramador, en la cual se presenta adicionalmente un caso de estudio. Con esta implementación se apoya la labor de conceptualización de los modelos por parte de los analistas y se mejora la consistencia, puesto que los modelos resultantes se elaboran a partir del mismo texto en lenguaje controlado.

PALABRAS CLAVE: UN-Lencep, Esquema Preconceptual, diagramas UML 2.0, Reglas de conversión, UNC-Diagramador.

ABSTRACT: Conventional Software Engineering tools have supported analysts in helping them for both model drawing and editing, but with little help in model conceptualization. Currently, there is a trend for automated conceptual schema generation and, despite of great advances, there's still work on few diagrams, and even there are some problems especially related to consistency of the obtained diagrams. In this paper, we define an environment for automatically obtaining of UML 2.0 diagrams, including a controlled language—UN-Lencep—, a Knowledge representation language—Preconceptual Schemas—, and a set of rules for UML diagrams translation; this environment has been implemented in a CASE tool—UNC-Diagramador—and a case study example has been provided. This implementation supports analysts in both model conceptualization, and consistency improvement, due to the fact that resulting models are elaborated from the same text in controlled language.

KEYWORDS: UN-Lencep, Pre-conceptual Schema, UML 2.0 diagrams, Conversion rules, UNC-Diagramador.

1. INTRODUCCIÓN

El desarrollo de software suele comenzar con una serie de entrevistas entre el interesado y el analista, donde se procura combinar el conocimiento que posee el interesado sobre el dominio del problema, con los conocimientos técnicos del analista para el desarrollo de software [1]. Una vez se ha consolidado un discurso que describe el dominio del problema, el analista representa mediante un modelo el ámbito del dominio y su solución, utilizando un

estándar de modelamiento, usualmente el Unified Modeling Language (UML), que es uno de los más reconocidos en la actualidad [2].

La Ingeniería del Software tradicionalmente ha apoyado las iniciativas de modelamiento con las denominadas herramientas CASE (Computer-Aided Software Engineering), desarrolladas desde mediados de la década de los setenta y que en la actualidad permiten el trazado y edición de diferentes tipos de diagramas (incluida la gama de diagramas UML), además de la generación parcial de código fuente [3]. Sin embargo, estas herramientas no brindan apoyo a los procesos previos a la realización de los modelos, y esta labor se ha dejado completamente en manos de los analistas, quienes deben interpretar el dominio del problema para plasmarlo posteriormente en los modelos.

Desde mediados de los noventa comenzó una tendencia que buscaba el apoyo a la gestión de los analistas colaborándoles en la obtención automática de ciertos elementos de los diagramas entidad-relación [5], los diagramas de clases de UML [4], [7], [8] y [9], los diagramas de actividades de UML [9] y un conjunto de diagramas correspondientes a los sistemas automáticos de control [6]. En esta tendencia se han realizado progresos importantes, pero aún existen diagramas que no se han trabajado (por ejemplo comunicación y máquina de estados, ambos de UML) y además los diagramas generados aún poseen problemas de coherencia e interrelación entre sus elementos, cuando esos diagramas pertenecen al mismo modelo [1]; comúnmente, esos problemas se suelen denominar "errores de consistencia").

Con el fin de superar algunas de estas limitaciones, en este artículo se presenta un entorno para la obtención automática de tres tipos de diagramas UML 2.0, que incluye los siguientes elementos:

- El lenguaje controlado UN-Lencep para la redacción de las especificaciones textuales.
- Un diagrama intermedio para la representación del discurso, denominado Esquema Preconceptual.
- Un conjunto de reglas de traducción a los diagramas de UML.

Además, se presenta y ejemplifica una herramienta denominada UNC-Diagramador que utiliza este entorno.

El artículo está organizado de la siguiente manera: en la Sección 2, se resumen algunos trabajos en obtención automática de esquemas conceptuales (entre los que se incluyen los diagramas UML); en la Sección 3, se definen los lineamientos del Esquema Preconceptual; en la Sección 4, se definen las reglas para el mapeo entre el Esquema Preconceptual y tres tipos de diagramas UML 2.0; en la Sección 5, se presenta la herramienta UNC-Diagramador y un caso de estudio para ejemplificar el uso del entorno. Las conclusiones y trabajos futuros se incluyen en las Secciones 6 y 7 respectivamente.

2. LA OBTENCIÓN AUTOMÁTICA DE DIAGRAMAS UML A PARTIR DE LENGUAJE NATURAL

En los inicios de la Ingeniería del Software, Chen propuso el diagrama entidad-relación (DER) [10] como un estándar para describir los datos de un problema; posteriormente, definió un conjunto de reglas que permitían extraer de un discurso en inglés el DER [11]. Esta iniciativa fue imitada posteriormente por Coad y Yourdon [12] para el diagrama de clases, en el diseño orientado por objetos. Estos trabajos previos nunca llegaron a ser implementados de manera directa, pero sirvieron de fundamento a la ola posterior de trabajos en obtención automática de esquemas conceptuales. Además, tanto las reglas de Chen como las de Coad y Yourdon poseían un alto grado de ambigüedad, ya que un mismo elemento del discurso podía ser utilizado por reglas diversas para ser transformado a diferentes elementos del DER o del Diagrama de Clases, según fuera el caso.

Posteriormente, la elaboración de los diferentes diagramas se apoyó en herramientas CASE, las cuales a la fecha continúan con su desarrollo y vigencia, especialmente en el trazado y edición de diagramas y la generación parcial de código a partir de los diagramas; sin embargo, estas herramientas no constituyen un avance sustancial en la obtención automática de los diagramas UML, puesto que dejan la interpretación del

discurso del interesado completamente en manos del analista [3].

Un paso más adelante en este campo se dio con el proyecto LIDA (LInguistic Domain Analysis) de Overmyer *et al.* [4], con el cual se realizaba una clasificación de las palabras incluidas en un discurso sobre el dominio, en verbos, sustantivos y adjetivos, con sus respectivas frecuencias de aparición; con esta información se buscaba suministrar elementos para que el analista tomara las decisiones pertinentes para el trazado del diagrama de clases. La herramienta desarrollada tenía capacidades para la edición de los diagramas resultantes, para que fueran completados por el analista, con base en su experiencia y criterio. El proceso completo en LIDA no se puede realizar sin la mediación del analista, convirtiéndolo en un proceso semiautomático, pero con alta dependencia del analista.

