DEFINICIÓN DE UN ESQUEMA PRECONCEPTUAL PARA LA OBTENCIÓN AUTOMÁTICA DE ESQUEMAS CONCEPTUALES DE UML

CARLOS MARIO ZAPATA J.

Tesis presentada como requisito parcial para optar al Título de Doctor en Ingeniería.

Este documento tiene únicamente propósitos de evaluación y no debería ser consultado o referido por cualquier persona diferente a los evaluadores.

Director: Ph.D. Fernando Arango Isaza – Universidad Nacional de Colombia
Co-Director: Ph.D. Alexander Gelbukh – Instituto Politécnico Nacional de México

Comité Doctoral:
Ph.D. Grigori Sidorov – Instituto Politécnico Nacional de México
Ph.D. Raquel Anaya – Universidad EAFIT de Colombia
Ph.D. Demetrio Ovalle C. – Universidad Nacional de Colombia

POSTGRADO EN SISTEMAS
FACULTAD DE MINAS
UNIVERSIDAD NACIONAL DE COLOMBIA
SEDE MEDELLÍN
2007

DEDICATORIA

A Vicky, Sebas y Pipe,
tres pruebas incontrovertibles de que
el amor es más fuerte que la adrenalina
cuando se busca alcanzar un sueño.

Carlos M.

AGRADECIMIENTOS

Son demasiadas las personas e instituciones que han contribuido a que esta Tesis se lleve a efecto. Si la memoria me falla y olvido mencionar a alguno, por favor mis disculpas. Mis mayores agradecimientos para mis tutores, Fernando Arango y Alexander Gelbukh, quienes a lo largo de estos años me han enseñado el quehacer del Investigador; sus enseñanzas las llevaré conmigo mientras viva. También valoro mucho las enseñanzas y apoyo del profesor Isaac Dyner, al igual que la motivación que a lo largo de mi vida académica me han suministrado los profesores Demetrio Ovalle y Jaime Tabares; ellos me alentaron a iniciar la vida académica de la cual ahora disfruto. Igualmente, agradezco el apoyo del Dr. Oscar Pastor por integrar el comité doctoral a lo largo de mis estudios; su esfuerzo también se refleja en los resultados de esta Tesis.

Muchas gracias a la DIME, la DINAIN, la Red de MacroUniversidades de América Latina, la Vicedecanatura de Investigación y Extensión de la Facultad de Minas y la Dirección Académica de la Sede Medellín, entidades que contribuyeron económicamente a que este proyecto se volviera una realidad y a la Escuela de Sistemas de la Facultad de Minas, Universidad Nacional de Colombia, Sede Medellín por su invaluable apoyo, al contribuir con tiempo y recursos para la realización de esta Tesis. Un agradecimiento muy especial también para Luis Fernando Londoño de Avansoft, por creer que este proyecto puede ser una realidad tangible para la industria del software.

Agradezco inmensamente a muchísimos alumnos de Maestría y pregrado que creyeron en este proyecto y se atrevieron a hacer cosas diferentes en pro de la Investigación. Estos son sólo algunos nombres: Luz Marcela Ruiz, David Cardona, Fernán Villa, Aldrin Fredy Jaramillo, Guillermo González, Betsy Mary Estrada, Paula Tamayo, María Clara Gómez, Liliana Garcés, Alfonso Lezcano, Andrés Muñetón, Roberto Manjarrés, Natalí Olaya, Carolina Palacio, Alejandra Grajales, Karla Cristina Palomino, Roberto Rosero, John Edison Mesa, Andrés Ignacio Baena, Nicolás Carmona, Mary Inés Duarte, Juan Carlos Hernández, Raúl Zuluaga, Sebastián Vallejo, Diego Figueroa.

Igualmente, quiero expresar mi gratitud a todas las personas que directa o indirectamente han apoyado estas ideas, a todos mis demás colaboradores, colegas, estudiantes y amigos y a quienes de alguna manera me ayudaron a salir adelante. Dentro de este inmenso grupo se destaca el profesor Juan David Velásquez, por su apoyo incondicional a este proyecto y a las iniciativas que estamos planteando, y el profesor Santiago Montoya (Q.E.P.D), quien con su aliento hizo más fáciles algunos de los difíciles momentos del Doctorado. Mi eterna gratitud también para personas muy importantes en todo este proceso como las profesoras María Teresa Berdugo y Salomé París, siempre con el consejo preciso en las más adversas circunstancias, Rosa Elvira Correa, por su apoyo en México, y Zorelly Jaramillo, por ser el soporte de todas nuestras labores. Muchas gracias también a Néstor Giraldo por sus enseñanzas, no solo lingüísticas sino también vivenciales, a Soraida Aguilar y a la profesora Norma Lucía Botero por su ayuda en los diseños experimentales.

CONTENIDO

CONTENIDO

INDICE DE FIGURAS

INDICE DE FIGURAS

INDICE DE FIGURAS

INDICE DE FIGURAS

INDICE DE TABLAS

ABREVIATURAS:

CASE: Computer-Aided Software Engineering – Ingeniería de Software asistida por computador.
CG: Conceptual Graphs – Grafos Conceptuales.
CGIF: Conceptual Graph Interchange Format – Formato de Intercambio de Grafos Conceptuales.
HPSG: Head-driven Phrase Structure Grammar – Gramática de Estructura de Frases Orientada por Encabezados.
KIF: Knowledge Interchange Format – Formato de Intercambio de Conocimientos.
LKB: Lexical Knowledge Base – Base de Conocimientos Léxica.
LN: Lenguaje natural.
LPPO: Lógica de Predicados de Primer Orden.
MRS: Minimal Recursion Semantics – Semántica de Recursión Mínima.
MTT: Meaning-Text Theory – Teoría Significado-Texto.
OMG: Object Management Group – Grupo de Manejo de Objetos.
PLN: Procesamiento del Lenguaje Natural.
TM: Traducción de máquina.
UML: Unified Modeling Language – Lenguaje Unificado de Modelamiento.

RESUMEN

La Elicitación de Requisitos de software es una parte de la Ingeniería de Requisitos donde se procura capturar, analizar, sintetizar y convertir a esquemas conceptuales las necesidades del interesado. Este proceso, que se realiza interactivamente con la participación de analistas e interesados en el desarrollo de la pieza de software, suele presentar problemas de comunicación originados en la diferencia de especialidades de los participantes en el desarrollo de la pieza de software. Tradicionalmente, en Ingeniería de software se han solucionado los problemas de este tipo empleando métodos de desarrollo.

Dado que los diferentes métodos no garantizan la solución de los problemas de comunicación, ha surgido una nueva tendencia para la generación automática de esquemas conceptuales desde lenguajes controlados. En esta nueva tendencia, existen aún problemas tales como los siguientes:

- Se sigue requiriendo una alta participación del analista, lo cual hace subjetivo el proceso.
- Se suelen enfocar los proyectos hacia la obtención de un solo diagrama (generalmente Clases o Entidad-Relación).
- Cuando los proyectos se enfocan a obtener varios diagramas de UML, se suelen emplear representaciones intermedias independientes para cada uno de los diagramas generados, lo que suele ocasionar problemas de consistencia entre los diagramas resultantes.

En esta Tesis se propone un entorno para la generación automática de esquemas conceptuales de UML a partir de un lenguaje controlado denominado UN-Lencep. Para ello, se define un nuevo tipo de esquemas intermedios—los Esquemas Preconceptuales—y se propone un conjunto de reglas heurísticas de transformación desde UN-Lencep hacia

Esquemas Preconceptuales y de allí a los diagramas de Clases, Comunicación y Máquina de Estados.

Los principales aportes de esta Tesis se pueden sintetizar así:

- La especificación de UN-Lencep, un nuevo lenguaje controlado que puede ser aplicable a cualquier dominio, pero que contiene los elementos necesarios para obtener automáticamente los denominados Esquemas Preconceptuales.
- La definición de los Esquemas Preconceptuales, su sintaxis y su forma de uso, además de las reglas para obtenerlos desde UN-Lencep.
- La conformación de un conjunto de reglas heurísticas que permiten la generación automática de diagramas de Clases, Comunicación y Máquina de Estados a partir de los Esquemas Preconceptuales.
- La implementación de los elementos descritos en una nuevo tipo de herramientas CASE, que se ocupa de la interpretación de un discurso en UN-Lencep para generar automáticamente los diagramas de UML mencionados. El prototipo de una herramienta CASE de este tipo, denominado UNC-Diagramador, también es un aporte de esta Tesis.

Con estos aportes se pretende la reducción del tiempo de elaboración de los diagramas de UML, el mejoramiento de la calidad de los diagramas que hacen parte de un mismo discurso en UN-Lencep y la creación de un conjunto de artefactos que permitan mejorar la comunicación entre analistas e interesados, acercando el lenguaje técnico del analista al lenguaje natural del interesado, y posibilitando la validación de los elementos que hacen parte de la descripción de un problema que requiere una solución informática. Finalmente, se pretende el mejoramiento de la calidad, la cual se entiende como la carencia de errores en corrección (la utilización de la sintaxis adecuada), consistencia (la representación de un mismo elemento en diferentes diagramas) y completitud (la adecuación de cada uno de los diagramas con el discurso en UN-Lencep).

Como trabajos futuros que se encuentran fuera del alcance de esta Tesis, pero que se pueden nutrir de sus resultados, se cuentan los siguientes:

- La generación automática de código ejecutable a partir de los diagramas que arroja el UNC-Diagramador.
- La definición de reglas heurísticas para la obtención de otros diagramas de UML, por ejemplo Casos de Uso o Secuencias.
- La complementación de la especificación de UN-Lencep, para acercarlo cada vez más a Lenguaje Natural.

ABSTRACT

Software Requirements Elicitation is the branch of Requirements Engineering to capture, analyze, synthesize, and convert the needs of the stakeholders into conceptual schemas. This process is made by means of the interactive participation of analysts and stakeholders in the software development process, and most of the times it presents communication problems, which can be originated in the differences of specialties among software development participants. Software development methods have been traditionally used in order to solve communication problems, and Requirements Engineering is an important part of these methods.

Due to the fact that Software development methods are not good enough for solving communication problems, a new trend for automatic generation of conceptual schemas from controlled languages has emerged. However, this new trend still has problems to be solved:

- Analysts are often required in the process, and their subjectivity affects the entire process.
- Projects of this new trend are focused on obtaining only one diagram (commonly class diagram or entity-relationship diagram).
- When projects are focused on several UML diagrams, they use intermediate representations oriented independently to every one of the target diagrams. Consequently, consistency problems among the resulting diagrams arise.

We propose, in this Thesis, a new environment for automatically generating UML conceptual schemas from UN-Lencep (a controlled language). We also define, in order to achieve this goal, a new kind of intermediate schemas, called Pre-conceptual Schemas, and

we propose a set of rules to transform a UN-Lencep discourse to these intermediate schemas, and then to Class, Communication, and State Machine diagrams.

The main contributions of this work are summarized as follows:

- The specification of UN-Lencep, a new controlled language applicable to any domain. UN-Lencep is suitable for automatically obtaining the so-called Pre-conceptual Schemas.
- The definition of Pre-conceptual Schemas syntax and the set of rules for generating them from UN-Lencep.
- The proposal of a set of heuristic rules for generating Class, Communication, and State Machine diagrams by means of Pre-conceptual Schemas.
- The implementation of the above defined elements in a new kind of CASE tool to interpret a UN-Lencep discourse and to automatically generate the mentioned UML diagrams. UNC-Diagrammer, the prototype of such CASE tool, is also a contribution of this Thesis.

We make these contributions in order to:

- Reduce the time period dedicated to UML diagrams making.
- Improve the quality of UML diagrams generated from one UN-Lencep discourse.
- Create a set of artifacts to improve the analyst-stakeholder communication. To achieve this goal, we pretend to bridge the gap between technical and natural language. Better communication facilitates validation of the modeling elements, which represent the information solution of a problem.
- Improve the quality of the models. Better quality is related to the reduction of errors in correction (the adequate use of syntax), consistency (the proper representation of the

same element in different diagrams), and completeness (the use, in the diagrams, of enough elements from the UN-Lencep discourse).

The results of this Thesis can generate the following future work:

- Automatic generation of a source executable code from the diagrams made by means of the UNC-Diagrammer.
- Definition of additional heuristic rules to obtain other UML diagrams, for example Sequence or Use Case diagrams.
- Addition of new elements to the UN-Lencep specification, in order to make it close to the natural language.

1. INTRODUCCIÓN

A finales de la década de los 60 el software afrontaba una de las peores crisis de su historia: las diferentes aplicaciones que se construían se retrasaban considerablemente, desbordaban los presupuestos y presentaban innumerables errores al momento de su entrega. Como una forma de solución, el comité científico de la OTAN lanzó como propuesta la creación de una ciencia que propugnara por la aplicación de esquemas metodológicos para el desarrollo de software, tal y como otras ciencias lo habían hecho siglos atrás para enfrentar sus problemas, y fundaron las bases para lo que hoy se conoce como Ingeniería del Software (Gibbs, 1994). Cuatro décadas después, el desarrollo de software continúa siendo caótico; el afán desenfrenado de los desarrolladores por culminar las diferentes aplicaciones continúa ocasionando problemas en su desarrollo. Sin embargo, la Ingeniería de Software se ramificó y comenzó a generar una nueva conciencia en los equipos de desarrollo. De esta manera surgió la Ingeniería de Requisitos, como una forma de garantizar que las piezas de software que se desarrollen satisfagan las necesidades y expectativas de sus interesados; con esto se busca evitar que se construya un software diferente al que los usuarios necesitan y enfrentar, de esta manera, la complejidad en el desarrollo de software.

Para lograr este resultado, la Ingeniería del Software ha aplicado dos enfoques:

- La definición de algunos marcos metodológicos para el desarrollo de software, tales como RUP (Rational Unified Process), CDM (Custom Development Method), FDD (Feature Driven Development), XP (Extreme Programming), y otros, que en el fondo buscan la formalización de los requisitos del cliente para convertirlos en especificaciones del software. La mayoría de estos métodos, deja la responsabilidad de la interpretación del discurso del interesado en manos del analista. Ello genera problemas de validación de la información por parte del interesado, quien sólo puede

realizar tales validaciones cuando, después de algunos meses de trabajo, puede apreciar las versiones iniciales del software construido. Dentro de este enfoque se ha incluido también el desarrollo de varias técnicas para la captura de requisitos, tales como las entrevistas, las técnicas para facilitar la especificación de aplicaciones, el despliegue de la función de calidad, las tormentas de ideas, los juegos de roles, la introspección, los tableros de historias y otras técnicas listadas en Pressman (2005) y Sommerville (2001). En estas técnicas, la responsabilidad de la interpretación de los resultados recae nuevamente sobre el analista.

- La realización de un proceso semiautomático o automático que, a partir de lenguaje natural controlado, genere los esquemas conceptuales. Esta tendencia surge de la necesidad de apoyar al analista en la comprensión de las necesidades del interesado, que se suelen expresar en discursos en lenguaje natural. Sin embargo, los trabajos hasta ahora realizados parten de lenguajes controlados y se suelen dirigir a un único diagrama (Clases o Entidad-Relación). Un único trabajo de esta tendencia permite la generación de varios diagramas de UML, pero emplea artefactos diferentes para cada diagrama, lo cual puede ocasionar problemas de consistencia entre los diagramas generados. Además, la validación de la completitud (información del dominio no descrita en el discurso del interesado) de los diagramas generados no se puede realizar, puesto que comúnmente los interesados no comprenden los esquemas conceptuales.

Las especificaciones del software, algunas de ellas traducidas en esquemas conceptuales y otras simplemente expresadas en lenguajes formales o informales, posteriormente permiten la toma de decisiones de diseño y finalmente la construcción de programas ejecutables. Entre más pronto se puedan tomar esas decisiones de manera conjunta entre el interesado y el analista, más se reducirá el tiempo de elaboración de la pieza de software. Ello implica dos problemas: demoras en la elaboración del software y falta de calidad en la información proveniente del interesado.

Una solución a los inconvenientes planteados deberá resolver las siguientes preguntas de investigación:

- ¿Es posible la creación de un esquema intermedio entre alguna forma de lenguaje controlado y los diagramas de UML, que contenga implícitamente la sintaxis de los tres tipos de diagramas de UML—estructural, comportamental y de interacción—y a la vez la información del dominio entregada por un interesado?
- ¿Ese esquema intermedio podrá ser obtenido a partir de un lenguaje controlado que sea entendible por el interesado y por el analista?
- ¿Es posible la definición de un conjunto de reglas heurísticas, que posibilite la generación automática de al menos un diagrama UML de cada tipo a partir del esquema intermedio obtenido?
- ¿Con el uso de los elementos anotados se podrá disminuir el tiempo de elaboración de los diagramas de UML, incrementando además su consistencia y suministrándole al interesado la posibilidad de completar la información disponible para el desarrollo de la pieza de software?
- ¿Es posible realizar los numerales anteriores para el idioma español, tomando en cuenta las características especiales que pueden existir en temas como la conjugación de verbos y los plurales?

Esta Tesis se puede enmarcar en el segundo enfoque de solución, y para ello se propone la definición de un esquema intermedio, denominado Esquema Preconceptual, que es obtenible desde un lenguaje controlado propuesto también en este trabajo, el UN-Lencep. Además, los denominados Esquemas Preconceptuales deben contener la información suficiente para generar automáticamente los siguientes diagramas de UML: Clases, que es un diagrama estructural, Máquina de Estados, que es un diagrama comportamental, y Comunicación, que es un diagrama de interacción. Se incluyen también en este trabajo las reglas que posibilitan los dos tipos de conversión y algunas plantillas que pretenden acercar

el UN-Lencep al lenguaje natural. Las primitivas de los diagramas UML que se pueden generar a partir de los Esquemas Preconceptuales se discuten en el Capítulo 2.

En este sentido, el aporte de esta Tesis es de corte teórico y conceptual, puesto que se trata de la definición de un nuevo tipo de diagrama que facilita el proceso de construcción de los esquemas conceptuales de UML a partir de un lenguaje cercano al lenguaje natural. Además, el resultado sirve como punto de partida para el apoyo a un método de desarrollo de software, el cual debería facilitar e incorporar los Esquemas Preconceptuales para la obtención de los esquemas conceptuales. Estos últimos describen el problema y sus posibles soluciones.

A manera de Hipótesis preliminar para esta Tesis, se puede establecer que es posible definir un Esquema Preconceptual, que expresa las frases del dominio del interesado en un lenguaje controlado cercano al lenguaje natural, y del cual es posible extraer los diferentes elementos que permiten el trazado de esquemas conceptuales UML de tipo estructural, comportamental o de interacción, empleando reglas heurísticas de transformación.

Esta Tesis está organizada de la siguiente manera: en el Capítulo 2 se realiza la definición del problema de investigación que motiva esta Tesis, definiendo de paso el Marco Conceptual del mismo; el Capítulo 3 se ocupa de un análisis descriptivo de los trabajos más relevantes que se han utilizado en el segundo enfoque mencionado, complementado con un análisis crítico del Estado del Arte con base en el Marco Conceptual, incluyendo una comparación de los diferentes métodos; la propuesta de solución se discute y presenta en el Capítulo 4; en el Capítulo 5 se presentan los casos de estudio y los experimentos realizados para validar la propuesta; en el Capítulo 6 se listan las publicaciones que se han realizado para la difusión de los resultados. Finalmente, en el Capítulo 7 se sintetizan los hallazgos y se presenta el trabajo futuro que se puede derivar de esta propuesta.

2. MARCO CONCEPTUAL DE LA PROBLEMÁTICA

2.1. Problemas Asociados con la Elicitación de Requisitos:

El proceso de la Ingeniería de Requisitos se inicia en la fase de definición con una serie de reuniones entre el equipo de desarrollo de software (particularmente los analistas del equipo de desarrollo) y los clientes y usuarios finales (llamados por lo general "interesados", que es un término proveniente del vocablo inglés *stakeholder*) para realizar lo que se suele denominar Elicitación de Requisitos. La elicitación no es otra cosa que el descubrimiento de las necesidades y expectativas de los interesados en relación con el problema que se desea sistematizar (Leite, 1987), de forma tal que se puedan posteriormente traducir a especificaciones del software durante la denominada fase de análisis. En estas reuniones, analistas e interesados estudian diversos aspectos del área de aplicación, tales como los objetivos del área, los procesos con los que se realizan los objetivos, las necesidades de información que involucran los procesos, y las posibles formas en que el computador podría suministrar la información y coordinar los procesos. La Elicitación de Requisitos se fundamenta, entonces, en un diálogo entre el analista y los interesados. En este diálogo, el analista aporta su conocimiento sobre la elaboración de software, y los interesados aportan su conocimiento sobre el área de aplicación. De este diálogo surge, por un lado, un texto en lenguaje natural que consigna el conocimiento de los interesados (o incluso otros documentos que incluyen listas de requisitos, objetivos y restricciones, que dependen de la técnica de captura de requisitos que se emplee) y, por el otro, un conjunto de diagramas semiformales que consignan la interpretación que hace el analista de dicho conocimiento.

La labor del analista en esta fase es doble: en primer lugar, interpreta el texto aportado por los interesados, eliminando las ambigüedades e imperfecciones propias del lenguaje natural y, en segundo lugar, traduce su interpretación a los diagramas semiformales propios del desarrollo de software (*v.g.* diagramas causa-efecto, modelos de procesos, diagramas de

casos de uso, diagramas de clases, diagramas de máquina de estados, diagramas de comunicación, diagramas de secuencias, etc.). Este proceso se ejemplifica en la Figura 1, donde se aprecia cómo la descripción en lenguaje natural del interesado la traduce el analista en sus modelos (para la figura, un diagrama de clases). En la Figura 2 se ejemplifican los dos problemas más importantes que se pueden presentar en la relación analista-interesado, que son:

Figura 1. El interesado (la dama) realiza una descripción en lenguaje natural que el analista (el caballero) convierte en sus diagramas.

Figura 2. Principales problemas de la Elicitación de requisitos.

- La capacidad del analista para interpretar correctamente el texto, que parte de los interesados, está severamente limitada por su desconocimiento del área de aplicación, el cual se presenta a pesar de que el discurso del dominio se exprese, como en algunos trabajos, mediante patrones lingüísticos o plantillas especiales que ayuden en su escritura. Este desconocimiento se traduce en que el analista es con frecuencia incapaz de darle el giro correcto a las frases y términos propios del área, e incapaz de guiar adecuadamente las reuniones para resolver las ambigüedades y faltantes de información.
- La capacidad de los interesados para interpretar los diagramas semiformales, que parten del analista, está severamente limitada por su desconocimiento de los métodos de desarrollo de software; a este respecto, Haumer *et al.* (2000) afirman que "los modelos conceptuales son difíciles de entender para alguien que no está involucrado en el proceso de definición de modelos", como es el caso de los interesados en el proceso de

desarrollo de software. Este desconocimiento se traduce en que ellos son incapaces de detectar tempranamente los errores de interpretación cometidos por el analista y el consecuente desacople del sistema informático, a ser elaborado, respecto de sus necesidades reales.

A partir de la década de los 90, ha surgido una tendencia que propugna por la construcción, automática o semiautomática, de esquemas conceptuales del software, tomando como base los requisitos del cliente expresados en un lenguaje cercano al lenguaje natural. Esta es una de las aplicaciones recientes de una técnica conocida desde la década de los 50, cuando la guerra fría obligó a la realización de traducciones automáticas para decodificar las comunicaciones; esta tendencia en adelante se denominaría Procesamiento del Lenguaje Natural (PLN). Diferentes investigadores han utilizado esta técnica, en especial para realizar la conversión propuesta en la Figura 1, es decir, la obtención automática de esquemas conceptuales a partir de lenguaje natural, aunque también hay antecedentes que se remontan a la Traducción de Máquina, una manera de realizar las traducciones entre diferentes lenguajes naturales empleando sistemas computarizados. Estos trabajos se revisan con mayor detenimiento en el Capítulo 3.

2.2. Procesamiento del Lenguaje Natural:

Ya sea que se procure la obtención de esquemas conceptuales a partir de lenguaje natural, o que mediante Traducción de Máquina se obtenga la traducción de un texto de un lenguaje a otro, el Procesamiento del Lenguaje Natural se realiza con similitudes entre los diferentes trabajos, lo cual permite la definición de un marco conceptual que posibilite la comparación de las propuestas (en el caso de la Traducción de Máquina lo que se obtiene es un texto en otro lenguaje en lugar de un esquema conceptual). En la Figura 3 se esquematiza el proceso general realizado para la obtención de esquemas conceptuales (particularmente UML); en dicha figura, las líneas discontinuas poseen un símbolo de interrogación para representar la

incertidumbre que posee uno de los actores (analista o interesado) en relación con cada uno de los resultados intermedios del proceso. Las flechas continuas que parten de los actores hacia cada resultado intermedio representan aquellos elementos del proceso sobre los cuales cada actor puede realizar una verificación adecuada a su nivel de conocimiento del problema.

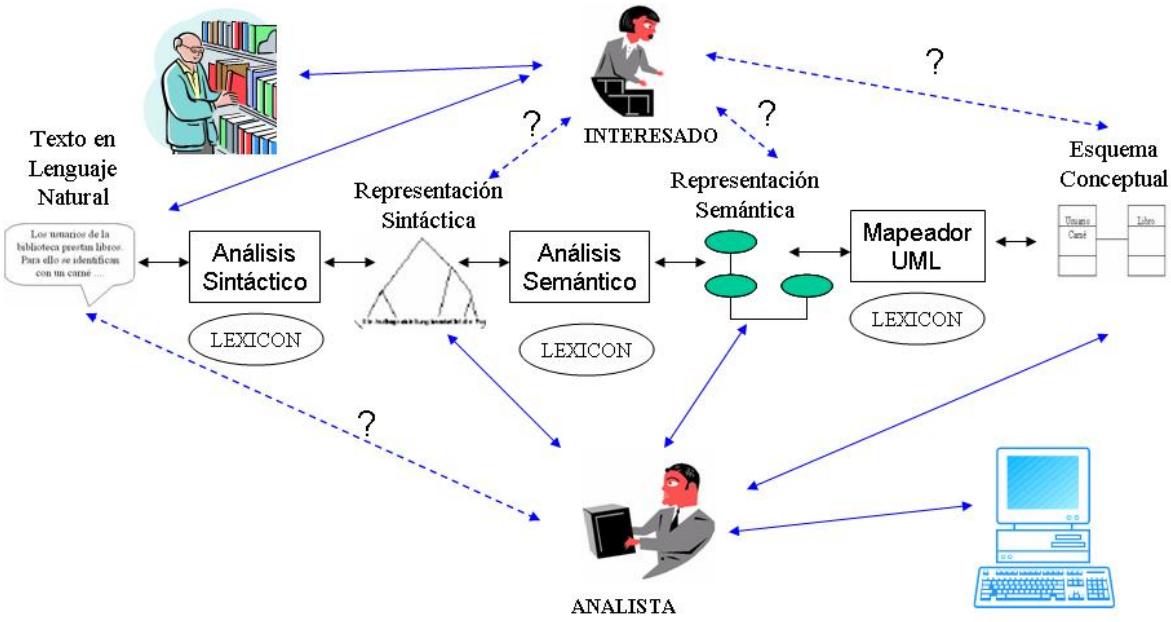


Figura 3. Procesamiento del Lenguaje Natural para la obtención de esquemas conceptuales a partir de textos en lenguaje natural. En línea discontinua y con una interrogación se señalan los aspectos que presentan incertidumbre para cada uno de los actores.

