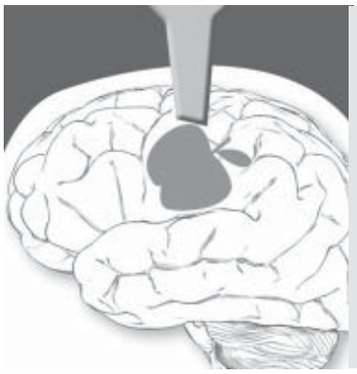
REVISTA Universidad EAFIT
Vol. 41. No. 137. 2005. pp. 77-95

Los modelos verbales
en lenguaje natural y su utilización en la elaboración de esquemas conceptuales para el desarrollo de software: una revisión crítica



Carlos Mario Zapata Jaramillo
Magíster en ingeniería de Sistemas. Profesor de la Escuela de Sistemas de la Facultad de Minas de la Universidad Nacional de Medellín. Integrante del Grupo de Investigación UN-INFO de la misma institución.
cmzapata@unalmed.edu.co

Fernando Arango Isaza
Profesor de la Escuela de Sistemas de la Facultad de Minas de la Universidad Nacional de Medellín. Integrante del Grupo de Investigación UN-INFO de la misma institución.
farango@perseus.unalmed.edu.co

Recepción: 15 de julio de 2004 l Aprobación: 22 de noviembre de 2004

Resumen
El desarrollo de software inicia con una serie de entrevistas realizadas a los usuarios potenciales con el fin de determinar los requisitos del software; como resultado de las entrevistas se obtienen modelos verbales en lenguaje natural. A partir de los modelos verbales es posible construir esquemas conceptuales, que son diagramas que permiten representar gráficamente los datos y funciones asociados con el problema para realizar el desarrollo del software. En este artículo se compendian los trabajos que en esta materia se han adelantado a nivel mundial, realizando un análisis de los posibles tópicos de investigación a partir de los problemas no resueltos.

Palabras Clave
Procesamiento del lenguaje natural
Ingeniería de requisitos
Lenguaje unificado de modelamiento
Elicitación de requisitos de software

Natural language verbal models and their use in the design of conceptual frameworks for software development: a critical review

Abstract
Software development begins with a series of interviews to potential users with the purpose of determining the software requirements; as a result of the interviews yield verbal models in natural language. Based on the verbal models, conceptual frameworks can be designed. These are diagrams that allow graphic data and functions related to the problem to develop software. This article covers worldwide work carried out in this field, with an analysis of the possible research topics based on the unsolved problems.

Key words
Natural language processing
Requirements engineering
Unified modeling language
Software requirements elicitation

Introducción

Los usuarios potenciales de una pieza de software suministran un origen al desarrollo de las mismas describiendo informalmente las características del área de aplicación mediante lenguaje natural. Tal como se ilustra en la Figura 1, a partir de esa descripción informal, que se suele compendiar en una historia del usuario denominada Modelo Verbal, los analistas deben elaborar los modelos conceptuales para representar de forma precisa las características del área que son relevantes al software.

A pesar de que, tanto usuarios como analistas, tratan de describir adecuadamente el área de aplicación, subsiste sin embargo una brecha en la comunicación, que le impide al usuario tener la seguridad de que el modelo elaborado por el analista representa su propia concepción. Presuman (2002) resume el problema generado al inicio del análisis del dominio de la siguiente manera:

El análisis y la especificación de los requisitos puede parecer una tarea relativamente sencilla, pero las apariencias engañan. El contenido de comunicación es muy denso. Abundan las ocasiones para las malas interpretaciones o falta de información. Es muy probable que haya ambigüedad. El dilema al que se enfrenta el ingeniero del software puede entenderse muy bien repitiendo la famosa frase de un cliente anónimo: “Sé que cree que entendió lo que piensa que dije, pero no estoy seguro de que se dé cuenta de que lo que escuchó no es lo que quise decir”.

Esta brecha de comunicación es consecuencia de las diferencias en las especialidades del usuario y el analista. Así, los usuarios son expertos en el dominio del problema, pero por lo general conocen poco de las diferentes técnicas de modelamiento, y los analistas, por su parte, son expertos en el lenguaje de modelamiento pero en términos generales poco conocen del problema. En la Figura 2 se representa la incapacidad manifiesta entre el usuario y el analista para entender el lenguaje del otro.

Esta brecha de comunicación impide que en la fase de requisitos se lleve a cabo, tanto la recolección correcta y completa de los aspectos relevantes del área de aplicación por parte del analista, como la detección de los problemas de dicha recolección por parte del usuario. En consecuencia, muchos defectos y carencias del software no serán detectados y corregidos hasta las fases de implantación y uso, con graves consecuencias sobre los costos y tiempos de entrega de la aplicación.

A la solución de este problema contribuyen técnicas y herramientas orientadas a apoyar los diversos aspectos de la comunicación analista - programador. Entre estos aspectos vale la pena destacar los siguientes:

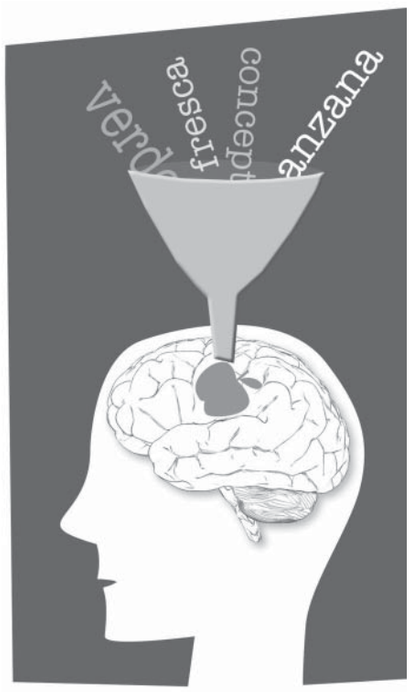
Figura 1. Formas de expresión de la realidad para usuarios y analistas

Figura 2. Incapacidad de entender el lenguaje del otro

• A la capacidad del analista para comprender el modelo verbal presentado por el usuario, contribuyen los diversos avances en la lectura e interpretación de textos en lenguaje natural mediante herramientas computarizadas. Con la interpretación automática es posible derivar la estructura y el significado más probable de las partes del modelo verbal, eliminando la influencia personal del analista. Existen varios ejemplos de esta tendencia (Baker, Fillmore y Lowe, 1998; Fillmore y Baker, 2001; Fellbaum, 1998; Meyer y Dale, 2002; Kamps et al., 2004 y Galicia, 2000).

• A la capacidad del analista de convertir el modelo verbal interpretado en un esquema conceptual apropiado, contribuyen diversos enfoques:

- Heurísticas a ser usadas a discreción del analista (Chen, 1983; Coad y Yourdon, 1990 y Nijssen, 1977).



- Herramientas de apoyo al analista para la aplicación de las heurísticas (Overmyer et al., 2001).

- Herramientas de apoyo que llevan a cabo tanto la interpretación automática del lenguaje natural como la aplicación automática de las heurísticas, para obtener versiones (preliminares) del esquema conceptual (Buchholz y Düsterhöft, 1994; Cyre, 1995; Mich, 1996; Jin, Bell y Wilkie, 1998; Harmain y Gaizauskas, 2000; Mich y Garigliano, 2002; Fliedl et al., 1999, 1999a, 2002, 2002a y 2003; Kop y Mayr, 2002; Mayr y Kop, 2002).

- Herramientas que apoyan la completitud del modelo verbal (Buchholz y Düsterhöft, 1994).

- Herramientas para apoyar la validación, por parte del usuario, del esquema conceptual elaborado por el analista (Fliedl et al., 1999, 2002, y 2003; Kop y Mayr, 2002; Mayr y Kop, 2002).

Para solucionar este problema se han desarrollado diferentes enfoques, los cuales pretenden realizar algunas de las siguientes actividades:

• Apoyar a los analistas en la construcción de los diferentes esquemas conceptuales, sugiriéndoles algunas ideas para elaborar los esquemas de forma semiautomática, pero dejando la decisión final de la construcción de los esquemas en manos del analista.

• Analizar los textos en lenguaje natural de manera que sean interpretables para las máquinas, pero sin establecer como utilidad específica la determinación de los elementos fundamentales de los esquemas conceptuales.