En el proyecto RADD, Buchholz y Düsterhöft [5] procuraron elaborar un DER a partir de especificaciones en lenguaje natural y luego promover la completitud del diagrama obtenido mediante un diálogo controlado con el usuario; en este proyecto se promovió únicamente el trazado del DER, sin abordar el tema de la elaboración de otros diagramas o la consistencia entre diagramas resultantes.

En el proyecto ASPIN, Cyre [6] procuró la consolidación de los diferentes diagramas del ámbito de los sistemas automáticos de control en un modelo único representado en un lenguaje común. Si bien Cyre no buscaba la obtención automática de diagramas, con ASPIN demostró que una representación unificada puede contener una gran cantidad de diagramas de manera simultánea. La desventaja principal de esta propuesta radica en lo restringido de su dominio, puesto que sólo se puede mostrar aplicabilidad para los sistemas automáticos de control.

Otras propuestas como NL-OOPS Mich [7] y CM-BUILDER de Harmain y Gaizauskas [8] se enfocan en la construcción de un único diagrama: el diagrama de clases de UML. Estos proyectos emplean redes semánticas como representaciones intermedias entre lenguaje natural y los esquemas conceptuales; algunas de sus desventajas son:

- Sólo obtienen el diagrama de clases (y no otros diagramas UML).
- La representación intermedia mediante redes semánticas no permite representar las características dinámicas del modelo del discurso.

Fliedl *et al.* [9] continuaron en esta tendencia presentando el proyecto NIBA, el cual busca la elaboración de diferentes diagramas UML (especialmente clases y actividades, aunque establecen que se podrían obtener otros como secuencias y comunicación), empleando para ello un conjunto de esquemas intermedios que denominaron KCPM (Klagenfurt Conceptual Predesign Model). KCPM posee formas diferentes de representación del conocimiento para los diferentes diagramas de UML, variando desde tablas con información especial para el diagrama de clases, hasta diagramas dinámicos especiales para el diagrama de actividades [9]; esto puede ocasionar cierta pérdida de información entre diagramas y, consecuentemente, fallas de consistencia entre los mismos. Así pues, elementos de los diagramas que requieran ser definidos a partir de información estática y dinámica (como por ejemplo los mensajes del diagrama de comunicación o los estados del diagrama de máquina de estados) pueden no ser obtenidos de manera consistente con otros diagramas.

Debido a las limitaciones anotadas (a veces con énfasis en un solo diagrama, en ocasiones generando varios diagramas pero con problemas de consistencia, o con procesos semiautomáticos con alta participación del analista) para los diferentes trabajos en el área, se plantea la siguiente hipótesis para este trabajo:

Hipótesis Inicial: A partir de un subconjunto del Lenguaje Natural (un lenguaje controlado), es posible definir un Esquema Preconceptual, del cual es posible extraer los diferentes elementos que permiten el trazado de esquemas conceptuales UML de tipo estructural, comportamental o de interacción, empleando reglas heurísticas de transformación.

El término "Esquema Preconceptual" [16] que se menciona en la hipótesis ha sido creado por el Grupo de Ingeniería de Software de la Escuela de Sistemas de la Universidad Nacional de Colombia y sus lineamientos teóricos se explican en la sección siguiente.

3. ESQUEMAS PRECONCEPTUALES: LINEAMIENTOS TEÓRICOS

En algunas de las propuestas analizadas en la sección anterior, se emplean formas intermedias de representación del conocimiento (redes semánticas, grafos conceptuales, tablas de información y diagramas dinámicos) que facilitan la transición hacia los diferentes diagramas (DER o UML). En esta propuesta se define un nuevo tipo de representación intermedia que se denomina "Esquema Preconceptual" [16].

Los orígenes del término "preconceptual" se remontan a la filosofía: para Heidegger [13] lo preconceptual aludía a información previa que se empleaba en la construcción de conceptos. El término fue posteriormente empleado por Piaget [14] en el ámbito educativo; en efecto, Piaget, dentro de sus etapas del conocimiento, incluye la que se denomina "Etapa Preconceptual" y la sitúa después de la adquisición lingüística, pero previa a la etapa de conceptualización del conocimiento. El proceso que se realiza en este artículo para la obtención automática de esquemas conceptuales parte del conocimiento lingüístico y se afianza con una representación intermedia que permite llegar a los esquemas conceptuales [16].

De las propuestas que emplean mecanismos intermedios de representación, la propuesta de Cyre [4], que emplea Grafos Conceptuales (GC) es la única que presenta una cierta unión de las características estructurales con las dinámicas. El Esquema Preconceptual (EP) posee ciertas similitudes con los GCs (tales como el uso de los conceptos en rectángulos y las relaciones en óvalos), pero también busca solucionar algunos inconvenientes de los GCs, tales como:

- Los GCs proveen el mecanismo denominado "correferencia" para la representación simultánea de varias frases, que hacen necesario replicar el mismo concepto varias veces en el grafo. Esto puede conducir a duplicidad de información y dificultades para establecer el comportamiento de un concepto.
- Los GCs representan de manera adecuada información de tipo estructural, pero requieren elementos especiales para representar las características dinámicas del discurso, que se requerirían para modelar un esquema intermedio unificado, alejándose, de esta forma, del estándar definido para ellos.
- En los GCs un mismo elemento puede pertenecer a categorías gramaticales diferentes; Por ejemplo, un rectángulo (concepto) puede ser un sustantivo, un verbo o un adjetivo, en tanto que un óvalo (relación) puede ser un caso semántico o un tipo de relación especial. La representación de diferentes tipos de palabras con los mismos símbolos puede ser contraproducente para el análisis porque puede generar o reproducir ambigüedades que estaban presentes en el discurso original.

Los símbolos que emplean los EPs se muestran en la Figura 1.