Los subprocesos intermedios que se suelen realizan son los siguientes:

2.2.1. Análisis Sintáctico:

En esta fase, el texto en lenguaje natural se transforma en las diferentes representaciones sintácticas que se pueden derivar, con el fin de determinar la categoría gramatical de cada palabra en cada frase del texto. La representación sintáctica puede ser un árbol de constituyentes o de dependencias, según sea el tipo de análisis sintáctico que se realice. En

todo este proceso se cuenta con el apoyo del lexicón, un recurso computacional que incluye información de tipo sintáctico y semántico; el lexicón también puede tener otros recursos opcionales como estadísticas para facilitar la desambiguación o algunas taxonomías de términos para contribuir a la identificación de elementos.

Para el análisis sintáctico, los trabajos existentes (véase el Capítulo 3) introducen el concepto de lenguaje "restringido" simplemente para obligar al usuario a hablar de la manera más inambigüa posible, permitiendo únicamente frases sencillas que posean estructuras que se puedan amoldar a sólo un árbol sintáctico. Sin embargo, en la realidad, las frases del usuario podrían amoldarse a varias representaciones sintácticas a la vez, por lo cual se requeriría un proceso de desambiguación que posibilitara la identificación (con algún grado de precisión) de la representación más próxima a lo que el usuario expresó.

Las representaciones sintácticas son resultados intermedios que se usan como insumos en fases siguientes del proceso.

2.2.2. Análisis Semántico:

El proceso continúa con la determinación del papel semántico que juega cada palabra que acompaña al verbo en la frase (Por ejemplo, el agente, el experimentador o paciente, la localización, el tema, etc.), el cual se determina con base en la gramática de Casos enunciada por Fillmore (1968). De allí se continúa con la elaboración de un esquema intermedio que el proyecto NIBA (que se describe en el próximo capítulo) denomina "prediseño conceptual" (Fliedl et al., 2002) para significar que se realiza de manera previa al conceptual; este esquema posibilita la transición de los elementos identificados de manera sintáctica y semántica hacia los diagramas conceptuales conocidos, con la ayuda nuevamente del lexicón. En este punto no existe un consenso metodológico, puesto que cada trabajo realizado en el área presenta diferentes propuestas de esquemas intermedios

que van desde ciertas tablas para la determinación del diagrama de clases (Kop y Mayr, 2002) hasta grafos dinámicos para el mapeo hacia los diagramas de comportamiento, como por ejemplo el diagrama de actividades (Mayr y Kop, 2002), pasando por redes semánticas en otros proyectos (Harmain y Gaizauskas, 2000, Mich, 1996).

El Análisis Semántico es útil para determinar una "interpretación" de cada palabra en la frase, que permita luego el mapeo de cada elemento en lenguaje natural a las componentes de una representación semántica y de allí a los correspondientes elementos en cada uno de los modelos UML.

2.2.3. Mapeador UML:

Posibilita la traducción de la representación semántica hacia el (los) esquema(s) conceptual(es) definido(s), particularmente los diagramas UML. Para ejecutar este paso se suelen emplear reglas heurísticas que utilizan los resultados de los análisis sintáctico y semántico, con el fin de realizar el mapeo de los elementos a sus contrapartes en los diagramas o esquemas seleccionados; además, para el reconocimiento de ciertos elementos de los esquemas conceptuales aún puede ser necesario el uso de un lexicón. Esas reglas heurísticas han sido comunes en la literatura técnica desde que Chen (1983) propusiera sus reglas para la obtención del diagrama Entidad-Relación y, posteriormente, Coad y Yourdon (1990) hicieran lo mismo para el diagrama de clases. En estos dos trabajos, las reglas heurísticas son bastante débiles, pues por lo general no logran realizar el mapeo de todos los elementos identificados en el modelo verbal hacia sus contrapartes en el modelo conceptual seleccionado y pueden presentar ambigüedad (se pueden aplicar dos o más reglas para encontrar elementos diferentes en el esquema conceptual dado).

Como resultado final, en esta fase se obtienen los esquemas conceptuales a partir de la representación semántica. Siendo el UML (Unified Modeling Language) uno de los

estándares de modelamiento de software más empleados en la actualidad, la mayoría de los trabajos estudiados procuran la identificación de tales diagramas.

Fowler (2004) define UML como una "familia de notaciones gráficas, apoyadas por un metamodelo sencillo, que ayuda en la descripción y diseño de sistemas de software, particularmente sistemas de software construidos usando el estilo orientado a objetos (OO)". La jerarquía de los diagramas de la versión UML 2.0 (OMG, 2006) se muestra en la Figura 4, tomada de Fowler (2004).

De la Figura 4, se pueden observar dos tipos de diagramas que hacen parte de la especificación de UML: los estructurales y los comportamentales; además, dentro de los comportamentales se incluyen los denominados diagramas de interacción. Con los tres tipos de diagrama, es posible construir la especificación completa de una pieza de software empleando UML.

Dado el carácter semiformal de los esquemas conceptuales de UML, es posible, mediante su uso, "precisar" los requisitos de los interesados durante el proceso de elicitación de requisitos. En el Capítulo 4 se presentan los principales diagramas UML con los cuales se trabajará en esta Tesis y que se podrán obtener a partir del Esquema Preconceptual.

Con este marco conceptual, se puede proceder a analizar el Estado del Arte, que se incluye en el capítulo siguiente.

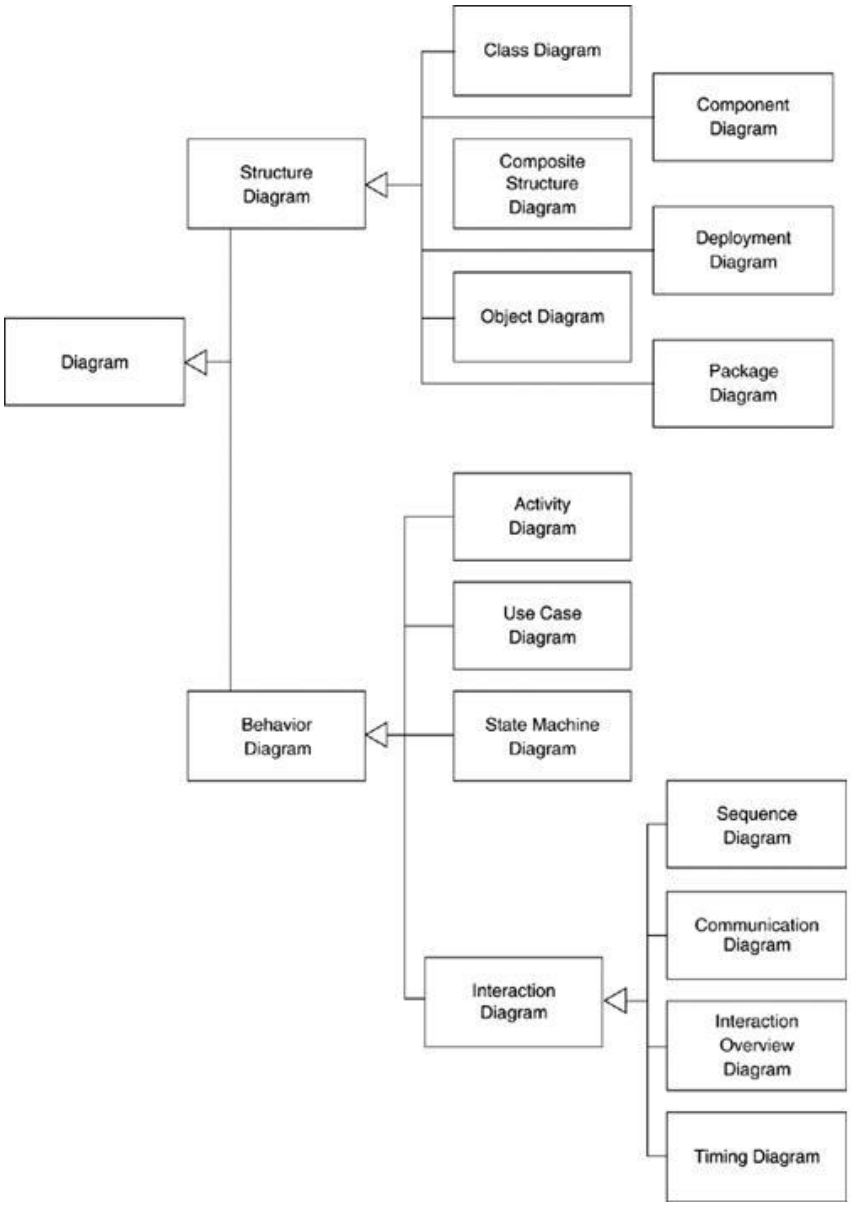


Figura 4. Jerarquía de Diagramas de la Superestructura de UML 2.0

3. ESTADO DEL ARTE:

3.1. Análisis Descriptivo:

3.1.1. Obtención Automática de Esquemas Conceptuales:

El proyecto LIDA (LInguistic assisstant for Domain Analysis; Overmyer, 2001) presenta una herramienta semiautomática que permite el ingreso de especificaciones textuales en lenguaje natural, tal como se muestra en la parte derecha de la Figura 5. Con base en el texto ingresado, la herramienta realiza un reconocimiento de las palabras y las clasifica en verbos, sustantivos y adjetivos, desconociendo todas las demás palabras que hacen parte de la sintaxis de las frases y que podrían suministrar pistas en relación con la forma de conversión a los diferentes esquemas conceptuales. Adicionalmente, la herramienta realiza un conteo de las frecuencias de cada palabra, de forma que el analista use la información que entrega el sistema para asignar un tipo de elemento a cada palabra mediante un código de colores, como se muestra en la Figura 5. El esquema conceptual de destino es el diagrama de clases de UML, por lo cual los posibles elementos para asignar a cada palabra son: clases, atributos, operaciones o roles. Una vez el analista ha asignado a cada palabra identificada uno de los elementos del esquema conceptual, la herramienta le suministra la posibilidad de trazar gráficamente el diagrama empleando los elementos identificados y de hacerle las correcciones que considere pertinentes, como se haría en una herramienta CASE (Computer-Aided Software Engineering) convencional para este proceso; incluso, la herramienta posee un módulo de exportación que posibilita el traslado del diagrama conseguido a una herramienta CASE para su culminación. La interfaz correspondiente al asistente gráfico se muestra en la Figura 6. Esta herramienta se puede considerar semiautomática puesto que requiere de una alta participación del analista en el proceso de creación del diagrama de clases, además de brindar posibilidades computacionales que apoyan ese proceso.

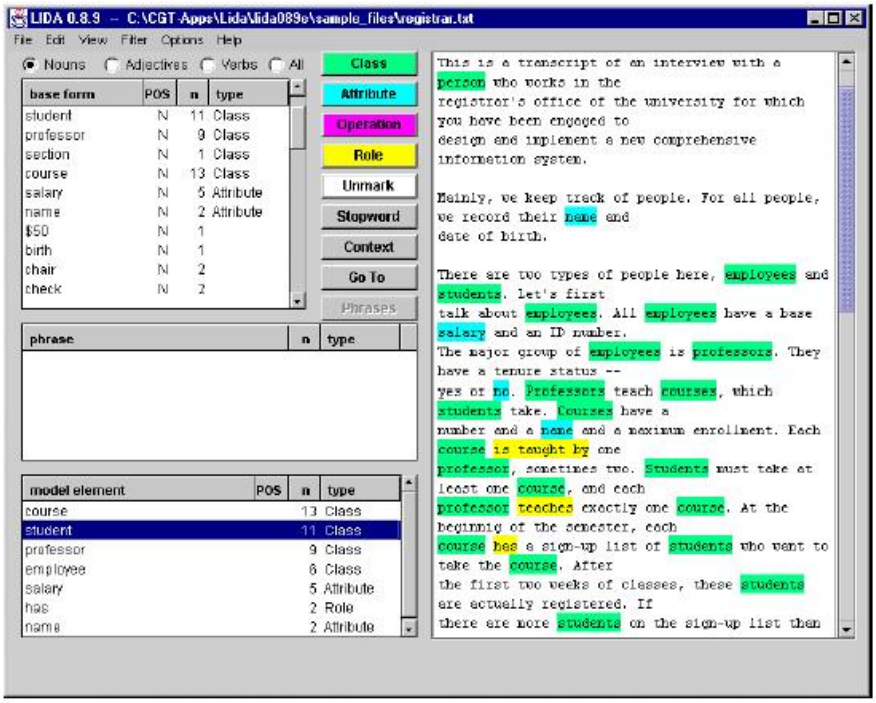


Figura 5. Imagen de la interfaz del proyecto LIDA, en la cual se aprecia a la izquierda la clasificación y conteo de las palabras del texto, en el centro los botones de asignación y a la derecha el texto resaltando la asignación por colores realizada por el analista para cada frase.

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 et al., 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 unos árboles gramaticales que clasifican las palabras en diferentes categorías, denominadas "sintagmas".

Posteriormente, RADD utiliza un modelo lingüístico que identifica el significado de una frase mediante los denominados 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; estos roles son similares a los definidos por Fillmore (1968). Finalmente, 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, utilizando otros elementos del análisis sintáctico (como los determinantes que acompañan a los sustantivos, por ejemplo).

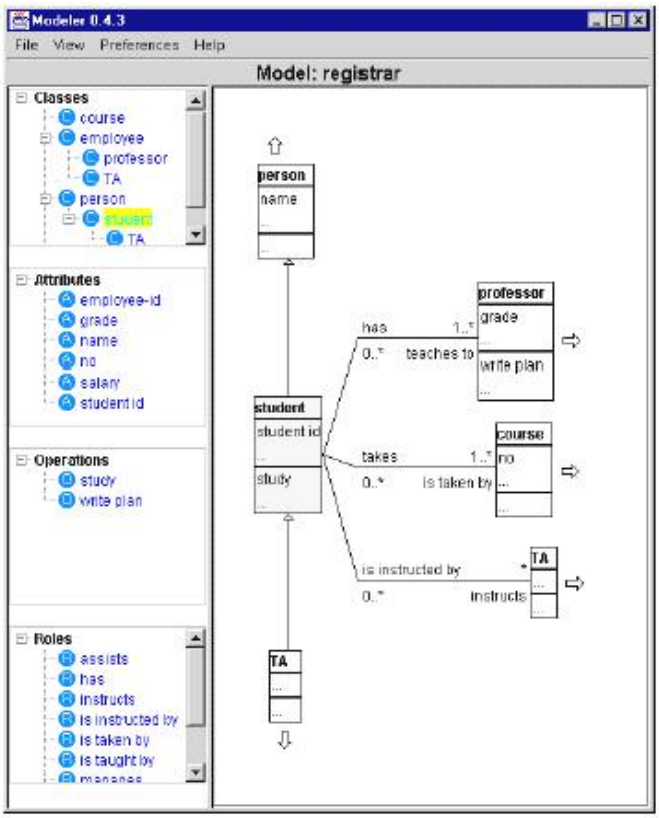


Figura 6. Imagen del asistente gráfico de la herramienta LIDA. A la izquierda se aprecian las clases, atributos, operaciones y roles según como los clasificó el analista; a la derecha se aprecia el diagrama ya realizado, también según la decisión del analista.

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, utilizando un tipo de análisis que ellos denominan “pragmático”. 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. En la Figura 7 se muestra gráficamente el proceso de obtención del esquema conceptual seleccionado por RADD (el modelo entidad-relación).

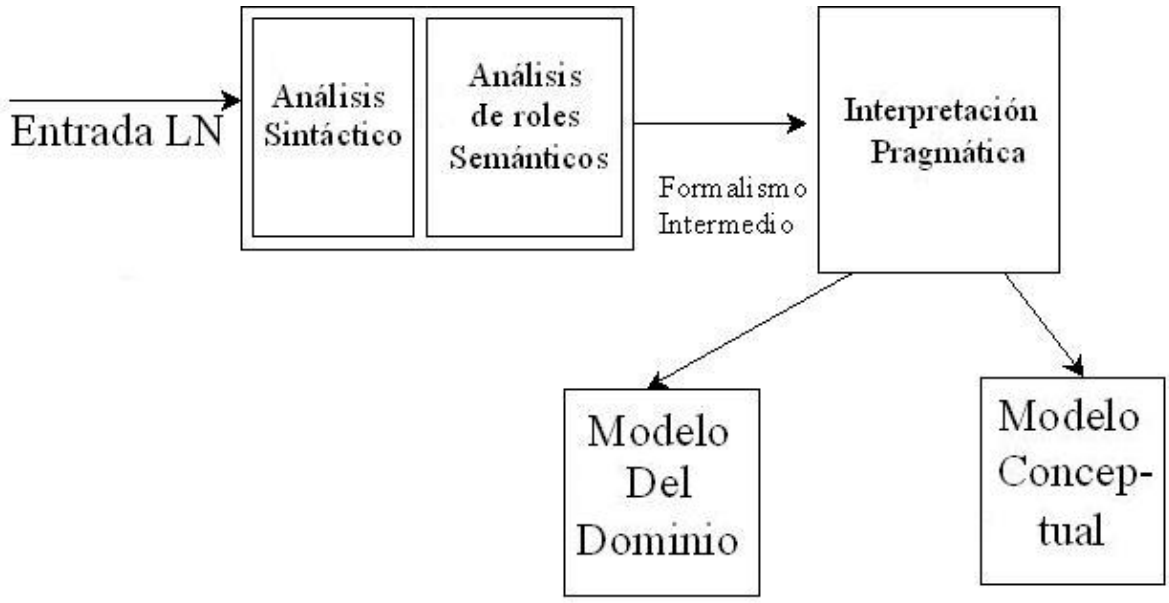


Figura 7. Proceso de obtención del diagrama entidad-relación para el proyecto RADD.

Un trabajo similar se realiza en la herramienta ER-Converter Tool (Omar *et al.*, 2004), que busca la construcción automática del diagrama entidad-relación a partir de textos en lenguaje aparentemente natural, pero que en los ejemplos resulta ser un poco controlado. En este trabajo se realiza el análisis sintáctico del texto combinado con un conjunto de reglas heurísticas que posibilitan la generación del diagrama correspondiente, para finalmente entregar una versión preliminar que sea refinada con la intervención del analista.

Díaz *et al.* (2004), han presentado recientemente una propuesta que pretende la obtención automática del diagrama de secuencias a partir de la descripción textual de un caso de uso,

que ha sido escrita siguiendo un esquema reglado previamente establecido, según el proceso que se muestra en la Figura 8. Esta propuesta es similar a RADD porque debe también realizar un análisis sintáctico cuidadoso y una especie de análisis semántico; el análisis semántico procura equiparar cada frase del texto con un patrón definido, que toma en consideración ciertas palabras clave, las categorías gramaticales y los sintagmas presentes en la frase, para determinar de qué manera se podrán incorporar al diagrama de secuencias. Por ejemplo, un Sintagma Preposicional de Pertenencia tiene un patrón como el siguiente: {'de' | 'del'} [determinante][adjetivo]sustantivo[adjetivo]; si en la descripción del caso de uso se encuentra una forma que equipare este patrón, se registra en una tabla como la que se muestra en la Figura 9; en dicha tabla se identifican instancias y parámetros utilizando varias combinaciones de patrones, como el ejemplificado, para determinar la forma de traducción al diagrama de secuencias. Esta propuesta emplea recursos léxicos muy estructurados, porque aquí se usa un lexicón que debe identificar todas las palabras presentes y los diferentes tipos de sintagmas que se presentan en un determinado patrón.

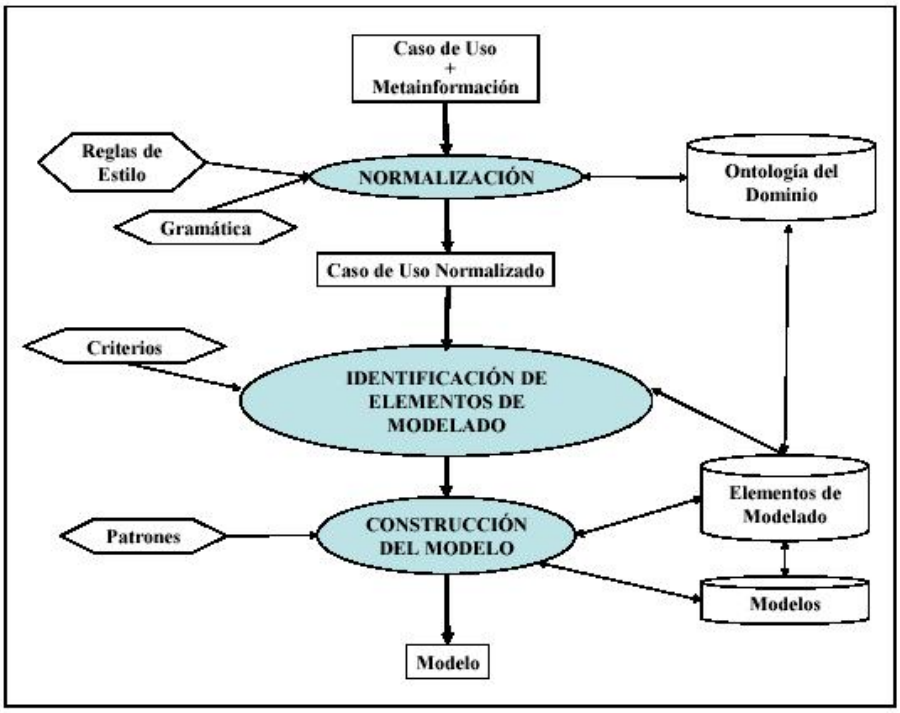


Figura 8. Proceso de la herramienta propuesta por Díaz et al. (2004)

		INSTANCIAS		PARÁMETROS	
PASO	PATRÓN	NOMBRE	CRITERIO	NOMBRE	CRITERIO
0	Actor-Sistema	Venta	SPP	Sesión	SNS
1	Sistema-Instancia	Venta	SPP	Sesión	SNEx
2	Sistema-Actor	Artículo	SPP	Código de Barras	SNE – SNInDet
				Cantidad Vendida	SNE - SNEs
3	Actor-Sistema	Artículo	SPP	Código de Barras	SNE – SNInDet
				Cantidad Vendida	SNE - SNEs
4	Sistema-Instancia	Artículo	SPP	Descripción	SNE
				Precio por Unidad	SNE – SNInDet
5	Sistema-Instancia	Venta	SPP	Monto	SNS
		Artículo	SPP		
6	Sistema-Instancia	Venta	SPP	Subtotal	SNS
7	Sistema-Actor	Venta	SPP	Subtotal	SNS

Figura 9. Fragmento de un análisis realizado mediante la herramienta propuesta por Díaz *et al.* (2004). Las columnas denominadas "criterio" tienen que ver con diferentes patrones identificados que combinan palabras clave, categorías gramaticales y sintagmas de diferente tipo.

Otro proyecto que trata de realizar la conversión de especificaciones textuales en esquemas conceptuales (particularmente en el diagrama de clases de UML) es CM-BUILDER (Conceptual Model Builder; Harmain y Gaizauskas, 2000). Este convertidor a diagramas de clases utiliza una jerarquía que agrupa en tres categorías los diferentes elementos y conceptos del mundo: objetos, eventos y atributos; a esa jerarquía, de la cual se puede visualizar un ejemplo en la Figura 10, la llaman "esquema del mundo" y requiere unos recursos léxicos que son difíciles de lograr en los lexicones existentes, puesto que la clasificación de los conceptos del mundo requiere el uso de una ontología previa del dominio, la cual no siempre está disponible. Para el ejemplo de la Figura 10, la ontología deberá saber que los conceptos "profesor" y "estudiante" se agrupan bajo la categoría de "cliente" y no bajo otra que no se haya considerado en esa ontología (como por ejemplo "persona"); esa agrupación obedece a que el ejemplo que se está manejando es una editorial. Además, en ningún lexicón común se puede encontrar la categorización específica en objetos, eventos y atributos que se plantea en el primer nivel de la jerarquía, la cual es claramente conducente a la elaboración del diagrama de clases; un análisis sintáctico

basado en el lexicón podría decir la categoría gramatical de las palabras o la manera de conformación de los diferentes sintagmas que hacen parte de la oración, pero una clasificación como la planteada requiere el uso de un conocimiento previo sobre el dominio. Además, una palabra no siempre tendrá la misma representación en diferentes modelos; por ejemplo, el teléfono será por lo general un atributo del objeto persona, pero tendrá la posibilidad de ser en sí mismo un objeto del mundo si se trata de la construcción de software para un call center. Esa información sería muy difícil de involucrar en los lexicones debido a que, dependiendo del contexto, podría variar sustancialmente de una categoría a otra.

A partir del "esquema del mundo", CM-BUILDER utiliza reglas de mapeo bastante potentes, pues emplea, como en LIDA, la frecuencia de aparición de las palabras como un indicio de clases del modelo; además, hace distinciones entre los diferentes tipos de verbos para identificar los diferentes tipos de relaciones presentes, define categorías asociadas con los adjetivos para descubrir los nombres de los atributos de una clase e identifica los determinantes para definir la multiplicidad en los extremos de una asociación.

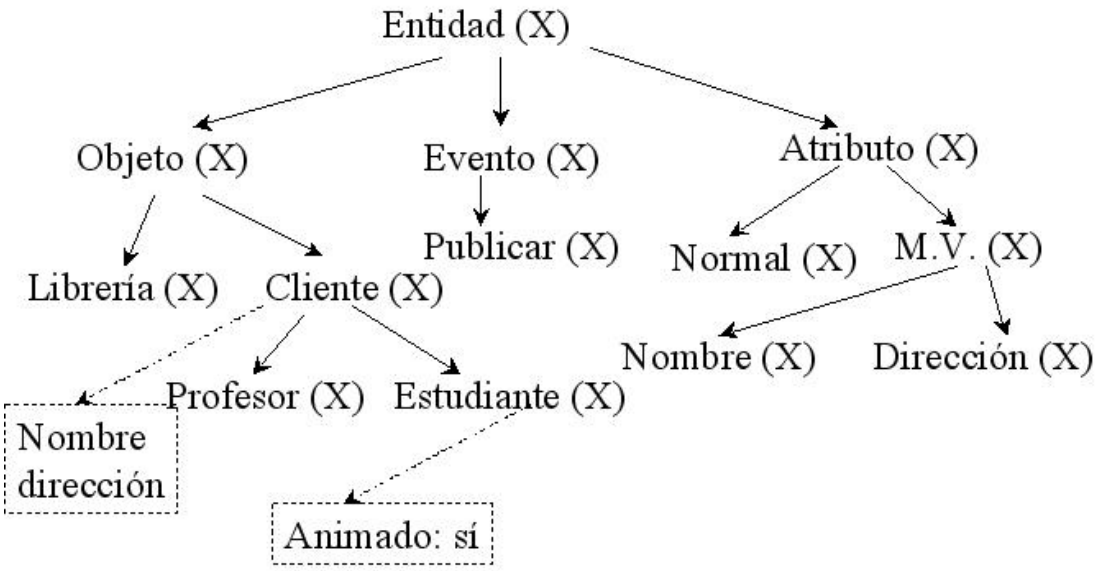


Figura 10. "Esquema del mundo" utilizado por el proyecto CM-BUILDER.