• Realizar la construcción de los esquemas conceptuales de manera automática, tomando como partida un modelo restringido en lenguaje natural, de las especificaciones del dominio de la pieza de software a construir.

En cualquiera de los casos anteriores, se presentan dificultades que impiden la obtención automática de modelos conceptuales a partir de modelos verbales.

En este artículo se realiza una revisión crítica del uso de los modelos verbales en la construcción de esquemas conceptuales, empleando para ello la siguiente estructura: en la Sección 2 se presentan los principales trabajos en el tópico en estudio; en la Sección 3 se analizan estos trabajos y se determinan los principales problemas que quedan aún por resolver; finalmente, en la Sección 4 se presentan las conclusiones y trabajos futuros que se pueden derivar de esta revisión.

1. Utilización de modelos verbales en la elaboración de esquemas conceptuales

Coad y Yourdon propusieron un método para el análisis del modelo verbal, con el fin de determinar los principales componentes del modelo de clases, principal esquema conceptual de la metodología de análisis orientado por objetos. De manera muy simple, su metodología de establece una identificación de los principales sustantivos presentes en el modelo, los cuales suministran algunas pistas en relación con las "clases" y "objetos" del modelo, y los principales verbos, los cuales podrían asimilarse con los conectores entre las diferentes entidades, es decir "asociaciones" del modelo (Coad y Yourdon, 1990).

En los albores del surgimiento del modelo entidad relación (Chen, 1976), se propuso una metodología con alguna ingerencia del Lenguaje Natural que parte de la determinación de algunas estructuras de los denominados "Modelos de relaciones binarias", previos al modelo entidad relación, utilizando para ello NIAM –Nijssen Information Analysis Methodology– propuesta por Nijssen (Nijssen, 1977). Esta metodología se llevó a nivel de implementación en diferentes lenguajes durante las décadas de 1970, 1980 y 1990 (ver, por ejemplo, Halpin, 1998), pero nunca de manera intensiva sobre analizadores sintácticos del lenguaje natural, sino como apoyo a la labor de los analistas en ciertos aspectos de la identificación de los elementos clave de los esquemas conceptuales.

Chen estableció de manera formal once reglas que permiten la realización de la traducción de las especificaciones, expresadas en lenguaje natural, al diagrama entidad–relación, identificando sustantivos comunes, verbos transitivos, adjetivos, adverbios, objetos de operaciones algebraicas y numéricas, gerundios, cláusulas y oraciones, además de algunas conversiones de frases en otras de más fácil análisis, como por ejemplo pasar de: "hay 200 empleados en ese departamento" a "el departamento tiene 200 empleados". Estas reglas las consideraba sugerencias de tipo heurístico para su utilización en la construcción del diagrama entidad–relación, no eran imperativos a seguir que pudiesen establecer de manera única el diagrama a partir de las especificaciones en lenguaje natural (Chen, 1983).

Kristen propuso el método denominado KISS (Kristen, 1994), que realiza en oraciones en lenguaje natural la identificación de los elementos que hacen parte de un esquema conceptual, particularmente objetos y acciones, y con ellos construye un esquema conceptual propio basado en grafos conceptuales. Este método en sí mismo no ha sido implementado, pero ha servido como base para otras implementaciones, tales como COLOR-X (Burg y Van de Riet, 1996) y Grammalizer (Hoppenbrouwers, 1997), los cuales, sin embargo, tienen objetivos diferentes al trazado automático de esquemas conceptuales.

El primer conjunto de actividades descrito en la sección anterior toma como punto de partida reglas de tipo heurístico, como las de Chen. Para su uso se desarrollaron algunos productos que buscaban apoyar la gestión de los analistas a partir de los modelos verbales, pero en las cuales el apoyo se limitaba principalmente a la clasificación de las palabras por su función gramatical, con la posibilidad de que el analista realizara la clasificación en diferentes categorías que le permitiesen elaborar los diagramas conceptuales (clases, atributos, métodos, relaciones,

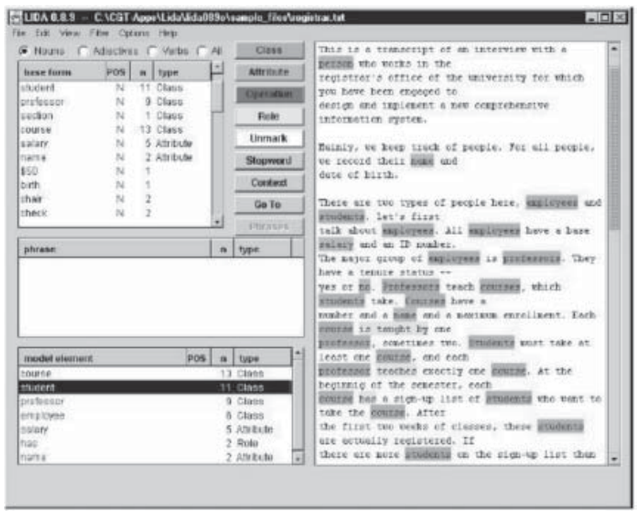
etc.). Esta motivación, sin embargo, deja en manos del analista la construcción de los diagramas y contribuye poco a la solución de la brecha que se plantea en la introducción, puesto que, en esencia, no contribuye a realizar una mejor comunicación entre el usuario y el analista; el rol del usuario es suministrar la información del dominio de la manera más completa posible en lenguaje natural, la cual es interpretada por el analista con la ayuda de diferentes herramientas (que pueden utilizar o no el modelo verbal, como se verá a continuación), pero sin establecer un lenguaje común con el usuario.

En este tipo de aplicaciones se destaca el proyecto LIDA –LInguistic assistant for Domain Analysis– (Overmyer, 2001), el cual utiliza un entorno gráfico que le permite al analista señalar con colores los diferentes elementos del modelo verbal, dependiendo de la elección de conversión hacia elementos tales como clases, atributos, operaciones o roles, suministrándole estadísticas para facilitar su elección, como el número de ocurrencias de una determinada palabra y el tipo de cada palabra. En la Figura 3 se muestra una de las interfaces del proyecto LIDA. Este producto permite al analista marcar con diferentes colores las clases, atributos, operaciones y roles, además de realizar el trazado inicial de los diagramas y exportarlos para complementarlos en otras herramientas.

Con menos participación del modelo verbal, pero con más capacidades gráficas, se pueden ubicar en esta categoría las diferentes herramientas CASE, disponibles en el mercado, de las cuales Rational Rose®, ArgoUML® (en la Figura 4 se muestra una interfaz de esta herramienta) y Designer de Oracle® son sólo algunos ejemplos. Estas herramientas no toman como punto de partida el modelo verbal de manera explícita, pues el analista debe leer el modelo verbal y traducirlo, según su criterio, en las estructuras correspondientes a los diferentes esquemas conceptuales disponibles, por lo general modelos de UML (OMG, 2004).

El segundo conjunto de actividades contiene innumerables ejemplos en la literatura especializada, pues se puede enmarcar dentro del área denominada "Procesamiento del Lenguaje Natural", la cual ha sido ampliamente estudiada por diferentes investigadores a nivel mundial. Esta es una categoría que tampoco contribuye sustancialmente al establecimiento de lazos de comunicación entre usuarios y analistas, ni a la definición de un lenguaje común. Sin embargo, los esfuerzos en el procesamiento del lenguaje natural posibilitan el entendimiento, por parte de las herramientas computacionales, de los diferentes significados asociados con las frases presentes en diferentes tipos de textos, y potencialmente podrían usarse para lograr

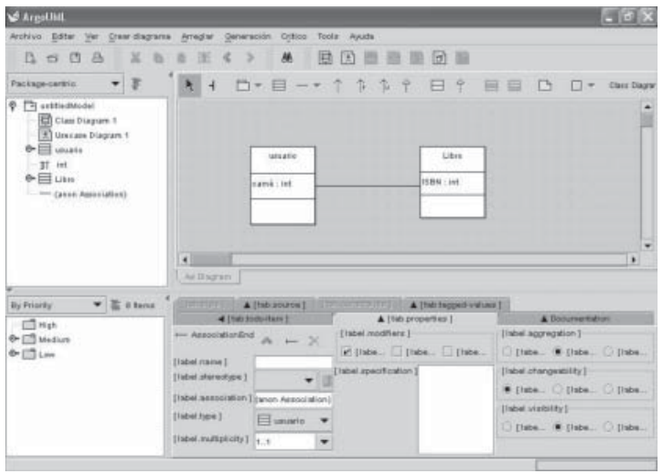
Figura 3. Imagen de una de las interfaces del proyecto LIDA.