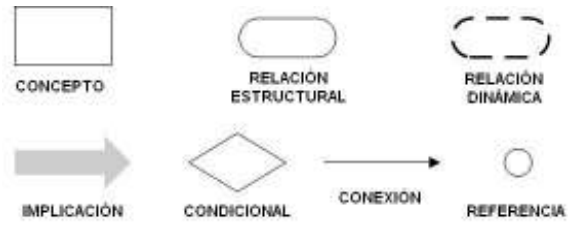


Figura 1. Símbolos de los Esquemas Preconceptuales.
Figure 1. Pre-conceptual Schemas Symbols.

El significado de los símbolos es el siguiente:

- Conceptos: Son sustantivos del discurso del interesado; también pueden ser sintagmas nominales del tipo sustantivo-preposición-sustantivo, como por ejemplo "departamento de pedidos". Cada concepto aparece sólo una vez en cada EP, por lo cual, a medida que se generan representaciones de un concepto, se van sumando relaciones a ese concepto. Esto

permite que el Esquema Preconceptual se grafique de manera unificada y no separando frases independientes como es el caso de los GCs.

- Relaciones estructurales: Son verbos que generan conexiones permanentes entre los conceptos. Básicamente se reconocen en esta categoría únicamente los verbos "es" y "tiene".

- Relaciones dinámicas: Son verbos que denotan acciones u operaciones en el mundo. Algunos ejemplos son: "registra", "paga", "presenta", etc. Una identificación de las relaciones dinámicas a partir de un discurso en lenguaje natural se puede lograr con el método descrito en [15].

- Implicaciones: Expresan relaciones de causa-efecto entre relaciones dinámicas o entre condicionales y relaciones dinámicas. Tienen el mismo significado de la implicación lógica, en la cual se requiere que el antecedente se cumpla (ya sea relación dinámica o condicional) para que el consecuente (siempre una relación dinámica) se cumpla.

- Condicionales: Son expresiones conformadas por conceptos y operadores entre ellos que sirven como precondición a una relación dinámica.

- Conexiones: Son flechas que unen los conceptos con relaciones dinámicas o estructurales y viceversa.

- Referencias: Son círculos numerados que permiten ligar elementos físicamente distantes en el diagrama.

Los Esquemas Preconceptuales se pueden obtener a partir del denominado UN-Lencep (Universidad Nacional de Colombia—Lenguaje Controlado para la Especificación de Esquemas Preconceptuales) [17].

En la Tabla 1 se muestra la construcción formal del UN-Lencep con algunas expresiones de equivalencia en lenguaje controlado; en la Figura 2 se muestran las reglas de conversión de la especificación básica de UN-Lencep en Esquemas Preconceptuales. A partir de las reglas incluidas en la Tabla 1, es posible realizar la descripción de un dominio cualquiera mediante expresiones en Lenguaje Natural Controlado; posteriormente, estas expresiones se pueden traducir a la construcción formal de UN-Lencep, para luego ser traducidas (empleando las equivalencias de la Figura 2) a Esquemas Preconceptuales. Por ejemplo, la expresión en lenguaje natural controlado "un curso está compuesto por estudiantes" se traduce en la construcción formal "curso tiene estudiante" y luego se transforma en Esquema Preconceptual como el concepto "curso" unido mediante la relación estructural "tiene" al concepto "estudiante". Si bien es posible describir cualquier dominio en UN-Lencep, se debe notar que las limitaciones estructurales del lenguaje (por ejemplo, las frases sujeto-verbo-objeto, la carencia de plurales y la limitación en el uso de adverbios pueden hacer difícil su utilización) pueden dificultar su uso.

4. REGLAS DE CONVERSIÓN ENTRE ESQUEMAS PRECONCEPTUALES Y ESQUEMAS CONCEPTUALES DE UML

Una vez obtenidos los Esquemas Preconceptuales, es posible obtener un conjunto de diagramas UML; para efectos de este trabajo se seleccionaron tres diagramas correspondientes a la versión 2.0 de UML: Clases, Comunicación y Máquina de Estados. La elección de estos diagramas obedece a la posibilidad de expresar las características de un dominio particular en términos de su estructura (clases), interacción (comunicación) y comportamiento (máquina de estados).

Es de notar que aún no es posible obtener, a partir de los Esquemas Preconceptuales, la totalidad de los elementos que hacen parte de los diagramas de UML mencionados. Algunos de los elementos faltantes son las cardinalidades, los roles de las relaciones, los estados compuestos y los ciclos.

Tabla 1. Equivalencia entre la construcción formal de UN-Lencep y algunas expresiones en Lenguaje Natural Controlado
Table 1. Equivalence between UN-Lencep formal construction and some controlled natural language expressions

Construcción Formal	***Expresión en Lenguaje Natural Controlado***	
A <ES> B	A *es una especie de* B A *es un tipo de* B	A *es una clase de* B B *se divide en* A
A <TIENE> B	A *incluye* B A *contiene* B A *posee* B A *está compuesto por* B A *está formado por* B B *pertenece a* A	B *es una parte de* A B *está incluido en* A B *está contenido en* A B *es un elemento de* A B *es un subconjunto de* A
A <R1> B	<R1> puede ser cualquier verbo dinámico, por ejemplo: A *registra* B, A *paga* B	
C <R2> D, <SI> A <R1> B	*si* A <R1> B *entonces* C <R2> D *Dado que* A <R1> B, C <R2> D *Luego de que* A <R1> B, C <R2> D	
<SI> {COND} <ENTONCES> A <R1> B, <SINO> C <R2> D	{COND} es una condición expresada en términos de conceptos. <R1> y <R2> son verbos dinámicos. <SINO> es opcional, por ejemplo: *si* M *es mayor que 100 entonces* A *registra* B	

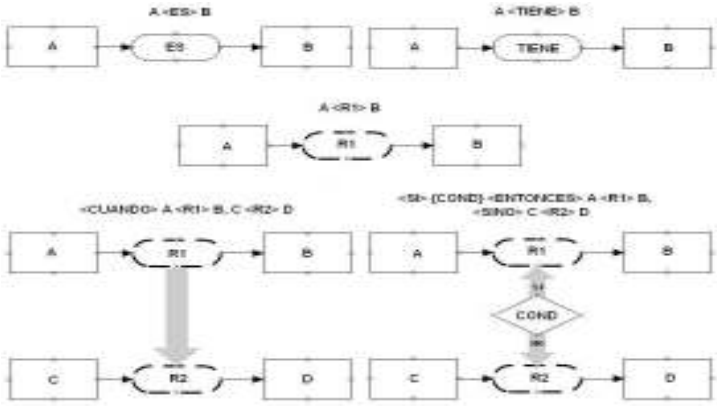


Figura 2. Reglas para la traducción de UN-Lencep a Esquemas Preconceptuales.
Figure 2. Rules for translating from UN-Lencep to Pre-conceptual Schemas.