Otro de los proyectos que realiza la transformación al diagrama de clases de UML a partir de lenguaje natural es el NL-OOPS (Natural Language—Object-Oriented Product System), un sistema desarrollado por Mich que se basa en el 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 la fase de análisis (Mich, 1996). Para obtener los esquemas conceptuales, NL-OOPS emplea una especie de red semántica llamada SemNet, de la cual se puede apreciar un ejemplo en la Figura 11; allí, se observan dos tipos de nodos: los de eventos (marcados con la palabra event), que pueden funcionar como una red, y los de entidad (en la Figura "clear", "transaction", "computer" y "Roberto"), que sólo pueden funcionar de manera jerárquica. Los enlaces tienen el nombre de la relación entre nodos, una especie de clasificación similar a los roles semánticos de Fillmore que cualifica la relación entre los nodos. Para el ejemplo de la Figura, el evento sería algo como "Roberto ejecuta una transacción de borrado en el computador". El número asociado con el nodo permite el ingreso a una información complementaria, que consiste, para los nodos de eventos, en una categoría que depende del tipo de evento (estático, cíclico, dinámico o instantáneo) y, para los nodos de entidad, en el rango (que se refiere a una cuantificación, es decir si la entidad es individual o universal) y en la familia (viviente, humano, organización humana, inanimado o hecho por el hombre).

De manera similar a CM-BUILDER, NL-OOPS emplea reglas de mapeo desde SemNet hasta el diagrama de clases. Sin embargo estas reglas son bastante simples pues carecen de universalidad, lo que las convierte más en propuestas que se deben seleccionar en un momento dado, que en reglas como tal. Por ello, NL-OOPS entrega como resultado un conjunto de las clases candidatas y las posibles instancias, atributos y métodos de las mismas, sin una certeza de que sean las más representativas para el diagrama de clases que pretende obtener.

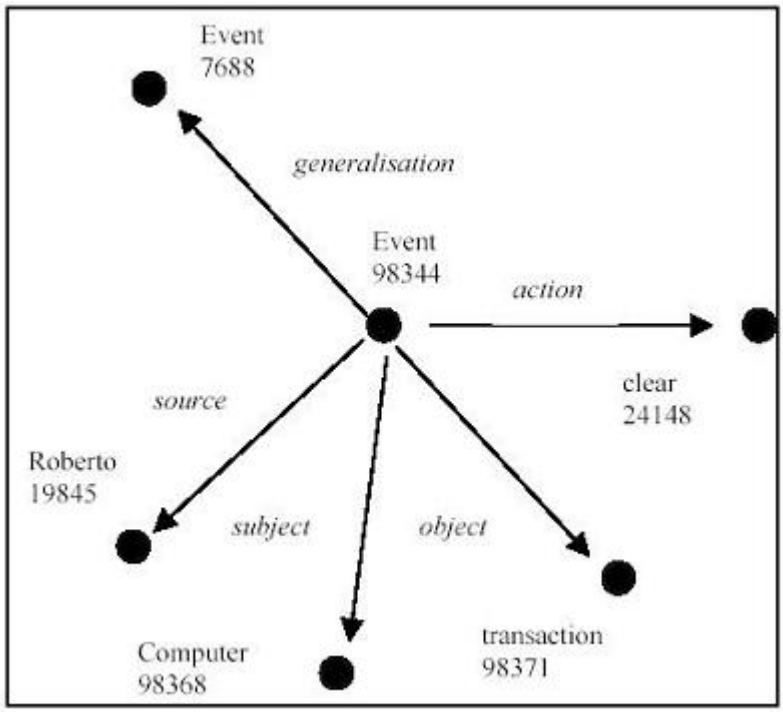


Figura 11. Imagen de SemNet, la red semántica utilizada por NL-OOPS.

Este análisis sirve como punto de partida para la elaboración del diagrama de clases, pero el analista tiene un alto grado de participación a partir de las salidas de NL-OOPS, sobre todo en la concreción de los elementos candidatos sugeridos por la herramienta.

En el dominio particular del desarrollo de Sistemas Digitales, se encuentra ASPIN (Automatic Specification Interpreter; Cyre, 1995), una aplicación de software para la interpretación de diferentes modelos relativos al diseño de este tipo de sistemas, tales como Diagramas de Bloques, Diagramas de Flujo de Datos, Estructuras de Datos, Diagramas de Flujo Estándar, Diagramas de Estado y Diagramas de Tiempo, y de especificaciones en un lenguaje natural restringido a términos aplicables únicamente a ese dominio particular. El objetivo de ASPIN es tomar todas estas especificaciones y llevarlas a un lenguaje común para realizar validaciones en relación con la completitud y consistencia de los diferentes esquemas para la realización del sistema digital. En ese sentido, el trabajo de Cyre realiza el

proceso contrario a esta Tesis, puesto tiene como insumo lo que para esta Tesis es uno de los resultados: los esquemas conceptuales respectivos; sin embargo, posee similitudes con esta Tesis en tanto que utiliza las descripciones de especificaciones en lenguaje natural y las traduce a un esquema intermedio que se basa en la teoría de Grafos Conceptuales (Sowa, 1984). Los grafos de Sowa tuvieron su origen en las redes semánticas utilizadas por la Inteligencia Artificial (o similares al SemNet que utiliza NL-OOPS), agregando elementos de los grafos existenciales de Pierce (1931-58) y los roles semánticos definidos por Fillmore (1968). Para la conversión de las especificaciones textuales en lenguaje natural a grafos conceptuales, 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 de los Sistemas Digitales, y luego un análisis semántico que utiliza la teoría de los roles semánticos. Un ejemplo particular de lo anterior es la frase "un programa es ejecutado por el procesador", para la cual se obtiene la equivalencia en grafos conceptuales que se muestra en la Figura 12 en dos de sus representaciones (textual y gráfica)

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

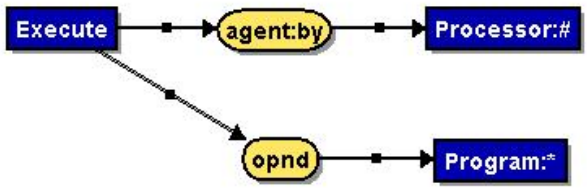


Figura 12. Representaciones textual y gráfica en grafos conceptuales de la frase "Un programa es ejecutado por el procesador".

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. Para los demás modelos, ASPIN define una serie de equivalencias a los grafos conceptuales, dependiendo de la forma de los diagramas. El principal inconveniente de esta representación consiste en que está limitada al ámbito de los sistemas digitales y por ello

sus reglas de transformación carecen de la generalidad necesaria; además, lo que se pretende en esta tesis es partir de esquemas intermedios (entre lenguaje natural y esquemas conceptuales) para llegar a esquemas conceptuales y no a la inversa como se realiza en ASPIN. Sin embargo, este trabajo es un buen punto de partida, porque demuestra que los grafos conceptuales de Sowa bien podrían describir varios tipos de esquemas conceptuales simultáneamente.

Entre los trabajos que buscan la generación automática de más de un diagrama a partir de lenguaje natural o controlado se encuentran: el trabajo de Juristo *et al.* (1999), KISS (Burg y Van de Riet, 1996) y NIBA (Fliedl *et al.*, 1999).

En el caso de Juristo *et al.* (1999), se procura la generación asistida por el analista del diagrama de objetos en notación OMT y el diagrama de comportamiento de Martin. Para ello, los autores segmentan el discurso en lenguaje natural controlado en dos tipos de expresiones: estáticos y dinámicos. Posteriormente, aplican un conjunto de heurísticas a cada conjunto de expresiones para obtener el diagrama de objetos (enunciado estático) y el diagrama de comportamiento (enunciado dinámico). En este caso, tampoco existe una representación intermedia y el análisis se realiza más desde el punto de vista sintáctico y semántico.

El trabajo que se realiza en el proyecto KISS (Burg y Van de Riet, 1996) es similar al anterior, puesto que se usa el análisis sintáctico y semántico para obtener los denominados diagramas CPL (Concept prototyping language) y porque se parte también de una forma de lenguaje controlado. Otra similitud es que los autores hacen una partición del texto, pero no en parte estática y dinámica, sino en estructura principal (sujeto, objeto y complemento directo) y elementos gramaticales especiales (adjetivos, adverbios, predicados nominales, etc.), con el fin de aplicar un conjunto de heurísticas especiales que permiten obtener los diagramas CPL. Tampoco se emplea en este caso una representación intermedia.

En la Universidad de Klagenfurt (Austria), los grupos de Lingüística y Sistemas de Información realizaron un proyecto conjunto llamado NIBA (Natürlichsprachliche InformationsBedarfs-Analyse, Análisis de Requisitos de Información en Lenguaje Natural), que pretende, a partir de un conjunto de frases especificadas en un lenguaje natural restringido (usan este término para incluir sólo frases con estructura sencilla y excluir frases de tipo imperativo o interrogativo, por ejemplo), realizar el trazado de diferentes esquemas conceptuales, particularmente el diagrama de clases y el diagrama de actividades, aunque plantean que sería posible derivar también otros esquemas comportamentales como el diagrama de secuencias o el de comunicación (Fliedl *et al.*, 1999, 1999a, 2002, 2002a y 2003; Kop y Mayr, 2002; Mayr y Kop, 2002). En este proyecto, el proceso es bastante completo pues, a partir del texto en lenguaje natural, realizan una serie de procesos adicionales como el análisis sintáctico, el análisis semántico y la aplicación de unas reglas de mapeo a los esquemas conceptuales definidos. El grupo de lingüistas tiene una alta participación en las fases iniciales del proceso, donde se define un esquema lingüístico denominado NTMS (Natürlichkeitstheoretische Morphosyntax—Morfosintaxis teórica del Lenguaje Natural) que permite la realización del análisis sintáctico para obtener los árboles sintácticos correspondientes, como se muestra en la Figura 13. El siguiente paso del análisis depende del tipo de diagrama que se quiera obtener, tomando como punto de partida los árboles sintácticos trazados. En general, el proyecto NIBA emplea una herramienta llamada KCPM (Klagenfurt Conceptual Predesign Model), que se encarga de elaborar unos esquemas necesarios para la conversión de cada uno de los diagramas destino, que en este caso son diagramas UML. Estos esquemas que se elaboran varían sustancialmente para diagramas estructurales y comportamentales; en el caso de diagramas estructurales, se trata de un conjunto de tablas que representan los diferentes conceptos del mundo, agrupados en lo que ellos llaman “tipo cosa” (correspondiente a los sustantivos del análisis sintáctico, y que pueden convertirse luego en clases o atributos, tomando como base las reglas heurísticas que definen) y “tipo conexión” (los verbos del análisis sintáctico, los cuales se

convierten principalmente en asociaciones). En las Figuras 14 y 15 se pueden apreciar las características de las tablas correspondientes al esquema para los dos tipos de elementos.

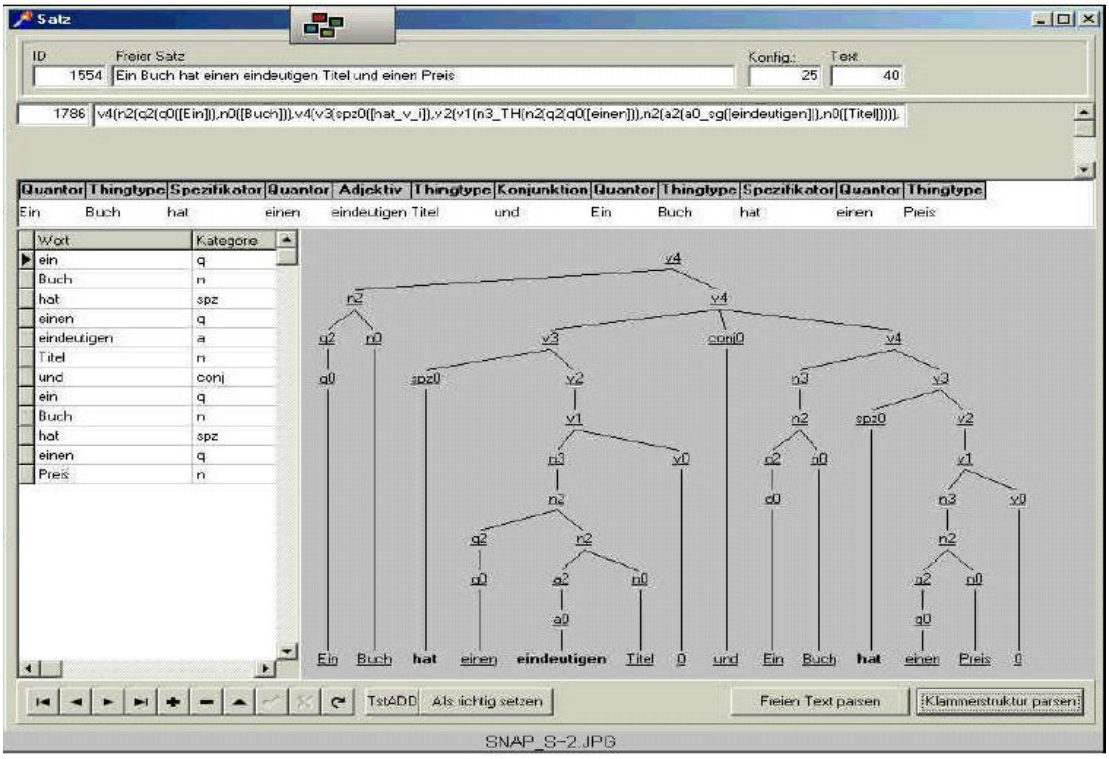


Figura 13. Resultado del análisis sintáctico realizado mediante el proyecto NIBA.

En el caso de los esquemas comportamentales, la herramienta genera un esquema gráfico como el que se muestra en la Figura 16, en la cual los círculos representan pre o postcondiciones, los óvalos representan tipos de operación y los conjuntos de círculos, flechas y rectángulos representan tipos de cooperación. Estos elementos se pueden mapear posteriormente al diagrama de actividades.

Con base en los esquemas tabulares o gráficos definidos, la conversión a los diferentes esquemas conceptuales de UML se logra mediante la definición y aplicación de un conjunto de reglas heurísticas de transformación, que realizan el mapeo de los diferentes elementos del esquema construido a los correspondientes en el esquema conceptual elegido.

UoD-area: publishing **UoD:** publishing companies **Project:** P5

id#	name	classifi-cation	quantity-description	examples	value domain	synonym	descri-ption	requirement source
D001	author	thing-type		e.g. „.. T. Mann“				S1, S2,
D002	book	thing-type	500					S1, S2, S3
D004	ISBN number	thing-type						S3
	...							

Figura 14. Ejemplo de una tabla para los “tipos cosa” en la herramienta KCPM.

UoD-area: publishing **UoD:** publishing companies **Project:** P5

c-id#	name	connection type determiner	perspective					req. source
			pers-pective#	involved thing-type	name	min/ max	persp-det.	
C001	write/ is_written		p001a	D001, author	write			S1, S2
			p001b	D002, book	is_written			
C002	identification		p002a	D004, ISBN number	identifies			S3
			p002b	D002, book	is_identified_by			
...	...							

Figura 15. Ejemplo de una tabla para los “tipos conexión” en la herramienta KCPM.

De los proyectos revisados, éste es el más completo en cuanto a los diagramas que obtiene desde las especificaciones textuales y en cuanto a los esquemas intermedios que emplea, puesto que posee una fundamentación lingüística importante y porque define una serie de recursos léxicos que contribuyen a la adecuada realización de la traducción (el grupo de Lingüística, por ejemplo, debió realizar la clasificación de casi 10.000 verbos en alemán en 12 categorías y 7 subcategorías definidas por ellos para poder realizar la construcción automática de los diferentes esquemas intermedios). Adicionalmente, emplean algunas reglas que se pueden considerar como leyes generales de conversión, en tanto que otras, y así lo dicen explícitamente, se presentan a nivel de propuesta y sirven para suministrar opciones de traducción que posteriormente el analista deberá discernir. A pesar de su

completitud, el proyecto aún posee ciertas fallas: en la elaboración de los diferentes esquemas intermedios se elimina cierta información que no es necesaria para el tipo de diagrama hacia el cual van dirigidos y que podría ser relevante para la conversión a otros diagramas (por ejemplo, el diagrama de clases no requiere la identificación de las pre y postcondiciones, puesto que es un diagrama estructural y por ello las tablas KCPM no guardan esta información); esto finalmente podría ocasionar problemas de consistencia entre los diferentes diagramas que se pueden obtener con la herramienta del proyecto NIBA. Además, los esquemas KCPM enfocados al diagrama de clases no guardan información respecto de las características de los tipos conexión de forma tal que se pudiera identificar si son herencias, agregaciones, composiciones o incluso operaciones de una clase.

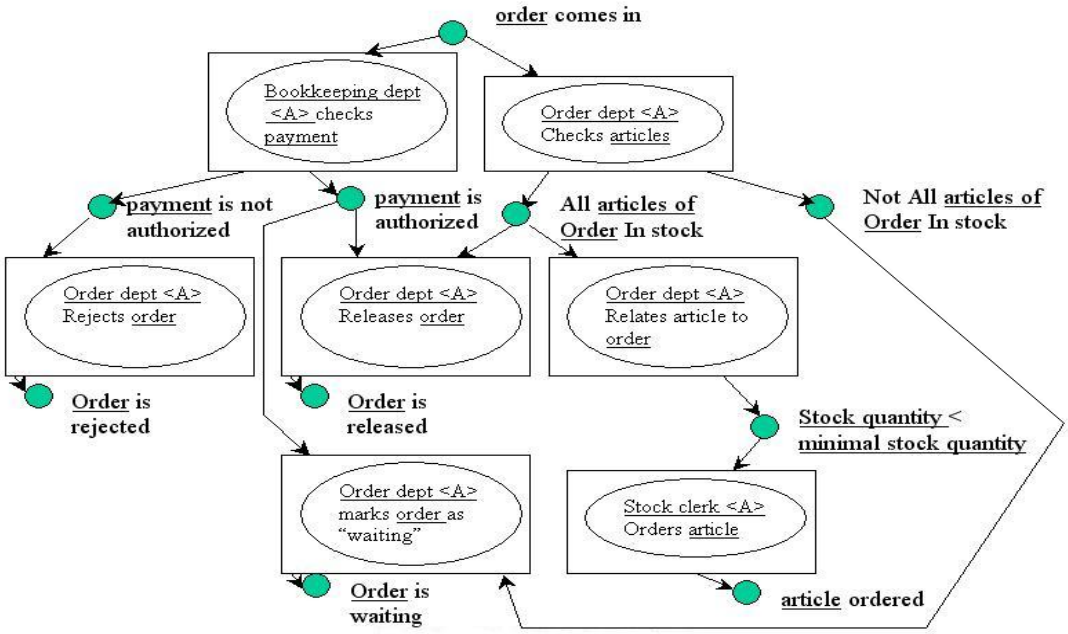


Figura 16. Ejemplo de un esquema gráfico generado con la herramienta KCPM para traducción a los esquemas comportamentales.

3.1.2. Traducción de Máquina:

Los proyectos que procuran obtención de esquemas conceptuales a partir de especificaciones textuales en lenguaje natural son relativamente recientes. Si bien las ideas iniciales datan de mediados de la década de los 80, sólo hasta finales de la década de los 90 se comenzaron a realizar trabajos al respecto. Sin embargo, existe un antecedente a este tipo de trabajos que data de la década de los 50 y puede exhibir el camino de su evolución: el Procesamiento del Lenguaje Natural (PLN) y una de sus áreas principales, como es la Traducción de Máquina (TM), la cual "se refiere a sistemas computarizados responsables por la producción de traducciones (... entre diferentes lenguajes...) con o sin la asistencia de humanos" (Hutchins, 2003).

La TM se inició en la década de los 50, cuando se comenzaron a plantear maneras de realizar la traducción entre diferentes lenguajes, y se pensaba que se podía hacer mediante el uso de computadores. Existen tres técnicas que se han aplicado desde entonces, con resultados diversos, para la Traducción de Máquina: Traducción directa o binaria, que se realiza previa al nivel morfológico, la Transferencia, que se realiza a nivel semántico y la Interlingua, que es la tendencia más reciente, y que se realiza a nivel metasemántico. En la Figura 17 se puede apreciar una gráfica comparativa de los niveles de actuación de cada una de las tendencias de Traducción de Máquina.

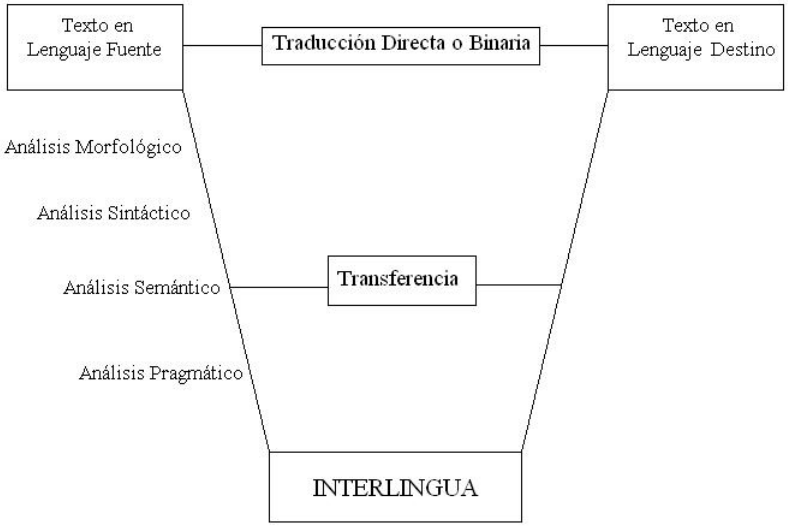


Figura 17. Esquema general de TM ubicando los diferentes enfoques para su realización.

Se puede notar la similitud de los procesos correspondientes a la Traducción de Máquina con los procesos esbozados en la Figura 3. En el nivel semántico se suelen emplear varias representaciones que podrían ser similares a los Esquemas Preconceptuales que se plantean en esta propuesta de Tesis.

Hausser (1999) suministra dos definiciones útiles de semántica cuando afirma: "en lingüística, semántica es un componente de la gramática que deriva representaciones de sentidos a partir de superficies naturales sintácticamente analizadas. (…) En ciencias de la computación, semántica consiste en la ejecución de comandos en un lenguaje de programación de manera automática como operaciones de máquina". Ambos conceptos se unen en la semántica computacional, cuyo fin, según Blackburn y Bos (2003) es "encontrar técnicas para la construcción automática de representaciones semánticas del lenguaje humano, representaciones que pueden ser usadas para realizar inferencias". Así, una representación semántica deberá consignar los significados lingüísticos mediante un conjunto de estrategias que pueden ser diversas, pero que por lo general incluyen

predicados lógicos y los roles desempeñados por las diferentes palabras en el contexto de la oración.

Un formalismo inicial de representación semántica es provisto por la lógica de predicados de primer orden (LPPO), definida por Mitkov (2003) como una "lógica cuyas expresiones incluyen: (i) constantes lógicas para conectores de sentencias, (ii) cuantificadores cuyos dominios son conjuntos o individuos, (iii) constantes individuales (nombres de individuos) y (iv) constantes de predicado que corresponden a propiedades y relaciones de individuos". Este es un formalismo básico de representación semántica, pues, como se verá más adelante, otros formalismos como el cálculo lambda, la semántica de recursión mínima y los grafos conceptuales pueden finalmente presentarse de forma equivalente en LPPO. En el último renglón de la Figura 18 se puede encontrar la representación en LPPO de la frase "A woman walks".

$$
\begin{aligned}
&((\overbrace{\lambda p.\lambda q.\exists x(p@x \wedge q@x)}^{\text{a}})@\overbrace{\lambda x.\textsc{woman}(x)}^{\text{woman}})@\overbrace{\lambda x.\textsc{walk}(x)}^{\text{walks}}) = \\
&(\lambda q.\exists x(\lambda x.\textsc{woman}(x)@x \wedge q@x)@\lambda x.\textsc{walk}(x)) = \\
&(\lambda q.\exists x(\textsc{woman}(x) \wedge q@x)@\lambda x.\textsc{walk}(x)) = \\
&\exists x(\textsc{woman}(x) \wedge \lambda x.\textsc{walk}(x)@x) = \\
&\exists x(\textsc{woman}(x) \wedge \textsc{walk}(x)).
\end{aligned}
$$

Figura 18. Transformación de λ-cálculo a lógica de predicados de primer orden para la frase "a woman walks".

Un segundo formalismo, muy cercano a la lógica de predicados de primer orden, es el denominado cálculo lambda (λ-cálculo), del cual Mitkov (2003) anota que es "un modelo universal de computación, usado ampliamente en semántica y ciencias de la computación para modelar comportamiento funcional de expresiones lingüísticas". El λ-cálculo se puede ver como una extensión de la lógica de predicados de primer orden que utiliza un operador λ para asignar valores a variables; con los valores se pueden colocar marcas sobre la fórmula para separar sus partes. En la Figura 18 se puede apreciar un ejemplo tomado de

Blackburn y Bos (2003) sobre la manera de conversión de una expresión en λ-cálculo a lógica de predicados de primer orden. Se han señalado algunas partes de la expresión inicial en λ-cálculo (a, woman y walks) para apreciar el punto de partida; además, se utiliza el símbolo @ para concatenar expresiones o para indicar en dónde hacer substituciones y por qué valores.

Un tercer formalismo a tomar en consideración es el denominado Gramática Estructural de Frases orientada por Encabezados (Head-driven Phrase Structure Grammar—HPSG) (Pollard y Sag, 1994), la cual, según Lappin (2003) "usa grafos dirigidos para representar toda la información lingüística como secuencias de características con tipos que toman valores de clases especificadas, donde esos valores pueden, en sí mismos, ser estructuras de características"; además, utilizan las denominadas matrices atributo-valor para realizar esa representación. En la Figura 19 se muestra la representación en HPSG del verbo ayudar (en inglés "help").

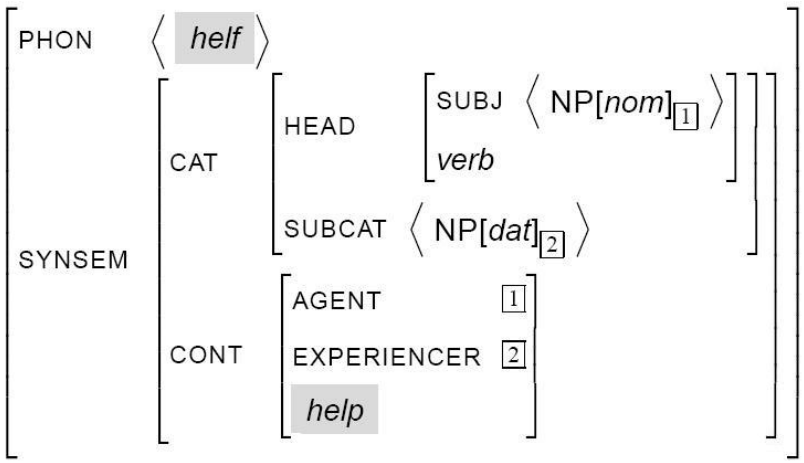


Figura 19. Matriz Atributo-Valor para la representación de HPSG del verbo inglés "help" (ayudar).

De la Figura 19, nótese que en esa matriz se consigna información fonológica, sintáctica y semántica en relación con la palabra, de la siguiente manera:

- Fonología: Helf.
- Sintaxis: El encabezado (Head) contiene un sujeto, que es una frase nominal, y el verbo ayudar. Contiene también una subcategoría que también será una frase nominal.
- Semántica: Debe tener un agente (que corresponde al sujeto del encabezado, de ahí la referencia 1) y un experimentador (que corresponde a la subcategoría, con referencia 2)

La importancia de este formalismo radica en la posibilidad de realizar operaciones con las matrices atributo-valor, de forma similar al λ-cálculo, pero registrando adicionalmente la información semántica de las matrices obtenidas para frases más complejas.

