UN–Lencep: Obtención Automática de Diagramas UML a partir de un Lenguaje Controlado

Carlos Mario Zapata J. Alexander Gelbukh Fernando Arango Isaza
Univ. Nacional de Colombia Instituto Politécnico Nacional Univ. Nacional de Colombia
cmzapata@unal.edu.co gelbukh@gelbukh.com farango@unal.edu.co

Resumen

La Elicitación de Requisitos de software es un proceso básico para garantizar la calidad del software y por lo general se realiza entre los Analistas y los Interesados en Lenguaje Natural, para obtener una especificación; dicha especificación suele estar conformada por un conjunto de diagramas (comúnmente de UML). El Procesamiento de Lenguaje Natural ha sido utilizado para la solución de problemas de Elicitación de Requisitos de software, pero aún utilizando lenguajes de corte técnico que los Interesados no dominan y por lo tanto no pueden validar adecuadamente. En este artículo se presenta UN–Lencep, una propuesta del uso de Lenguajes Controlados para especificación de Esquemas Preconceptuales, que se utilizan en la automatización del proceso de elaboración de diagramas UML. Adicionalmente, se muestra un ejemplo de la aplicación de UN-Lencep.

1. Introducción

La elaboración de una pieza de software usualmente comienza con un proceso en el cual los Interesados (personas con algún interés en el desarrollo de la pieza de software) y los Analistas se comunican en Lenguaje Natural. Sin embargo, esa comunicación se ve entorpecida por el hecho de que los Interesados conocen muy bien el dominio de su problema, pero desconocen lenguajes técnicos como el UML (Unified Modeling Language) [1], que comúnmente se emplean para la especificación de los requisitos; por el contrario, los Analistas conocen los lenguajes técnicos, pero pueden tener problemas para la interpretación del dominio del interesado.

Como respuesta a estas limitaciones, algunos investigadores en Procesamiento del Lenguaje Natural (PLN) han propuesto trabajos que ayudan a la Elicitación de Requisitos [2–8], en los cuales se realizan descripciones de las especificaciones del software que se va a realizar. Sin embargo, el Interesado por lo general desconoce cómo será el mundo del software que se va a construir, y mucho menos será capaz de describirlo de esa manera. Además, los lenguajes que se emplean en la descripción son lenguajes técnicos que en el fondo tratan de describir el sistema y poco se relacionan con el dominio del interesado.

Como una forma de solución a estos problemas, en este artículo se propone UN–Lencep, un Lenguaje Controlado que posibilita la elaboración de los denominados Esquemas Preconceptuales [9] y de allí la obtención automática de diagramas UML.

Este artículo está organizado así: en la Sección 2 se describe el proceso de Elicitación de Requisitos y su relación con el lenguaje UML y con los Esquemas Preconceptuales, en la Sección 3 se realiza un análisis crítico del uso de Lenguajes Controlados en Elicitación de Requisitos, en la Sección 4 se propone UN-Lencep como un lenguaje controlado que posibilita la obtención automática de diagramas UML y finalmente en la Sección 5 se presentan las Conclusiones y Trabajos Futuros.

2. La Elicitación de Requisitos y el lenguaje UML

Leite [10] acuñó el término “Elicitación de Requisitos”, cuando definió que los requisitos del Interesado debían ser recolectados, depurados y traducidos a un lenguaje de especificación que posibilitara su continuidad en el proceso de desarrollo del software. A través de los años, la Elicitación de Requisitos ha sido un proceso crucial para la elaboración de software de calidad: si los errores potenciales del software se pueden detectar en etapas tempranas de su elaboración, la corrección de los mismos puede ser mucho menos costosa y puede redundar en incrementos sustanciales en la calidad del

producto final. Sin embargo, la Elicitación de Requisitos es un proceso que se ve entorpecido por los problemas de comunicación que se generan entre los Analistas y los Interesados, quienes tienen énfasis diferentes dentro del proceso de desarrollo del software, y cuyas especialidades usualmente difieren de manera sustancial: los Analistas se concentran en los lenguajes técnicos de modelamiento, en tanto que los interesados tratan de describir su mundo en lenguajes cercanos al Lenguaje Natural.

