Analizador Sintáctico de Lenguaje Natural con Reglas Editables para la Generación de Primitivas UML

Natural Language Syntactic Analyzer with Editable Rules for Generating UML Primitives

Carlos M. Zapata, Ph.D. y Juan C. Hernández, Ing.
Grupo de Ingeniería de Software, Escuela de Ingenieria de Sistemas, Universidad Nacional de Colombia
cmzapata@unalmed.edu.co, jcherna0@unalmed.edu.co

Recibido para revisión 26 de Marzo de 2007, aceptado 15 de Junio de 2007, versión final 19 de julio de 2007

***Resumen*— En este artículo se presenta un analizador sintáctico automático de lenguaje natural basado en constituyentes, con la capacidad de modificar las reglas que utiliza para realizar el árbol sintáctico. El parser utiliza el método bottom-up para generar el árbol sintáctico y una verificación derecha-izquierda, izquierda-derecha, para la formación de los niveles del árbol. El parser fue construido como el primero de los módulos de una aplicación realizada para la obtención automática de primitivas UML a partir de lenguaje natural, de ahí la necesidad de optar por la edición de las reglas sintácticas para la ágil inclusión de nuevas formas oracionales y facilitar el procesamiento de los datos entregados a los demás módulos de análisis.**

***Palabras Clave*— Ambigüedad, Análisis Sintáctico, Diagramas UML, Lenguaje Natural.**

***Abstract*— We propose, in this paper, a constituent-based automated syntactic analyzer from natural language; it has capabilities for grammatical rules modification. The parser uses bottom-up method for syntactic tree generation and right-left, left-right verifying for level tree building. The parser itself was built as a part of an application for automatic obtaining of UML primitives from natural language; due to this fact, we need syntactic rules edition for incorporing new phrases and for easing data processing to other analysis modules.**

***Key words*— Ambiguity, Natural Language, Syntactic Analysis, UML Diagrams.**

I. Introducción

El análisis sintáctico es el encargado de realizar la verificación de las distintas reglas de formación de un lenguaje, siendo los resultados de dichos análisis representados gráficamente a través de una estructura jerárquica o árbol sintáctico. Por medio de estos árboles se define si una expresión realmente pertenece a un lenguaje [1]. El análisis sintáctico es el punto de partida del Procesamiento del Lenguaje Natural (NLP por sus siglas en inglés) y sus aplicaciones; en el contexto de este artículo, el análisis sintáctico se emplea como el paso inicial para la detección de primitivas conceptuales de UML. En una aplicación tal, la calidad del resultado final (los diagramas de UML para este ejemplo), dependerá de la versatilidad que tenga el analizador sintáctico empleado para la generación de las diferentes opciones de interpretación que se deriven de una frase; este es el insumo las decisiones correctas del analista en términos de los diagramas definitivos.

La realización del análisis sintáctico es una actividad tediosa que implica la revisión de las reglas de producción, las cuales definen un lenguaje, y la verificación de la frase que se analiza para que cumpla con estas reglas de formación. Considerando la realización de este proceso por parte de un ser humano, se vería implicada la mala interpretación de los elementos sintácticos del lenguaje, generando problemas de subjetividad e incompletitud que estarían reflejados a medida que incrementa la masa de los elementos a analizar.

Existen varios métodos para la realización automática o semiautomática del análisis sintáctico. Por ejemplo, en [2] se muestra una forma de realizar el análisis pero aún sin ser totalmente automático, dependiendo del juicio del usuario para tomar la mejor opción; en otros trabajos se realiza un análisis automático, pero su enfoque sólo va hacia las gramáticas asociativas por la izquierda, dejando de lado los demás métodos de asociación [3].

En este artículo se presenta una propuesta para la realización automática del análisis sintáctico de lenguaje natural, con la posibilidad de modificar las reglas de

producción utilizadas en el proceso. Este analizador es el punto de partida de un proyecto que busca la obtención automática de diagramas UML a partir de lenguaje natural[1].