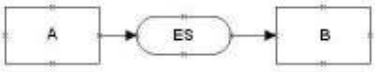
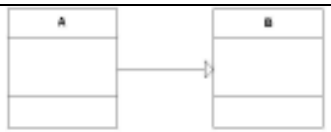
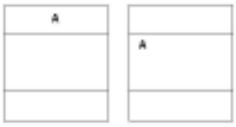
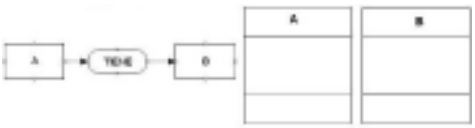
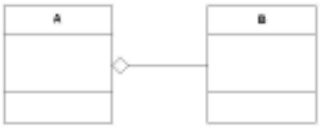
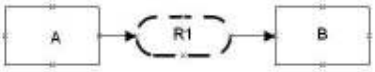
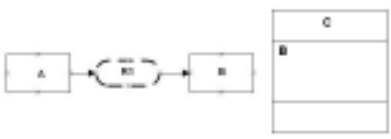
Figura 3. Símbolos de UML 2.0
Figure 3. UML 2.0 Symbols.

En la Figura 3 se muestran los elementos básicos de los diagramas de clases, comunicación y máquina de estados correspondientes a UML 2.0 que actualmente es posible identificar mediante las reglas de conversión.

Una descripción más detallada de estos diagramas y los restantes que hacen parte de la especificación de UML 2.0 se puede consultar en [18]; la especificación completa se puede consultar en [2].

Adicionalmente, la expresión gráfica de las reglas que permiten la obtención de los diagramas de UML 2.0 a partir de los Esquemas Preconceptuales se compendian en la Tabla 2. En la columna "Precondición" se encuentran combinados elementos de los Esquemas Preconceptuales con elementos de otros diagramas, puesto que algunas reglas emplean de manera recursiva los elementos que se van identificando a lo largo del proceso (véanse, por ejemplo, las reglas 3, 4, 6, 7, 8, 11 y 15). En las reglas 5 y 6 la identificación de la relación dinámica como operación del diagrama de clases se garantiza utilizando adecuadamente los verbos que denotan acciones desde la descripción en UN-Lencep; un método automático de reconocimiento de estos verbos se puede consultar en [15]. Las reglas se aplican haciendo varios barridos del diagrama (a excepción de la regla 3 que puede ser considerada una metaregla) hasta que todos los elementos del esquema preconceptual se han intentado mapear a sus equivalencias en los demás diagramas.

Tabla 2. Reglas para la obtención de los diagramas de UML 2.0 (Continúa).
Table 2. Rules for obtaining UML 2.0 diagrams (To be continued).

No.	*Precondición*	*Resultado*
1		
2		
3		
4		
5		
6		

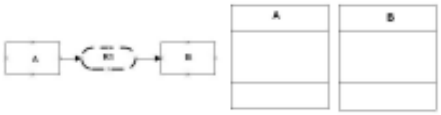
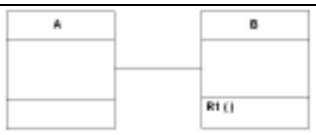
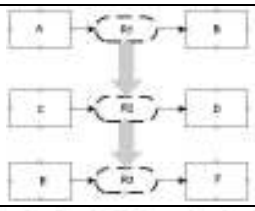
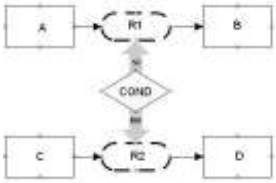
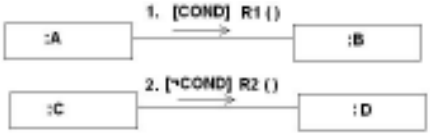
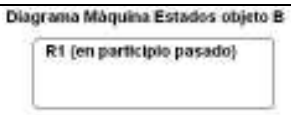
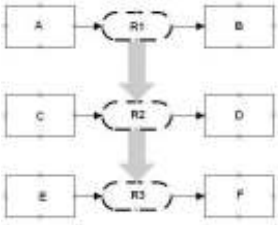
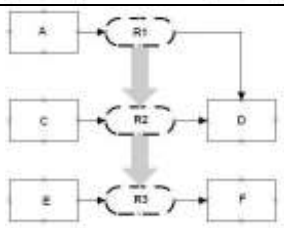
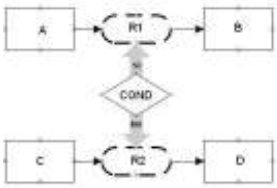
No.	*Precondición*	*Resultado*
7		
8		
9		
10		
11		
12		
13		
14		
15		

Tabla 2. Reglas para la obtención de los diagramas de UML 2.0
Table 2. Rules for obtaining UML 2.0 diagrams

5. UNC-DIAGRAMADOR: UNA HERRAMIENTA CASE BASADA EN UN-LENCEP Y ESQUEMAS PRECONCEPTUALES

5.1 Descripción del UNC-Diagramador

La herramienta CASE UNC-Diagramador emplea la hipótesis definida en la Sección 2 para intentar solucionar algunas de las limitaciones que aún subsisten en la obtención automática de Esquemas Conceptuales de UML 2.0 a partir de lenguaje natural. Esta herramienta ha sido desarrollada por el Grupo de Ingeniería de Software de la Escuela de Sistemas de la Universidad Nacional de Colombia. Para su construcción se han empleado las ventajas de la tecnología .NET de Microsoft® y su lenguaje C#, combinándolas con las posibilidades gráficas de Microsoft Visio®. El módulo de manejo del UN-Lencep se desarrolló en lenguaje PHP.