Fuente: (Overmyer, 2001)

que diferentes herramientas computacionales dieran interpretaciones automáticas a los modelos verbales del usuario, suministrando a los analistas pistas en relación con el dominio de los diferentes modelos que se pueden trazar.

Figura 4. Imagen de la interfaz gráfica de ArgoUML®.



Fuente: A partir del uso de la herramienta por parte de los autores (2004).

Dentro de esta categoría se pueden ubicar trabajos como FrameNet de la Universidad de Berkeley (Baker, Fillmore y Lowe, 1998; Fillmore y Baker, 2001), en el cual se desarrolla un análisis sintáctico y semántico de las posibles frases en inglés con miras a una interpretación de las mismas, empleando la denominada "Gramática de Casos" (Fillmore, 1968). Dentro de esta categoría también se puede ubicar el proyecto WordNet (Fellbaum, 1998), que es un léxico de tipo semántico construido para el análisis genérico de diferentes frases en inglés y que, a su vez, ha motivado la aparición de diferentes trabajos en desambiguación (solución a las ambigüedades presentes en el lenguaje natural, Meyer y Dale, 2002), sinonimia y otros tópicos relacionados (Kamps, 2004).

En esta orientación se encuentran trabajos para el idioma inglés y el alemán, especialmente, que intentan recopilar léxicos que posibilitan la interpretación posterior de las diferentes frases que hacen parte de un determinado texto para analizar. Para el español existe una gran cantidad de trabajos en temas como la desambiguación (ver, por ejemplo, Galicia, 2000 o las publicaciones de la Revista de la Sociedad Española para el Procesamiento del Lenguaje Natural SEPLN, disponibles en formato electrónico en www.sepln.org), pero poco se ha trabajado en proyectos como WordNet y FrameNet.

El tercer conjunto de actividades también ha sido ampliamente trabajado para diferentes idiomas, como el inglés, el alemán y el italiano. La mayoría de los ejemplos analizados de este tipo, para la conversión automática del texto en lenguaje natural en modelos conceptuales, utilizan algunos de los elementos de la estructura siguiente:

- Un intérprete sintáctico que permite asignar una categoría lingüística a cada frase y trazar los árboles sintácticos respectivos, en los que se suele señalar la importancia de los verbos dentro de las diferentes oraciones.

- Un analizador semántico en el que, mediante gramáticas de casos o determinación de las valencias de los verbos, se examinan las partículas que normalmente acompañan un verbo, para categorizarlas y determinar los diferentes roles que pueden desempeñar en las frases.

- Léxicos propios del dominio de la aplicación, con el fin de limitar el lenguaje disponible para el

análisis y para realizar ciertos chequeos de consistencia de la información.

- Esquemas preconceptuales de diferente índole, que permiten realizar un paso intermedio entre la especificación en lenguaje natural y el esquema conceptual definido.

- Un conjunto de reglas heurísticas, que permite determinar a cuál de los elementos pertenecientes a un esquema conceptual se puede traducir un elemento dentro del esquema preconceptual definido o, en su defecto, de la especificación en lenguaje natural. Esas reglas tienen por lo general formas similares a las expresadas por Chen.

Galatescou presenta un conjunto de pasos similar al descrito, aunque incluye el cálculo de predicados para la definición y descripción de objetos, acciones y relaciones entre ellos. En este trabajo se identifican las principales habilidades de integración del lenguaje natural y se hace una caracterización del potencial descriptivo del lenguaje natural, finalizando con la descripción en cálculo relacional, que puede luego ser traducida a cualquier esquema conceptual (Galatescou, 2002).

En esta categoría se pueden encontrar algunos ejemplos, que buscan la solución de la brecha que se genera entre los usuarios y los analistas, especialmente mediante productos que chequean la completitud de los modelos generados mediante herramientas automáticas construidas y le devuelven al usuario preguntas que buscan los complementos de información necesarios para culminar el trazado de los modelos; otros productos, en cambio, presentan al usuario pantallas con alguna información correspondiente a los modelos buscando su aprobación, pero sin tomar en consideración que los usuarios no poseen el suficiente conocimiento en el manejo de los diferentes tipos de modelos que se pueden generar para un problema específico.

A continuación se presentan algunos de los ejemplos de esta categoría de herramientas y métodos:

Buchholz y Düsterhöft presentan una metodología para la extracción del diagrama entidad - relación a partir de especificaciones en lenguaje natural. Este proyecto, denominado RADD (Rapid Application and Database Development), emplea lo que ellos denominan una “Herramienta de diálogo moderado” que posibilita la comunicación con el diseñador de la base de datos en lenguaje natural (Buchholz y Düsterhöft, 1994 y Buchholz, 1995).

En RADD, el proceso se inicia con un texto en lenguaje natural, en alemán, que se ingresa a un analizador sintáctico, el cual identifica los diferentes elementos gramaticales presentes en el texto y elabora los árboles sintácticos correspondientes. Posteriormente, el RADD utiliza un modelo lingüístico que inserta un nivel semántico entre los niveles sintáctico y conceptual, identificando el significado de una frase mediante los roles semánticos asociados con los verbos; estos roles se usan para verificar la completitud lingüística de la frase, pues cada verbo puede estar acompañado por un número determinado de roles semánticos (causa, tema, resultado/meta, fuente, localización, tiempo, modo, voz/aspecto) que pueden desempeñar cada una de las palabras que acompañan al verbo. Finalmente, el RADD utiliza una serie de reglas heurísticas para realizar la conversión del resultado del análisis sintáctico–semántico en los diferentes elementos del modelo entidad–relación.

En RADD, la herramienta de diálogo se basa en un programa en PROLOG, que activa una serie de preguntas dependiendo del nivel de análisis que se haya alcanzado con el texto ingresado. La activación de las preguntas al diseñador se produce con un análisis sintáctico incompleto o un modelo de diseño incompleto, realizando preguntas de contenido (“¿Existen más detalles sobre la aplicación?”), clarificación lingüística (“¿Cómo se realiza el acto-prestar-?”) y clarificación pragmática (“¿Cómo se caracterizan los –libros-?”). Con base en las respuestas suministradas también en lenguaje natural por parte del diseñador, se completa el modelo. Los resultados se presentan en un modelo textual con la siguiente nomenclatura (Buchholz, 1995):

- Entity (EName): describe una entidad con el nombre Ename.

- Relship (RName, EName1, [EName2]): describe una relación RName entre la entidad EName y la lista de entidades (EName2 describe un conjunto de entidades).

- Attre (EName, AName): la entidad EName tiene un atributo AName.
- Attrr (RName, AName): la relación RName tiene un atributo AName.
- Keycand (EName/RName, AName): el atributo AName es una clave candidata de la entidad EName o la relación RName.
- Cardcand (NR, RName, EName, MinCard, Max Card): la relación RName tiene unas cardinalidades MinCard: MaxCard correspondientes a la entidad EName en el extremo NR.
- Inclcand (EName1, EName2): describe una dependencia de inclusiones de dos entidades (EName1 y EName2), donde un EName1 incluye un EName2.
- Exclcand ([EName]): describe una lista de entidades excluidas EName.

De manera similar al trabajo anterior, el proyecto CIRCE utiliza una herramienta llamada CICO (Gervasi, 2001), la cual utiliza un enfoque de lógica difusa para realizar el análisis sintáctico de las frases; además, dentro del proyecto CIRCE se puede realizar un análisis de la completitud, consistencia y corrección de los textos en lenguaje natural, realizando, entre otras, las siguientes funciones: elicitación de requisitos (determinación de los principales conceptos del dominio), selección de requisitos (para reducir la complejidad), identificación de conflictos entre requisitos y para forzar un buen estilo en los requisitos (Gervasi, 1999; Zowghi y Gervasi, 2002; Gervasi y Nuseibeh, 2002; Zowghi et. al 2001; Ambriola y Gervasi, 1997). Ambriola y Gervasi muestran la forma como el CICO parser puede llegar a generar también diferentes tipos de diagramas a partir de especificaciones en lenguaje natural (Ambriola y Gervasi, 2001).