Este artículo está organizado así: en la Sección 2 se describen algunos conceptos preliminares utilizados en el proceso del análisis; en la Sección 3 se presentan de manera crítica algunos trabajos relacionados; en la Sección 4 se detalla la propuesta de análisis sintáctico de lenguaje natural con reglas editables; en la Sección 5 se presenta un caso de estudio y en las Secciones 6 y 7 se presentan las conclusiones y los trabajos futuros respectivamente.

II. ANÁLISIS SINTÁCTICO DE UN TEXTO

La sintaxis es el estudio de los principios y de los procesos por los cuales las oraciones son construidas en idiomas particulares [4]. El análisis sintáctico de un texto es el proceso mediante el cual se pretende lograr la formación del árbol sintáctico, aplicando las reglas de producción de un lenguaje.

A. Gramáticas de contexto libre

Una gramática de contexto libre se puede entender como sigue:

$G = \{T, N, P, S\}$

$R \rightarrow E$

$R \subset N$

$E \subset \{R \cup T\}$

Donde G representa la gramática del lenguaje, T los símbolos terminales, N los símbolos no terminales, P las reglas de producción y S el símbolo objetivo del lenguaje.

En los lenguajes que implementan estas gramáticas se definen dos conjuntos finitos disjuntos; el primer conjunto está formado por los símbolos terminales T, equivalentes a las palabras y el segundo conjunto está formado por los símbolos no terminales N, equivalentes a los roles sintácticos. Este conjunto incluye una cantidad finita de reglas de producción P, correspondientes a las reglas gramaticales, y un símbolo objetivo S, que para el lenguaje natural es la oración.

Siendo R el símbolo no terminal que crea la regla y E los símbolos terminales o no terminales que la regla usa como argumentos para dar paso al símbolo R. El sistema utiliza los roles sintácticos asociados a cada palabra del lenguaje, además de las reglas sintácticas de los roles para la adquisición del árbol sintáctico. Este proceso requiere conocer previamente qué rol o roles puede estar jugando un símbolo del lenguaje, es decir cuáles son las relaciones directas entre los conjuntos de palabras y los roles sintácticos.

B. Árbol Sintáctico

El árbol sintáctico es un grafo jerárquico con la propiedad de que cada uno de sus nodos representa un rol sintáctico. El nodo raíz del árbol simboliza la unidad de expresión del lenguaje, que para el caso del lenguaje natural equivale a la oración; de igual manera, las hojas del árbol representan las ocurrencias textuales de las palabras, y los nodos intermedios, los diferentes roles sintácticos presentes para cada palabra, junto con los que se pueden generar a partir de las reglas de producción definidas para el lenguaje [4]. La forma como se conectan los nodos del árbol, está dada por un conjunto de reglas que definen qué secuencia de hijos puede formar un padre en un momento determinado. El objetivo de la realización del árbol sintáctico es, a partir de un conjunto de palabras, concluir si dicho conjunto está bien formado y equivale a una estructura que es reconocida para el lenguaje, el lenguaje natural en este caso. Dentro de la generación de los diagramas, el analizador sintáctico ejecuta el primer paso del análisis al cumplir el papel de verificar si lo que el usuario ha ingresado, es una estructura acorde con el lenguaje natural; además define las diferentes partes que conforman el árbol, para su posterior análisis por las demás aplicaciones que así lo requieran.

C. Ambigüedad

La ambigüedad en el contexto lingüístico es el fenómeno que ocurre cuando se puede obtener más de una interpretación a partir de una misma expresión dada [5]. Este fenómeno puede ser generado por la ambigüedad léxica (una palabra en un lenguaje determinado puede tener más de un rol sintáctico asociado) o por la ambigüedad estructural, (se puede representar una misma expresión con dos o más árboles sintácticos). Existen otros tipos de ambigüedad como la morfológica y la semántica que producen el mismo efecto en este tipo de aplicaciones, pero se salen del contexto de este artículo.