UNC-Diagramador realiza las siguientes funciones:

- Permite el ingreso de frases en lenguaje controlado que se convierten en un archivo en Un-Lencep.
- Convierte el archivo en UN-Lencep en el Esquema Preconceptual correspondiente.
- Obtiene automáticamente los diagramas de clases, comunicación y máquina de estados de UML 2.0, correspondientes al Esquema Preconceptual generado.

5.2 Ejemplo de Aplicación

Seguidamente, y a manera de ejemplo, se presenta un conjunto de frases en lenguaje natural controlado que describen parcialmente el dominio de una clínica veterinaria. Este caso de estudio se emplea para mostrar el funcionamiento del UNC-Diagramador para la generación automática de esquemas conceptuales de UML 2.0.

- *El propietario posee una mascota*
- *La identificación es un elemento de una mascota*
- *Un nombre pertenece a una mascota*
- *Una mascota posee una historia_clínica*
- *El número es un elemento de una historia_clínica*
- *El detalle es un elemento de una historia_clínica*
- *La fecha es un elemento de detalle*
- *El detalle contiene un diagnóstico*
- *El detalle contiene un medicamento*
- *Dado que el propietario pide una cita, la secretaria asigna la cita*
- *Si el propietario cumple la cita, el veterinario revisa la mascota*
- *Luego de que el veterinario revisa la mascota, el veterinario registra el diagnóstico*
- *Si el veterinario registra el diagnóstico, entonces el veterinario receta un medicamento*

Estas frases se deben introducir una a una en la interfaz que se muestra en la Figura 3, correspondiente al módulo de procesamiento de UN-Lencep.

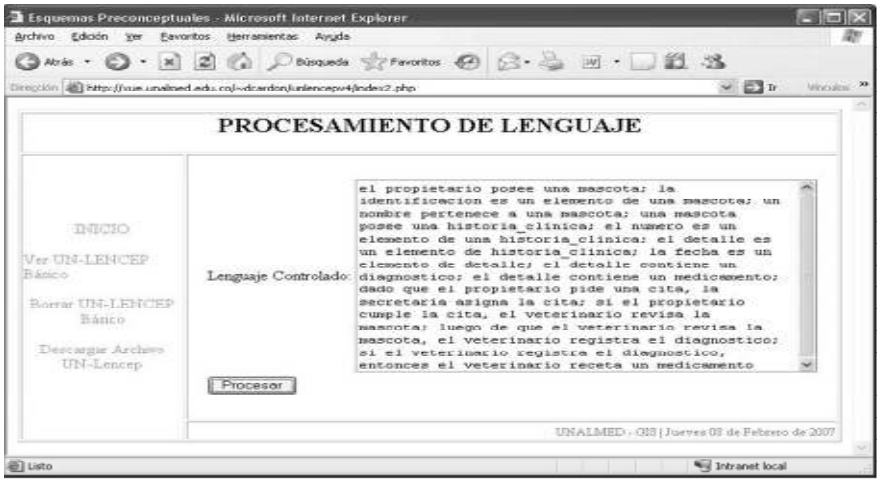


Figura 3. Interfaz para la generación de UN-Lencep a partir de un Lenguaje Controlado.
Figure 3. Graphical User Interface for UN-Lencep generation from a controlled language.

Una vez se han ingresado todas las frases, se oprime el vínculo "Descargar Archivo UN-Lencep" para obtener la especificación siguiente:

ST propietario TIENE mascota
ST mascota TIENE identificacion
ST mascota TIENE nombre
ST mascota TIENE historia_clinica
ST historia_clinica TIENE numero
ST historia_clinica TIENE detalle
ST detalle TIENE fecha
ST detalle TIENE diagnostico
ST detalle TIENE medicamento
IM Cuando PROPIETARIO PIDE CITA entonces SECRETARIA ASIGNA CITA
IM Cuando PROPIETARIO CUMPLE CITA entonces VETERINARIO REVISA MASCOTA
IM Cuando VETERINARIO REVISA MASCOTA entonces VETERINARIO REGISTRA DIAGNOSTICO
IM Cuando VETERINARIO REGISTRA DIAGNOSTICO entonces VETERINARIO RECETA MEDICAMENTO

La abreviatura ST denota una frase de tipo estructural e IM denota una frase de implicación. Esta especificación se puede leer con el UNC-Diagramador para obtener el Esquema Preconceptual que se muestra en la Figura 4, el cual posibilita una revisión visual de la estructura de las frases con el fin de subsanar posibles errores aún presentes en el UN-Lencep; en caso de encontrar errores u omisiones, es posible modificar nuevamente el UN-Lencep o simplemente realizar las modificaciones pertinentes en el esquema preconceptual resultante. Finalmente, luego de presionar el ícono de UML ubicado en la parte superior derecha de la Figura 4, se obtienen los tres diagramas de UML 2.0 que se mencionan en las reglas de la Sección 4; estos diagramas se muestran en las Figuras 5, 6 y 7.