Harmain y Gaizauskas introducen el CM-Builder, una herramienta CASE que permite la elaboración de diagramas de clases a partir de textos de requisitos en inglés, utilizando como modelo intermedio para su obtención una red semántica. Si bien manifiestan no imponer restricciones de ningún tipo al lenguaje, no tienen una respuesta clara para el manejo de las ambigüedades que pueden existir en el modelo verbal; además, la información correspondiente a los métodos del diagrama de clases no se emplea para incluirlos en la que ellos llaman primera versión del diagrama, que deberá ser depurada y complementada en otras herramientas (Harmain y Gaizauskas, 2000).

Cyre presenta un trabajo aplicable en el dominio particular del desarrollo de Sistemas Digitales, en el cual combina los diferentes diagramas que se pueden trazar para este desarrollo en particular (tales como Diagramas de Bloques, Diagramas de Flujo de Datos, Estructuras de Datos, Diagramas de Flujo, Diagramas de Estado y Diagramas de Tiempo), con especificaciones expresadas en lenguaje natural. El prototipo de su sistema, denominado ASPIN (Automatic Specification Interpreter), toma como entradas los modelos definidos y la especificación en lenguaje natural y los convierte en grafos conceptuales (que son especies de redes semánticas), que emplean un lenguaje de representación común (independiente del modelo que le sirve de origen), con el fin de integrarlos y representarlos de manera gráfica para que el usuario pueda verificar la ausencia de inconsistencias y omisiones. Las especificaciones en lenguaje natural emplean una forma de inglés restringido, que proviene de la ontología derivada de una colección de notaciones de modelos y ejemplos de oraciones en lenguaje natural, usadas en el dominio de los Sistemas Digitales. Un ejemplo particular de la aplicación del lenguaje definido para los grafos conceptuales en la oración “Un programa es ejecutado por el procesador”, tiene la forma siguiente (Cyre, 1995):

```
[execute]-
        (agnt : by) -> [processor : #]
        (opnd) -> [program : *]
```

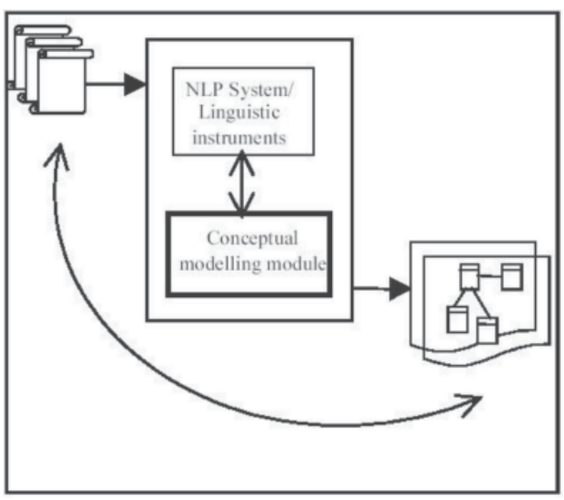
Para lograr este resultado, ASPIN realiza un análisis sintáctico, en el que básicamente se determinan los verbos y los sustantivos de las especificaciones, tomando como restricción aquellos pertenecientes al dominio del desarrollo de Sistemas Digitales, y luego un análisis semántico que utiliza la teoría de los roles semánticos (Fillmore, 1968). Para los demás modelos, ASPIN define una serie de equivalencias a los grafos conceptuales, dependiendo de la forma de los diagramas.

Jin, Bell y Wilkie presentan un trabajo similar al de Cyre, empleando grafos conceptuales para representar una ontología del dominio, pero ampliando el dominio de aplicación, el cual no está restringido a los Sistemas Digitales, como en el caso de Cyre, sino que es de aplicación general (Jin, Bell y Wilkie, 1998). Al igual que Cyre, utilizan los grafos conceptuales en forma de lenguaje y no de manera gráfica; además, parten únicamente de especificaciones expresadas en lenguaje natural y no de otros tipos de diagramas.

NL-OOPS (Natural Language Object – Oriented Product System), es un sistema propuesto por Mich y que se basa en un sistema de procesamiento del lenguaje natural denominado LOLITA (Large-scale Object-based Language Interactor, Translator and Analyser), el cual contiene una serie de funciones para el análisis del lenguaje natural, con el fin de detectar incluso ambigüedades en el texto que sirve de entrada para el análisis (Mich, 1996). NL-OOPS entrega como resultado un conjunto de las clases candidatas y las posibles instancias, atributos y métodos de las mismas. Utiliza para la entrega de los resultados una estructura en forma de árbol, pero no se establecen en él las diferentes relaciones que pueden estar presentes en el diagrama (agregaciones, generalizaciones, dependencias, etc.). El resultado de este análisis sirve como punto de partida para la complementación del diagrama de clases (Mich y Garigliano, 2002). En la Figura 5 se puede apreciar un esquema del propósito de NL-OOPS.

Los documentos en lenguaje natural son examinados mediante un sistema de procesamiento del lenguaje natural que emplea instrumentos lingüísticos y que interactúa con un módulo de modelamiento conceptual que finalmente traza los diagramas requeridos.

Figura 5. El propósito de NL-OOPS.



Fuente: (Mich, 1996)

El KCPM (Klagenfurt Conceptual Predesing Model), fue elaborado en la Universidad de Klagenfurt, en Austria, por un grupo de lingüistas (Fliedl et al., 1999, 1999a, 2002, 2002a y 2003) y un grupo de ingenieros de sistemas (Kop y Mayr, 2002; Mayr y Kop, 2002), como parte del proyecto NIBA, que busca la comprensión de textos en lenguaje natural en alemán restringido para convertirlos en esquemas conceptuales que apoyen la producción automática de software. En la Figura 6 se muestra el conjunto de las herramientas con que cuenta el proyecto, que son:

- Un analizador sintáctico, basado en NT(M)S (Morfosintaxis teórica del lenguaje natural), una teoría lingüística diseñada y complementada por el grupo lingüístico de la Universidad de Klagenfurt, que posibilita la realización de los árboles sintácticos.

- Un intérprete semántico que, basado en la teoría de los Roles Theta (Gruber, 1965), una forma previa de la semántica de casos propuesta luego por Fillmore, determina las funciones de las distintas palabras que acompañan a un verbo.

- Una herramienta de KCPM, la cual, mediante un conjunto de reglas de tipo heurístico, puede realizar la traducción de las oraciones, analizadas de manera sintáctica y semántica, en los denominados esquemas preconceptuales, definidos como esquemas gráficos o tabulares que posibilitan la construcción de los diferentes esquemas conceptuales, disponibles para un determinado problema de desarrollo de software.

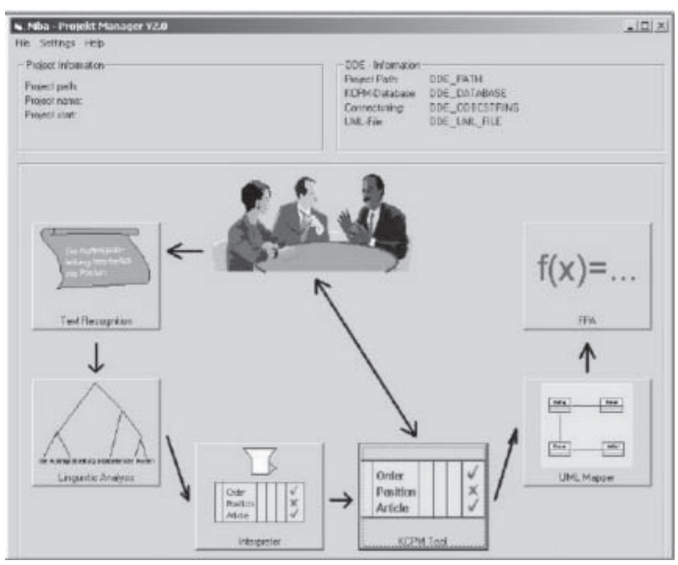
- Una herramienta para el cálculo de los puntos de función, actualmente en desarrollo.