La ambigüedad es uno de los más grandes problemas en el desarrollo de las aplicaciones para el procesamiento del lenguaje natural, dada la naturaleza inherentemente ambigua de este tipo de lenguajes [5]. Es importante que un analizador sintáctico genere todos los posibles árboles sintácticos que sean permitidos por las reglas, con el fin de posibilitar la completitud necesaria que se requiere en los procesos subsecuentes, ya sea para continuar la representación de las frases en otras fases del proceso de análisis o para la realización de aplicaciones basadas en procesamiento del lenguaje natural como es el caso de la generación automática de diagramas de UML. En este último caso, los demás módulos de la aplicación reciben toda esta información e interpretan estos resultados para sus correspondientes análisis y la generación de los diagramas UML; teniendo en cuenta que existe más de una representación para cada oración, estos componentes pueden formar varias versiones de lo que sería un elemento del mundo, según se haya escrito inicialmente en el texto ingresado.

1 Este artículo se realizó en el marco de los siguientes proyectos de investigación: “Construcción automatica de esquemas conceptuales a partir de len-guaje natural”, financiado por la DIME y “Definición de un esquema pre conceptual para la obtención automática de esquemas conceptuales de UML”, financiado por DINAIN y administrado por la DIME

III. TRABAJOS RELACIONADOS

Los analizadores sintácticos de lenguaje natural buscan formar el árbol sintáctico de una oración, comprobando las reglas de formación del lenguaje. Para ello, existen métodos en los que se realiza la comparación de las reglas pero con la intervención del usuario a la hora de definir cómo se aplican las mismas [2]; esto constituye un problema cuando se desea que la aplicación que realiza el análisis sintáctico alimente otras aplicaciones automáticamente, como es el caso del analizador sintáctico que se propone en este artículo.

En otros métodos, se implementan las reglas gramaticales y se realiza el proceso de análisis sintáctico, pero sólo haciendo un recorrido de izquierda a derecha [3]; en otras palabras, cuando se presenta una regla de producción se compara cada vez con el elemento ubicado más a la izquierda y se procede con este mismo principio hasta que se termine de construir el árbol. Esto acarrea la desventaja de que las reglas siempre deberían estar definidas con asociación por la izquierda, lo que restringe el proceso de edición de las reglas y desecha muchos árboles sintácticos que pueden no cumplir con esta característica.

Otras propuestas trabajan con analizadores sintácticos con una estrategia top-down en la que el árbol se forma desde la raíz, intentando predecir los hijos que en un momento determinado deba asumir un nodo. Este método, sin embargo, exige un conjunto de reglas menos ambiguo que en otras estrategias [6]; esta rigidez es también un limitante para un analizador con reglas editables.

Tanto en [3] como en [6] la formación del árbol sintáctico depende de la forma de construcción de las reglas y no tanto de su aplicación, por lo cual se desechan diferentes opciones de representación para una determinada oración. Además, las aplicaciones que se generaron poseen un conjunto finito de reglas, que no puede ser modificado por los usuarios de la aplicación, limitando de esta forma el crecimiento futuro de las mismas.

IV. UNA PROPUESTA PARA EL ANÁLISIS SINTÁCTICO DEL LENGUAJE NATURAL CON REGLAS EDITABLES

El analizador sintáctico con reglas editables que se propone en este artículo cumple con dos funciones principales:

1) Realiza el análisis sintáctico del lenguaje natural de una oración, permitiendo incluir nuevas reglas de formación.
2) Permite manipular reglas sintácticas ambiguas con el fin de representar los posibles casos de ambigüedad, y con ello soportar la gramática ambigua del lenguaje. Debido a que los módulos podrían crecer en cantidad y complejidad, se definió un analizador sintáctico lo suficientemente dinámico para poder suplir la demanda de datos que pueden exigir los demás módulos.

A. El Analizador sintáctico

El analizador sintáctico utiliza el método bottom-up para generar el árbol sintáctico en la medida en la que se equiparen las reglas definidas y los roles sintácticos obtenidos para cada palabra. Dentro de la aplicación se decidió implementar este método, para evitar la rigidez que ofrece su contraparte, el método top-down, que para su eficiente desarrollo requiere menos ambigüedad por parte de las reglas [6]. El método utiliza un recorrido derecha-izquierda e izquierda-derecha según las reglas gramaticales que puedan ser aplicadas en un momento dado, para evitar los problemas esbozados en la asociación a izquierda [3].