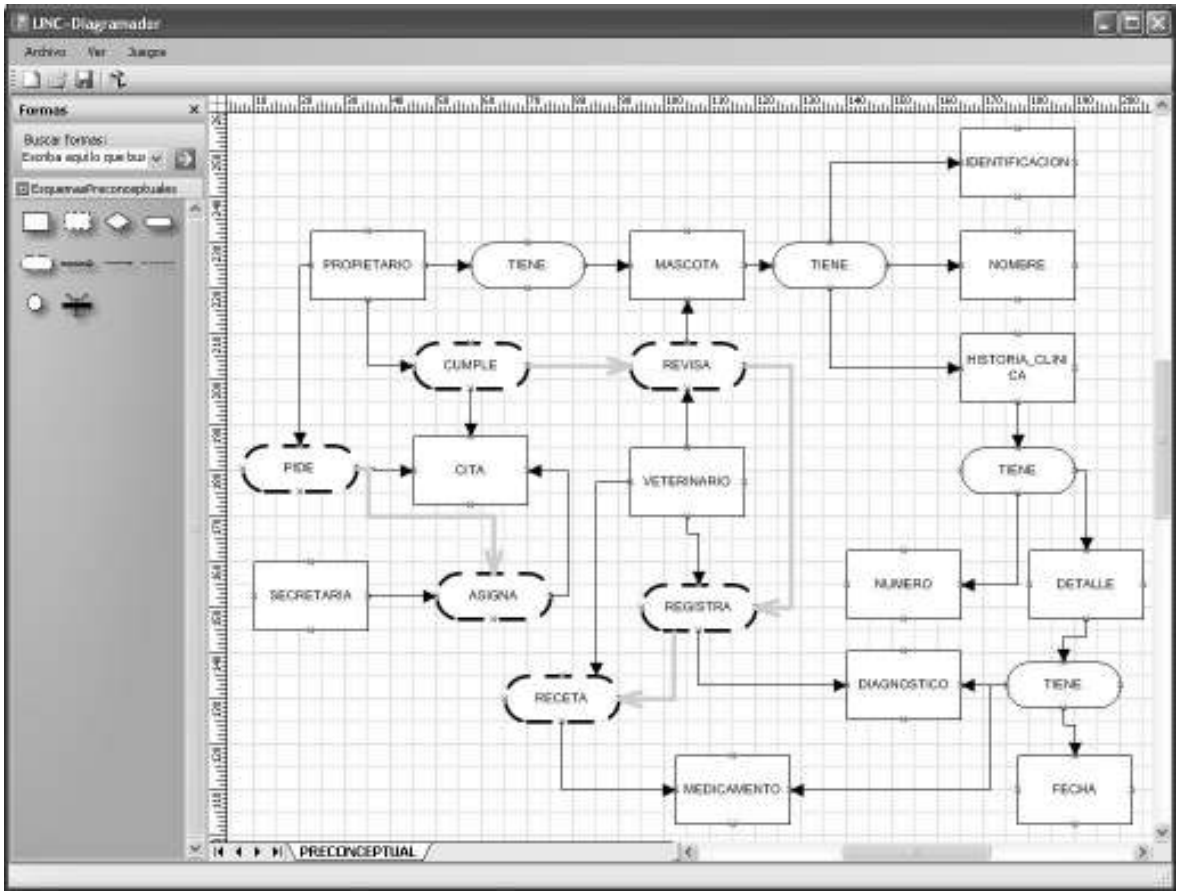


Figura 4. Esquema Preconceptual de una clínica veterinaria en UNC-Diagramador.
Figure 4. Pre-conceptual Schema for a pet clinic in UNC-Diagramador.

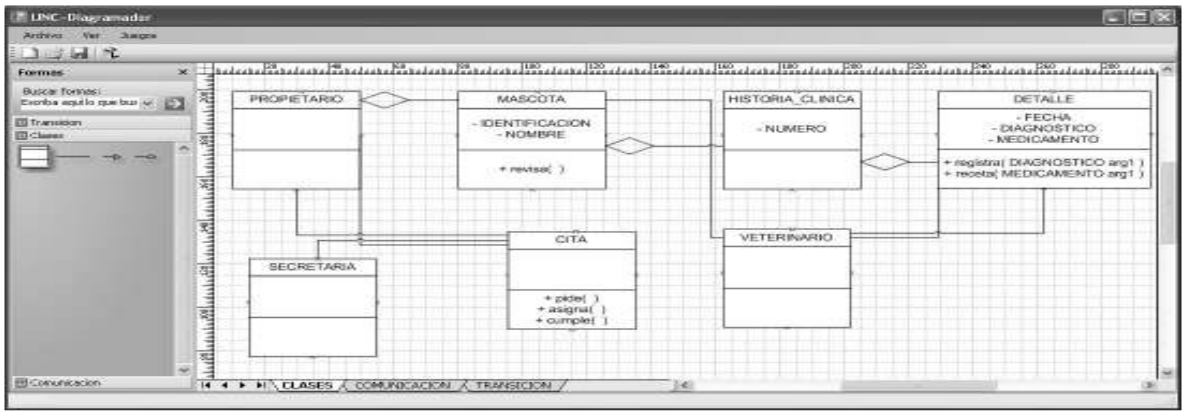


Figura 5. Diagrama de clases correspondiente al Esquema Preconceptual de la Figura 4.
Figure 5. UML class diagram corresponding to Pre-conceptual Schema from Figure 4.

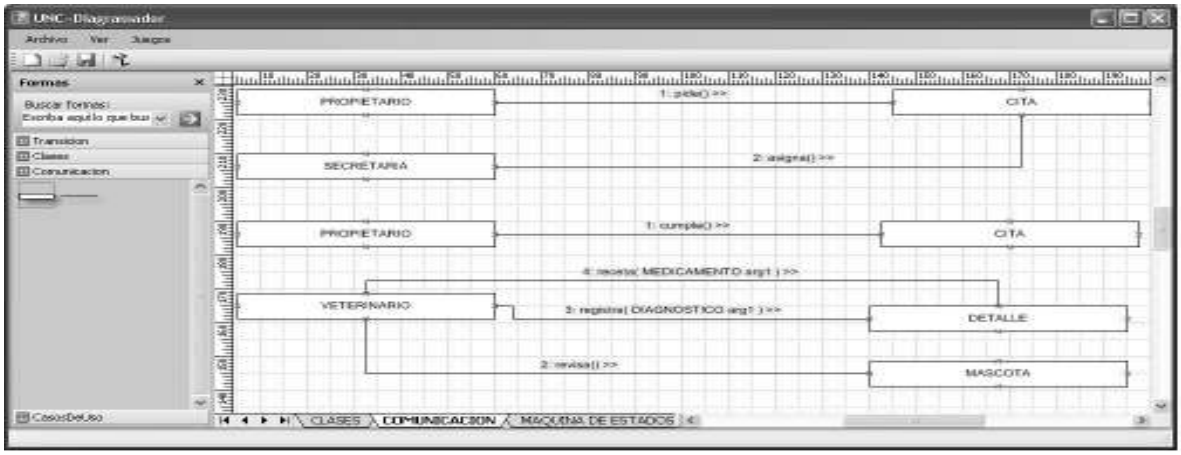


Figura 6. Diagramas de comunicación correspondientes al Esquema Preconceptual de la Figura 4.
Figure 6. UML communication diagrams corresponding to Pre-conceptual Schema from Figure 4.