El analizador de NT(M)S emplea un diccionario lingüístico que identifica los valores gramaticales de las palabras, con el fin de facilitar la construcción de los árboles semánticos. Como resultado del analizador, se obtienen los árboles semánticos expresados de manera gráfica para comprensión de los usuarios de la herramienta NIBA, y de manera textual mediante paréntesis balanceados, con el fin de lograr la interpretación automática por parte de la máquina.

En el intérprete semántico se encuentran clasificados 10.000 verbos del alemán, tomando en consideración 19 tipos de verbos definidos en NT(M)S y categorizados según las palabras que pueden acompañarlos en un caso dado, con los diferentes roles que pueden desempeñar (agente, experimentador, tema, meta, fuente, locación). Esos roles suministran pistas en relación con los tipos de elementos con los cuales se pueden asociar en el posterior desarrollo de los esquemas conceptuales.

La herramienta KCPM es diferente según el tipo de esquema conceptual a construir. Cuando se trata de esquemas de tipo estructural (diagrama de clases de UML), se emplean tablas de conversión que categorizan las palabras de una frase en “tipo cosa” (elementos que posteriormente pueden ser clases o atributos) y “tipo conexión” (los cuales posiblemente serán relaciones o incluso atributos). Cuando se trata de esquemas de comportamiento, se emplean diagramas preconceptuales (Figura 7), que categorizan los elementos en Actividades o acciones, propiedades o estados, eventos, finales de actividad y restricciones. En ambos casos se utiliza un conjunto de reglas heurísticas que permiten realizar la conversión de los diferentes modelos preconceptuales (tabulares o gráficos), en los respectivos modelos conceptuales estructurales o comportamentales. Particularmente,

Figura 6. Conjunto de herramientas del proyecto de la Universidad de Klagenfurt

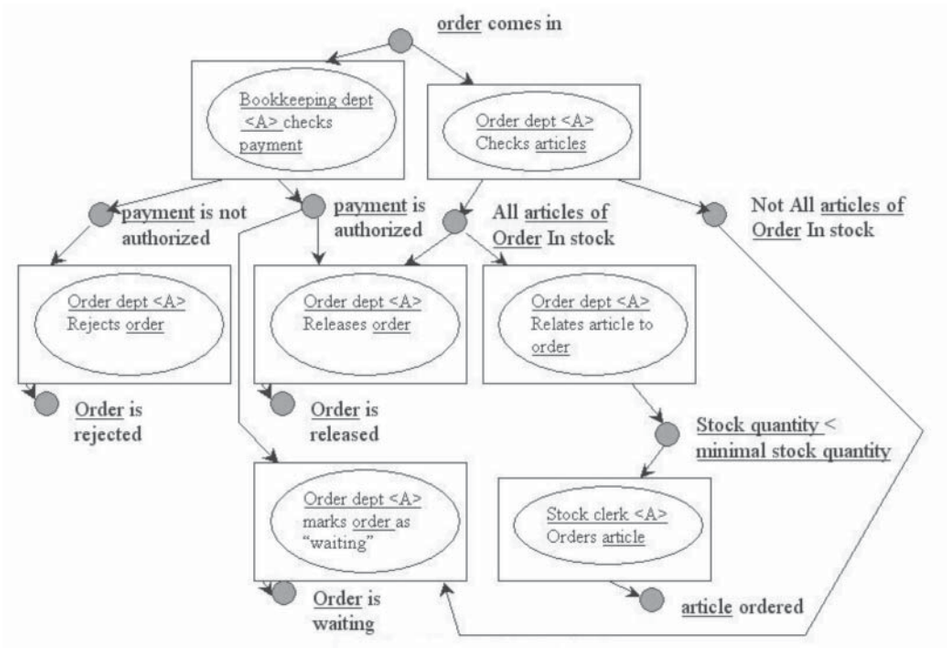


Fuente: (Fliedl, 1999)

para los artículos reseñados, se muestra la conversión en el diagrama de clases y en el diagrama de actividades.

En el idioma español se han comenzado a realizar estudios sobre modelos conceptuales, que poseen un alto contenido textual, para acercarlos al nivel de implementación (Díaz, Sánchez y Matteo, 2003; Díaz et al., 2004). Este mismo tema para el idioma inglés es tratado por Ben Achour (Rolland y Ben Achour, 1998), (Ben Achour, 1999). No se conocen trabajos que tomen como punto de partida el lenguaje natural en español para la construcción de esquemas conceptuales.

Figura 7. Esquema preconceptual de la Universidad de Klagenfurt para el trazado de diagramas de actividades



Los trabajos hasta aquí reseñados involucran la descripción en lenguaje natural de los requisitos funcionales de las organizaciones. A diferencia de ellos, el Proyecto KAOS (Knowledge Acquisition in autOmated Specification) presentado por Dardenne incorpora requisitos no funcionales, expresados en lenguaje natural, en un esquema conceptual construido a partir de un lenguaje formal (Dardenne, 1993). En este esquema, el analista debe realizar la traducción del lenguaje natural al lenguaje formal (Van Lansweerde y Letier, 1998; Letier, 2001; Van Lansweerde, 2001). En una línea similar se encuentra el trabajo de Koubarakis y Plexousakis, que también involucra los requisitos no funcionales en forma de metas de la organización expresadas en un lenguaje llamado cálculo situacional, que es un formalismo para la representación del conocimiento y razonamiento en dominios que evolucionan dinámicamente (Koubarakis y Plexousakis, 1999 y 2000).

En la sección siguiente se compendian algunos de los problemas que subsisten, en lo que respecta al uso de modelos verbales para la construcción de esquemas conceptuales.

2. Problemas por resolver

Como se expresó en la introducción, el punto de partida para la construcción de cualquier pieza de software está constituido por su modelo verbal. En la sección anterior se listaron varios trabajos que emplean ese modelo verbal para tratar de identificar

partes de diferentes esquemas conceptuales. En esta sección se analizan algunos vacíos que son susceptibles de explorar:

2.1 No se entabla un diálogo entre los usuarios y los analistas, para verificar la comprensión del dominio o la completitud de los diagramas

Existen trabajos que propugnan por la construcción de modelos preconceptuales, a partir de modelos verbales en lenguaje natural restringido (RADD, ASPIN, KCPM, NL-OOPS), que posteriormente sirven para realizar modelos conceptuales de diferente índole (diagrama Entidad-Relación, diagramas de Clases o diagramas de Interacción de UML). En estos trabajos se plantea el procesamiento del lenguaje natural (PLN) mediante un proceso, primero de tipo sintáctico, en el que se ubica el papel de cada una de las partículas de una oración y luego, mediante un proceso de tipo semántico, que busca el significado de cada una de las partículas en un contexto dado, trabajado por lo general mediante Ontologías. En esos trabajos se establece un diálogo que los diferentes autores llaman abierto entre el usuario final y los analistas y desarrolladores, mediante el esquema preconceptual, el cual afirman que es mucho más claro que los diferentes esquemas conceptuales existentes y que por ello será de más fácil interpretación para los diferentes usuarios finales del software. Sin embargo, los esquemas preconceptuales usados tienen también una simbología gráfica (por ejemplo, en el caso KCPM) o de un lenguaje particular definido para el uso en traducción automática (como ASPIN), cuya lectura puede ser clara para los analistas y desarrolladores después de un mínimo de entrenamiento en la sintaxis de la simbología, pero que aún sigue siendo un tanto obscura para los usuarios finales.

Exceptuando RADD, en ese diálogo que se entabla con el usuario falta la vía de regreso en la comunicación, es decir, que los elementos definidos como esquemas preconceptuales se vuelvan a traducir a lenguaje natural, para confrontarlos con lo que inicialmente el usuario final definió como características del problema, cuya solución se está tratando de implementar con los esquemas preconceptuales como paso inicial; cabe anotar, que herramientas semiautomáticas como LIDA sí realizan esta conversión y la presentan al usuario del sistema (suponen que el usuario del sistema es el diseñador y no el usuario final que entrega el modelo verbal). Se requiere, entonces, que el problema expresado en lenguaje inicial, y traducido a los esquemas preconceptuales, vuelva a traducirse en lenguaje natural, como en una especie de: "lo que usted quiso decir fue...". De esa manera, la comunicación no se rompería y se podría establecer un diálogo productivo en relación con el software, que posibilitara la depuración de los modelos verbales hasta obtener algo que, tanto los analistas y desarrolladores como los usuarios finales, puedan encontrar útil para la construcción del software.