Para la especificación de requisitos, desde mediados de los años noventa se ha venido imponiendo como un estándar de-facto el UML [1], un lenguaje gráfico que posee un conjunto de diagramas para la especificación de las principales características de una pieza de software. UML posee tres tipos de diagramas: Estructurales, de Interacción y Comportamentales. Los Estructurales incluyen los elementos de una especificación que no cambian con el tiempo; los de Interacción se enfocan en las interacciones entre los objetos del sistema; los Comportamentales se centran en el comportamiento de los objetos del sistema. Con los diagramas de los tres tipos se puede construir una especificación completa del software. Una descripción completa de los diagramas de UML se puede consultar en [11].

En [9] se propone el uso de Esquemas Preconceptuales como diagramas que compendian las características de tres de los diagramas característicos de UML, como son Clases, Comunicación y Máquina de Estados. Los Esquemas Preconceptuales utilizan una notación gráfica para la expresión de los diferentes elementos del discurso de un Interesado y constituyen un paso intermedio en la obtención automática de los diagramas de UML a partir de un Lenguaje Controlado. La sintaxis básica de los Esquemas Preconceptuales se muestra en la Figura 1.

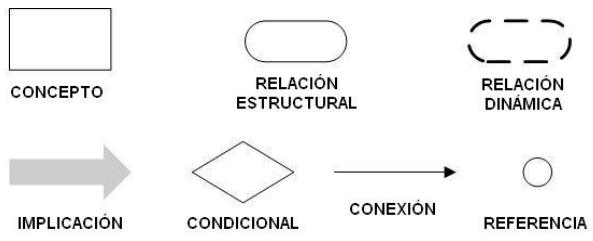

Figura 1. Especificación Básica de los Esquemas Preconceptuales

La explicación de los diferentes elementos de la Figura 1 es la siguiente:

- Un Concepto es un sustantivo o frase nominal del discurso del interesado. En los Esquemas Preconceptuales cada concepto se menciona sólo una vez, de forma que es posible identificar las diferentes relaciones de un concepto con los demás.
- Una Relación estructural se asocia con los verbos “ser” y “tener”. Es una relación permanente entre conceptos.
- Una Relación dinámica se asocia con los denominados “verbos de actividad”, que generan relaciones de tipo temporal entre los conceptos.
- La implicación es una relación causa-efecto entre relaciones dinámicas, en la cual la primera relación dinámica es el antecedente y la segunda es el consecuente.
- El condicional es también una relación de causalidad, pero en la cual la condición consta de un conjunto de conceptos ligados con operadores algebraicos (suma, resta, multiplicación, división) y operadores de comparación (igual, mayor o igual, menor o igual, diferente).
- La conexión es la forma en que se unen los conceptos con las relaciones, las relaciones con los conceptos y los condicionales con las relaciones dinámicas.
- Una referencia es un círculo numerado que sirve para ligar conceptos o relaciones que se hallan físicamente distantes en el Esquema Preconceptual.

3. Los Lenguajes Controlados en la Elicitación de Requisitos

En Procesamiento del Lenguaje Natural se han tenido los Lenguajes Controlados como un área de interés, por su utilización en diferentes aplicaciones, como la escritura y traducción de manuales técnicos, la representación del conocimiento, y la extracción de información, entre otras. Uno de los Lenguajes Controlados más conocidos pertenece a la Asociación Europea de Industrias Aeroespaciales (AECMA) [12], que se ha utilizado ampliamente para la elaboración de manuales técnicos para esa organización. En representación del conocimiento se han empleado lenguajes cercanos a la lógica de predicados, como por ejemplo el KIF (Knowledge Interchange Format) [13]. En extracción de información se han utilizado lenguajes como el CLIE (Controlled Language for Information Extraction) [14].