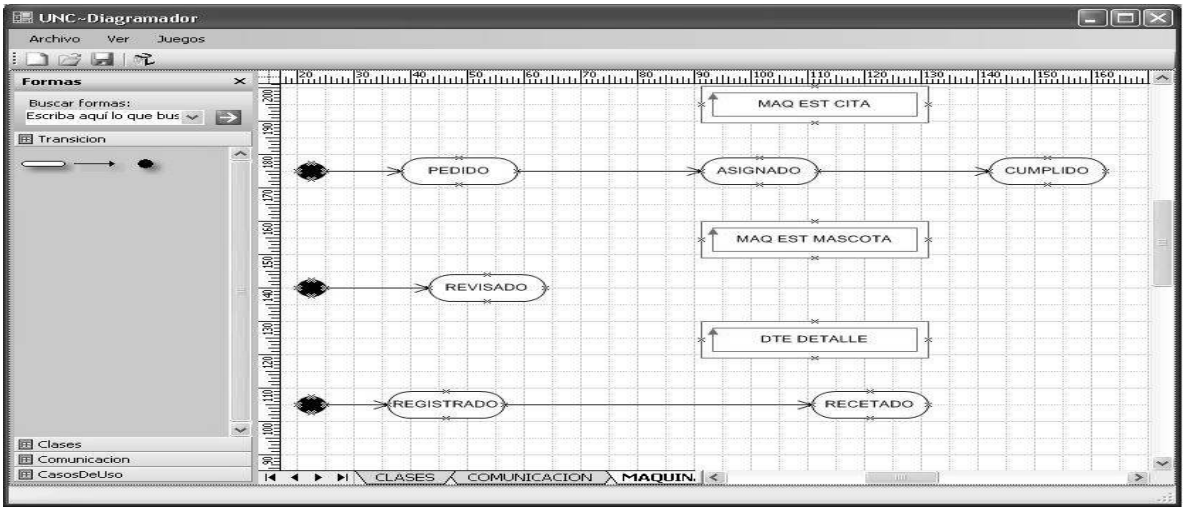


Figura 7. Diagramas de máquina de estados correspondientes al Esquema Preconceptual de la Figura 4.
Figure 7. UML state machine diagrams corresponding to Pre-conceptual Schema from Figure 4.

Las reglas se aplican por ciclos que únicamente culminan cuando ya no quedan reglas pendientes por evaluar. Por ejemplo, en el ciclo inicial el concepto "mascota" se identifica como un atributo mediante la aplicación de la regla 1 sobre la frase "propietario TIENE mascota", pero luego se identifica como clase cuando se aplica la regla 1 a la frase "mascota TIENE identificación". En un ciclo posterior se define que efectivamente es una clase aplicando la regla 3 y por la aplicación de la regla 4 se determina que tanto "propietario" como "mascota" son clases y que poseen una relación de agregación entre ellas. En la Figura 5 se puede apreciar el resultado de la aplicación de las reglas para estos dos conceptos.

En el caso del diagrama de comunicación, se identifican dos secuencias de implicaciones independientes que generan dos diagramas diferentes de comunicación utilizando la regla 9. En efecto, el conjunto de relaciones dinámicas "pide" y "asigna", ambas correspondientes al concepto "cita", constituyen el primer diagrama de comunicación, en tanto que "cumple cita", "revisa mascota", "registra diagnóstico" y "receta medicamento" constituyen el segundo diagrama. Nótese que los conceptos "diagnóstico" y "medicamento" pertenecen ambos al concepto "detalle", lo que genera una variación de la regla 9 en la cual los mensajes se asignan a la clase de la cual otros conceptos son atributos. Los diagramas correspondientes se pueden visualizar en la Figura 6.

Un fenómeno similar ocurre en el diagrama de máquina de estados, en el cual los estados "registrado" y "recetado" provienen de elementos que fueron identificados en un ciclo anterior como atributos de la clase "detalle", lo cual constituye una variación a la aplicación de la regla 12. Los diagramas completos se pueden apreciar en la Figura 7.
Nótese adicionalmente que los diagramas resultantes son consistentes entre sí; por ejemplo, las clases de los objetos del diagrama de comunicación existen como clases en el diagrama de clases y los estados del diagrama de máquina de estados tienen su correspondencia en los mensajes del diagrama de comunicación. Otras reglas de consistencia entre los tres diagramas también se cumplen de manera automática, sin que el analista deba preocuparse por ello.

6. CONCLUSIONES

Siguiendo la tendencia hacia la obtención automática de diagramas UML, se ha presentado en este artículo un ambiente que persigue este fin empleando el lenguaje UN-Lencep, los Esquemas Preconceptuales y un conjunto de reglas heurísticas para realizar la transformación a tres diagramas específicos de UML 2.0 (Clases, Comunicación y Transición de Estados). Dichos diagramas se obtienen en una versión muy preliminar, lejos de las especificaciones detalladas del diseño, pero con sus principales primitivas conceptuales.
El ambiente se integró en la herramienta CASE UNC-Diagramador, que actualmente está en desarrollo en la Escuela de Sistemas de la Universidad Nacional, sede Medellín. Igualmente, se ejemplificó la herramienta con un caso de estudio correspondiente a una clínica veterinaria.