Los lineamientos iniciales para ese diálogo pueden ser los que posee RADD para verificar completitud, clarificación lingüística y clarificación semántica, pero se deben revisar cuidadosamente las reglas heurísticas que lanzan las preguntas de verificación al usuario, puesto que en RADD se establece, para la conversión a modelo Entidad–Relación, sólo uno de los modelos conceptuales disponibles; como esas reglas dependen de las características de los diferentes modelos, habría que determinarlas para cada uno de ellos.

2.2 La mayoría de estos trabajos parte de modelos verbales restringidos que eluden las ambigüedades

Esas "restricciones" del lenguaje en ocasiones son limitaciones a la manera como se estructuran las frases, para evitar problemas como las anáforas o las ambigüedades originadas, por ejemplo en el uso de pronombres para aludir objetos o actores de frases anteriores (este es el caso, por ejemplo, de KCPM). En estos casos, el lenguaje en el que deben ser escritos los modelos verbales limita la interpretación y recolección de las características de la organización, labor ésta que debe ser realizada conjuntamente por los analistas y los usuarios finales. Para el lenguaje español, por ejemplo, ya se han realizado diferentes trabajos para solucionar problemas en cuestiones de anáforas y ambigüedades, que se pueden incorporar en cualquiera de las soluciones que se implementen para la construcción automática de software a partir de modelos verbales. Además,

como establece el proyecto LIDA, en los modelos verbales sin restricción todavía se puede encontrar información relevante a la forma de uso de la aplicación futura y no pueden ser desechados de manera tan inmediata como plantean otros modelos.

Otra forma de restricción la constituyen los subconjuntos del lenguaje, aplicables a ciertos dominios específicos -como es el caso de ASPIN-, que le restan universalidad a la aplicación de la solución. No es claro todavía que pueda ser posible una Ontología propia de todos los dominios disponibles, pero una herramienta que utilice el lenguaje natural para capturar características de los esquemas conceptuales, bien podría ir recabando la información correspondiente a un dominio particular e ir conformando, también de manera automática, la Ontología que pueda dar claridad a los términos que se involucran en un modelo verbal, perteneciente a un dominio particular.

2.3 En los modelos resultantes no se expresan ciertas características propias del dominio, conocidas en la literatura especializada como "reglas del negocio", presentes en los modelos verbales

Los modelos verbales también pueden contener un número no determinado de reglas del negocio, las cuales pueden no tener equivalencias en la mayoría de los modelos conceptuales. Por lo general, las reglas del negocio que no puedan ser expresadas en la sintaxis de los esquemas conceptuales, por referirse a requisitos no funcionales, terminan siendo incluidas mediante notas de tipo textual en las herramientas CASE, y no se pueden traducir de manera directa a código; entonces, esas reglas deberán ser manejadas por los desarrolladores mediante triggers (piezas de código que ejecutan ciertos procesos de manera automática al presentarse ciertos eventos en la aplicación) o subrutinas de programación que puedan dar cuenta de ellas, pero, por su falta de traducción formal a algún tipo de modelo, pueden ser muy fácilmente dejadas de lado en el desarrollo, especialmente en proyectos de gran envergadura que posean muchas restricciones a nivel de organización.

2.4 Se realizan modelos con alto contenido gráfico, pero se dejan de lado modelos con alto contenido textual

Algunos de los modelos empleados para el desarrollo de software, tienen grandes cantidades de lenguaje natural incorporadas en su estructura. El modelo causa–efecto o el diagrama de procesos o la descripción de los diagramas de casos de uso, por citar algunos ejemplos, poseen aún muchos textos en lenguaje natural, que son indispensables para la adecuada comprensión del sistema futuro. Lo que se ha trabajado hasta ahora en el procesamiento del lenguaje natural (PLN), enfocado a la Ingeniería de requisitos, realiza simplificaciones que pueden traducir a modelos gráficos, pero que difícilmente pueden traducirse a otros textos en lenguaje natural que hagan parte de modelos, como los que se han ejemplificado. Esos modelos hasta ahora generados, se enfocan casi todos en diagramas tipo UML, pero no permiten la realización de diagramas con alto contenido textual y menos contenido gráfico. Además, presentan como complicación adicional el hecho de que sus textos pueden generar problemas de consistencia para su construcción, de forma tal que no se puede garantizar que una parte de uno de los diagramas se refiera específicamente a una parte de otro de los diagramas, como ocurre en el caso de los diagramas causa–efecto, que se deberían asociar en cada una de sus partes a uno de los procesos del Diagrama de Procesos de la Organización. No se han definido ni incluido en los diferentes trabajos analizados, reglas de consistencia que permitan deducir este tipo de relaciones de manera automática, con el fin de que el software las incorpore y valide en el proceso de desarrollo.

2.5 No se verifica la consistencia intra e intermodelos

Si bien la construcción de los diferentes diagramas parte de la misma descripción en lenguaje natural, cada uno de los diagramas se elabora por separado, lo que genera ciertas reglas internas de consistencia, pero por otra parte no se puede realizar un proceso dinámico de comunicación entre los diferentes

diagramas, puesto que la retroalimentación entre ellos sólo es posible vía el modelo verbal. Estas reglas de consistencia entre modelos aún no han sido completamente definidas para los diferentes diagramas que hacen parte de un mismo proyecto de desarrollo de software, y sólo con la especificación de UML 1.4 y 2.0 (OMG, 2004) se han venido trabajando en lenguaje OCL. Una herramienta que tome como punto de partida la especificación de los requisitos en lenguaje natural, debe tener la capacidad de verificar la consistencia intra e intermodelos, de los esquemas conceptuales pertenecientes a un mismo proyecto.

2.6 Existen pocos trabajos para el español

Se han realizado estudios para idiomas como el Alemán, el Inglés o el Italiano, pero no se conocen trabajos a este respecto en español –haciendo la salvedad, claro está, de que Díaz, Sánchez y Matteo trabajan en lenguaje natural pero desde un esquema conceptual como es el modelo de Casos de Uso para facilitar la traducción automática a código-. El Español tiene características que dificultan aún más la comprensión de los textos en lenguaje natural, por la gran cantidad de verbos irregulares que se pueden encontrar en su estructura y por las diferentes clases de ambigüedades que posee; estos problemas hacen del Español un lenguaje altamente complejo para su interpretación. Los lenguajes como el inglés y el alemán tienen estructuras más sintéticas que pueden generar una interpretación más adecuada de los diferentes textos. En estos aspectos sería importante la vinculación de trabajos previos sobre la desambiguación (Galicia, 2000) o que realicen propuestas para el manejo de los verbos en español (Zamorano, 2002).

2.7 Existen problemas asociados con la actualización del software existente

Los trabajos analizados, contemplan la creación de los esquemas preconceptuales a partir del modelo verbal de la organización; sin embargo, ninguno de ellos evalúa qué ocurre cuando el software ya existe y se requiere realizar una modificación al mismo partiendo de un modelo verbal. En este caso es fácil elaborar un esquema preconceptual con las características de los que se hacen en la literatura especializada, pero ¿cómo se puede garantizar que ese modelo se acomode con las características del software existente?, ¿será posible expresar un software existente (o, en su defecto, la documentación que le dio origen) en términos de lenguaje natural, que permita realizar las modificaciones pertinentes y reexpresarlas en términos del software ya construido (o de los modelos que lo conformaron)?

2.8 Falta complementar las reglas heurísticas que sirven de punto de partida para los diagramas

Las reglas heurísticas que posibilitan la transformación de los esquemas preconceptuales en conceptuales, aún pueden ser un motivo de complementación. Herramientas como CM-Builder y KCPM, presentan primeras versiones de los diagramas de clases; por ejemplo, trazan el diagrama de clases pero no incluyen los métodos, que son elementos pertenecientes a las clases y que definen las acciones que puede realizar una determinada clase. Además, con el tipo de diálogo planteado en 3.1., incluso se podría llegar a preguntarle al usuario de qué manera se realiza el proceso, conformando una especie de pseudocódigo que facilite la posterior programación de los métodos.