En Elicitación de Requisitos poco se han usado los Lenguajes Controlados. Algunos ejemplos que se pueden citar son los siguientes:

- Fuchs y Schwitter [2] propusieron Attempto Controlled English (ACE) para formular especificaciones de requisitos y luego traducirlas al lenguaje Prolog.
- Smith *et al.* [3] desarrollaron la herramienta para escribir y comprender las propiedades de un sistema y la denominaron "Propel Elucidation" (PROPEL). PROPEL usa plantillas para capturar detalles de los patrones de las propiedades del sistema. De manera similar, Konrad y Cheng [4] desarrollaron SPIDER (Specification Pattern Instantiation and Derivation Environment), una herramienta para la especificación de sistemas que emplea gramáticas de lenguaje natural estructurado y sistemas de patrones de especificación.
- Richards *et al.* [5] desarrollaron RECOCASE, un enfoque basado en el desarrollo de puntos de vista que usa un lenguaje controlado para capturar los requisitos de los interesados en forma de descripciones de casos de uso y luego los traduce a lattices. En esa misma línea de trabajo Diaz *et al.* [6] propusieron formatos generales para la escritura efectiva de descripciones de casos de uso que fueran inambiguas y con esas descripciones obtenían el diagrama de secuencias de UML.
- Cooper [7] definió una notación denominada SRRS (Stimulus Response Requirements Specification), que es un lenguaje natural estructurado legible por humanos y tratable computacionalmente.
- Mueckstein [8] propuso INTERPRET, un método para el diseño de interfaces gráficas de usuario que mezcla SQL y lenguajes naturales.

En todos los proyectos descritos, las especificaciones de requisitos del software se capturan en lenguajes técnicos (especificaciones de sistemas o descripciones de casos de uso) durante la fase de análisis del software. Esto genera dos tipos de problemas:

- Los interesados usualmente no comprenden ese tipo de lenguajes técnicos.
- Durante la fase de análisis, el Analista realiza interpretaciones del discurso del interesado que se involucran finalmente en las especificaciones y que no están incluidas en el discurso del Interesado.

Si se pudieran obtener automáticamente las especificaciones del software directamente a partir del discurso del Interesado, se reduciría el riesgo de introducción de información extraña por parte del Analista y se garantizaría que los diagramas UML resultantes serían reflejo del discurso del Interesado. Por otra parte, no se requeriría que los Interesados comprendieran lenguajes técnicos para validar los requisitos. Convendría apoyar este tipo de sistemas en lenguaje Español, ya que de los trabajos analizados únicamente el trabajo de Díaz *et al.* [6] incluye este lenguaje.

4. UN–Lencep: un Lenguaje Controlado para la obtención automática de diagramas UML

El Interesado suele expresar el dominio del problema durante las fases iniciales de la Elicitación de Requisitos; en ese discurso del Interesado se pueden encontrar algunos de los elementos fundamentales que se incorporarán posteriormente a la pieza de software que constituye la solución al problema que motiva su aparición. UN–Lencep (Universidad Nacional de Colombia—Lenguaje Controlado para la Especificación de Esquemas Preconceptuales) es un Lenguaje Controlado diseñado para que el Interesado pueda expresar las ideas del dominio del problema, con el fin de realizar su traducción automática hacia los denominados Esquemas Preconceptuales [9]. Con dichos esquemas, utilizando el enfoque descrito en [9], es posible realizar la transformación automática del discurso en tres de los diagramas de UML: Clases, Comunicación y Máquina de Estados, los cuales pertenecen respectivamente a los diagramas Estructurales, de Interacción y Comportamentales descritos en la Sección 2. En otras palabras, si el Interesado es capaz de describir el dominio de su problema en UN–Lencep, es posible elaborar automáticamente los tres diagramas mencionados de UML.

UN–Lencep es similar al lenguaje CLIE (Controlled Language for Information Extraction) [14], que se elaboró en el entorno GATE (General Architecture for Text Engineering) [15], desarrollado en la Universidad de Sheffield. Un ejemplo de CLIE es el siguiente (las palabras clave del lenguaje CLIE se han señalado con negrilla):

> **There are** projects. **There are** workpackages, tasks, and deliverables.
> SEKT **is a** project.