Otra ampliación de la lógica de predicados de primer orden, y que se puede representar también en HPSG, es la denominada "Semántica de Recursión Mínima" (Minimal Recursion Semantics—MRS) (Copestake et. al., 1995, 2001), la cual "en sí, no es una teoría semántica, pero puede ser más simple pensar en ella como un lenguaje de meta-nivel para la descripción de estructuras semánticas en algún lenguaje de objetos subyacente"; ese lenguaje subyacente puede ser cálculo de predicados o λ-cálculo. MRS define un conjunto de encabezados, similares a los de HPSG, que se van sustituyendo en la expresión en lógica de predicados de primer orden, aplanando la estructura hasta generar una jerarquía semántica que permite interpretar la frase correspondiente. En la Figura 20 se muestra una frase en lógica de predicados de primer orden y su transformación a MRS, tomada de (Copestake et al., 2001), donde se añade la estructura en forma de árbol con el único fin de comprender la transición de LPPO (en la parte superior) a MRS (en la parte inferior).

Un formalismo similar de representación semántica, que ha trabajado paralelamente el mismo grupo de investigación, con miras a su automatización para el PLN es la base de conocimientos léxica (Lexical Knowledge Base—LKB) (Copestake, 2002). Es más un lenguaje para la representación semántica que se utiliza en el software "Lexical Knowledge Builder".

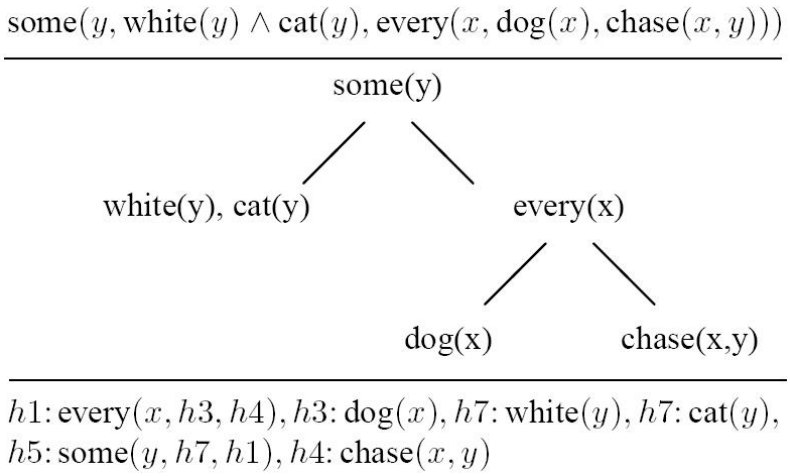


Figura 20. Representación en lógica de predicados de primer orden y su correspondiente traducción a MRS de la frase "Every dog chases some white cat".

Otro formalismo de representación semántica es la Teoría Significado-Texto (Meaning-Text Theory, MTT) de Mel'čuk y Žolkovskij (1970), que consta de un conjunto de diagramas para representar una frase a través de la fonética, la morfología (superficial y profunda), la sintaxis (superficial y profunda) y la semántica; su nombre se origina precisamente del conjunto de diagramas que se pueden realizar desde el texto hasta llegar a la interpretación del significado (semántica). La MTT utiliza diversos enfoques para la representación de los diferentes diagramas, desde la teoría de dependencias, para la sintaxis, hasta la teoría de los actuantes semánticos, una teoría similar a la de las valencias. Los formalismos gráficos que elabora son redes semánticas en las que cada nodo representa una palabra y cada arco posee diferentes etiquetas dependiendo del nivel del diagrama (relaciones sintácticas de superficie o números, por ejemplo). En la Figura 21 se pueden apreciar algunos de los diferentes diagramas realizados para la MTT, en un ejemplo particular tomado de Mel'čuk y Polguère (1987).

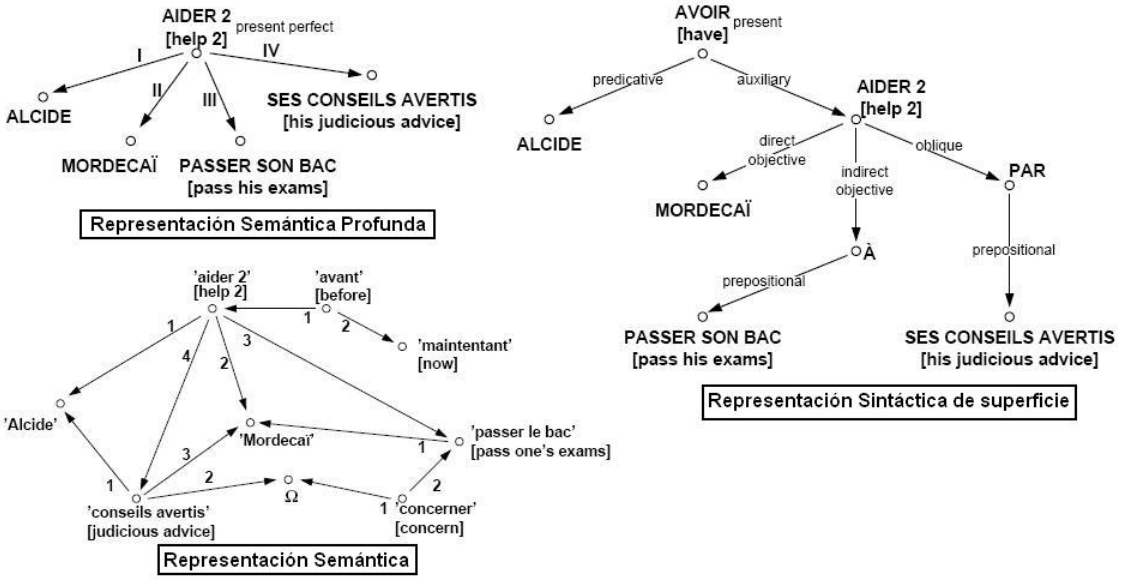


Figura 21. Tres de los diagramas de MTT para representar la frase: "Alcide a aidé Mordecaï à passer son bac par ses conseils avertis". [Alcide helped Mordecaï pass his high school leaving exams with his judicious advice.]

Un último formalismo semántico por revisar en la Traducción de Máquina lo constituyen los Grafos Conceptuales (Conceptual Graphs—CG, Sowa, 1984), que ya se habían ejemplificado en la Figura 12, correspondiente al proyecto ASPIN. Los CG son grafos dirigidos que poseen conceptos en los nodos (rectángulos) y relaciones (círculos) en los arcos, que se basan, entre otras teorías, en los casos semánticos enunciados por Fillmore (1968), que tienen ciertas similitudes con las valencias de los verbos o con la teoría de dependencias. Por su forma de representación, los CG se pueden traducir directamente a LPPO y poseen otras formas de representación para intercambio de información entre aplicaciones, particularmente el Formato de Intercambio de Conocimiento (Knowledge Interchange Format—KIF) y el Formato de Intercambio de Grafos Conceptuales (Conceptual Graph Interchange Format—CGIF).

3.2. Análisis Crítico:

3.2.1. Obtención Automática de Esquemas Conceptuales:

El Marco Conceptual definido en el capítulo anterior permite que se retome el análisis descriptivo del estado del arte para evaluar más detenidamente las diferentes propuestas, realizando además un análisis crítico de sus ventajas y desventajas en relación con la utilización de representaciones intermedias para lograr su cometido.

La construcción de los esquemas conceptuales para el desarrollo de software ha sido un proceso encargado por lo general a los analistas quienes, de forma manual o asistidos por diferentes herramientas CASE, han cumplido esta función sin la mediación de representaciones. Los primeros pasos hacia la construcción automática de esquemas conceptuales fueron dados por Chen (1983), quien definió un conjunto de reglas para la obtención del diagrama entidad-relación a partir de una descripción de un problema en lenguaje natural; Chen mismo definía esas reglas como simples propuestas, pues se basaban en un análisis sintáctico muy simple para determinar únicamente los tipos de las palabras que componían la descripción y porque se podían encontrar muchos contraejemplos para las reglas definidas. Posteriormente Coad y Yourdon (1990) definieron un método basado en un análisis sintáctico, también muy preliminar para la realización del diagrama de clases. Cuando el analista aplica directamente estas reglas o, en su defecto, cuando elabora directamente los esquemas conceptuales en las herramientas CASE disponibles, la representación intermedia está presente en la mente del analista, quien sintetiza la información consignada en la especificación en lenguaje natural y realiza internamente su análisis sintáctico y semántico.

Lo que se busca con la introducción de la representación intermedia es la automatización en la elaboración de los esquemas conceptuales, con el fin de que se pueda extraer la

información relevante del problema que permita el trazado de los esquemas conceptuales; en ausencia de estas representaciones, como es el caso de LIDA, RADD, ER-Converter Tool, Juristo *et al.* y KISS, el proceso, o bien no se ejecuta de manera automática y requiere una alta participación del analista (convirtiendo de paso estos productos en herramientas CASE avanzadas más que en generadores automáticos de esquemas conceptuales), o bien se reemplaza la representación intermedia con una combinación de recursos lingüísticos muy robustos con reglas supremamente complejas. Si bien la representación intermedia es un paso adicional para el proceso de conversión, desde el punto de vista computacional representa un producto que suministra el punto de partida para una transición más suave hacia el esquema conceptual, y que puede eliminar o minimizar la participación del analista en dicho proceso.

Particularmente en LIDA, como las decisiones para la obtención del esquema conceptual que emplea (el diagrama de clases) las toma el analista según su criterio personal, la herramienta no utiliza ninguna forma de análisis semántico y, consecuentemente, tampoco define una representación intermedia que permita la automatización de dichas decisiones. En esta herramienta se puede apreciar cómo la carencia de esa representación o, en su defecto, de un análisis semántico cuidadoso, genera ciertas desventajas, tales como:

- Impide la realización automática de la asignación de los elementos finales a cada palabra dentro del diagrama de clases (clase, atributo, operación o rol).
- Desecha gran cantidad de palabras que podrían dar indicios para la asignación de elementos correspondientes a otros tipos de diagramas (mensajes, objetos, estados, acciones, etc.) o incluso del mismo diagrama de clases (herencias, agregaciones, composiciones, cardinalidades, etc.).

Como aspecto positivo se resalta el conteo que realiza de cada palabra para determinar su frecuencia de aparición. Este elemento, como se verá en los siguientes trabajos, poco se

toma en cuenta, pero puede suministrar algunas pistas en relación con la forma de asignación de cada palabra al esquema conceptual definitivo. Además, por la forma semiautomática de obtención del esquema conceptual, la estructura requerida para recursos léxicos es mínima, puesto que lo único que se necesita saber de cada palabra es la categoría gramatical, eliminando por ejemplo de la lista de palabras del lexicón los adverbios, preposiciones, artículos, etc. También se discutirá más adelante cómo el grado de automatización que se quiera lograr en el proceso irá demandando estructuras mucho mayores en los recursos léxicos, necesarios para la elaboración de las diferentes representaciones intermedias.

Ahora, si bien en RADD, ER-Converter Tool, Juristo *et al.* y KISS no se presenta de manera explícita una representación que pueda considerarse intermedia, sí se realizan los análisis sintáctico y semántico. Pese a ello, estos proyectos aún presentan algunas dificultades, tales como:

- RADD y ER-Converter Tool sólo generan el diagrama entidad-relación, lo cual justifica la ausencia de una representación intermedia como tal, reemplazándola con el análisis de los pocos roles semánticos necesarios para la generación de ese diagrama, en el caso de RADD, o el manejo de reglas heurísticas, en el caso de ER-Converter Tool. La complejidad del análisis semántico para la realización de varios esquemas conceptuales es la que justifica la aparición de las representaciones intermedias. En el caso de Juristo *et al.* y KISS, que generan varios diagramas se requiere un análisis sintáctico y semántico complejo para determinar los elementos a los cuales se va a traducir el discurso; es por ello que deben realizar la segmentación del discurso, con el fin de reducir la complejidad y permitir que la transición a los diagramas sea más fácil.
- RADD posee un "Formalismo Intermedio" que permite la realización de la interpretación pragmática; sin embargo, ese formalismo también es incompleto porque el texto de hecho contiene información suficiente para el trazado de otros diagramas. Si

se hubiese usado una representación intermedia, la definición de ese formalismo podría haber sido mucho más sencilla y sería más fácilmente ampliable para la realización de los diagramas adicionales.

Sin embargo, los fundamentos teóricos de esta herramienta constituyen un insumo importante en la conceptualización de esta propuesta de Tesis, por razones como las siguientes:

- El proyecto RADD busca algo que es escaso dentro de los diferentes esquemas que pretenden la obtención de esquemas conceptuales a partir de especificaciones textuales en lenguaje natural: la completitud del diagrama resultante. La interpretación pragmática misma es uno de los elementos que deben ser fundamentales para la representación intermedia que se plantee, puesto que los textos en lenguaje natural pueden poseer ciertas imprecisiones que será pertinente refinar en un proceso de diálogo con el usuario, y por ello la comunicación de doble vía deberá ser posible en cualquier instante. La definición de reglas de completitud de los diferentes esquemas se puede ejemplificar dentro del ámbito de esta Tesis, pese a que está fuera del alcance de la misma.
- Utiliza más características que las que emplea LIDA en su análisis sintáctico, y por ello puede sugerir ciertos elementos adicionales al esquema conceptual, que en LIDA serían impensables; elementos tales como las cardinalidades (obtenidas a partir de los determinantes que acompañan los sustantivos) y los identificadores únicos de las entidades, se pueden sugerir a partir del análisis que se realiza en RADD y ER-Converter Tool.

En el trabajo de Díaz y otros (Díaz *et al.*, 2004), donde se busca la obtención del diagrama de secuencias a partir de la especificación formal en lenguaje natural de los casos de uso, no se encuentra un esquema gráfico que sirva como representación intermedia, pero éste

puede ser sustituido por un conjunto de tablas que consignan información sintáctica y semántica, como se muestra en la Figura 9. Estas tablas *per-se* podrían ser consideradas como representación intermedia. Las principales desventajas de este método son:

- Se realiza para un tipo de diagrama único: el diagrama de secuencias. Nuevamente, si se requiriese la obtención de otros diagramas, los patrones necesarios para lograr establecer los criterios deberían incrementarse.
- Aún es un proceso semiautomático. Se puede estudiar la posibilidad de su automatización completa mediante la incorporación de algún tipo de representación intermedia, que pueda ayudar en la determinación de los criterios de comparación con los diferentes patrones.
- El punto de partida no es tan "libre" como en RADD, ER-Converter Tool o LIDA. La descripción textual del caso de uso debe escribirse según una plantilla definida, restringiendo el lenguaje de manera apreciable. Además, para la elaboración de la descripción textual, se requiere que la solución informática para las necesidades del interesado ya se haya conceptualizado, puesto que los casos de uso describen las interacciones entre el usuario y el sistema futuro; en general, al inicio de la elicitación de requisitos esta solución aún no se ha conceptualizado.

Esta última desventaja tiene también un lado positivo, puesto que al obligar al usuario a hablar de una manera reglada o canónica se eliminan muchas de las ambigüedades comunes que se pueden presentar, tratándose aún de lenguaje natural, pero en una versión controlada. Otros proyectos como NIBA utilizan esta estrategia con el fin de simplificar los conjuntos de reglas heurísticas que se requerirían para lidiar con especificaciones textuales que debieran lidiar también con las imprecisiones del lenguaje natural.

Ahora, en los proyectos analizados se nota una tendencia a realizar un solo esquema conceptual a la vez. Sólo los proyectos de Juristo y otros, KISS y NIBA involucraN la

creación de varios esquemas conceptuales, pero sólo NIBA emplea diferentes representaciones intermedias para lograrlo. Para el modelamiento del software se recomienda siempre la representación del universo del discurso en diferentes modelos, por lo cual una representación intermedia que permita la obtención de únicamente un esquema conceptual no sería completamente útil. En este sentido, la representación intermedia que se proponga debe tener la capacidad de albergar las características de diferentes esquemas conceptuales, con el fin de que se puedan definir a partir de él reglas de mapeo adecuadas y sencillas para cada uno de esos esquemas conceptuales. En este caso, el esquema realizado para el proyecto ASPIN, basado en los grafos conceptuales de Sowa (1984), si bien no busca el proceso de conversión que se plantea en esta Tesis, se convierte en una alternativa viable, puesto que con él se pueden representar los diferentes esquemas conceptuales del ámbito de los Sistemas Digitales; una representación intermedia de este tipo podría servir incluso para las verificaciones de completitud que se requerirían en los esquemas conceptuales resultantes, facilitando el análisis pragmático que plantea el proyecto RADD, en busca de la completitud ya no de un diagrama solamente como en ese caso, sino de un conjunto de esquemas conceptuales que se hayan definido para la elaboración de software para un dominio particular.

Los grafos conceptuales usados como representación intermedia en ASPIN presentan ventajas sobre los esquemas utilizados por CM-BUILDER y NL-OOPS:

- Si bien el punto de partida es el dominio de los Sistemas Digitales, y por ello las reglas de conversión son precisas para las especificaciones en el lenguaje natural restringido por la ontología y los esquemas conceptuales propios de ese dominio, una vez la representación se logra, ésta es muy autónoma y su comprensión es casi inmediata, lo cual no ocurre con el SemNet de NL-OOPS que oculta información ligada con el número del nodo.

- La duplicidad de representaciones permite que la representación gráfica se use para la comprensión de los usuarios humanos, en tanto que la representación textual puede ser interpretable mediante diferentes programas.
- Los nodos de eventos de NL-OOPS, que aparecen ocultos en la representación gráfica de SemNet, se hacen explícitos como conceptos en los grafos conceptuales. En la Figura 11 el concepto "ejecutar" no aparece explícito sino dentro del evento 98344; por el contrario, en la Figura 12 el concepto *execute* se encuentra explícito en la representación del grafo conceptual.
- Los grafos conceptuales de ASPIN son representaciones de múltiples esquemas conceptuales, a diferencia de SemNet y las jerarquías de CM-Builder, que tienen utilidad únicamente para el diagrama de clases.

Sin embargo, subsisten ciertos inconvenientes que impiden una representación plena de los esquemas conceptuales:

- Los grafos conceptuales empleados por ASPIN se obtienen a partir de un ámbito muy restringido, supeditado a los conceptos de los Sistemas Digitales. En el desarrollo de software, si bien los dominios son siempre muy restringidos, esos dominios poco se repiten entre aplicaciones diferentes. Por ello, se pretende tener una representación muy amplia que dependa poco del contexto en el cual se utilicen.
- No se pueden representar adecuadamente en el grafo muchos elementos de las frases y de los diagramas que podrían ser relevantes en el momento de hacer una conversión. Elementos como frecuencias de aparición de conceptos, categorías verbales o adjetivales, que se utilizan en SemNet para NL-OOPS, no tienen una equivalencia en grafos conceptuales. Por esta razón, su uso para la construcción de esquemas conceptuales del software, como los diagramas UML, requeriría un uso intensivo de recursos léxicos que complementen los grafos conceptuales y permitan definir adecuadamente las reglas de formación de los mismos.

Una opción como la presentada por el proyecto NIBA, con diversas representaciones intermedias que se utilizan en familias de diagramas particulares (estructurales o interacción), no suministra una representación completa del universo del discurso, sino visiones parcializadas que podrían incluso cumplir un papel similar al de los diferentes esquemas conceptuales que apoyan el desarrollo de software para ese universo específico del discurso. Hay quienes afirman (Juristo y Moreno, 2000, por ejemplo) que las especificaciones textuales que sirven de base al desarrollo del software deberían fraccionarse en una parte estática y una dinámica, de forma tal que se pudieran atender los diagramas estáticos con la primera y los dinámicos con la segunda; el hecho de excluir las características estáticas o las dinámicas de uno u otro tipo de especificación genera una desconexión que podría afectar la consistencia entre los diagramas. Además, entre una especificación y otra se podrían presentar pérdidas de información y no sería posible la realización de una validación cruzada entre ambas, para detectar problemas de sinonimia o de significado en el uso de las palabras. Uno de los usos de una representación intermedia única en este caso (que sirva para los tres tipos de diagramas que se han mencionado), se aprecia cuando, en lugar de traducir una especificación textual particular a un esquema conceptual, lo que se desarrolla es un diálogo entre el usuario y el sistema para resolver los problemas de interpretación que puedan surgir (en este caso las posibles opciones de representación intermedia que se puedan generar para un problema dado), y para realizar el análisis pragmático planteado por el proyecto RADD, con el fin de que los esquemas conceptuales adquieran un refinamiento mayor que el que se establece únicamente con la traducción de la especificación textual. Sin una herramienta de diálogo tal, se dificulta la justificación de una representación intermedia única, puesto que el software para la obtención de los esquemas conceptuales no requeriría interpretar en los dos sentidos (interesado-analista y viceversa) sino solamente en uno.

Adicionalmente, en los tres proyectos que se orientan a múltiples diagramas se presentan problemas de consistencia en los diagramas resultantes originados en la forma de fraccionar el discurso o el esquema intermedio para encontrar los elementos correspondientes a los diferentes esquemas conceptuales. Es importante recalcar que los esquemas conceptuales son vistas del mismo modelo, pero pueden incluir características estructurales o dinámicas en diferente proporción.

En lo que respecta a las características de las diferentes representaciones intermedias, se presentan problemas aún no resueltos en los trabajos analizados. En primer lugar, algunos de esos trabajos emplean un conocimiento "previo" en relación con el dominio de la aplicación; la jerarquía de CM-BUILDER o los esquemas de prediseño conceptual de KCPM son ejemplos de esta afirmación. En la fase de análisis del desarrollo de una pieza de software, es común la utilización de este tipo de información, muy basado en las experiencias anteriores del analista en desarrollo de otros programas en dominios similares, que incluso le otorgan la facultad de anticiparse a ciertas preguntas que debe realizar al interesado para complementar la información pertinente al software; por ejemplo, si a un analista se le menciona la palabra "factura" inmediatamente la asocia con una fecha de realización, un cliente y un número de factura, sin que el usuario se lo mencione expresamente en principio. Si no existiera el conocimiento previo, se debería entablar un diálogo analista-interesado que permitiese la complementación de las especificaciones en búsqueda de todos los elementos pertinentes al desarrollo; ese diálogo tiene un conocimiento previo común que sería, como mínimo, el manejo del mismo lenguaje, que tiene un cuerpo de conceptos y significados tal que permita la comunicación entre ambos. A ese cuerpo de conceptos se le suele denominar "ontología" y podrá ser tan restringido o tan amplio como el conocimiento de los elementos de un área así lo permitan. Para el caso del desarrollo automático de software a partir de representaciones intermedias, la ontología no debería ser muy restringida, puesto que es difícil contar con ontologías ya desarrolladas y no se tiene la certeza de que el desarrollo que se va a hacer tenga una ontología ya

definida. Una ontología como la planteada en CM-BUILDER es difícil de conseguir, porque debe entregar un conocimiento previo en relación con la posibilidad de tener un concepto como un objeto o como un atributo, lo cual puede ser muy variable de problema a problema. La ontología que usa ASPIN es mucho más acertada en cuanto a los conceptos propios de los Sistemas Digitales, pero, nuevamente, es supremamente restringida, aunque su uso facilita enormemente la interpretación de las representaciones intermedias que obtienen.

Ahora, las representaciones intermedias analizadas utilizan un porcentaje significativamente bajo de las palabras presentes en la especificación textual en lenguaje natural (excepto KCPM), puesto que se concentran en tipos de palabras específicos para la realización de la conversión a los esquemas conceptuales. Esto es altamente inconveniente, porque se están desechando palabras que podrían suministrar claves importantes en relación con la conversión que se realiza, como ya se discutió antes. Esto se presenta porque, en la mayoría de las representaciones intermedias que se analizaron en el capítulo anterior, se estaba realizando la conversión a un solo diagrama (entidad-relación, clases, secuencias u otros) y por ello no todas las palabras eran de utilidad para la identificación. Sin embargo, si se quisiera reconstruir el texto a partir de la representación intermedia, el tema sería complicado incluso para KCPM, porque se sufren pérdidas importantes en palabras no convertidas a dichas representaciones; esta reconstrucción tiene la utilidad antes planteada para el diálogo de refinamiento de esquemas con el usuario, por lo cual las pérdidas entre las especificaciones en lenguaje natural y la representación intermedia deberían ser mínimas. Esta situación no se presenta porque las representaciones intermedias analizadas eligen como estrategia separar la parte estática de la dinámica en las especificaciones textuales, tal como se puede apreciar en las Figuras 14, 15 y 16. Una opción interesante de tratamiento del discurso, aunque sin representación intermedia, la tiene el proyecto KISS, en el cual se realiza una "depuración" de los conceptos y relaciones importantes del mundo (mediante la determinación de frases sujeto-predicado-objeto directo), y luego se

complementa con información adicional que puede ayudar en la conversión a los esquemas conceptuales. Además, la representación intermedia debería ser tal que permitiese una forma "canónica" de hablar que se le pudiese enseñar al usuario, para que paulatinamente se superen las barreras del idioma originadas por formas ambiguas del lenguaje y sea menos necesario, por tanto, realizar la desambiguación del discurso.

Además, en relación con la ambigüedad, se debe resaltar que los recursos léxicos disponibles actualmente no tienen un consenso que permita eliminar este tipo de defectos del idioma. Existe una dependencia muy marcada de la ambigüedad en las frases y su posibilidad de eliminarla con el tipo de lexicón que esté disponible, puesto que las clasificaciones de los sentidos gramaticales de los verbos varían sustancialmente entre los trabajos analizados en el capítulo anterior. En KCPM se optó por un lexicón nuevo, realizando la clasificación de 10.000 verbos del alemán para incluirlos de esa manera en el lexicón; sería preferible en este caso utilizar recursos léxicos ya existentes y que posibiliten el acceso a otras clasificaciones como la de Vendler (1957). Este tipo de clasificaciones son de común uso en los recursos léxicos disponibles y posibilitan tareas como la determinación del sentido de los verbos, la cual se basa en el contexto que se encuentra en la especificación textual en lenguaje natural.

El proyecto CM-BUILDER como tal tiene una serie de desventajas que ya se han discutido:

- La dificultad de construcción de una ontología universal que establezca la categoría de cada palabra en los grupos definidos, especialmente entre objeto y atributo, puesto que el contexto puede sugerir que un sustantivo que se encuentra en un texto sea en unos casos atributo y en otros objeto.
- Los grupos de las jerarquías son claramente enfocados hacia el diagrama de clases y excluyen características que impiden su utilización para otros tipos de diagrama, restándole generalidad a las redes semánticas como representaciones intermedias.

Características como las pre o postcondiciones, o incluso los estados, impiden que esta representación se pueda usar en la obtención de diagramas de máquina de estados o diagramas de actividades.

Dentro de los aspectos positivos del proyecto, se resalta la identificación que realiza de diferentes tipos de relaciones (agregaciones y composiciones), las multiplicidades en los extremos de las asociaciones, las categorías de los adjetivos y la frecuencia de las palabras. Realizando estas identificaciones, las reglas de transformación entregan diagramas de clases bastante completos, aun cuando la representación intermedia que usa es insuficiente para realizar estas identificaciones, que deben ser complementadas con el uso de otros recursos léxicos.

En cuanto a NL-OOPS, las desventajas del proyecto nuevamente tienen relación con las de otros proyectos que se han analizado; se pueden mencionar las siguientes:

- También es una herramienta enfocada al diagrama de clases, lo que nuevamente le resta generalidad de resultados. La representación intermedia que utiliza, y sobre todo la información complementaria ligada con los nodos, puede suministrar elementos para la elaboración de otros diagramas, pero sólo se centra en la construcción de un diagrama de clases propuesto.
- Otras características del diagrama de clases, como puede ser la identificación del tipo de relación (agregación, generalización, dependencia o herencia) se podrían lograr también de una manera sencilla, lo cual tampoco se realiza.