2.9 No se trabajan modelos diferentes al Diagrama Entidad–Relación o a los diagramas de UML

Los modelos analizados muestran la realización de esquemas estructurales (Entidad–Relación, diagrama de Clases) y esquemas dinámicos (diagrama de actividades) y se establece que de manera similar se pueden obtener los diagramas de colaboración, secuencias y estados. No se mencionan diagramas de otros tipos, como los de componentes, los de casos de uso o los de despliegue, y menos aún diagramas diferentes a UML, como el diagrama de procesos, el diagrama causa–efecto o el diagrama CRC. Puede ser un tópico de investigación, la inclusión de un conjunto de diagramas que permita tener una visión completa del software que facilite su traducción posterior a código.

Conclusiones

Los requisitos expresados mediante modelos verbales, pueden dar origen a la construcción de los diagramas conceptuales de una aplicación, lo cual se ejemplifica en el artículo con diferentes tendencias que trazan diagramas Entidad–Relación, diagramas de Clases, diagramas de Actividades y otros para aplicaciones específicas. Sin embargo, todavía existen vacíos en la literatura especializada sobre los que se puede trabajar: el diálogo entablado con el usuario para refinar la especificación en lenguaje natural de los requisitos, la incorporación de desambiguación, la expresión de las reglas del negocio en los esquemas conceptuales, la elaboración automática de modelos conceptuales altamente textuales, los chequeos de consistencia entre modelos generados con la misma especificación en lenguaje natural, la elaboración de las herramientas de traducción para el español, el manejo de especificaciones adicionales para software ya existente, la complementación de las reglas heurísticas para generar esquemas conceptuales más completos y la elaboración de modelos diferentes a UML.

Las características de los diferentes productos y métodos analizados se compendian en la Tabla 1. Las convenciones son las siguientes:

- Análisis sintáctico, semántico y pragmático: aluden a los diferentes tipos de análisis que realizan los diferentes productos y lenguajes, tal como se definen en el contenido del artículo.
- Conversión del modelo verbal (M. V.) en esquemas conceptuales (E. C.): se analiza según la sección 1, colocando heurísticas para ser usadas por el analista, herramientas de apoyo a la aplicación de reglas, herramientas para obtener versiones preliminares del esquema conceptual, herramientas de apoyo a la completitud del modelo y herramientas para apoyar la validación del esquema conceptual.
- Los esquemas preconceptuales pueden ser: grafos conceptuales, redes semánticas y jerarquías clase–atributos.
- Diálogo con el usuario: permite saber si las herramientas usan algún tipo de comunicación en lenguaje natural con el usuario.
- Los modelos conceptuales que se obtienen al aplicar los diferentes métodos, son: modelo de relaciones binarias, diagrama E-R, diagrama de clases, diagrama de actividades y otros.
- Se coloca una X en lenguaje si el método o herramienta se trata de un lenguaje, ya sea implementado o teórico.
- Método manual, semiautomático o automático se marca según el nivel de automatización que presente la herramienta o método que se esté analizando y dependiendo de la manera en que apoya la gestión de los analistas para la determinación de diagramas a partir de los modelos verbales.

Existen varios aspectos en los cuales se pueden iniciar trabajos para el idioma español y que pueden complementar los trabajos analizados en las secciones 2 y 3, tales como:

- Construcción de un analizador sintáctico.
- Complementación de las reglas heurísticas esbozadas en algunos de los productos, que permitan la implementación de algunos elementos como métodos de los diagramas de clases o mensajes de los diagramas de colaboración, los cuales aún no han sido implementados en los diferentes trabajos.
- Definición de reglas heurísticas para el trazado de diagramas causa–efecto o diagramas de procesos, los cuales no están aún incluidos en los productos analizados.
- Establecimiento de conversiones de los diferentes diagramas conceptuales preliminares a lenguaje natural, con el fin de que los usuarios puedan comprender su significado y contribuir en el refinamiento posterior de los mismos.

Complementación de las reglas de validación de los diferentes modelos, que permitan la realización de preguntas automáticas al usuario, para completar los diagramas respectivos.

Tabla 1. Comparación de las características de los modelos presentados

	MODELO												
CARACTERÍSTICA	**Coad-Yourdon**	**Chen**	**NIAM**	**KISS**	**LIDA**	**Herramientas CASE**	**FrameNet**	**WordNet**	**RADD**	**CM-Builder**	**ASPIN**	**NL-OOPS**	**KCPM**
Análisis sintáctico					X		X	X	X	X	X	X	X
Análisis semántico							X	X	X	X	X	X	X
Análisis pragmático									X				
Interpretación automática LN.							X	X					
Conversión M. V. en E. C.	X	X	X		X				X	X	X	X	X
Heurísticas usadas por analista	X	X	X										
Herramientas apoyo a reglas					X								
Herramientas versiones preliminares					X				X	X	X	X	X
Herramientas completitud modelo									X				
Herramientas validación esquema													X
Esquemas preconceptuales			X	X						X	X	X	X
Grafos Conceptuales			X	X							X		
Redes Semánticas										X			
Jerarquías clases, atributos												X	
Otros esquemas preconceptuales													X
Diálogo con el usuario en L. N.					X				X				
Modelos Conceptuales	X	X	X		X	X			X	X	X	X	X
Modelo Relaciones Binarias			X										
Diagrama E-R	X	X				X			X				
Diagrama de clases					X	X				X		X	X
Diagrama actividades						X							X
Otros Diagramas						X					X		X
Lenguaje			X	X							X		
Herramienta computacional					X	X	X	X	X	X		X	X
Método Manual	X	X	X	X							X		
Método Semiautomático					X								
Método Automático							X	X	X	X		X	X

Bibliografía

Ambriola, V. y Gervasi, V. (1997). "An environment for cooperative construction of natural language requirements bases". En: *Proceedings of the Eight International Conference on Software Engineering Environments* (SEE '97), Cottbus. pp. 124-130.

_____ (2001). "On the parallel refinement of NL requirements and UML diagrams". En: *Proceedings of the ETAPS 2001 Workshop on Trasformations in UML*, Genova. 5 p.

Baker, C., Fillmore, Ch. y Lowe, J. (1998). "The Berkeley FrameNet Project". En: *COLING-ACL '98: Proceedings of the Conference of the Association for Computational Linguistics,* Montreal, pp. 86-90.

Ben Achour, C. (1999). *Extraction des bensoins par analyse de Scénarios textuels*. Tesis de Doctorado de la Universidad de París. 294 p.

Buchholz, E., et al. (1995). "Applying a Natural Language Dialogue Tool for Designing Databases". En: *Proceedings of the First International Workshop on Applications of Natural Language to Databases.* 15 p.

Buchholz, E. y Düsterhöft, A. (1994). "Using Natural Language for Database Design". En: *Proceedings Deutsche Jahrestagung für Künstliche Intelligenz.* 5 p.

Burg, J.F.M. y Van de Riet, R.P. (1996). "Analyzing Informal Requirements Specifications: A First Step towards Conceptual Modeling". En: *Proceedings of NLDB'96, 2nd International Workshop Applications of Natural Language to Information Systems*, Amsterdam. 13 p.

Chen, P. P. (1976). "The Entity–Relationship Model: Toward a Unified View of Data". En: *ACM Transactions on DataBase Systems*, Vol. 1, No. 1.

Chen, P. P. (1983). "English Sentence Structure and Entity–Relationship Diagrams". En: *Information Science*, No. 29, Vol. 2. pp. 127-149.

Coad, P. y Yourdon, E. (1990). *Object – Oriented Analysis*. New Jersey: Yourdon Press. 233 p.

Cyre, W. (1995). "A requirements sublanguage for automated analysis". En: *International Journal of Intelligent Systems*, Vol. 10, No. 7. pp. 665-689.

Dardenne, A., van Lamsweerde, A. y Fickas, S. (1993). "Goal-Directed Requirements acquisition". En: *Science of Computer Programming*, Vol. 20. pp. 3-50.