'MUSING', 'Knowledge Web', and Presto Space **are** projects.
Projects **have** workpackages.
Workpackages **can have** tasks.
WP1, WP2, WP3, WP4, WP5 and WP6 **are** workpackages.
SEKT **has** WP1.
'MUSING' **has** WP2, WP3, and WP4.
'Knowledge Web' **has** WP5 and WP6

UN–Lencep posee un conjunto básico de plantillas, similares a las que se emplean en CLIE, y que se amplían para permitir una mejor interacción con el interesado en un subconjunto del lenguaje natural, obteniéndose una versión de UN-Lencep avanzado. Algunas de las palabras de CLIE se conservan (como en el caso de "have", "has", "are" o "is" (en este caso en su equivalencias en Español), en tanto que se agregan nuevas plantillas para la representación de condicionales e implicaciones, que no eran consideradas en CLIE.

El conjunto básico de las plantillas de UN–Lencep y el resultado de la traducción correspondiente a Esquemas Preconceptuales se muestran en la Figura 2.

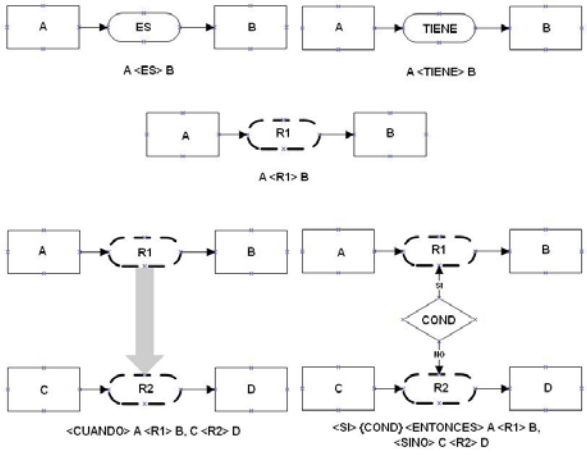

Figura 2. Especificación Básica de UN–Lencep y su traducción a Esquemas Preconceptuales

La ampliación de las plantillas hacia un subconjunto del lenguaje natural para obtener UN–Lencep avanzado, se muestra en la Tabla 1.

Para la Figura 2 y la Tabla 1, las convenciones son las siguientes:

- A, B, C y D son conceptos.
- <ES> y <TIENE> son relaciones estructurales, que tienen sus equivalencias como se muestra en la Tabla 1.
- <R1> y <R2> son relaciones dinámicas, como por ejemplo: registra, revisa, valida, chequea, califica, aprueba, presenta, llama, despacha, asigna, regresa, paga, realiza, elabora, prepara, entrega, recibe, solicita, etc. Estas relaciones son por lo general diferentes para los diferentes dominios, pero se suelen asociar con verbos de actividad.

Tabla 1. Equivalencias de la Especificación Básica de UN–Lencep en un subconjunto del Lenguaje Natural

Especificación Básica de UN–Lencep	Equivalencia en un subconjunto del Lenguaje Natural
A <ES> B	A <ES UN TIPO DE> B A <ES UNA CLASE DE> B A <ES UNA ESPECIE DE> B B <SE DIVIDE EN> A
A <TIENE> B	A <INCLUYE> B A <CONTIENE> B A <POSEE> B A <CONSTA DE> B A <ESTA CONFORMADO POR> B A <ESTA FORMADO POR> B A <SE COMPONE DE> B A <ESTA COMPUESTO POR> B B <PERTENECE A> A B <ES UNA PARTE DE> A B <ESTA INCLUIDO EN> A B <ESTA CONTENIDO EN> A B <ES UN ELEMENTO DE> A B <ES UN SUBCONJUNTO DE> A
<CUANDO> A <R1> B, C <R2> D	<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

- {COND} es una condición, que incluye un conjunto de conceptos relacionados mediante operadores de comparación (igual, mayor o igual que, menor o igual), además de operadores algebraicos (suma, resta, multiplicación, división).
- <CUANDO>, <SI>, <ENTONCES>, <DADO QUE>, <LUEGO DE QUE> son palabras reservadas para el uso de condicionales e implicaciones.