Los roles sintácticos de cada palabra son obtenidos a partir de un lexicón que posee dichos roles junto con los roles semánticos; además, se constituye en un punto de partida para la generación de diagramas UML. El lexicón puede ser enriquecido con las distintas ocurrencias que se puedan presentar en un lenguaje, teniendo en cuenta que requiere no sólo información sintáctica sino también semántica.

Las aplicaciones que intentan analizar la estructura sintáctica de un lenguaje se enfrentan al fenómeno de la ambigüedad, que se puede dar debido a fenómenos como: redundancia en la definición de las reglas gramaticales y/o variedad de roles sintácticos que pueda tener una palabra; se pueden generar así varias representaciones sintácticas para una misma oración. El analizador muestra los distintos árboles sintácticos que puedan ser generados como consecuencia de la ambigüedad presente.

Este proceso tiene a su favor la sencillez del algoritmo para la generación del árbol sintáctico, puesto que sólo necesita la secuencia de roles sintácticos permitidos por las reglas en un lenguaje. Posteriormente, se le permite al usuario definir sus propias reglas, modificándolas a su conveniencia, sin restricciones por la ambigüedad que genere el conjunto de reglas, y también le permite modificar las propiedades de las palabras que componen el lenguaje, sin tener que cambiar el código en el que se escribió el analizador sintáctico para definir un lenguaje o idioma totalmente nuevo.

B. El Proceso

La manera en que opera el analizador es la siguiente: después de haber ingresado el texto a analizar, se procede a separarlo en oraciones y cada una de éstas, a su vez, se separa en palabras. A cada palabra se le asignan consecutivamente los roles sintácticos que para ella se hayan encontrado en el lexicón. Luego, para cada oración se realiza la productoria de roles, que consiste en tomar todos los roles sintácticos de las palabras de la oración y formar todas las posibles combinaciones de roles sintácticos para ella; a esta combinación de roles sintácticos se le denomina nivel 1 del árbol. En este punto del análisis es donde se comienza a generar la ambigüedad de la que se hablaba anteriormente. Debido a que una palabra puede tener asociados varios roles sintácticos, como consecuencia se generan tantos niveles 1 por oración, como lo indica la siguiente fórmula:

$$V = \prod N_i$$

Siendo V el número de versiones posibles de árboles sintácticos y N_i el número de roles sintácticos de la palabra i.

Luego de obtener las distintas versiones del primer nivel de una oración, se realiza la aplicación de las reglas de formación por nivel.

Dentro de la aplicación una regla está definida de la siguiente forma:

$$R_i \rightarrow E_{i1} + E_{i2} + \ldots E_{in}$$

Donde R_i es el resultado de la regla i, que es un rol sintáctico, y E_{i1}, E_{i2} y E_{in} son los argumentos de la regla i, que también son roles sintácticos. Junto con todo esto cada regla posee otros tres atributos:

1) **Final:** indica si el rol sintáctico producido por dicha regla puede ser la raíz del árbol o no.
2) **Completa:** señala si la regla puede ser la última que se aplique en un nivel; esto se hace con el fin de facilitar la recursión de las reglas finales.

Teniendo esto en cuenta, el analizador sintáctico busca qué secuencias de roles sintácticos cumplen la misma secuencia de alguna regla que ya se haya definido; si se encuentra una secuencia de roles que coincida, se posiciona en el siguiente nivel el resultado de la regla, R_i, descartando la secuencia de caracteres del nivel anterior. Este proceso se repite hasta que ya no haya más elementos en el nivel actual, que es justo cuando se pasa al siguiente nivel. Cada vez que se aplican las reglas a un nivel, se verifica si ha cambiado la secuencia de roles sintácticos; si no ha cambiado dicha secuencia, es porque para la lista de roles de ese nivel no hay ninguna regla que la pueda convertir a un nivel superior y por ende se desecha la versión de la oración, en caso de que el nivel cambie, se compara con todos los niveles anteriores, buscando comportamientos cíclicos para también desechar esa versión de la oración, y se vuelve a empezar con la versión siguiente. Para la confirmación de la secuencia de roles, la búsqueda de estas secuencias se hace de izquierda a derecha, tomando el primer rol, viendo luego qué regla tiene como primer argumento este rol; si lo tiene, se mira si el segundo argumento de la regla coincide con el segundo rol del nivel y así sucesivamente, formando una secuencia; luego se aplica el mismo principio, pero de derecha a izquierda, buscando si alguna regla posee como último argumento el último elemento del nivel actual, repitiendo el mismo procedimiento. Si las versiones de un nivel producen resultados diferentes entre las secuencias de derecha a izquierda y de izquierda a derecha, se crea una nueva versión del árbol, duplicando todos los nodos de niveles inferiores; sino, se toma cualquiera de las dos secuencias. La verificación hacia delante y hacia atrás se realiza con el ánimo de mostrar si las reglas que han sido definidas son ambiguas, dado que en cualquiera de las dos direcciones que se realice el análisis sintáctico, el resultado debe ser el mismo, puesto que la ambigüedad depende del lenguaje y no de las reglas como es en el caso de [3] y [6].