Díaz, I., Sánchez, J. y Matteo, A. (2003). "Criterios de Análisis Sintáctico para la Deducción de Patrones de Interacción de Instancias a partir de Casos de Uso". En: *Memorias de la Conferencia Latinoamericana de Informática*, La Paz. 11 p.

Díaz, I., Pastor, O., Moreno, L. y Matteo, A. (2004) "Una aproximación lingüística de Ingeniería de Requisitos para OO-Method". En: *Memorias del Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes Software* (IDEAS2004), Arequipa. pp. 270-281.

Fellbaum, C. (1998). *WordNet: An Electronical Lexical Database*. MIT Press.

Fillmore, Ch. (1968). "The case for case". En: *Universals in Linguistics*, Bach and Harms, editors. pp. 1- 88.

Fillmore, Ch., Baker, C. (2001). "Frame Semantics for Text Understanding". En: *Proceedings of WordNet and Other Lexical Resources Workshop, NAACL*, Pittsburgh. 6 p.

Fliedl, G. et al. (2002). "The NIBA workflow: From textual requirements specifications to UML-schemata". En: *Proceedings of the ICSSEA '2002 - International Conference "Software & Systems Engineering and their Applications"*, Paris.

_____ (1999). "Linguistically Based Requirements Engineering - The NIBA Project". En: *Proceedings 4th Int. Conference NLDB'99*

Applications of Natural Language to Information Systems, Klagenfurt. pp. 177 – 182.

Fliedl, G., Kop, Ch. y Mayr, H. C. (2003) "From Scenarios to KCPM Dynamic Schemas: Aspects of Automatic Mapping". En: *Proceedings of the Natural Language Processing and Information Systems Conference NLDB'2003,* Lecture Notes in Informatics P-29. pp. 91 - 105.

Fliedl, G., Mayerthaler, W. y Winkler, Ch. (1999). "The NT(M)S-Parser: An Efficient Computational Linguistic Tool". En: *Proceedings of the 1st International Workshop on Computer Science and Information Technologies*, Moscow.

Fliedl, G. y Weber, G. (2002). "Niba-Tag - A Tool for Analyzing and Preparing German Texts". En: *Proceedings of DataMining 2002,* Bologna.

Galatescu, A. (2002). "Reifying Model Integration Abilities from Natural Language". En: *Journal of Conceptual Modeling* No. 24. 19 p.

Galicia, S. (2000). *Análisis sintáctico conducido por un diccionario de patrones de manejo sintáctico para lenguaje español*. Tesis Doctoral del Instituto Politécnico Nacional, México.

Gervasi, V. (1999). *Environment support for requirements writing and analysis*. PhD Thesis, Universidad de Pisa. 172 p.

Gervasi, V. (2001). "The CICO domain-based parser". Reporte técnico TR-01-25, Departamento de Informática, Universidad de Pisa. 52 p. (Consulta 15 de Julio de 2004). http://circe.di.unipi.it/Cico/

Gervasi, V. y Nuseibeh, B. (2000). "Lightweight validation of natural language requirements: a case study". En: *Proceedings of the 4th International Conference on Requirements Engineering.* pp. 140-148.

_____ (2002). "Lightweight validation of natural language requirements". En: Software Practice and Experience, Vol. 32. pp. 113-133.

Gruber, J. (1965). *Studies in Lexical Relations*. Disertación Doctoral, Cambridge, MIT.

Halpin, T. (1998). "Object-Role Modeling (ORM/NIAM)". En: *Capítulo 4 de Handbook on Architectures of Information Systems*, eds P. Bernus, K. Mertins & G. Schmidt, Springer-Verlag, Berlin. 13 p.

Harmain, H. y Gaizauskas, R. (2000). "CM-Builder: An Automated NL-based CASE Tool". En: *Proceedings of the fifteenth IEEE International Conference on Automated Software Engineering* (ASE'00), Grenoble. 9 p.

Hoppenbrouwers, J. (1997). *Conceptual Modeling and the lexicon.* Tesis Doctoral de la Universidad Católica de Brabant. 200 p.

Jin, Z., Bell, D. y Wilkie, F. (1998). "Automatically acquiring the Requirements of Business Information Systems by using Business Ontology". En: *Proceedings of the 13th european conference on Artificial Intelligence*, Brighton. 15 p.

Kamps, J. *et al*. (2004). "Using WordNet to measure semantic orientation of adjectives". En: *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC 2004)*, volume IV, París. pp. 1115-1118.

Kop, Ch. y Mayr, H. C. (2002). "Mapping functional requirements: from natural language to conceptual schemata". En: *Proceedings of the 6th IASTED International conference Software Engineering and applications.* 5 p.

Koubarakis, M. y Plexousakis, D. (1999). "Business Process modelling and design: a formal model and methodology". En: *BT Technology Journal*, Vol. 17, No. 4. pp. 23-35.

_____ (2000). "A formal model for business process modeling and design". En: *Proceedings of CAiSE*00*, Stockholm. 15 p.

Kristen, G. (1994). *Object Orientation: The KISS Method. From Information Architecture to Information Systems.* Addison-Wesley.

Letier, E. (2001). Reasoning *about Agents in Goal-Oriented Requirements Engineering*. Tesis Doctoral, Universidad de Louvaine, 283 p.

Mayr, H. C. y Kop, Ch. (2002) "A user centered approach to Requirements Engineering". En: *Proceedings of "Modellierung 2002"*, Lecture Notes in Informatics P-12. pp. 75-86.

Meyer, J. y Dale, R. (2002). "Using the Wordnet Hierarchy for Associative Anaphora Resolution". En: *Proceedings of the Coling 2002 Workshop ''SemaNet'02: Building and Using Semantic Networks''*, Taipei. 7 p.

Mich L. (1996). "NL-OOPS: From Natural Natural Language to Object Oriented Requirements using the Natural Language Processing System LOLITA". En: *Journal of Natural Language Engineering*, Cambridge University Press, Vol. 2, No. 2. pp. 161-187.

Mich, L. y Garigliano, R. (2002). "NL-OOPS: A Requirements Analysis tool based on Natural Language Processing". En: *Proceedings of the 3rd International Conference on Data Mining 2002*, Bologna. pp. 321-330.

Nijssen, G.M. (1977). "Current issues in conceptual schema concepts". En: *Proceedings of the IFIP Working Conference on Modelling in Data Base Management Systems*. pp. 31-66.

OMG. (2004) OMG Unified Modeling Language Specification. *Object Management Group*. (Consulta 28 de junio de 2004). http://www.omg.org/UML/.

Overmyer, S.P., Lavoie, B., y Rambow, O. (2001) "Conceptual modeling through linguistic analysis using LIDA". En: *Proceedings of ICSE 2001*, Toronto, Canada. 10 p.

Pressman, R. (2002). Ingeniería del Software: Un enfoque práctico. Madrid: McGraw–Hill. 601 p.

Rolland, C. y Ben Achour, C. (1998). "Guiding the Construction of Textual Use Case Specifications". En: *Data & Knowledge Engineering Journal*, Vol. 25, No. 1-2. pp. 125-160.

Van Lamsweerde, A. (2001). "Goal-Oriented Requirements Engineering: A guided tour". *Proceedings 5th IEEE Symposium on Requirements Engineering*, Toronto. pp. 249-263.

Van Lamsweerde, A. y Letier, E. (1998). "Integrating obstacles in goal-driven Requirements Engineering". En: *Proceedings 20th Conferecence on Software Engineering*, Kioto. 10 p.

Zamorano, J. (2000) "La morfología verbal del español y la generación automática". En: *Procesamiento del Lenguaje Natural*, No. 28. pp. 35-43.

Zowghi, D. y Gervasi, V. (2002) "The Three Cs of Requirements: Consistency, Completeness and Correctness". En: *Proceedings of 8th International Workshop on Requirements Engineering: Foundation for Software Quality*, (REFSQ'02), Essen. pp. 155-164.

Zowghi, D., Gervasi, V. y McRae, A. (2001) "Using Default Reasoning to Discover Inconsistencies in Natural Language Requirements". En: *Proceedings of the Eighth Asia-Pacific Software Engineering Conference (APSEC'01)*, Macao. pp. 133-142.