Adicionalmente, y para permitir una mayor legibilidad del UN-Lencep avanzado, se permite al Interesado el uso de artículos definidos e indefinidos, que se eliminan del UN–Lencep avanzado para obtener la versión del UN–Lencep básico.

Con la Tabla 1 y la Figura 2 es posible traducir una especificación en UN–Lencep en su correspondiente Esquema Preconceptual. A modo de ejemplo, se presenta la siguiente especificación, en la cual las

palabras pertenecientes a las plantillas se señalan en negrilla:

Un estudiante **es una clase de** persona
Un profesor **es un tipo de** persona
El profesor **posee** un curso
El estudiante **pertenece a** un curso
Luego de que el estudiante **presenta** un examen, el profesor **califica** el examen
Si nota es mayor que 3 entonces el estudiante aprueba el curso
La nota **pertenece a** un examen

Actualmente se ha desarrollado una herramienta que toma un discurso del Interesado como el que se ejemplifica y arroja la especificación básica del UN-Lencep, que es la siguiente para el ejemplo anterior:

Estudiante **es** persona
Profesor **es** persona
Profesor **tiene** curso
Curso **tiene** estudiante
Cuando estudiante **presenta** examen, profesor **califica** examen
Si nota>3 **entonces** estudiante **aprueba** curso
Examen **tiene** nota

El Esquema Preconceptual resultante de esta especificación se puede apreciar en la Figura 3. Adicionalmente, se cuenta con una herramienta que realiza la conversión a los diagramas de Clases, Comunicación y Máquina de Estados.

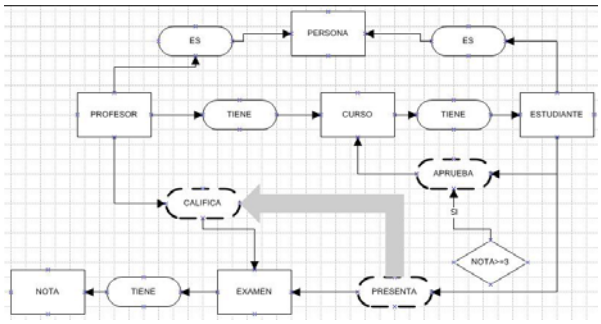

Figura 3. Esquema Preconceptual resultante de la Especificación en UN–Lencep del ejemplo

Con el esquema Preconceptual generado, se pueden elaborar los diagramas de Clases, Comunicación y Máquina de Estados correspondientes. Para ello, se emplea actualmente la herramienta UNC-Diagramador, que se basa en el proceso de conversión que se detalla en [9].

5. Conclusiones y Trabajos Futuros

En este artículo se presenta UN–Lencep, como una manera de emplear los Lenguajes Controlados del PLN en la labor de Elicitación de Requisitos. En UN–Lencep se puede describir el discurso de un Interesado que desea desarrollar una pieza de software, pero sin que necesariamente se refiera al software mismo; este último ha sido el caso de los diferentes enfoques que se han utilizado para esta labor (lenguajes de Especificaciones Técnicas y Descripciones de Casos de Uso). A partir del discurso del interesado en UN–Lencep avanzado se puede obtener una especificación de UN–Lencep básico, que corresponde de manera inambigua a un Esquema Preconceptual único, con el cual se pueden generar tres diagramas UML: Clases, Comunicación y Máquina de Estados. De esta manera, a partir de una especificación UN–Lencep elaborada por el interesado, es posible obtener, de manera automática, los tres diagramas UML mencionados. Dichos diagramas serán completamente consistentes entre sí por partir del mismo discurso del Interesado y estarán ya validados por éste, puesto que la especificación será provista por el Interesado mismo.