Este procedimiento se realiza una y otra vez hasta que en un nivel determinado se llegue a que exista sólo un nodo o rol sintáctico; luego, se pregunta si este elemento puede ocupar el puesto de raíz del árbol. Esta información es suministrada cuando se define la regla, indicando que la regla es final; si el rol encontrado puede ocupar el puesto de raíz, se acepta el árbol sintáctico; de lo contrario, se rechaza la versión de la oración. Sea cual sea el resultado se continúa con la siguiente versión del árbol; si no hay más versiones, se continua con la siguiente oración, comenzando de nuevo el proceso. Si no hay más oraciones, el proceso termina. En las figuras 1, 2 y 3 se pueden apreciar algunas vistas del analizador sintáctico, que incluye el lexicón empleado para el proceso.

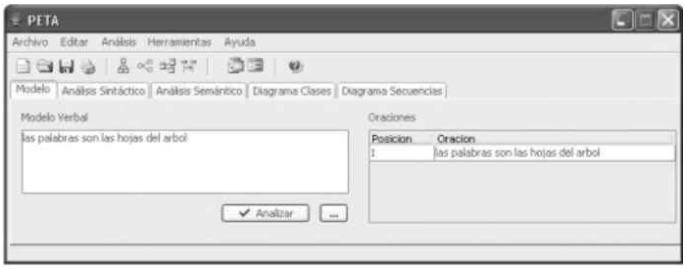

Figura 1. Ingreso del texto: cuando se ingresa el texto debe definirse un proyecto de trabajo.

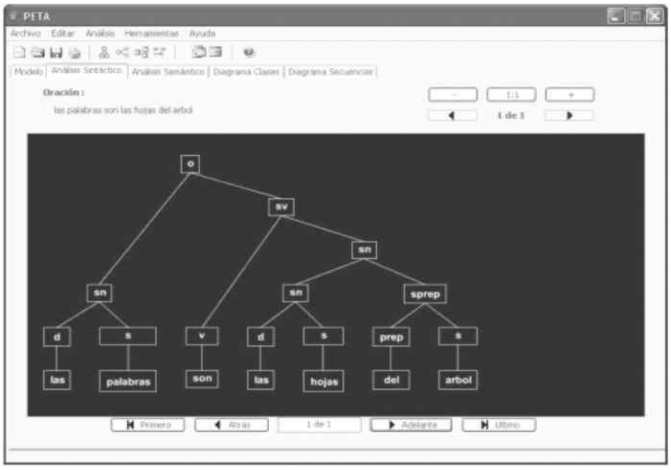

Figura 2. Gráfica del árbol sintáctico: Si la oración posee más de una posible representación sintáctica, la interfaz permite navegar por cada una de las versiones analizadas para cada oración.

V. CASO DE ESTUDIO

En esta Sección se muestra un ejemplo paso a paso del procedimiento utilizado para la obtención del árbol sintáctico. El primer paso del proceso está constituido por la adquisición de los roles sintácticos de cada palabra. Los pasos siguientes se pueden resumir en la iteración entre nivel y nivel, aplicando las reglas que hayan sido definidas para el lenguaje.