Como aspecto positivo se puede señalar la información complementaria a los nodos que, si se involucrara activamente dentro de la representación intermedia, podría servir de base para la elaboración de otros esquemas conceptuales diferentes; además, se podría esclarecer gráficamente el sentido de las frases involucradas. Esa información es obtenible en algunos

de los lexicones que actualmente se están elaborando, como por ejemplo el de la Universidad de Maryland (Dorr, 2001).

Ahora, KCPM posee algunas ventajas sobre las demás representaciones intermedias, entre las que se destacan:

- Se cubre una amplia gama de diagramas UML con las representaciones intermedias de KCPM, pese a que cada tipo de esquema conceptual de UML emplea una diferente. En general, los proyectos analizados sólo realizan la conversión a un solo diagrama.
- Utilizan otros recursos lingüísticos además de la clasificación de los verbos; por ejemplo, utilizan la teoría de los roles theta propuesta por Gruber (1965)—una teoría que antecedió a los roles semánticos de Fillmore (1968)—para establecer las características sintácticas y semánticas de las frases. Además, emplean otros recursos que pueden esclarecer la identificación de las clases, como el determinante de tipo conexión (véase Figura 15), que consiste en buscar en las especificaciones en lenguaje natural si una palabra se ha usado como verbo y a la vez como sustantivo en diferentes partes del texto; por ejemplo, si en las especificaciones aparecen las palabras “pedido” y “pedir” en diferentes oraciones, es un indicio de que “pedido” es una clase.
- Los autores afirman que la representación intermedia tabular que emplean para los diagramas estructurales se convierte en una buena alternativa para verificar la completitud de las especificaciones, de forma similar a la manera como lo hace RADD. Esto, sin embargo, no lo emplean con ese fin para su desarrollo.

Sin embargo, y pese a las grandes ventajas que se encuentran en este enfoque, también se pueden encontrar ciertas desventajas desde el punto de vista de las representaciones intermedias, tales como:

- El hecho de utilizar representaciones intermedias diferentes para los esquemas estructurales y comportamentales de UML puede ocasionar errores de consistencia entre los modelos generados. Si bien las especificaciones textuales en lenguaje natural que dan origen a los diagramas pueden ser únicas, se emplean subconjuntos de ellas cuando se trata de realizar los diferentes tipos de diagramas, de forma tal que la información contenida en una representación tabular no se puede reconstruir a partir de una representación dinámica para el mismo problema. Por el hecho de representar un mismo Universo del discurso, las representaciones intermedias deberían ser equivalentes para obtener una a partir de la otra, lo cual no es posible para KCPM.
- La conformación de la representación tabular aún no es completamente unívoca, y depende de los criterios internos que tenga la herramienta para realizarlos. En otras palabras, es posible que un mismo conjunto de especificaciones textuales en lenguaje natural pueda entregar dos o tres representaciones tabulares diferentes, pero la herramienta sólo entrega una, en la cual adicionalmente el diseñador no tuvo posibilidad de intervención, descartando sus preferencias a la hora de interpretación de las especificaciones. Algo similar ocurre con las representaciones dinámicas.
- Para obtención de otros esquemas conceptuales diferentes a los de UML no es claro de qué manera habría que hacer la ampliación a las representaciones intermedias, ni cuál de ellas usar. Por ejemplo, un diagrama de procesos, que conjuga tanto elementos estructurales como comportamentales, sería muy difícil de traducir a partir de las representaciones intermedias independientes que es posible construir con KCPM.

La revisión de los diferentes trabajos que procuran la obtención de esquemas conceptuales a partir de especificaciones textuales en lenguaje natural muestra que aún existen inconvenientes en la realización de esta labor que motivan la elaboración de esta Tesis. Particularmente se resalta lo siguiente:

- Ninguno de los proyectos tiene una representación intermedia de tipo único que sirva de base para la elaboración automática de esquemas conceptuales estructurales, comportamentales y de interacción simultáneamente. Las visiones son parciales, como en el caso de KCPM, o simplemente se dedican a la obtención de un solo tipo de diagrama.
- Las reglas de mapeo entre las representaciones intermedias y los esquemas conceptuales todavía presentan inconvenientes que hacen que no puedan ser consideradas como reglas generales. Los diferentes trabajos presentan diferencias sustanciales, pues entregan soluciones preliminares, las cuales se pueden considerar únicamente como un punto de partida para la realización de los esquemas conceptuales; apoyados en este hecho, no está(n) entregando el (los) mejor(es) esquema(s) conceptual(es) que es posible obtener a partir del conjunto de especificaciones textuales entregadas. La representación intermedia debería ser tal que fuese posible entregar el diagrama, realizar la validación de completitud que plantea el proyecto RADD y facilitar la comunicación con el diseñador en lenguaje natural también para realizar el refinamiento de los diagramas.
- A excepción del trabajo de Díaz *et al.* (2004), los trabajos analizados se han realizado para lenguajes diferentes al español: LIDA soporta italiano e inglés, al igual que NL-OOPS, CM-BUILDER y ASPIN se realizaron para el inglés y RADD y KCPM funcionan para el inglés y el alemán. El hecho de realizar un análisis similar al descrito para el español tiene algunas complicaciones adicionales, por las características especiales que presenta este idioma en relación con la conjugación de los verbos y el manejo de los plurales, por sólo mencionar algunas.
- Para la consistencia entre los diferentes modelos que se puedan obtener de un conjunto de especificaciones textuales, la mayoría de los trabajos sólo atiende un tipo de diagrama y en KCPM ya se discutió que se pueden presentar problemas por la presencia de múltiples representaciones intermedias. Si fuese posible definir una representación

intermedia de tipo único para los diferentes tipos de diagrama, se podría controlar de manera más adecuada el tema de la consistencia.

En la Tabla 1 se realiza una comparación de los diferentes métodos desde la óptica del marco conceptual definido en el Capítulo 2. Allí se resumen las características de los esquemas conceptuales analizados en relación con la propuesta de solución que se plantea, tomando como base los elementos de la Figura 3.

3.2.2. Traducción de Máquina:

También los formalismos descritos en la sección 3.1.2. para el análisis semántico, en el caso de la Traducción de Máquina, presentan ventajas y desventajas cuando se proponen como posibles candidatos a representación intermedia, tal y como se pretende en el contexto de esta Tesis.

La LPPO, por ejemplo, para su uso en semántica computacional, y particularmente para su utilidad como representación intermedia para la obtención automática de esquemas conceptuales UML, tiene ventajas como las siguientes:

- En general posee buenos mecanismos de inferencia (se pueden demostrar teoremas a partir de axiomas básicos). En particular, puede ser posible la representación de las reglas de transformación, ya sea de lenguaje natural a lógica, o de allí a diagramas UML mediante el mapeador.

Tabla 1. Comparación de los diferentes proyectos con base en la representación intermedia

	PROYECTO										
Característica	**LIDA**	**RADD**	**ER-Converter Tool**	**Díaz *et al.***	**CM-Builder**	**NL-OOPS**	**ASPIN**	**Juristo *et al.***	**KISS**	**KCPM**	**UNAL (Propuesta)**
Realiza Análisis Sintáctico?	Muy preliminar	SI	SI	SI	SI	SI	NO	SI	SI	SI	SI
Realiza Análisis Semántico?	NO	SI	NO	SI	SI	SI	SI	SI	SI	SI	SI
Emplea reglas de transformación?	NO	SI	SI	SI	SI	SI	SI	SI	SI	SI	SI
Cuál es la orientación de las reglas de transformación?	-	Análisis Sintáctico y Semántico	Análisis sintáctico y heurísticas	Análisis Sintáctico y Semántico	Análisis Sintáctico y Semántico	Análisis Sintáctico y Semántico	Análisis Sintáctico y Semántico	Análisis Sintáctico y Semántico	Análisis Sintáctico y Semántico	Análisis Sintáctico y Semántico	Análisis Sintáctico y Semántico
Utiliza Representación intermedia?	NO	NO	NO	SI	NO	SI	SI	NO	NO	SI	SI
Tipo de Representación intermedia	-	-	-	Tablas resultantes del análisis sintáctico y semántico	La Jerarquía Propia no puede ser considerada representación intermedia.	SemNet	Grafos Conceptuales de Sowa	-	-	Tablas para esquemas estructurales Diagrama para esquemas dinámicos	Grafo único (Esquema Preconceptual) para atender todos los esquemas
Idioma	Italiano	Alemán	Inglés	Español	Inglés	Inglés – Italiano	Inglés	Inglés	Inglés	Alemán – Inglés	Puede ser Inglés o español

Tabla 1 (Continuación). Comparación de los diferentes proyectos con base en los requisitos de la propuesta de Esquema Preconceptual

	PROYECTO										
Característica	**LIDA**	**RADD**	**ER-Converter Tool**	**Díaz *et al.***	**CM-Builder**	**NL-OOPS**	**ASPIN**	**Juristo *et al.***	**KISS**	**KCPM**	**UNAL (Propuesta)**
Atiende simultáneamente los diferentes esquemas?	Sólo atiende esquemas estructurales (diagrama clases)	Sólo atiende esquemas estructurales (diagrama entidad-relación)	Sólo atiende esquemas estructurales (diagrama entidad-relación)	Sólo atiende diagramas de secuencias a partir de casos de uso	Sólo atiende esquemas estructurales (diagrama clases)	Sólo atiende esquemas estructurales (diagrama clases)	SI	SI	SI	Atiende esquemas estructurales y dinámicos pero por separado	SI
Completitud de los esquemas conceptuales	Se realiza de manera manual	Se realiza con una herramienta de diálogo	-	Le faltan elementos	Le faltan elementos	Le faltan elementos	-	Le faltan elementos	Le faltan elementos	Le faltan elementos	Los diagramas se deben obtener automáticamente e incluir la mayor cantidad posible de información
Elementos adicionales para conversión a esquemas conceptuales	Herramienta de edición	Interpretación pragmática usando un formalismo	-	Interpretación sintáctico – semántica mediante una tabla	Frecuencia palabras, relaciones posesivas, categorías adjetivales, cuantificación	Cuantificación, clasificación actores, categoría de evento	-	-	-	Determinantes, pre y post condiciones, categoría verbal	Deberá incluir la mayor cantidad posible.
Tipo de Ontología	General	General	General	General	Dominio	General	Dominio	General		Dominio	General
Lexicón	Cualquiera	Especializado	Especializado	Cualquiera	Especializado	Especializado	Especializado	Especializado	Wordnet	Especializado	Cualquiera
Posee formas gráficas y textuales?	SI	No se conoce	No se conoce	No se conoce	No se conoce	No se conoce	SI	No se conoce	No se conoce	SI	SI

- Presenta una sintaxis completamente inambigua, una característica muy deseable para el desarrollo de piezas de software para procesamiento del lenguaje natural; se podrían, eso sí, realizar sutiles cambios para mejorar su legibilidad.
- Como consecuencia de lo anterior, puede representar mejor que algunos otros formalismos ciertos fenómenos del lenguaje que son de difícil comprensión, tales como las ambigüedades de ámbito de cuantificadores, que es un tipo de ambigüedad que se origina en presencia de cuantificadores, operadores lógicos, operadores modales y adverbios en diferentes frases (Allen, 1995). Este tipo de ambigüedad puede persistir aún después de eliminar ambigüedades de tipo sintáctico y semántico.
- Es una representación completamente funcional, puesto que se pueden evaluar las condiciones asociadas con las fórmulas que con este tipo de lógica se pueden definir, para determinar si lo que está representando la fórmula es cierto o no.

Y presenta también algunos inconvenientes, como son:

- Se requiere entrenamiento cuidadoso para su uso. Para un usuario convencional, la LPPO presenta problemas de elaboración y validación, lo que hace que sólo pudiera ser usado como mecanismo interno de representación. Para esta Tesis, se requiere un formalismo que pueda estar más ligado con la actividad de los interesados, pues se puede requerir su intervención para realizar algunas validaciones.
- No es fácil representar varias frases simultáneamente, lo cual puede proveer mecanismos de desambiguación o de aclaración de los modelos UML resultantes.

Como extensión de la LPPO, para el objetivo de esta propuesta de Tesis, el λ-cálculo presenta las mismas ventajas y desventajas de la LPPO, pero además sobre ésta tiene como ventaja adicional la posibilidad de construir fórmulas semánticas a partir de representaciones sintácticas de elementos. De esta manera se podría hacer una transición suave entre la representación sintáctica y la semántica.

Ahora, si se fuese a elegir el formalismo en HPSG como representación intermedia para la obtención automática de esquemas conceptuales UML, se tendrían algunas ventajas como:

- Conjuga una representación simultánea de la información sintáctica y semántica, generando una transición suave entre los dos tipos de análisis.
- Es una representación más gráfica que la LPPO o el λ-cálculo; por ello, un usuario entrenado podría comprenderla relativamente bien, como para poder hacer validaciones si fuese necesario.
- También es un enfoque susceptible de utilización en piezas de software. La representación matricial facilita su expresión para lenguajes orientados a la elaboración de software.

Sin embargo, también presenta ciertas desventajas:

- El nivel de complejidad de las matrices atributo-valor va creciendo con los diferentes elementos que se vayan sumando a las frases. Si bien una representación intermedia será manejada casi exclusivamente por el software de transformación, se requiere también que tenga un equivalente que sea entendible por humanos, puesto que el analista podría requerir la realización validaciones en un momento dado sobre el esquema.
- Como en el caso de la lógica de predicados y el λ-cálculo, se dificulta enormemente la representación de varias frases simultáneamente. Incluso, la representación mediante HPSG puede ser similar a los árboles de componentes del análisis sintáctico, pero sustituyendo las categorías intermedias por matrices atributo-valor. La representación se va haciendo muy grande si se van sumando frases a ella.

MRS, por ser una ampliación de la LPPO, y por ser representable en HPSG, hereda las ventajas y desventajas de los dos enfoques, con miras a su utilización como representación intermedia. Particularmente se destacan la funcionalidad para realizar cálculos y la combinación de información sintáctica y semántica (cuando se representa con HPSG), como ventajas. Como principales desventajas, se arguyen el incremento en la complejidad a medida que se suman palabras a las frases y las dificultades para la representación de más de una frase a la vez.

Como lenguaje de representación, LKB presenta las mismas ventajas y desventajas de la LPPO, enfatizando en las dificultades inherentes para los usuarios que tendrían que aprender su sintaxis.

El uso de la MTT como representación intermedia para la obtención automática de esquemas conceptuales UML tendría las siguientes ventajas:

- Posee un proceso de conversión completo desde texto en lenguaje natural hasta la representación semántica, que puede entregarle consistencia para la realización de una transición adecuada a ese proceso. Las transiciones entre los diferentes diagramas van llevando la frase paulatinamente a disponer de los diferentes elementos morfológicos, sintácticos y semánticos que podrían ser indispensables para realizar la traducción a esquemas conceptuales UML.
- Por ser una representación gráfica, es fácilmente interpretable, y por ser una red semántica, su equivalencia legible por máquinas es prácticamente directa.

Pero también presenta desventajas, tales como:

- Como todos los formalismos revisados, la MTT tiene dificultades para representar más de una frase a la vez, lo cual sería necesario para resolver algún tipo de ambigüedades,

cuando el interesado suministra más información sobre los términos relacionados en una frase.

- La equivalencia entre significados y textos es muchos a muchos, es decir que un texto puede tener múltiples interpretaciones y una interpretación puede coincidir con muchos textos. Por la gran cantidad de esquemas que hay que elaborar, la complejidad tiende a crecer enormemente cuando la frase en lenguaje natural es muy ambigua.
- Algunos de sus diagramas eliminan información que puede ser importante al momento de identificar los elementos fundamentales de los esquemas conceptuales UML, tales como las preposiciones o los artículos.

Los trabajos que se analizaron en el presente capítulo revelan problemas aún no completamente solucionados en el ámbito de las representaciones intermedias para el desarrollo automático de software, que motivan la realización de esta Tesis. En el capítulo siguiente se realiza una propuesta de solución que incluye un Esquema Preconceptual que facilite la obtención automática de esquemas conceptuales de UML.

4. PROPUESTA DE SOLUCIÓN

En este capítulo se discute la propuesta de solución, que incluye varios elementos: los Esquemas Preconceptuales, el lenguaje UN-Lencep que permite especificarlos y las reglas heurísticas para la obtención automática de los esquemas conceptuales de UML. Estos elementos se implementan en el entorno de la herramienta CASE UNC-Diagramador, que se ejemplifica también en este capítulo.

4.1. Requisitos del Método Propuesto.

Para la automatización del proceso de obtención de esquemas conceptuales de UML, se propone la elaboración de una representación intermedia basada en un nuevo tipo de grafos, denominados Esquemas Preconceptuales. En este capítulo se mostrará cómo estos esquemas se pueden especificar en un subconjunto del lenguaje natural denominado UN-Lencep. Los Esquemas Preconceptuales deberán capturar las características sintácticas y semánticas necesarias del texto de especificaciones, así como otras características especiales que faciliten la obtención de esquemas conceptuales de tipo estructural, comportamental y de interacción.

Algunos de los requisitos que debe cumplir el Esquema Preconceptual que se propone en esta Tesis son los siguientes:

- Debe utilizar la sintaxis y semántica del idioma español, aunque podría ser válido también para otros idiomas.
- Debe atender simultáneamente diagramas estructurales, comportamentales y de interacción. Con ello se espera que la consistencia entre los diagramas resultantes mejore por partir de un núcleo común.

- Debe permitir la definición de reglas heurísticas que conduzcan a la obtención automática de esquemas conceptuales de UML, con la mayor completitud posible, dadas las especificaciones textuales en lenguaje natural.
- Debe ser obtenible a partir de un subconjunto del lenguaje natural. Ese subconjunto debe ser un lenguaje controlado en el cual se pueda expresar el discurso del interesado.
- No debe limitarse a una ontología de un dominio específico, sino que debe utilizar los conceptos del idioma español como ontología para realizar las conversiones a esquemas conceptuales de UML. Además, no debería tener elementos preclasificados o mapeados previamente hacia algunos elementos de los esquemas conceptuales; por ejemplo, un libro podría ser clase o atributo dependiendo del contexto en que se utilice. El Esquema Preconceptual debería apoyar la identificación de ese contexto; este esquema debería ser lo único que se requiriese para capturar la sintaxis y semántica del dominio.
- Debe poderse construir con alguno de los recursos léxicos disponibles, realizando las adaptaciones necesarias para ello.
- Debe poseer formas equivalentes de representación que puedan ser entendibles por humanos (por ejemplo de forma gráfica) y máquinas (con una forma equivalente en texto).

Ahora, la complejidad de la especificación de UML dificulta considerablemente el objeto de estudio de esta Tesis, puesto que la cantidad de primitivas conceptuales que se podrían manejar es enorme. A este respecto, la Ingeniería de Software ha presentado tradicionalmente estrategias para lidiar la complejidad; en efecto, en Booch et al. (1999) se afirma:

> "Mediante el modelamiento, estrechamos el problema que estamos estudiando al enfocarnos en sólo un aspecto a la vez. Este es esencialmente el enfoque de 'divide y vencerás' que Edsger Dijkstra hablaba hace unos años: atacar un problema difícil por la división en una serie de subproblemas que se puedan resolver"

Si bien la Tesis se enfoca más en los lineamientos teóricos para la definición del Esquema Preconceptual, de alguna manera estos lineamientos se convierten en los requisitos de una herramienta CASE que realiza el trazado automático de los esquemas conceptuales UML, partiendo del texto en lenguaje controlado. Tomando esto en consideración, y dada la complejidad del objeto de estudio, Leffingwell y Widrig (1999) reafirman el hecho de que hay que concentrarse en unas pocas características del sistema para comprender adecuadamente el nivel de abstracción. Esa es la estrategia seleccionada para lidiar con la complejidad de la superestructura de UML.

Dado que el objetivo general de la Tesis tiene un alcance muy amplio, se decide realizar la caracterización en el Esquema Preconceptual de un diagrama de cada tipo, seleccionados por la frecuencia y representatividad que pueden presentar en el análisis. Los diagramas elegidos son: clases, máquina de estados y comunicación, que representan estructura, comportamiento e interacción, respectivamente.

Algunas consideraciones adicionales al respecto son:

- Se excluyen de la Tesis los diagramas estructurales considerados de implementación o despliegue (en la Figura 4 son: *component, composite structure, deployment* y *package*), puesto que se considera que para su realización hay que tomar decisiones de diseño que aún no están definidas en el discurso que plantea el usuario.
- Se excluye también el diagrama de objetos puesto que algunos de sus elementos se pueden visualizar en los diagramas de clases y comunicación.
- Entre los diagramas de comportamiento, se excluyen el diagrama de casos de uso y el de actividades, puesto que algunos de sus elementos representativos hacen parte de los diagramas de comunicación y de máquina de estados.

- Entre los diagramas de interacción se excluyen: el diagrama de secuencias (por ser un equivalente sintáctico del diagrama de comunicación), el diagrama de revisión de interacción (porque, como Fowler, 2004, lo anota, es demasiado reciente y aún no se tiene una idea muy clara de cuál será su uso en el desarrollo de software), y el diagrama de tiempos (porque parte de su información se incluye en el diagrama de máquina de estados y la información restante puede ser difícil de obtener a partir de las especificaciones textuales en lenguaje natural o controlado entregadas por el interesado).

Ahora, si bien es cierto que UML posee una sobrecarga semántica importante, el manejo que se dará al Esquema Preconceptual será la representación de los conceptos básicos de estos diagramas que posibiliten una descripción conceptual mínima del problema esbozado por el interesado. Esos conceptos básicos (denominados *constructs* en la superestructura de UML, versión 2.0), son los siguientes:

- Diagrama de clases:
  En la Figura 22 se pueden apreciar las primitivas conceptuales básicas que se trabajarán para el diagrama de clases, que son: clase, atributos, operaciones, asociación, generalización y agregación.

- Diagrama de máquina de estados:
  En la Figura 23 se muestran las primitivas conceptuales seleccionadas, correspondientes al DME, que son: estado, condición de guarda y estado inicial.

- Diagrama de comunicación:
  Similarmente, en la Figura 24 se muestran las primitivas conceptuales a considerar para este diagrama, que son: clase a la que pertenecen los objetos, conector entre objetos, condición de guarda y mensaje.

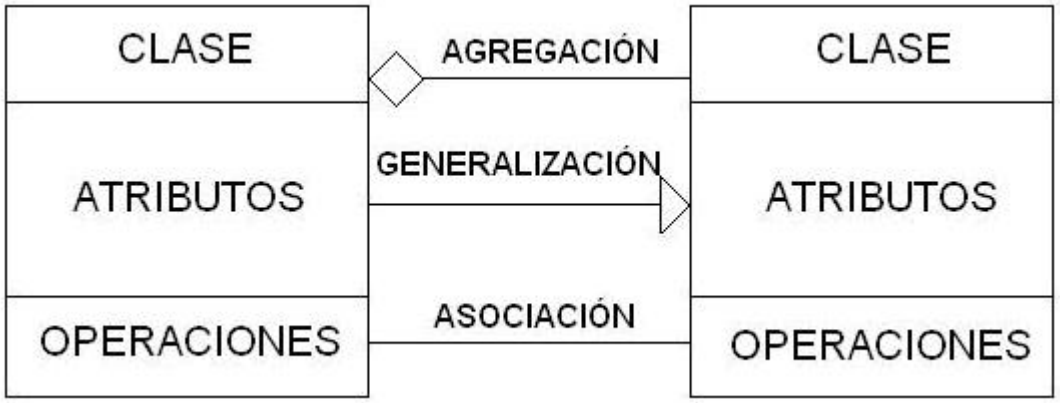


Figura 22. Primitivas conceptuales básicas del diagrama de clases que se incorporarán en el Esquema Preconceptual.

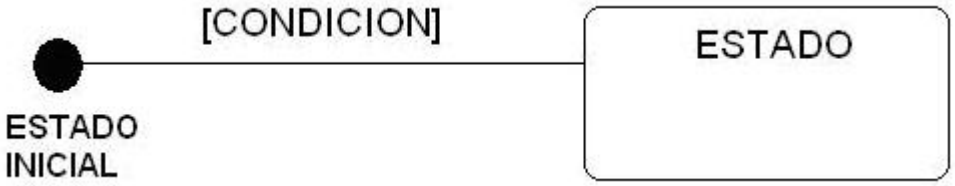


Figura 23. Primitivas conceptuales básicas del diagrama de máquina de estados que se incorporarán en el Esquema Preconceptual.

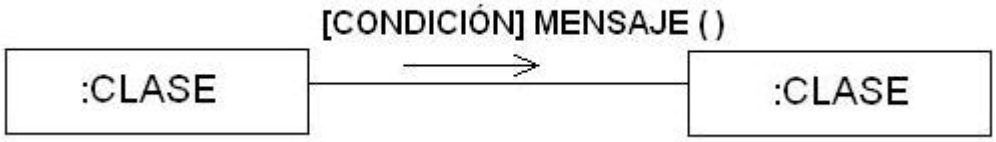


Figura 24. Primitivas conceptuales básicas del diagrama de comunicación que se incorporarán en el Esquema Preconceptual.

En general, el Esquema Preconceptual deberá estar en capacidad de representar como mínimo este conjunto de *constructs*, apoyándose para ello en la descripción del dominio del interesado expresada en un lenguaje controlado. Adicionalmente, las reglas heurísticas de transformación deberán ser tales que, únicamente con el Esquema Preconceptual, se puedan mapear todos estos conceptos básicos a los diagramas de UML descritos.

En principio, lo que se pretende con esta Tesis es representar las primitivas conceptuales mostradas en los tres diagramas seleccionados, sin incluir toda la semántica asociada con ellos. Para cada primitiva conceptual, las características que no se representarán son:

- Atributos: tipos de dato, cardinalidades y valores iniciales.
- Operaciones: Lista de argumentos y tipo de retorno.
- Asociaciones: nombres de los roles.
- Generalización: conjunto de generalización.
- Estados: acciones internas (*on entry, do, exit, event*), estados compuestos.
- Mensajes: argumentos.

Las reglas heurísticas de transformación se definirán desde el Esquema Preconceptual. Para ello, este esquema deberá conservar la mayor cantidad posible de elementos sintácticos y semánticos, así como ciertas características de las especificaciones textuales en lenguaje natural, de forma tal que esos elementos se incorporen adecuadamente en las reglas heurísticas de transformación para lograr la mejor conversión posible.

4.2. El origen del término "Preconceptual"

Los primeros usos del término "Preconceptual" se pueden asociar con la Filosofía y la Pedagogía. Para el filósofo alemán Martin Heidegger (1976), al tratar de definir el significado del "ser" era necesario entender intuitivamente ese concepto; es decir, era necesario construir ese concepto con un conjunto de conocimientos previos que suministraran la información necesaria para entenderlo. Por "Preconceptual" Heidegger trataba de calificar la información previa que se empleaba en la construcción de conceptos.

El término "Preconceptual" fue también usado por Piaget (1952) en el ámbito pedagógico. Piaget definió un conjunto de etapas del conocimiento, para representar la manera en que se desarrolla la inteligencia en los niños; en la que denominó la "Etapa Preconceptual", que sitúa después de la adquisición lingüística, pero previa a la etapa de conceptualización del conocimiento, el niño construye un conjunto de interpretaciones intuitivas del mundo, que posteriormente emplea para definir categorías de los elementos que percibe. Esas interpretaciones intuitivas se denominan "Preconceptos" y sirven de base para la elaboración de complejas taxonomías que incrementan el conocimiento del mundo por parte de los niños.

En Sistemas de Información, el término "Preconceptual" ha sido poco usado. Tan sólo se puede ubicar una referencia a Pylyshyn (2001) en el ámbito de la visión artificial, empleando los denominados "índices visuales". Según Pylyshyn, para la captura de la información visual de una escena, se requiere un procesamiento preconceptual de la misma, que ubique previamente ciertos patrones de reconocimiento; esos patrones se encargan de determinar elementos conocidos en la escena a capturar, con el fin de facilitar la identificación de la escena completa.