Actualmente existe una herramienta para la obtención del UN–Lencep básico a partir de la especificación del UN–Lencep avanzado y se está elaborando una herramienta para la obtención de los Esquemas Preconceptuales en el formato requerido por UNC–Diagramador. Es de notar que esta última herramienta ya realiza de manera automática los tres diagramas mencionados.

Como Trabajos Futuros, se plantean los siguientes:

- Complementación de las plantillas de UN–Lencep avanzado con el fin de acercarlas cada vez más a Lenguaje Natural. Esto contempla dos tipos de trabajos:
  - Análisis sintáctico y semántico de los artículos, que en este momento se permiten en UN-Lencep avanzado para facilitar la legibilidad de la especificación, pero que se eliminan para la especificación de UN–Lencep básico. Con este análisis se pretende determinar la posible utilidad de los artículos definidos e indefinidos y su representación en UN–Lencep básico.
  - Análisis sintáctico y semántico de otros elementos del discurso que aún no se incluyen en UN–Lencep ni en los Esquemas Preconceptuales, tales como adverbios y adjetivos.

- Conducción de experimentos con Interesados reales y simulados que permitan determinar las posibilidades de utilización de UN–Lencep en la especificación de diferentes piezas de software.

Referencias

[1] OMG, UML 2.0 Specification, www.omg.org/UML

[2] N. E. Fuchs y R. Schwitter, "Attempto Controlled English (ACE)", Proceedings of the First International Workshop on Controlled Language Applications, Leuven 1996.

[3] R. L. Smith, G. S. Avrunin, y L. Clarke, "From natural language requirements to rigorous property specifications", Proceedings of the Workshop on Software Engineering for Embedded Systems (SEES 2003): From Requirements to Implementation, Chicago, 2003, pp. 40–46.

[4] S. Konrad, y B. H. C. Cheng, "Facilitating the Construction of Specification Pattern-based Properties", Proceedings of the IEEE International Requirements Engineering Conference (RE05), Paris, 2005.

[5] D. Richards, K. Boettger, y O. Aguilera, "A Controlled Language to Assist Conversion of Use Case Descriptions into Concept Lattices", Proceedings of AI 2002, Canberra, LNCS 2557, 2002.

[6] I. Díaz, F. Losavio, A. Matteo y O. Pastor, "A Specification Pattern for Use Cases", Information & Management, Vol. 41, No. 8, 2004, pp. 961–975.

[7] K. Cooper, "Stimulus Response Requirements Specification Notation: An Empirically Evaluated Requirements Specification Notation", Ph.D. Thesis, University of British Columbia, 2001.

[8] E. M. Mueckstein, "Controlled Natural Language Interfaces: The Best of Three Worlds", Terry M. Walker, Wayne D. Dominick (Eds.), Proceedings of the 13th ACM Annual Conference on Computer Science, New Orleans, 1985, pp. 176–178.

[9] C. M. Zapata, A. Gelbukh, y F. Arango, "Pre-conceptual Schema: a UML Isomorphism for Automatically Obtaining UML Conceptual Schemas", Research in Computing Science: Advances in Computer Science and Engineering, Vol. 19, 2006, pp. 3–13.

[10] J. C. Leite, "A survey on requirements analysis", Advanced Software Engineering Project Technical Report RTP-071, Department of Information and Computer Science, University of California at Irvine, 1987.

[11] Fowler, M., *UML Distilled*, Addison-Wesley, Boston, 2003.

[12] AECMA, "AECMA Simplified English: A Guide for the Preparation of Aircraft Maintenance Documentation in the International Aerosace Maintenance Language", AECMA, Brussels, 1995.

[13] "Knowledge Interchange Format. Draft proposed American National Standard (dpANS) NCITS.T2/98-004". logic.stanford.edu/kif/dpans.html

[14] T. Polajnar, H. Cunningham, V. Tablan, y K. Bontcheva, "Controlled Language IE Components Version 1", EU–IST Integrated Project (IP) IST–2003–506826 SEKT, D2.2.1 Report, Sheffield, 2006.

[15] H. Cunningham, D. Maynard, K. Bontcheva, y V. Tablan, "GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications", Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02), Philadelphia, 2002.