Este ejemplo muestra cómo la aplicación crea los distintos niveles del árbol. Por razones de espacio, en la Tabla I sólo se

muestra una lista reducida de las reglas necesarias para efectuar el proceso de análisis sintáctico y poder construir el árbol de la oración ejemplo. Las Figuras 4 a 9 muestran el proceso paso a paso.

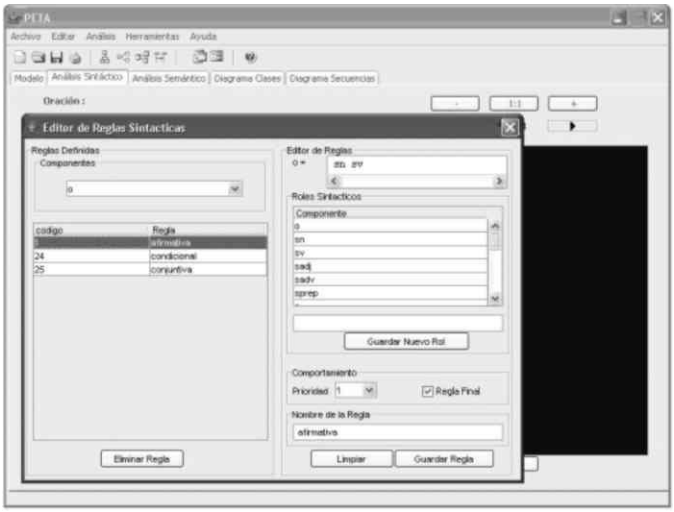

Figura 3. Interfaz para la edición de reglas: A la hora de definir una regla, la interfaz ofrece las opciones de eliminar reglas anteriores y de ingresar nuevos roles sintácti-cos que no estén presentes en el lexicón.

TABLA 1 Reglas Utilizadas para el Análisis sintáctico de la Oración "El hombre pinta la puerta de la casa de madera"

Código	Regla	Completa	Terminal
1	o → sn + sv	Si	Si
2	sn → d +s	No	No
3	sn → sn + sprep	No	No
4	sv → v + sn	Si	No
5	sprep → sprep + sprep	No	No
6	sprep → prep + sn	No	No
7	sprep → prep + s	No	No

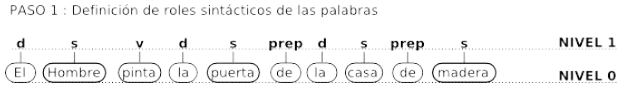

Figura 4. El primer paso del algoritmo: la obtención de los roles sintácticos desde el lexicón.

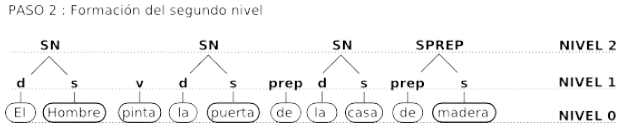

Figura 5. El segundo paso del algoritmo: aplicación de las reglas 2 y 7.

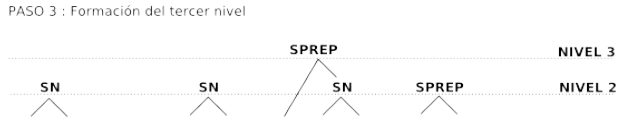

Figura 6. El tercer paso del algoritmo: aplicación de la regla 6.

Una de las aplicaciones futuras que se proponen para este analizador sintáctico es servir de base para la obtención automática de diagramas UML. En este caso, en la Figura 7 se muestra el efecto que podrían tener las dos versiones de la frase representadas en el diagrama de clases de UML.

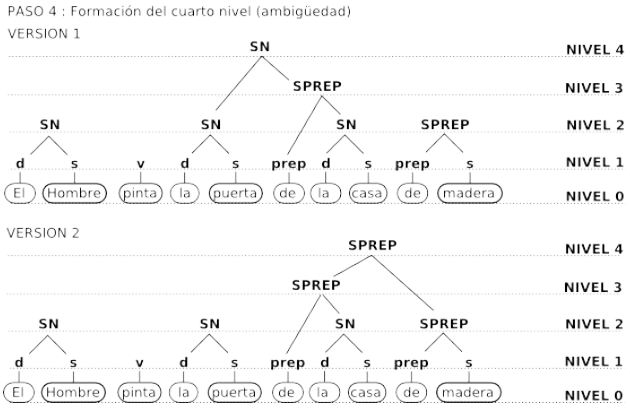

Figura 7. El cuarto paso del algoritmo: aplicación de la regla 3 (version 1) o de la regla 5 (version 2).