Según se pudo establecer en el Capítulo 3, el proceso que se realiza en la obtención automática de esquemas conceptuales parte de descripciones en lenguajes controlados y se afianza con una representación intermedia, que finalmente permite llegar a los esquemas conceptuales (Zapata *et al.*, 2006). A esa representación intermedia se le denominará en adelante "Esquema Preconceptual", para expresar el conocimiento previo que se debe representar para poder alcanzar los esquemas conceptuales.

En el trabajo de la Universidad de Klagenfurt se define el KCPM como un modelo de prediseño conceptual, que hace las veces de un lenguaje intermedio entre el lenguaje natural y los esquemas conceptuales (Fliedl *et al.*, 2003), o, de manera similar, como una

herramienta para armonizar los puntos de vista del usuario final y del desarrollador en un universo del discurso dado (Mayr y Kop, 2002). Otros esquemas, como el SemNet de NL-OOPS (Mich, 1996), la jerarquía empleada por CM-BUILDER (Harmain y Gaizauskas, 2000) y los grafos conceptuales utilizados por ASPIN (Cyre, 1995), se podrían agrupar como representaciones intermedias, para significar que son esquemas transitorios que posibilitan la elaboración de los esquemas conceptuales, representando las especificaciones textuales en un lenguaje controlado cercano al natural. Sin embargo, para evaluar la conveniencia de esta agrupación, se deben establecer las características que debería cumplir un esquema para ser considerado "preconceptual". Bajo las suposiciones básicas de esta Tesis, un Esquema Preconceptual deberá ser tal que:

- Sirva de intermediación entre una especificación textual en lenguaje controlado y los esquemas conceptuales de UML. Esto implica dos cosas:
  - El Esquema Preconceptual debe poderse obtener a partir de las especificaciones en lenguaje controlado.
  - Los diferentes esquemas conceptuales deberán tener contenidos sus diferentes elementos en el Esquema Preconceptual; dichos elementos se deberán obtener a partir de los elementos del Esquema Preconceptual empleando reglas heurísticas de transformación.
- Contenga información proveniente de los análisis sintáctico y semántico que le preceden. Como mínimo se considera que deberá tener información sobre la categoría gramatical de las palabras; esto, sin embargo, no impide que el Esquema Preconceptual pueda contener otros elementos sintácticos o semánticos que faciliten la obtención posterior de esquemas conceptuales.
- Posea un formalismo gráfico o textual, sobre el cual se puedan verificar reglas de "buena formación".

Se prefiere el nombre "Esquema Preconceptual" sobre el "Esquema de Prediseño Conceptual" establecido por los investigadores de la universidad de Klagenfurt (Fliedl *et al.*, 2002), puesto que se está obteniendo un esquema a partir de lenguaje controlado cercano al natural, pero es más un modelo del dominio de aplicación que una conceptualización de la solución a un problema de ese dominio, que sería lo que le adicionaría de más la palabra "prediseño". Un Esquema Preconceptual debería ser tal que permitiese la definición de un vocabulario del área, más que establecer las características previas de diseño de una solución informática al problema de esa área.

4.3. Los Esquemas Preconceptuales

De las representaciones intermedias analizadas en el Capítulo 3, se encontró que los Grafos Conceptuales se habían empleado para representar simultáneamente diferentes tipos de diagramas en el ámbito de los Sistemas Digitales. Como en esta Tesis se pretende la agrupación de varios diagramas de UML en una representación común, los Grafos Conceptuales parecen ser una opción viable para realizarlo. Sin embargo, los Grafos Conceptuales poseen algunos inconvenientes adicionales que es necesario solucionar para esta representación intermedia, tales como:

- Es difícil representar simultáneamente varias frases. Los Grafos Conceptuales deben, para ello, replicar el mismo concepto varias veces en el grafo y utilizar un mecanismo conocido como "correferencia", que consiste en unir los diferentes conceptos que tienen el mismo nombre con una línea discontinua. Esto puede conducir a duplicidad de información y dificultades para establecer el comportamiento de un concepto.
- En los Grafos Conceptuales un mismo elemento puede pertenecer a categorías gramaticales diferentes; por ejemplo, un rectángulo (concepto) puede ser un sustantivo, un verbo o un adjetivo, en tanto que un óvalo (relación) puede ser un caso semántico o un tipo de relación especial. La representación de diferentes tipos de palabras con los

mismos símbolos puede ser contraproducente para el análisis porque puede generar o reproducir ambigüedades que estaban presentes en el discurso original.

- Los Grafos Conceptuales sólo pueden representar información estructural y no pueden representar frases con características dinámicas, que se requerirían para un esquema intermedio unificado.

A partir de la sintaxis de los Grafos Conceptuales, es posible realizar algunas modificaciones que contribuyan a solucionar las limitaciones anotadas. Posteriormente, se puede verificar si efectivamente el esquema resultante cumple con los requisitos que se anotan en la Sección 4.1. Las siguientes modificaciones a los Grafos Conceptuales permiten la definición de la sintaxis básica de los Esquemas Preconceptuales (véase Figura 25) (Zapata et al., 2006a y 2006b):

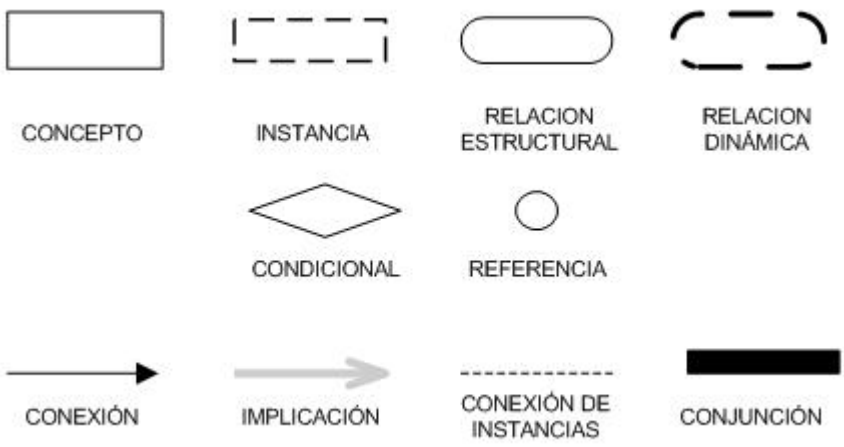


Figura 25. Sintaxis básica de los Esquemas Preconceptuales.

- Se conserva la notación correspondiente a los conceptos (un rectángulo), pero se restringen los elementos que se pueden considerar conceptos. Para los Esquemas Preconceptuales, los conceptos sólo pueden ser sustantivos (por ejemplo “profesor”, “factura”, “solicitud”, etc.) o frases nominales del tipo sustantivo+”de”+sustantivo (por ejemplo “departamento de pedidos”, “hora de salida”, “horario de atención”, etc.) o del

tipo sustantivo+adjetivo ("hora límite", "diagrama final", etc.). Además, cada concepto debe aparecer sólo una vez en el Esquema Preconceptual, por lo cual no se requieren mecanismos como la correferencia para la representación conjunta de frases individuales.

- Se conserva también la notación correspondiente a las relaciones (un óvalo), pero sólo se permite que las relaciones sean verbos. En este punto, se debe establecer una diferenciación entre los verbos (relaciones) que hacen parte del Esquema Preconceptual:
  - Relaciones estructurales (óvalos con línea continua): Son verbos que generan una relación permanente entre dos conceptos. Para los Esquemas Preconceptuales únicamente se reconocen como relaciones estructurales los verbos "ser" y "tener".
  - Relaciones dinámicas (óvalos con línea discontinua): Son verbos que denotan acciones u operaciones en el mundo, y que, por tanto, generan relaciones de tipo transitorio entre los conceptos. En otras palabras, la relación dinámica que une dos conceptos genera una conexión entre esos conceptos que sólo dura mientras la acción se ejecute, tal como lo establece Vendler (1957) cuando los clasifica como "verbos de actividad". Algunos ejemplos de este tipo de verbos son: "registrar", "pagar", "presentar", etc. Desde el punto de vista de los esquemas conceptuales, las relaciones dinámicas pueden conducir a definir ciertas relaciones permanentes entre los conceptos que unen; por ejemplo, cuando "el cliente paga la factura" se establece en el diagrama de clases un vínculo que puede ser incluso más fuerte que las relaciones estructurales definidas, pues se liga indisolublemente la factura al cliente que la paga.
- Se conservan las conexiones (flechas sencillas unidireccionales) empleadas en los Grafos Conceptuales y con su misma simbología: sirven para unir conceptos con relaciones estructurales o dinámicas y viceversa. Al conjunto conformado por un concepto, una relación estructural o dinámica y otro concepto, ligados con conexiones

en la misma dirección, se le denomina "triada", y es útil para la posterior definición del lenguaje controlado para la especificación de los Esquemas Preconceptuales.

Con el fin de dotar los Esquemas Preconceptuales con la capacidad de representar situaciones dinámicas, se proponen dos nuevos elementos para su sintaxis básica:

- Los condicionales (denotados por rombos) son conjuntos de conceptos separados por operadores básicos (suma, resta, multiplicación, división) y de comparación (mayor que, menor que, igual a, mayor o igual que, menor o igual que, diferente de), que constituyen enunciados boléanos. Ejemplos de condicionales son los siguientes: "hora de entrega mayor que hora límite", "nota mayor que tres", "diagrama inicial diferente de diagrama final", etc. Por pertenecer a un lenguaje controlado, en los condicionales aún no es posible representar reglas del negocio mucho más complejas que las anotadas.
- Las implicaciones (denotadas por flechas unidireccionales gruesas) son conexiones que expresan causalidad, en el mismo sentido de la implicación lógica; en otras palabras, se requiere que la cabeza de la flecha sea cierta para que la punta de la flecha se pueda realizar. Las implicaciones sirven para conectar condicionales con relaciones dinámicas o relaciones dinámicas entre sí. En el ejemplo "si el estudiante presenta el examen, entonces el profesor califica el examen", la implicación se establece entre las relaciones dinámicas "presenta" y "califica".

Otros elementos que se agregan a la sintaxis básica de los Esquemas Preconceptuales son:

- Las instancias (denotadas por rectángulos en línea discontinua) son conjuntos de valores que puede tomar un concepto y que sirven para aclararlo. Un conjunto de instancias se une al concepto que lo origina mediante una línea discontinua. Ejemplos de instancias para el concepto "estado"—el cual se puede ligar al concepto "solicitud"—pueden ser "aprobada", "en trámite" y "rechazada".

- Las conexiones de instancias (denotadas por una línea discontinua) son conexiones que permiten ligar un conjunto de instancias con el concepto del cual son valores.
- Las conjunciones (denotadas por líneas gruesas) son elementos que sirven para unir implicaciones. Si dos o más implicaciones llegan a una relación dinámica sin que medie una conjunción, se entiende que esas implicaciones son disjuntas.
- Las referencias (denotadas por círculos numerados) son elementos que permiten ligar conexiones desde o hacia conceptos físicamente distantes en el Esquema Preconceptual, con el fin de facilitar la legibilidad.

Se debe notar que los Esquemas Preconceptuales poseen una lógica de permisos y no de obligaciones en lo que respecta a elementos como relaciones dinámicas, condicionales e implicaciones; por ejemplo, en la Figura 26 el Esquema Preconceptual se traduce de la siguiente manera "cuando el estudiante presenta el examen, el profesor puede evaluar el examen". En otras palabras, no se establece en qué momento se realiza la acción sino que se conceden permisos para que la acción se pueda ejecutar.

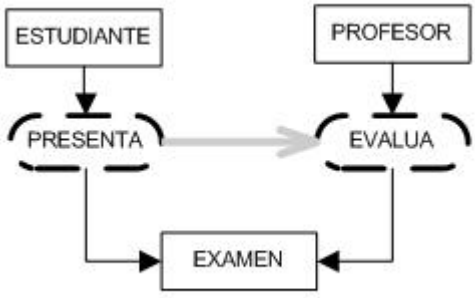


Figura 26. Ejemplo de permisos en Esquemas Preconceptuales.

4.4. UN-Lencep

El lenguaje natural ha sido durante siglos objeto de investigación, debido a la gran cantidad de irregularidades lingüísticas que presenta; fenómenos como la ambigüedad ocupan la

atención de los investigadores en el mundo y parecen ser una fuente aún no explorada en su totalidad. En lo que respecta al Procesamiento del Lenguaje Natural, algunos trabajos se han enfocado en procurar la comprensión del lenguaje natural, mientras que otros trabajos han eludido completamente esta dificultad y se han centrado en la utilización de lenguajes más precisos, menos ambiguos. Tendencias de este último tipo son las que propugnan por el uso de lenguajes controlados en algunas labores como escritura y traducción de manuales técnicos, representación del conocimiento y extracción de información, entre otras. Un lenguaje controlado es un subconjunto del lenguaje natural (generalmente en inglés) que posee restricciones en la terminología, la sintaxis y/o la semántica. Lenguajes como los grafos conceptuales empleados en el proyecto ASPIN (Cyre, 1995), para el caso de los Sistemas Digitales, poseen los tres tipos de restricciones. Otros ejemplos de lenguajes controlados son:

- Escritura y traducción de manuales técnicos: Por lo general poseen restricciones en la terminología y en algunos casos también se limita la sintaxis que se puede utilizar en este tipo de elementos. La Asociación Europea de Industrias Aeroespaciales (AECMA, 1995) posee uno de los más famosos lenguajes controlados para la elaboración de manuales técnicos de vuelo.
- Representación del conocimiento: En este tipo de aplicaciones se suelen emplear lenguajes cercanos a la lógica de predicados, dada la necesidad de contar con mecanismos de inferencia para el procesamiento de los enunciados escritos en esos lenguajes. Uno de tales lenguajes es el Knowledge Interchange Format (Formato de Intercambio de Conocimiento, KIF, 2006), el cual se suele emplear para intercambiar información entre diversas aplicaciones.
- Extracción de información: En este caso se suele limitar la sintaxis del lenguaje para buscar patrones lingüísticos que faciliten tareas como la identificación de términos o acciones que se suelen encontrar inmersas en un texto en lenguaje natural. Bajo este tópico, uno de los ejemplos que se puede citar es el CLIE (Controlled Language for

Information Extraction—Lenguaje controlado para la Extracción de Información) (Polajnar *et al.*, 2006).

- Elicitación de Requisitos: En este campo, objetivo de la presente Tesis, se han realizado pocos trabajos, algunos de los cuales se listan seguidamente:
  - Fuchs y Schwitter (1996) especificaron el lenguaje Attempto Controlled English (ACE) y lo emplearon para formular especificaciones de requisitos que posteriormente traducían al lenguaje Prolog.
  - Smith *et al.* (2003) elaboraron una aplicación que permitía escribir y comprender las propiedades de un sistema. Esta herramienta, denominada "Propel Elucidation" captura detalles de las propiedades de un sistema empleando plantillas que sirven como patrones de extracción.
  - Konrad y Cheng (2005) desarrollaron una herramienta para la especificación de sistemas, denominada SPIDER (Specification Pattern Instantiation and Derivation Environment). Esta herramienta utiliza gramáticas de lenguaje natural estructurado y sistemas de patrones de especificación.
  - Richards *et al.* (2005) propusieron RECOCASE, un entorno que se fundamenta en el desarrollo de puntos de vista. Este entorno usa un lenguaje controlado para escribir descripciones de casos de uso (capturando de esta manera las necesidades de los interesados) y posteriormente traducirlas a lattices.
  - Un uso similar al anterior le dieron Diaz *et al.* (2004b) a un lenguaje controlado sintácticamente para la escritura efectiva de descripciones de casos de uso. Se procuraba que las descripciones fueran inambiguas y se definieron reglas heurísticas para la obtención automática del diagrama de secuencias de UML.
  - Cooper (2001), empleando un lenguaje controlado sintácticamente, definió una notación denominada SRRS (Stimulus Response Requirements Specification), un lenguaje natural estructurado legible por humanos y tratable computacionalmente.

- Mueckstein (1985) propuso un método para el diseño de interfaces gráficas de usuario que mezcla SQL y lenguajes naturales. El método se denominó INTERPRET.

La totalidad de los proyectos mencionados para el uso de lenguajes controlados en Elicitación de Requisitos, restringen el lenguaje en cuanto a terminología y/o sintaxis para referirse a la solución informática a las necesidades del interesado. En otras palabras, para obtener las descripciones en lenguajes controlados se requiere un análisis previo por parte del analista, que es lo que se suele hacer en la actualidad con un proceso manual y subjetivo. Si se usaran lenguajes controlados para atender la tendencia de elaboración automática de Esquemas Conceptuales, estos lenguajes en principio deberían reflejar el discurso del dominio del interesado y no el de la solución. Además, las restricciones también apuntan a escribir discursos muy técnicos (especificaciones de sistemas o descripciones de casos de uso), que son lenguajes supremamente técnicos y para los cuales se requiere un entrenamiento adicional; por ello, los interesados poco entienden de este tipo de lenguajes. Además, únicamente el trabajo de Díaz *et al.* (2004b) se realiza en idioma Español y los demás son en inglés. Dado que el interesado es quien debe validar la información que servirá de base para la elaboración del software, es necesario para la Elicitación de Requisitos que el interesado comprenda el lenguaje que servirá de base para la elaboración de los Esquemas Preconceptuales; ello implica la eliminación de la terminología técnica relativa a la solución (descripciones de casos de uso, por ejemplo), para incorporar y respetar los términos propios del dominio del interesado.

Es de notar que no se pretende, en principio, que los interesados escriban en el lenguaje controlado ni que elaboren los Esquemas Preconceptuales; simplemente, se requiere que el interesado esté en capacidad de entender lo que se escribe en el lenguaje controlado, y validar si la información está correctamente utilizada o no. En este caso, la labor del analista será contribuir en la escritura del lenguaje controlado o la conformación de los

Esquemas Preconceptuales, reflejando el discurso del interesado y el conocimiento técnico que posee como analista, para encauzar la generación automática de los Esquemas Conceptuales.

El lenguaje controlado que se propone en esta Tesis para la superación de estas limitaciones se denomina UN-Lencep (Zapata *et al.*, 2006c), que significa Universidad Nacional de Colombia—Lenguaje Controlado para la Especificación de Esquemas Preconceptuales. A partir de un discurso expresado en UN-Lencep, debería ser posible la construcción automática de los Esquemas Preconceptuales.

UN–Lencep es similar al lenguaje CLIE (Polajnar *et al.*, 2005), que se elaboró en el entorno GATE (General Architecture for Text Engineering) (Cunningam *et al.*, 2002), 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*

De manera similar a CLIE, UN–Lencep posee un conjunto básico de plantillas que poseen diferentes formas de ser expresadas en lenguaje natural. Estas formas de expresión posibilitan una mejor interacción con el interesado y pueden considerarse una versión

"avanzada" de UN-Lencep. En UN-Lencep básico se conservan algunas de las palabras clave de CLIE, por ejemplo "have", "has", "are" o "is"; además, se agregan nuevas plantillas para la representación de elementos que no se consideraban inicialmente en CLIE, tales como condicionales e implicaciones.

La construcción formal de las palabras clave del UN-Lencep, junto con sus expresiones equivalentes en lenguaje natural controlado, se muestran en la Tabla 2. Las reglas para la traducción del UN-Lencep a Esquemas Preconceptuales están incluidas en la Figura 27. Para la Figura 27 y la Tabla 2, 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 2.
- <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.
- {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.

La Tabla 2 y la Figura 27 posibilitan la transformación de un discurso en UN-Lencep en el correspondiente Esquema Preconceptual. En oposición a lo que ocurre con los Grafos Conceptuales, se debe señalar que para un discurso en UN-Lencep existe uno y sólo un Esquema Preconceptual posible. De esta manera, se evita la ambigüedad en la generación de los Esquemas Preconcepuales. Se debe señalar, sin embargo, que la construcción del discurso que se realiza en UN-Lencep está ampliamente influenciada por el conocimiento

del dominio que expresa el interesado (o el grupo de interesados) y el conocimiento de los esquemas conceptuales que el analista (o el grupo de analistas) intenta traducir; en este sentido, pueden existir varios posibles discursos en UN-Lencep que describan las mismas ideas, pero todos ellos podrán ser validados por analistas e interesados simultáneamente. El grado de detalle de los diagramas resultantes dependerá de la forma en que analistas e interesados estructuren el diálogo correspondiente.

Tabla 2. Construcción formal de UN-Lencep y su equivalencia en expresiones en Lenguaje Natural Controlado.

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

4.5. Reglas Heurísticas de Transformación a Esquemas Conceptuales de UML

Una vez el analista y el interesado han concertado un discurso en UN-Lencep que describa el dominio del problema, y que se haya obtenido el Esquema Preconceptual correspondiente, es posible realizar una conversión de dicho esquema en los tres diagramas de UML mencionados en la sección 4.1., con las limitaciones también allí anotadas. Para

ello, se requiere un conjunto de reglas heurísticas que van utilizando tanto la información inicial del Esquema Preconceptual, así como los resultados intermedios que se van generando. Esta parte del proceso se diferencia del proyecto NIBA, en tanto que los tres diagramas se van obteniendo a partir del mismo esquema de partida (el Preconceptual) y se van retroalimentando para conseguir que los elementos de los tres diagramas sean completamente consistentes en cada uno de ellos. El conjunto completo de reglas heurísticas de transformación se puede consultar en la Tabla 3, y emplea la notación consignada en las Figuras 22 a 25.

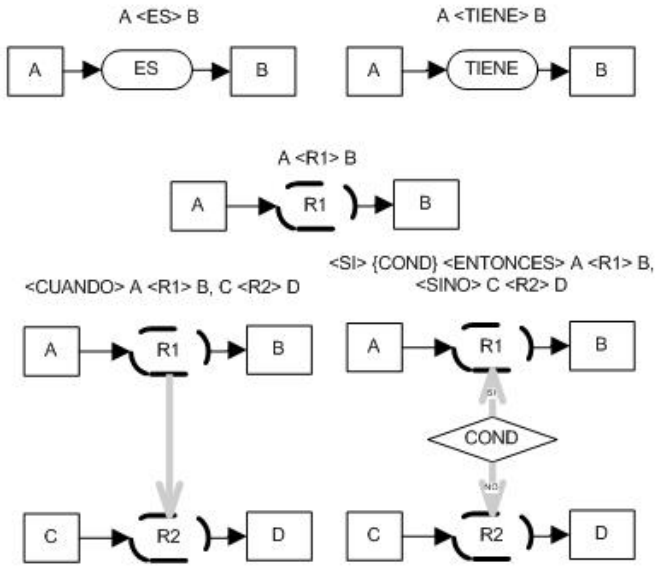
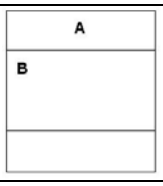
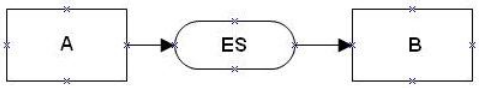
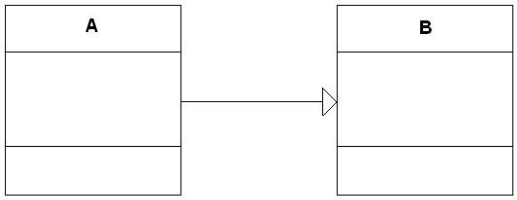
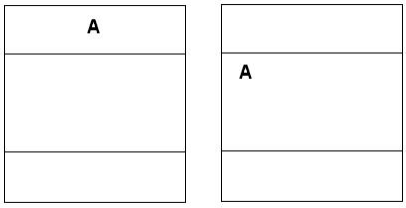
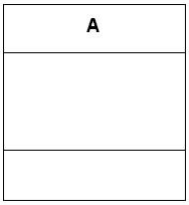
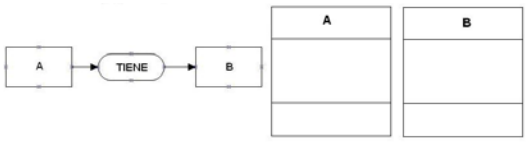
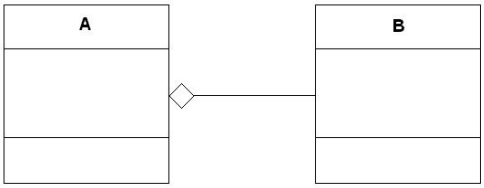
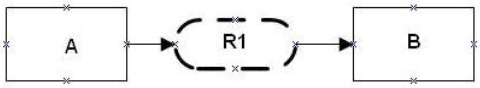
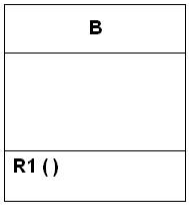
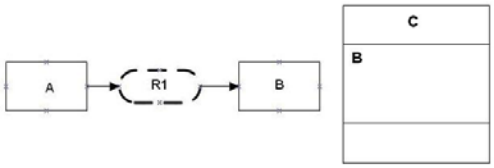
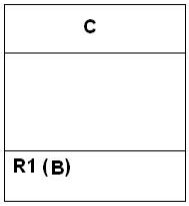
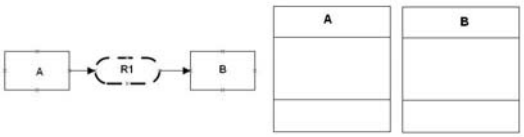
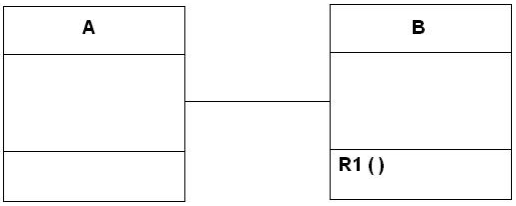
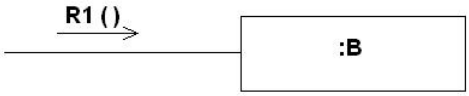
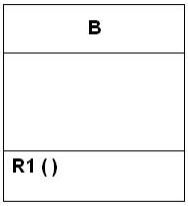
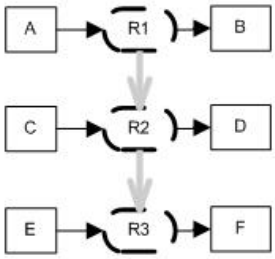
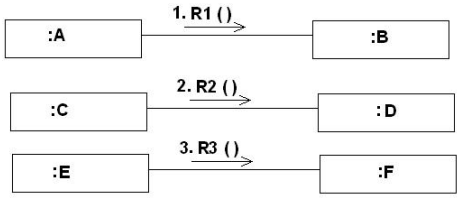
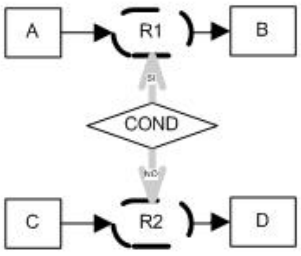
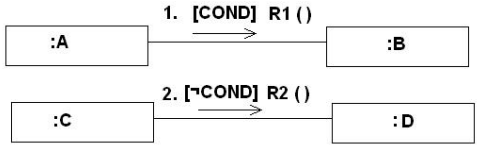
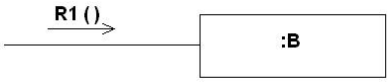
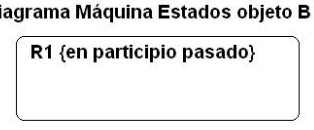


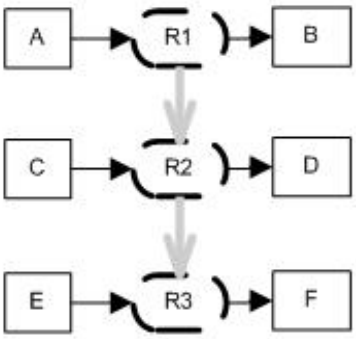
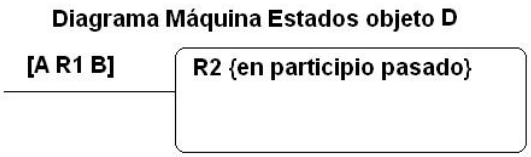
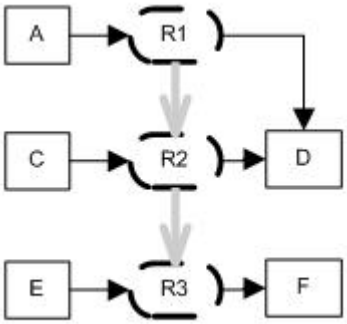
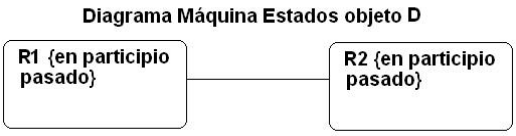
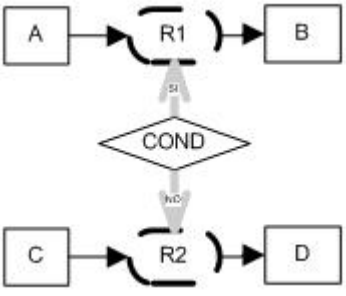
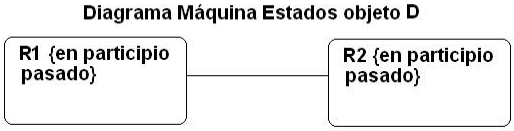
Figura 27. Reglas para la traducción de UN-Lencep en Esquemas Preconceptuales.