Mediante este ambiente, conjuntamente con la herramienta UNC-Diagramador que lo materializa, es posible demostrar la hipótesis planteada en la Sección 2. En este caso, el lenguaje controlado es el UN-Lencep y los tres diagramas resultantes representan los elementos correspondientes a los tres tipos mencionados de UML (estructura, comportamiento e interacción). Las reglas heurísticas presentadas en este artículo son un subconjunto de la totalidad de las reglas presentes en el UNC-Diagramador. El Esquema Preconceptual definido se emplea como punto de partida para las reglas de traducción a los diagramas de UML y como una manera de validar y corregir la información que se especifica en UN-Lencep. De esta manera, el UN-Lencep posibilita a los interesados la comunicación con los analistas en un lenguaje no técnico y a los analistas les permite expresar los modelos en términos entendibles por los interesados. La ganancia total de ambos es la agilización del proceso de desarrollo, al facilitar sus procesos de

comunicación en un lenguaje que, aunque sencillo, permite una gran parte de la expresividad de los diagramas de UML de manera entendible para los interesados.

7. TRABAJO FUTURO

Existen algunas líneas de interés que pueden dar continuidad al presente trabajo, entre las que se cuentan:

- La definición de reglas heurísticas que permitan la identificación de nuevos elementos de los diagramas descritos, tales como las multiplicidades de las clases en el diagrama de clases o las acciones "on entry" al interior de un estado del diagrama de transición de estados.
- La definición de reglas heurísticas que posibiliten la conversión del Esquema Preconceptual a otros diagramas UML (tiempos, secuencias, casos de uso, etc.).
- El enriquecimiento sintáctico de los Esquemas Preconceptuales con nuevos elementos que permitan la representación de otros tipos de palabras, tales como adjetivos y adverbios; estas palabras podrían incrementar la gama de los discursos representables mediante los Esquemas Preconceptuales. En esta línea de trabajo también se podrían considerar otros tipos de verbos diferentes a los ya considerados (estructurales y dinámicos), tales como los verbos de logro, que podrían dar la idea de objetivos.
- La definición de reglas heurísticas que posibiliten la obtención automática de los Esquemas Preconceptuales a partir del discurso en lenguaje natural. Sería particularmente importante el análisis de documentación de tipo técnico suministrada por los interesados en el desarrollo de una pieza de software, como por ejemplo reglamentos, manuales de funciones u otros.

8. AGRADECIMIENTOS

Este artículo se realizó en el marco de los siguientes Proyectos de Investigación: "Construcción Automática de Esquemas Conceptuales a partir de Lenguaje Natural", financiado por la DIME y "Definición de un Esquema Preconceptual para la Obtención Automática de Esquemas Conceptuales de UML", financiado por DINAIN y administrado por la DIME

REFERENCIAS

[1] ZAPATA, C. M. Y ARANGO, F. Los Modelos Verbales en Lenguaje Natural y su utilización en la elaboración de esquemas conceptuales para el desarrollo de software: Una revisión crítica, Revista Universidad EAFIT, Vol. 41, No. 137, 77–95, 2005.

[2] OMG: Object Management Group. OMG Unified Modeling Language Specification. Available: http://www.omg.org/UML/. [Citado 20 de Septiembre de 2006].

[3] PRESSMAN, R. Ingeniería del Software: Un enfoque práctico, McGraw–Hill, Madrid, 2002.

[4] OVERMYER, S.P., LAVOIE, B., AND RAMBOW, O. Conceptual modeling through linguistic analysis using LIDA. Proceedings of ICSE, Toronto, Canada, May 2001.

[5] BUCHHOLZ, E. AND DÜSTERHÖFT**, A.** Using Natural Language for Database Design. Proceedings of Deutsche Jahrestagung für Künstliche Intelligenz, Saarbrücken, Germany, September 1994.

[6] CYRE, W. A requirements sublanguage for automated analysis, International Journal of Intelligent Systems, Vol. 10, No. 7, 665–689, 1995.

[7] MICH L. NL-OOPS: From Natural Natural Language to Object Oriented Requirements using the Natural Language Processing System LOLITA, Journal of Natural Language Engineering, Cambridge University Press, Vol. 2, No. 2, 161–187, 1996.

[8] HARMAIN, H. Y GAIZAUSKAS**,** R**.** CM-Builder: An Automated NL-based CASE Tool. Proceedings of the fifteenth IEEE International

Conference on Automated Software Engineering (ASE'00), Grenoble, France, 2000.

[9] FLIEDL, G., KOP, CH., MAYR, H., MAYERTHALER, W. Y WINKLER, CH**.** Linguistically Based Requirements Engineering—The NIBA Project. Proceedings 4th Int. Conference NLDB'99 Applications of Natural Language to Information Systems, Klagenfurt, Austria, 177–182, June 1999.

[10] CHEN, P. P. The Entity–Relationship Model: Toward a Unified View of Data, ACM Transactions on DataBase Systems, Vol. 1, No. 1, 9–36, 1976.

[11] CHEN, P. P. English Sentence Structure and Entity–Relationship Diagrams, Information Science, No. 29, Vol. 2, 127–149, 1983.

[12] COAD, P. AND YOURDON, E. Object-Oriented Analysis, Yourdon Press, New Jersey, 1990.

[13] HEIDEGGER. M. Protokoll zu einem Seminar über den Vortrag "Zeit und Sein". Zur Sache des Denkens, Tübingen, Germany, 1976.

[14] PIAGET, J. The origins of intelligence in children (2nd ed.), International Universities Press, New York, 1952.

[15] JARAMILLO, A. F., ZAPATA, C. M. Y ARANGO, F. Una propuesta para el Reconocimiento Semiautomático de Operaciones utilizando un enfoque lingüístico, Revista Ingeniería Universidad de Antioquia, No. 34, 42–51, 2005.

[16] ZAPATA, C. M., ARANGO, F., AND GELBUKH, A. Pre-conceptual Schema: a UML Isomorphism for Automatically Obtaining UML Conceptual Schemas, Research in Computing Science: Advances in Computer Science and Engineering, Volume 19, 3–14, 2006.

[17] ZAPATA, C. M., GELBUKH, A. Y ARANGO, F. UN-Lencep: Obtención Automática de Diagramas UML a partir de un Lenguaje Controlado, Taller de Tecnologías del Lenguaje, Encuentro Nacional de Computación ENC06, México, 2006.

[18] FOWLER, M. UML Distilled: A brief guide to the Standard Object Modeling Language Addison–Wesley, Reading, 2004.