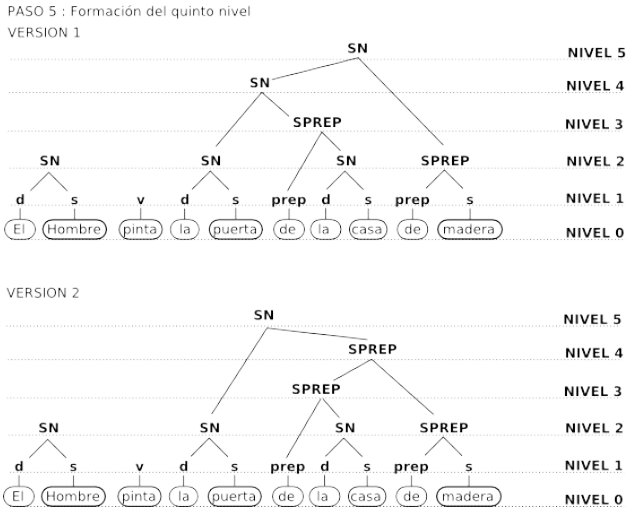

Figura 8. El quinto paso del algoritmo: aplicación de la regla 3 para ambas versiones.

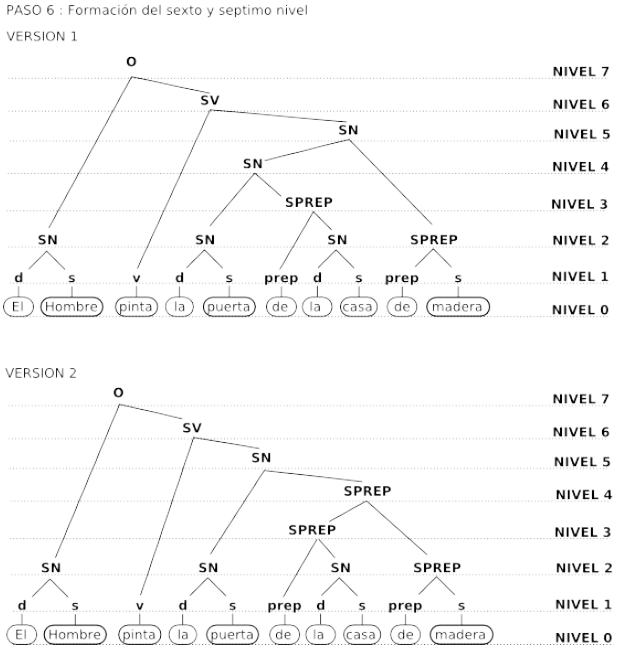

Figura 9. El sexto paso del algoritmo: aplicación de la reglas 4 y 1.

Si se toman en consideración reglas simples de conversión para el diagrama de clases (siendo éste un ejemplo de la aplicabilidad en la generación automática de diagramas UML), desde el análisis sintáctico se pueden convertir los sustantivos en clases, los verbos en las operaciones de las clases o las relaciones entre ellas, los adjetivos en los atributos de dichas clases o en los valores de tales atributos, las preposiciones en las distintas relaciones entre las clases y la ocurrencia de las palabras 'ser un' en una relación de herencia entre las clases implicadas.

Aplicando las reglas antes mencionadas, se puede ver que en la primera versión, el árbol indica que el hombre pinta (la puerta) que es 'de' (la casa) pero este primer algo (la puerta) es 'de' (madera); en la segunda versión se muestra que el hombre pinta (la puerta) que es 'de' (la casa) la cual es 'de' (madera). El resultado de la aplicación de las reglas se puede visualizar en la Figura 10.