Tabla 3. Reglas Heurísticas de Transformación entre Esquemas Preconceptuales y Esquemas Conceptuales de UML.

No	*Precondición*	*Resultado*
1	En una relación estructural con el verbo “tiene” que liga dos conceptos A y B, el primer concepto A es una clase candidata y el concepto B es un atributo candidato de la clase A.	
		

No	*Precondición*	*Resultado*
2	En una relación estructural con el verbo "es" que liga dos conceptos A y B, ambos conceptos son clases candidatas y existe una relación de generalización en la que la clase B es la clase padre de la clase A.	
		
3	Un concepto A que simultáneamente se haya identificado como clase y como atributo por diferentes reglas será una clase.	
		
4	Si en la regla 1 ambos conceptos han sido identificados como clases candidatas, se presenta una relación de agregación entre ellas, siendo A el agregado y B la parte.	
		
5	En una relación dinámica que une dos conceptos A y B, el concepto B es una clase candidata y la relación dinámica es una operación candidata de la clase B.	
		
6	En la regla 5, si B ha sido identificado previamente como atributo candidato de una clase C, entonces la relación dinámica es una operación candidata de la clase C y el concepto B es un parámetro de dicha operación.	
		

No	*Precondición*	*Resultado*
7	En la regla 5, si los dos conceptos A y B han sido identificados como clases candidatas, se genera adicionalmente una relación de asociación entre las clases.	
		
8	Un diagrama de comunicación con un mensaje R1 que llega a una clase de objeto B, se genera una clase B en el diagrama de clases con una operación R1.	
		
9	Una cadena de implicaciones que une relaciones dinámicas genera una secuencia numerada de mensajes en el diagrama de comunicación.	
		
10	Las implicaciones que salen de un condicional generan condiciones de guarda en los mensajes del diagrama de comunicación.	
		
11	Un mensaje R1 que llega a un objeto de clase B en el diagrama de comunicación, genera un mensaje R1 (expresado en participio pasado) en el diagrama de máquina de estados del objeto B	
		

No	*Precondición*	*Resultado*
12	Una implicación sobre relaciones dinámicas R1 y R2 genera en el diagrama de máquina de estados una condición de guarda R1 y un estado R2 expresado en participio pasado, siempre y cuando los conceptos de llegada de R1 y R2 sean diferentes.	
		
13	Si en la regla 12 los conceptos de llegada de las relaciones dinámicas R1 y R2 son los mismos (en este caso, D), se generan dos estados (expresados en participio pasado) sobre la clase de objeto D.	
		
14	La implicación negativa de un condicional genera en el diagrama de máquina de estados una condición de guarda que se compone de la condición y su negación y que se aplica a la clase de objeto denotada por el concepto de llegada (en este caso, D).	
		
15	En un diagrama de máquina de objetos completamente identificado se debe adicionar un estado inicial.	
		

4.6. UNC-Diagramador: una Herramienta CASE basada en UN-Lencep y Esquemas Preconceptuales para el Trazado automático de Diagramas de UML

Basada en la Hipótesis preliminar planteada en el Capítulo 1, la herramienta CASE UNC-Diagramador procura resolver algunas de las limitaciones que aún subsisten en la obtención automática de Esquemas Conceptuales de UML 2.0 a partir de lenguaje natural. Esta herramienta viene siendo desarrollada por el Grupo de Ingeniería de Software de la Escuela de Sistemas de la Universidad Nacional de Colombia, empleando las ventajas de la tecnología .NET de Microsoft® y su lenguaje C#, y combinándolas con las posibilidades gráficas de Microsoft Visio®. Además, el módulo de manejo del UN-Lencep se desarrolló en lenguaje PHP.

UNC-Diagramador realiza las siguientes funciones:

- Permite el ingreso de frases en lenguaje controlado que se convierten en un archivo en Un-Lencep.
- Convierte el archivo en UN-Lencep en el Esquema Preconceptual correspondiente.
- Obtiene automáticamente los diagramas de clases, comunicación y máquina de estados de UML 2.0, correspondientes al Esquema Preconceptual generado, con las limitaciones establecidas en la Sección 4.1.

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

El propietario posee una mascota
La identificación es un elemento de una mascota
Un nombre pertenece a una mascota
Una mascota posee una historia_clínica
El número es un elemento de una historia_clínica
El detalle es un elemento de una historia_clínica

La fecha es un elemento de detalle
El detalle contiene un diagnóstico
El detalle contiene un medicamento
Dado que el propietario pide una cita, la secretaria asigna la cita
Si el propietario cumple la cita, el veterinario revisa la mascota
Luego que el veterinario revisa la mascota, el veterinario registra el diagnóstico
Si el veterinario registra el diagnóstico, entonces el veterinario receta un medicamento

El módulo inicial de UNC-Diagramador se encarga del procesamiento de este tipo de frases para obtener una versión del discurso en UN-Lencep; para ello, se vale de una interfaz como la que se muestra en la Figura 28. Cuando se ha ingresado la totalidad de las frases, al oprimir el vínculo "Descargar Archivo UN-Lencep" se puede conseguir el archivo que contiene el discurso expresado en UN-Lencep. Para el ejemplo, ese archivo tiene la forma siguiente:

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

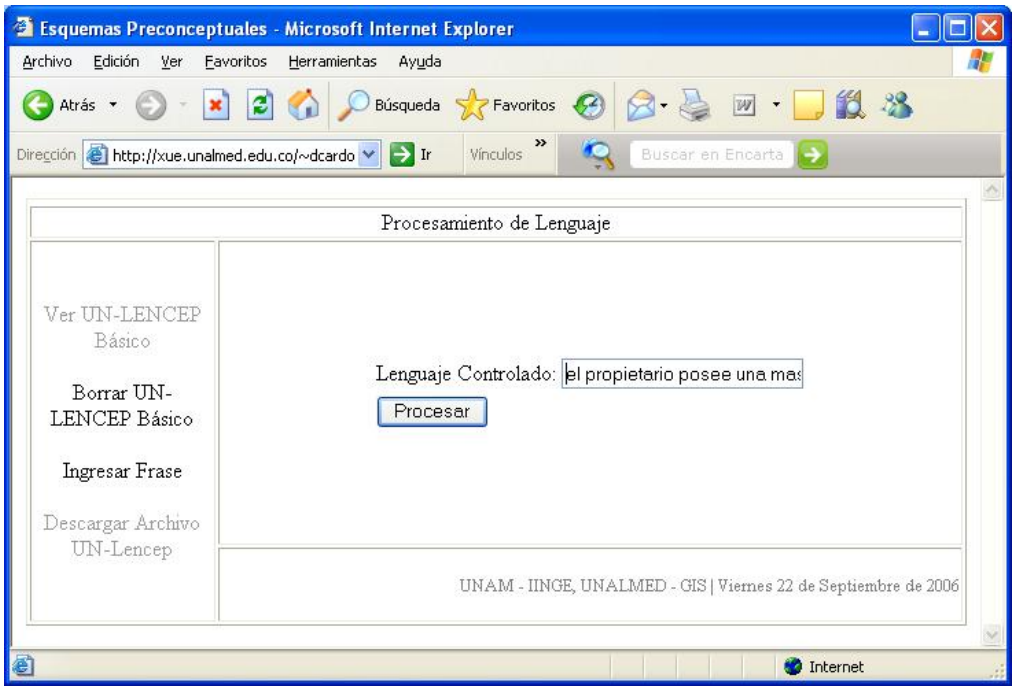


Figura 28. Imagen del primer módulo de UNC-Diagramador para el procesamiento de frases en lenguaje natural controlado.

La abreviatura ST denota una frase de tipo estructural e IM denota una frase de implicación; otras frases que se pueden traducir son DI para frases dinámicas e IN para instancias. Esta especificación se puede leer con el UNC-Diagramador para obtener el Esquema Preconceptual que se muestra en la Figura 29. Finalmente, luego de presionar el ícono de UML ubicado en la parte superior derecha de la Figura 29, se aplican las reglas heurísticas para obtener los tres diagramas de UML 2.0 que se mencionan en la Sección 4.1; estos diagramas se muestran en las Figuras 30, 31 y 32.

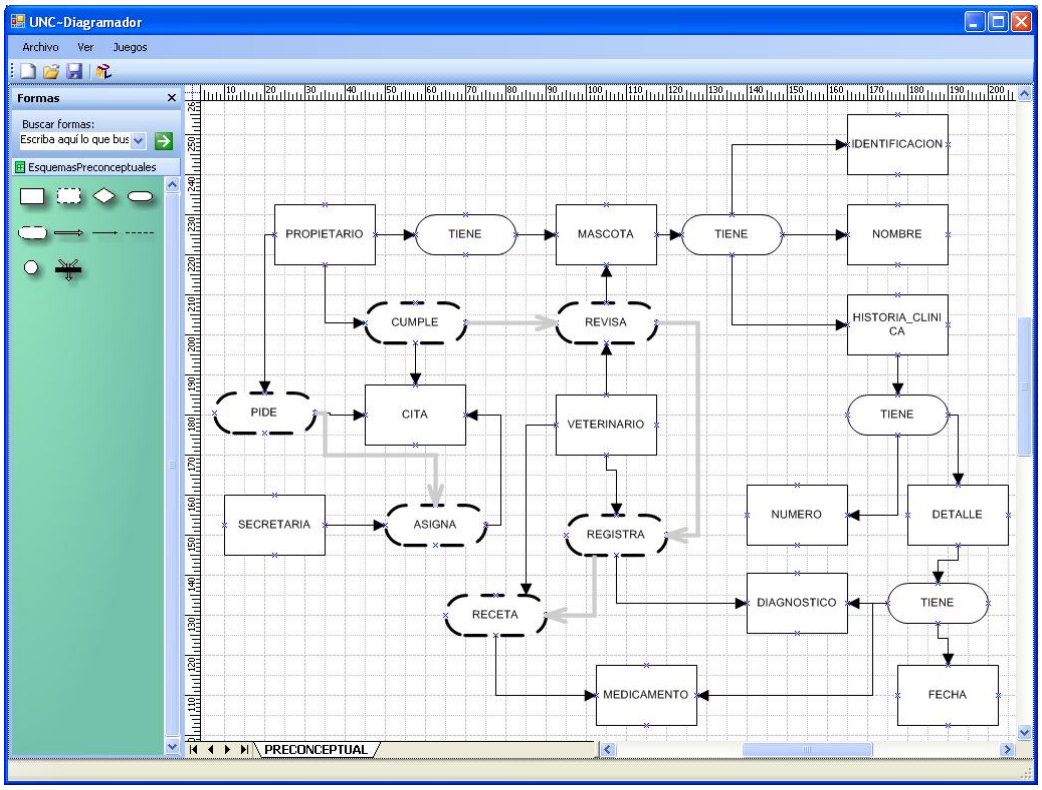


Figura 29. Imagen del Esquema Preconceptual resultante para la Clínica Veterinaria.

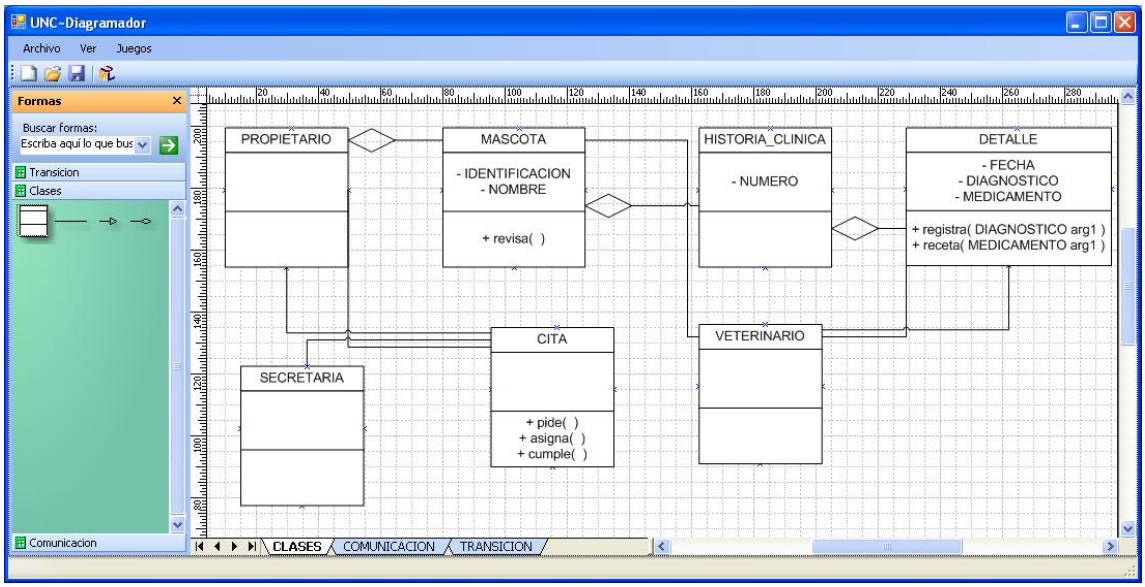


Figura 30. Diagrama de clases resultante en UNC-Diagramador para la Clínica Veterinaria.

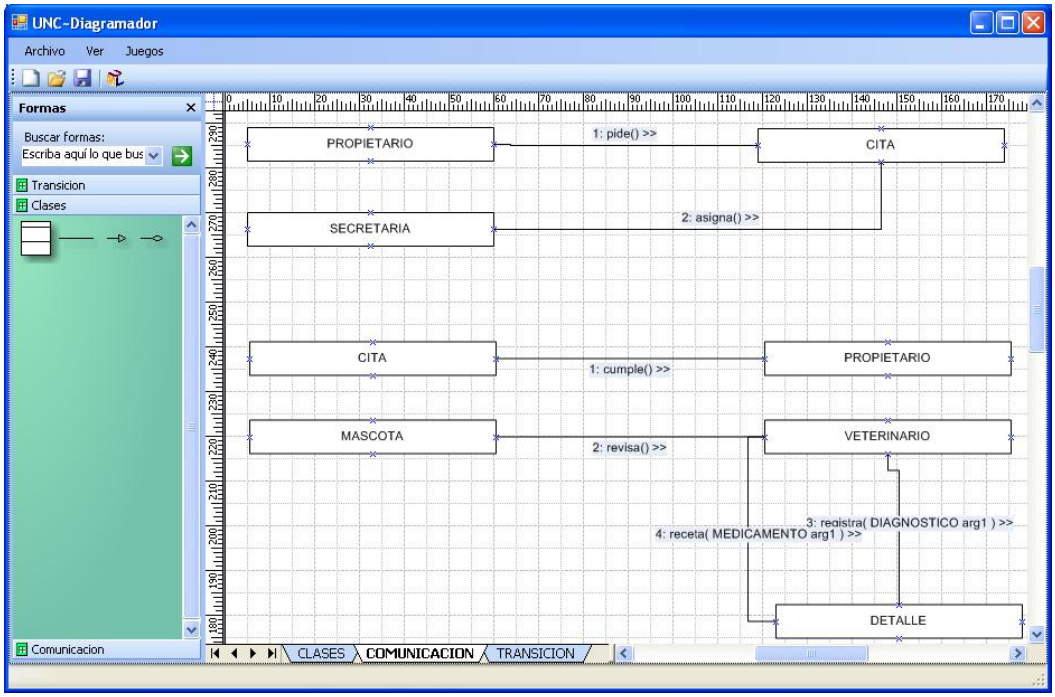


Figura 31. Diagrama de comunicación resultante en UNC-Diagramador para la Clínica Veterinaria.

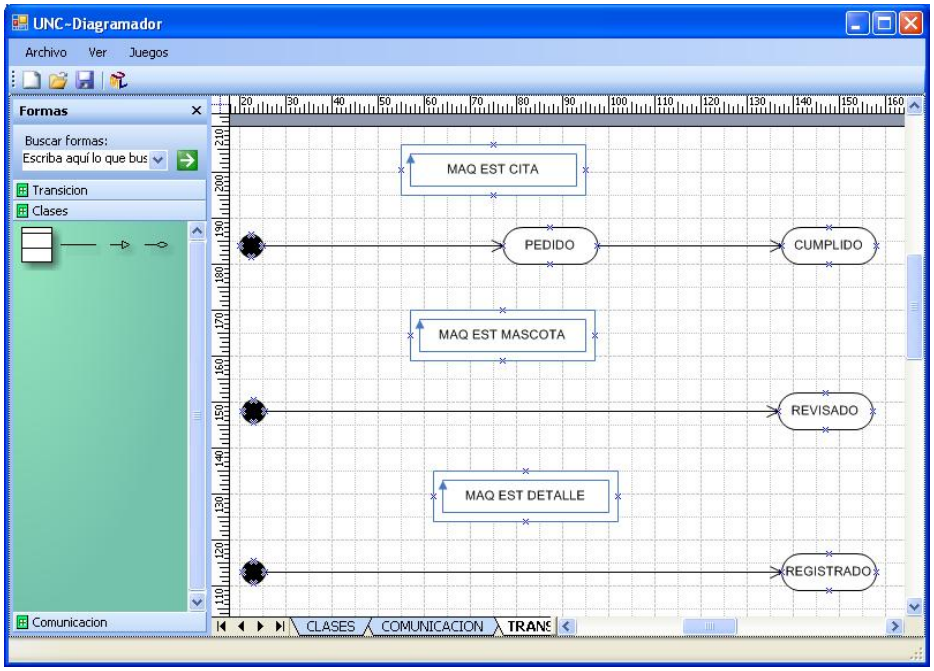


Figura 32. Diagramas de máquina de estados resultantes en UNC-Diagramador para la Clínica Veterinaria.

5. VALIDACIÓN DE LA PROPUESTA DE SOLUCIÓN

En este Capítulo se muestran los casos de estudio para establecer una comparación entre los trabajos del estado del arte y el entorno que se propone en esta Tesis para la obtención automática de esquemas conceptuales de UML a partir de lenguaje natural; para ello, se muestran los enunciados que hacen parte de las diferentes propuestas y los diagramas que se generan en cada caso, para luego presentar el discurso en UN-Lencep que se podría obtener (en este caso de manera manual) a partir del enunciado dado, para mostrar los diagramas UML que se pueden generar y establecer las comparaciones en cada caso. En la segunda parte de este Capítulo se discuten los resultados de un conjunto de experimentos realizados para mostrar la utilidad del UN-Lencep y los Esquemas Preconceptuales para la elaboración automática de diagramas de UML a partir de un lenguaje controlado. Finalmente, en la tercera parte del Capítulo se presentan las publicaciones que se han realizado de este trabajo en diferentes eventos y revistas.

5.1. Casos de Estudio

5.1.1. RADD

El ejemplo tomado del artículo de Albrecht *et al.* (1998) tiene el siguiente enunciado, conducido en una especie de diálogo simulado entre un analista y un interesado:

A water has a concentration of iones.
How is –water- characterized?
A water has a name and a site.
How is –concentration of ions- characterized?
The type, the kind and the term.
Are there any more details?
Lakes, rivers and ponds are waters.
Does –lake- have to characterized any more?
A lake has a size and a depth

Does –river- have to characterized any more? If yes, how?
A river has a length.
Are there any more details?
Several points of entry are defined for waters
In what way is the action –define- carried out?
With a number of a data paper.
How is point of entry characterized?
Place, depth, vector of flowing.
Are there any more details?
Ions are measured on every point of entry.
In what way is the action –measure- carried out?
A concentration is measured on a particular day and a particular depth.
Are there any more details?
Waters are populated by faunae

En la Figura 33 se puede apreciar el diagrama entidad-relación que fue reportado por los autores como generado a partir de este enunciado.

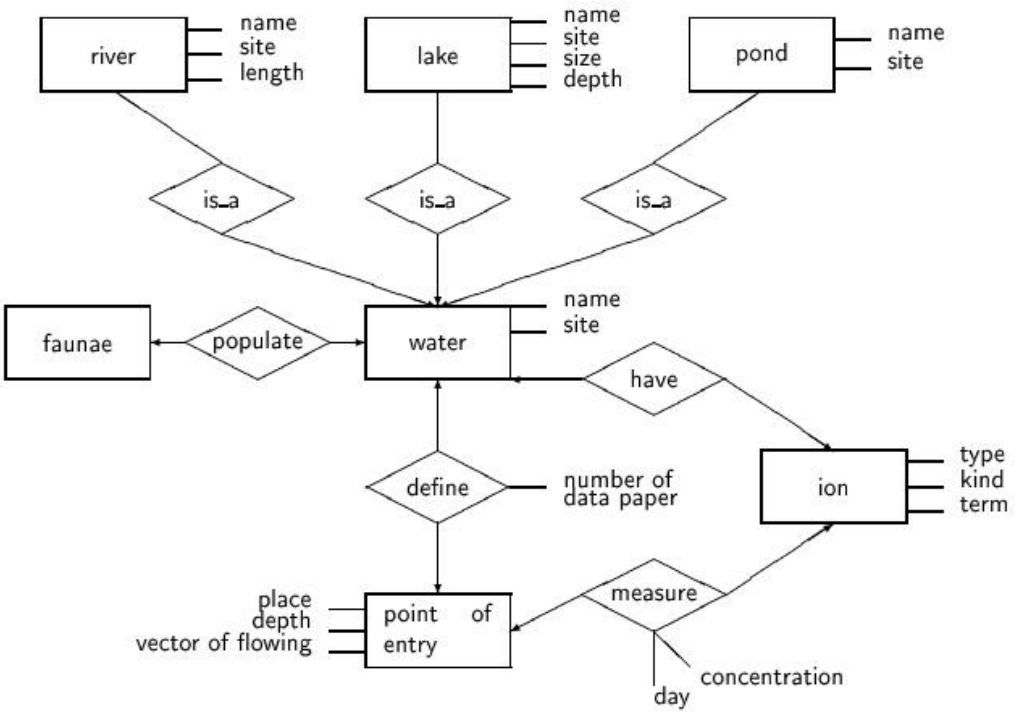


Figura 33. Un diagrama entidad-relación generado en RADD a partir de un enunciado en forma de diálogo.

Una traducción en lenguaje controlado del anterior enunciado, y que se puede emplear para ser ingresado en el módulo UN-Lencep del UNC-Diagramador, es el siguiente:

> *Un acuifero esta compuesto por ion; un acuifero posee un nombre; un acuifero posee un emplazamiento; un ion posee tipo; un ion posee clase; un ion posee concepto; un lago es un tipo de acuifero; un rio es una clase de acuifero; un estanque es una clase de acuifero; un lago incluye un tamano; un lago posee una profundidad; un rio posee longitud; un punto_de_entrada pertenece a un acuifero; un punto_de_entrada posee un registro; un punto_de_entrada posee un lugar; una profundidad pertenece a un punto_de_entrada; un vector_de_flujo pertenece a un punto_de_entrada; el recolector realiza una medicion; un punto_de_entrada pertenece a una medicion; un ion pertenece a una medicion; una medicion posee dia; una medicion posee concentracion; la fauna es un elemento de un acuifero*

El resultado arrojado por este módulo es el siguiente discurso, que tiene el formato de la construcción formal de UN-Lencep, y que se puede utilizar para los siguientes pasos del proceso:

> *ST ACUIFERO tiene ION*
> *ST ACUIFERO tiene NOMBRE*
> *ST ACUIFERO tiene EMPLAZAMIENTO*
> *ST ION tiene TIPO*
> *ST ION tiene CLASE*
> *ST ION tiene CONCEPTO*
> *ST LAGO ES ACUIFERO*
> *ST RIO ES ACUIFERO*
> *ST ESTANQUE ES ACUIFERO*
> *ST LAGO tiene TAMANO*
> *ST LAGO tiene PROFUNDIDAD*
> *ST RIO tiene LONGITUD*
> *ST ACUIFERO tiene PUNTO_DE_ENTRADA*
> *ST PUNTO_DE_ENTRADA tiene REGISTRO*
> *ST PUNTO_DE_ENTRADA tiene LUGAR*
> *ST PUNTO_DE_ENTRADA tiene PROFUNDIDAD*
> *ST PUNTO_DE_ENTRADA tiene VECTOR_DE_FLUJO*
> *RD RECOLECTOR REALIZA MEDICION*

ST MEDICION tiene PUNTO_DE_ENTRADA
ST MEDICION tiene ION
ST MEDICION tiene DIA
ST MEDICION tiene CONCENTRACION
ST ACUIFERO tiene FAUNA

Este discurso en UN-Lencep genera el Esquema Preconceptual de la Figura 34 y posteriormente el diagrama de clases de la Figura 35.