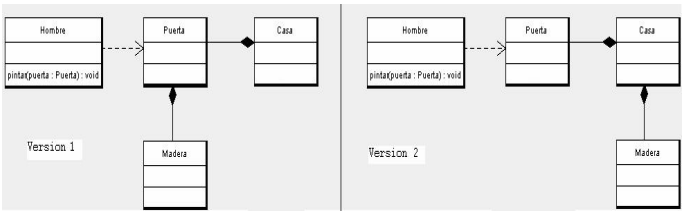

Figura 10. Primera etapa de generación del diagrama de clases (análisis de árboles sintácticos).

VI. CONCLUSIONES

En el artículo se muestra un analizador sintáctico automático que utiliza el método buttom-up para generar el árbol sintáctico; este método ofrece un manejo especial para la gestión de la ambigüedad, permitiéndole a la aplicación generar varias versiones del árbol sintáctico cuando está aplicando recorridos de izquierda a derecha y de derecha a izquierda, a medida que el árbol es construido. Para la edición de las reglas de producción, se requiere cierta habilidad por parte del usuario para escribir las reglas que requiere el lenguaje a ser usado. La aplicación en su conjunto habilita la opción, además, de editar el lexicón que es la base de datos donde residen los roles sintácticos asociados a cada palabra, permitir la inclusión de nuevas palabras y los roles sintácticos que estas puedan tener asociados. Dentro de las ventajas principales de la aplicación, figuran:

1) La generación de todos los árboles sintácticos posibles debido a la ambigüedad del lenguaje.
2) La edición de las reglas sintácticas en las que, a pesar de no hacer una gestión para la optimización de las mismas, se ofrece la posibilidad de adquirir nuevas formas oracionales y figuras dentro del árbol.
3) La gestión de la completitud y el reconocimiento de los distintos roles por parte de los demás módulos analizadores de la aplicación.
4) La complementación del lexicón por parte del usuario para incluir los roles sintácticos asociados con una palabra.

VII. TRABAJOS FUTUROS

Existen algunos trabajos que podrían dar continuidad al analizador sintáctico que se ha presentado en este artículo, a saber:

1) La aplicación podría incluir un módulo de predicción para los casos en los que se presente una palabra sin roles sintácticos asociados y poder 'intuir' qué roles puede estar jugando la palabra y sugerir una posible corrección, basándose en las reglas sintácticas previamente definidas y analizando cual(es) reglas serían las aplicables en esa instancia.
2) Dentro de la aplicación no se hace gestión de las reglas ingresadas por el usuario; esto puede llegar al punto en que el usuario genere reglas redundantes, generando así árboles muy extensos. Una opción para este tipo de dificultades sería la implementación de un algoritmo de reducción del árbol, como se propone en [8].
3) El analizador sintáctico es el punto de partida de una aplicación que pretende la generación de diagramas UML tomando como punto de partida textos en lenguaje natural y a través del método de reglas editables, se puede independizar del idioma origen de los textos, graduar la complejidad y la estructura de los textos a analizar.

REFERENCIAS

[1] Mitkov R, The Oxford Handbook of Computational Linguistics. Oxford University Press, New York, 2003.
[2] A. Sanchez Manbrilla, Un generador de parsers para micros, Procesamiento del Lenguaje Natural, vol 3. 1985.
[3] J. M. Gómez, J. M. Goñi, J. C. González, Un analizador sintáctico para gramáticas asociativas por la izquierda. X Congreso SEPLN. Procesamiento del Lenguaje Natural vol 15. 1994.
[4] N. Chomsky, Aspects of the Theory of Syntax. MIT Press, Cambridge, Mass. 1965.
[5] J. Allen, Natural Language Understanding. The Benjamin/Cummings Publishing Company, Inc., Redwood City. 1995.
[6] R. Moore, J. Dowding, Efficient Bottom – Up Parsing, Speech and Natural Language: Proceedings of a Workshop Held at Pacific Grove, California. 1991.
[7] K. Martin, Experiments with a powerfull parser, Conference Internationale Sur Le Traitement Automatique Des Langues, vol 1, 1967, pp 1 - 20.
[8] Y. Sakakibara, Learning context-free grammars using tabular representations, Pattern Recognition, vol 38, 2004, pp 1372-1383.