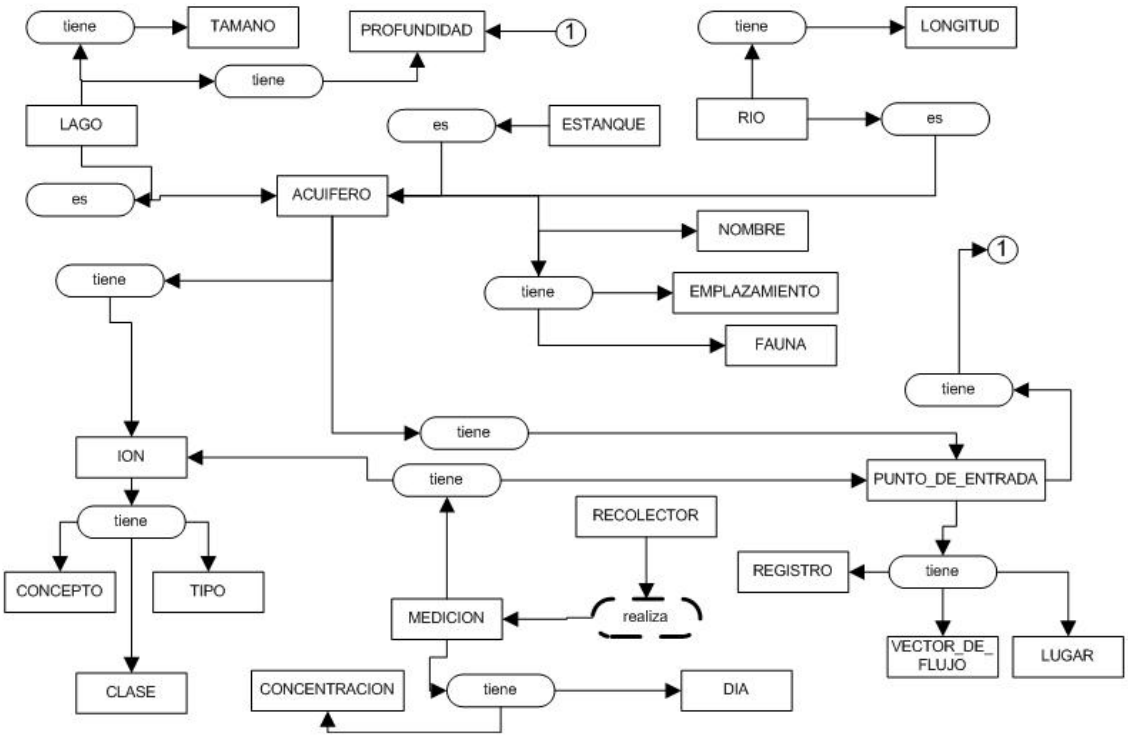


Figura 34. Esquema Preconceptual resultante en UNC-Diagramador para el ejemplo de RADD.

El diagrama de clases de la Figura 35 tiene muchas similitudes con el diagrama entidad-relación de la Figura 33. Sin embargo, existen diferencias que es importante resaltar:

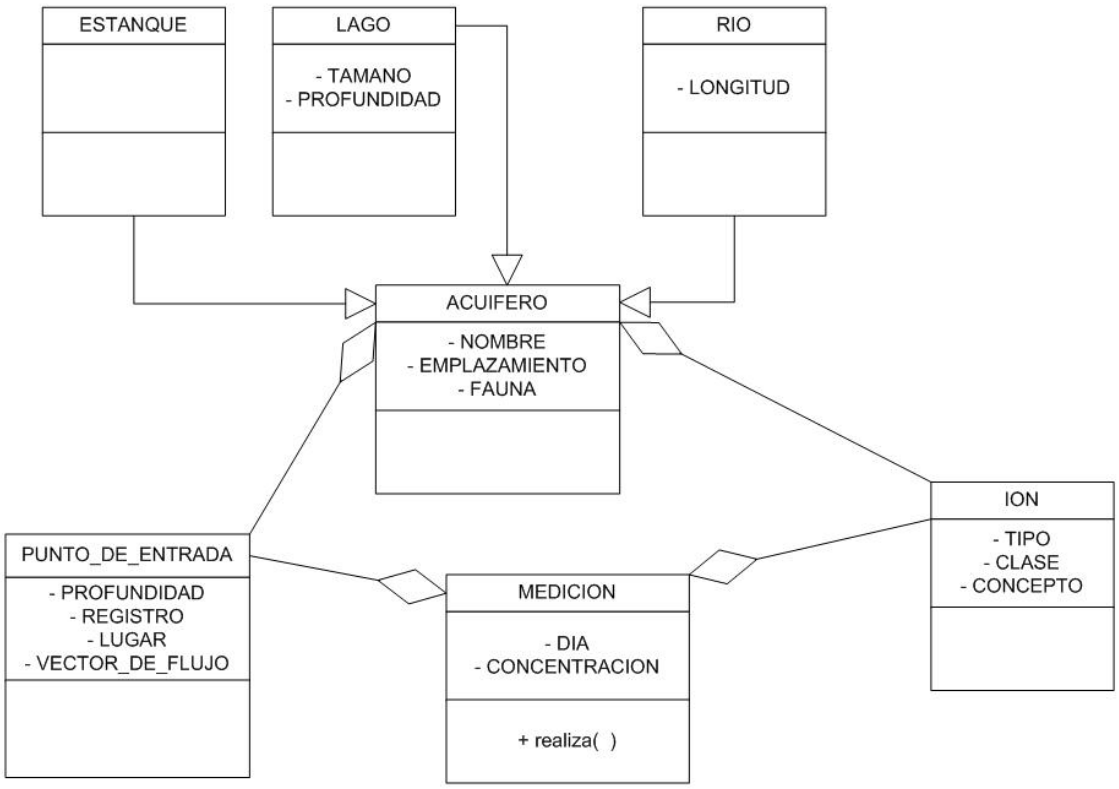


Figura 35. Diagrama de clases resultante en UNC-Diagramador para el ejemplo de RADD.

- El discurso tiene como característica principal el hecho de que únicamente se refiere a la descripción que se denomina "estructural", en el sentido definido por Juristo y Moreno (2000). Por esta razón, el único de los diagramas de UML que se genera en UNC-Diagramador es el diagrama de clases. En el caso de RADD, el diagrama entidad-relación de la Figura 33 posee como entidades las mismas clases que el diagrama de clases de la Figura 35, a excepción de "Medición", la cual implícitamente en el discurso se puede entender como el resultado de la acción "medir", cuando se establece que "una concentración es medida" en el discurso original. El hecho de que aparezca esa acción, implica que se debería saber quién la realizó (en el discurso en lenguaje controlado se supuso un "Recolector", que finalmente no modifica la estructura del diagrama, pero sí

complementa la información de su construcción), lo cual da origen a la acción "realiza" pertenciente a la clase "medición".

- La relación entre "fauna" y "acuífero", que se define en el enunciado de RADD mediante la acción "poblar" se refiere a una relación de pertenencia, que es de tipo estructural. Por ello, en el diagrama de clases resultante dejó de ser una clase para convertirse en un atributo.
- Igual ocurre con la relación "define" entre acuífero y punto_de_entrada, en la cual el segundo es una característica que describe el primero, entregando una relación de tipo estructural, lo cual se representa en el diagrama resultante como una relación de agregación entre ambas clases.
- En el diagrama entidad-relación, los atributos comunes entre "acuífero" y sus hijos podrían indicar que se pudieran redefinir esos atributos (que son "nombre" y "emplazamiento") en cualquiera de los hijos. En el caso del discurso en UN-Lencep se evitan esas repeticiones y es por ello que en el diagrama de clases resultante se aprovecha la estructura de generalización para sólo endilgarle a la clase "acuífero" los atributos comunes a sus subclases. Esto podría dar lugar a una norma futura de refinamiento, para determinar relaciones de generalización que no se declaren explícitamente dentro del discurso.

5.1.2. Juristo *et al.*

El ejemplo que presentan estos autores (Juristo *et al*, 1999, Moreno *et al.*, 2000) para ejemplificar su entorno de generación automática de esquemas conceptuales se divide en dos tipos de enunciados: uno estático y uno dinámico. El enunciado estático es el siguiente:

> *Vendors may be sales employees or companies.*
> *Employees receive a basic wage and a commission.*
> *Companies receive a commission.*
> *Each order corresponds to one vendor.*
> *Each vendor has made at least one order.*

An order is identified by an order number.
One commission may be paid to several employees and several companies.
One basic wage may be paid to several employees.

Y el enunciado dinámico es el siguiente:

Order Management:
If and only if a vendor makes a sale, then the vendor reports the order to the system.
If and only if a vendor reports the order to the system, then the system confirms the order to the customer.
If and only if the system confirms the order to the customer and the week ends, then the company delivers the order to the customer.

Cada uno de estos enunciados genera un diagrama diferente: el enunciado estático genera el diagrama de objetos en la notación OMT de Rumbaugh y el enunciado dinámico genera el diagrama de comportamiento de Martin. Esos diagramas se pueden apreciar en las Figuras 36 y 37 respectivamente.

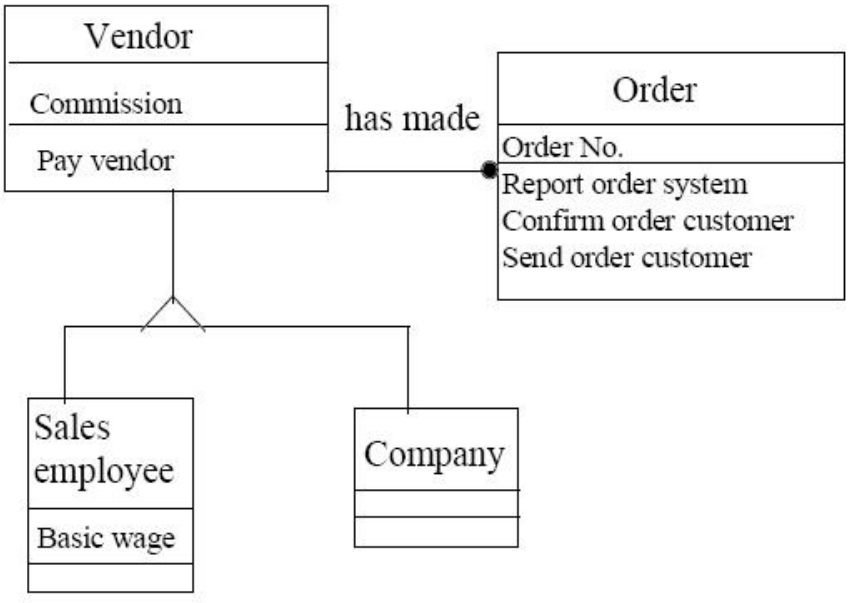


Figura 36. Diagrama OMT para el enunciado estático de Moreno *et al.* (2000).

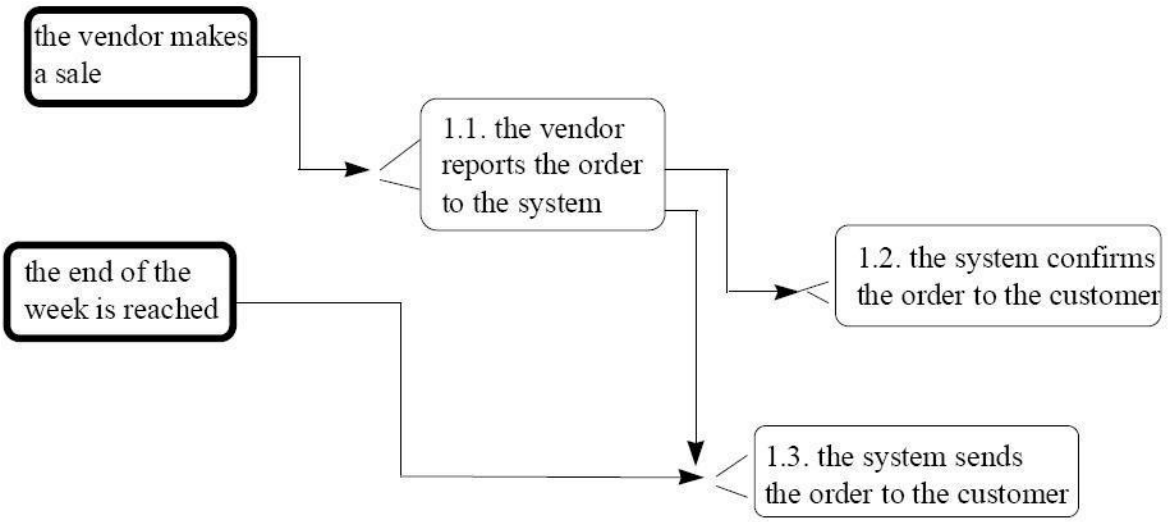


Figura 37. Diagrama de comportamiento para el enunciado dinámico de Moreno *et al.* (2000).

En lenguaje controlado, el discurso se puede traducir de la siguiente manera:

> *Un empleado es una clase de vendedor; una compania es un tipo de vendedor; un vendedor posee una comision; un salario_basico pertenece a un empleado; una orden incluye un vendedor; un numero es un elemento de una orden; cuando un vendedor realiza una venta, el vendedor reporta la orden; cuando el vendedor reporta la orden, el sistema confirma la orden; un cliente pertenece a una orden; si dia="viernes" entonces la compania entrega la orden; si el sistema confirma la orden entonces la compania entrega la orden*

Además, el discurso resultante en UN-Lencep es el siguiente:

> *ST EMPLEADO ES VENDEDOR*
> *ST COMPANIA ES VENDEDOR*
> *ST VENDEDOR tiene COMISION*
> *ST EMPLEADO tiene SALARIO_BASICO*
> *ST ORDEN tiene VENDEDOR*
> *ST ORDEN tiene NUMERO*
> *IM Cuando VENDEDOR REALIZA VENTA entonces VENDEDOR REPORTA ORDEN*

IM Cuando VENDEDOR REPORTA ORDEN entonces SISTEMA CONFIRMA ORDEN
ST ORDEN tiene CLIENTE
CO Si DIA=VIERNES, COMPANIA ENTREGA ORDEN
IM Cuando SISTEMA CONFIRMA ORDEN entonces COMPANIA ENTREGA ORDEN

El Esquema Preconceptual resultante del anterior discurso en UN-Lencep se presenta en la Figura 38. Los diagramas resultantes en UNC-Diagramador se muestran en las Figuras 39, 40 y 41.

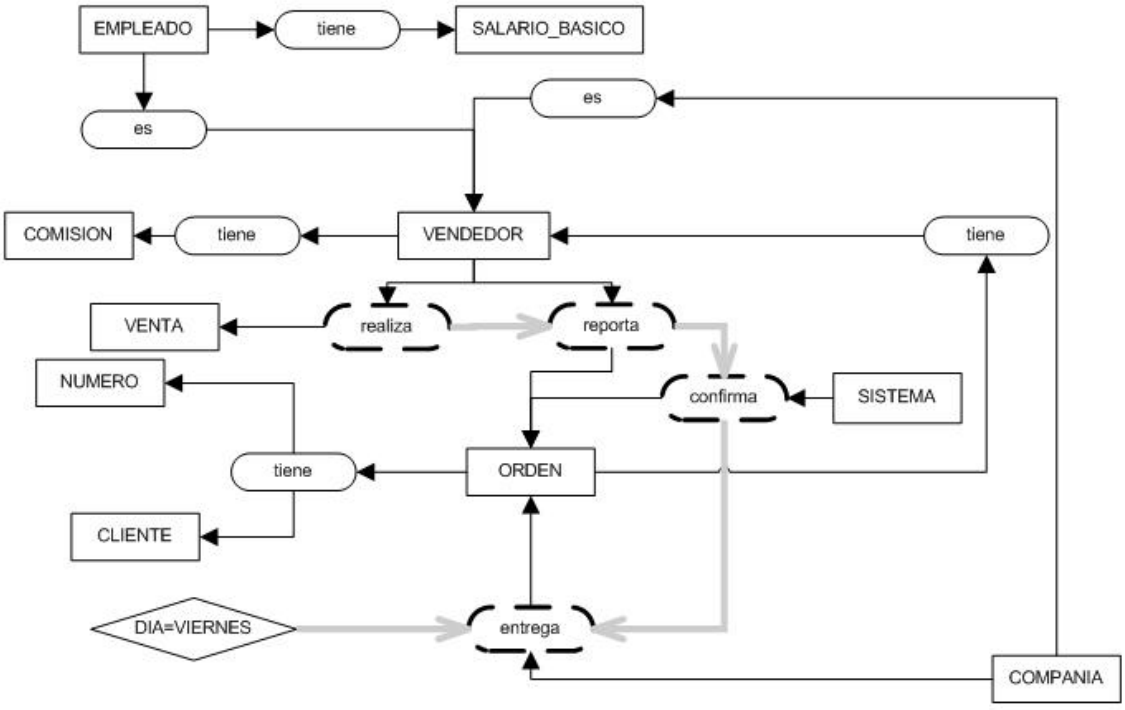


Figura 38. Esquema Preconceptual resultante en UNC-Diagramador a partir del discurso en UN-Lencep basado en Moreno *et al.* (2000).

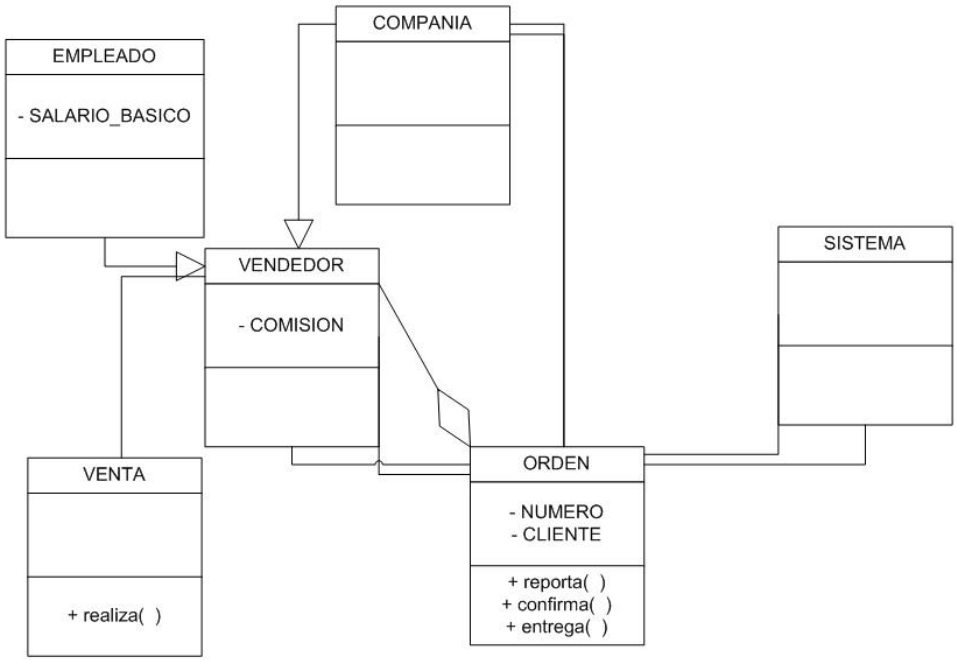


Figura 39. Diagrama de clases resultante del Esquema Preconceptual de la Figura 38.

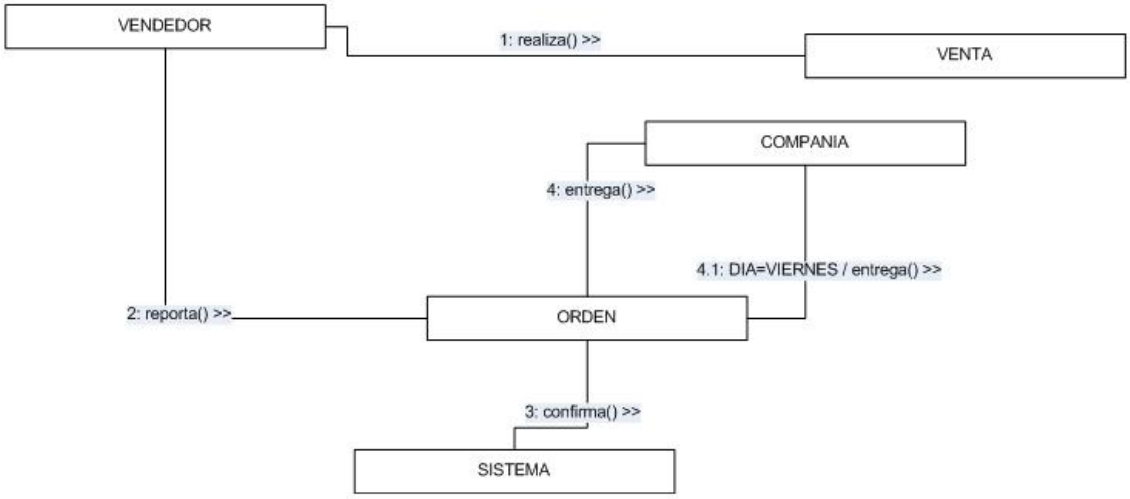


Figura 40. Diagrama de comunicación resultante del Esquema Preconceptual de la Figura 38.

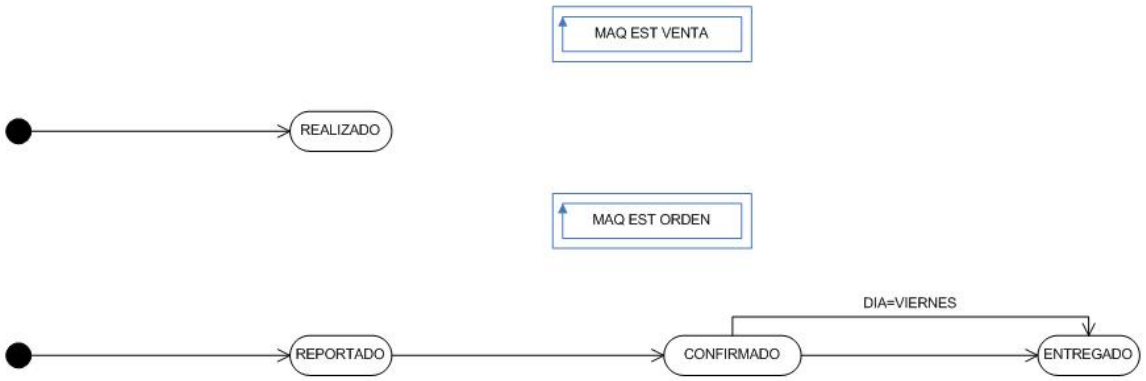


Figura 41. Diagrama de máquina de estados resultante del Esquema Preconceptual de la Figura 38.

Entre el proceso descrito por Moreno *et al.* (2000) y los resultados entregados por el UNC-Diagramador, existen algunas diferencias básicas, a saber:

- A partir del discurso en UN-Lencep se generan los diagramas de clases, comunicación y máquina de estados. En el trabajo de Moreno *et al.* (2000) se genera el diagrama OMT, que es una especie de diagrama de clases, y el diagrama de comportamiento, que tiene similitud con los diagramas de interacción de UML, por ejemplo el diagrama de comunicación o el de secuencias, pero con la desventaja de que no se pueden agrupar los mensajes que llegan al mismo objeto.
- Las reglas de Moreno *et al.* (2000) podrían generar la cardinalidad del diagrama OMT, pero no la muestran en el diagrama. Las reglas de conversión a partir de Esquemas Preconceptuales aún no incluyen la cardinalidad.
- La frase de Moreno *et al.* (2000) "el sistema confirma la orden al cliente" realmente revela dos frases en el discurso en UN-Lencep: "el sistema confirma la orden" y "la orden tiene cliente".
- Se revela un posible problema de consistencia en el hecho de que la palabra compañía que se encuentra presente en "los vendedores pueden ser compañías" es diferente de "la

compañía entrega la orden". En el primer caso, parece referirse a una empresa contratista; en el segundo caso, parece referirse a la compañía que agrupa los diferentes vendedores. Si ese es el caso, la última frase podría cambiarse por el rol específico que hace la entrega al cliente al interior de la compañía para eliminar la ambigüedad; por ejemplo, podría cambiarse en el discurso UN-Lencep por: "El asistente entrega la orden".

- Nótese que en UN-Lencep no es necesario dividir el discurso en parte estática y parte dinámica. A este respecto, Moreno *et al.* (2000) defienden que los dos tipos de discurso (el estructural y el dinámico), no pueden coexistir. En el discurso en UN-Lencep y en los Esquemas Preconceptuales esos discursos no sólo coexisten, sino que contribuyen mutuamente para la elaboración de los diagramas.
- Moreno *et al.* (2000) desechan parte de la información con sus reglas. Por ejemplo, la frase "el vendedor realiza una venta" no tiene una equivalencia en el diagrama de clases resultante. En el caso del UN-Lencep y el consecuente Esquema Preconceptual, aparece una clase "venta" en el mundo, con una operación "realiza". Igualmente, surge el cliente al interior de la orden, que en el diagrama de clases de Moreno *et al.* (2000) ni siquiera se vislumbraba; cabe anotar que parecería ser que "cliente" tendiera a clase y no a atributo, pero el discurso no presenta más información al respecto.
- En los diagramas provenientes de los Esquemas Preconceptuales surge una clase "Sistema". Esto se debe a que el discurso se refiere a un sistema, pero este concepto podría ser sustituido por el responsable de ejecutar la acción en el dominio (por ejemplo el contador o el director administrativo). En tanto la palabra "sistema" surja en el mundo, será difícil eliminarla de los diagramas, ya sea como clase del diagrama de clases u objeto del diagrama de comunicación y podría enmascarar las funciones de actores que podrían suministrar información valiosa para la construcción de otros diagramas. Si bien los Esquemas Preconceptuales se han concebido para describir el dominio del problema antes del surgimiento de la aplicación de software, es posible

también elaborar un discurso que hable del sistema y realizar el proceso de traducción que se ha ejemplificado en esta Tesis.

5.1.3. KISS

Con el fin de ilustrar su trabajo en interpretación del lenguaje natural hacia esquemas conceptuales (particularmente, el esquema denominado CPL—Concept Prototyping Language—que es una especie de lenguaje similar a las máquinas de estados pero con más información), Burg y Van de Riet (1996) presentan el siguiente enunciado:

> *A member can borrow a book from the library with his pass. The user should return the book within three weeks.*

Los diagramas en CPL que se originan a partir de este enunciado se muestran en la Figura 42. El primero de ellos consiste en una separación de la frase en roles. El segundo, en cambio, es una especie de diagrama de estados, que contiene información en un lenguaje declarativo al interior de los estados. Por ejemplo, para el caso del verbo "borrow" está conformado por un agente, una fuente, una meta y un tiempo. Esos roles se usan en el segundo diagrama para complementar las acciones al interior de los "estados".

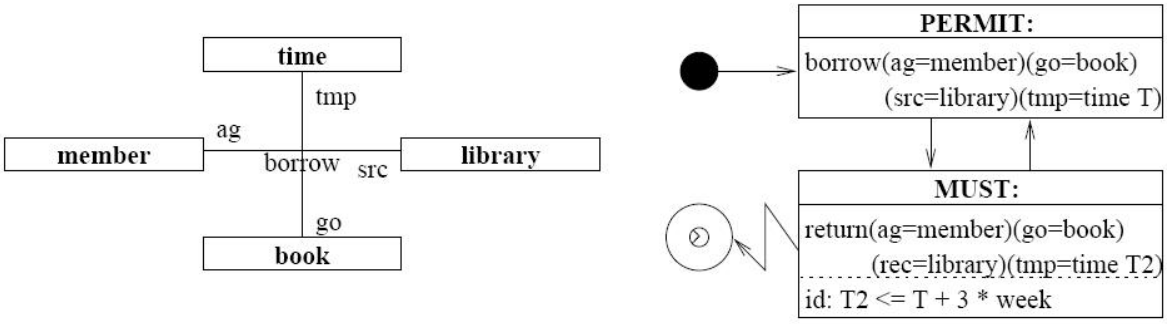


Figura 42. Diagramas en CPL resultantes del enunciado de Burg y Van de Riet (1996).

Ese enunciado se puede transformar en el siguiente discurso en lenguaje controlado:

> *un miembro tiene carnet; un miembro realiza un prestamo; un prestamo posee un miembro; un prestamo tiene fecha; un prestamo posee un libro; si fecha_del_sistema=prestamo.fecha+21 entonces el miembro regresa el libro*

El discurso en UN-Lencep correspondiente al enunciado anterior, es:

> *ST MIEMBRO tiene CARNET*
> *RD MIEMBRO REALIZA PRESTAMO*
> *ST PRESTAMO tiene MIEMBRO*
> *ST PRESTAMO tiene FECHA*
> *ST PRESTAMO tiene LIBRO*
> *CO Si FECHA_DEL_SISTEMA=PRESTAMO.FECHA+21, MIEMBRO REGRESA LIBRO*

El Esquema Preconceptual, así como los respectivos diagramas de clases, comunicación y máquina de estados, generados por el UNC-Diagramador se presentan en las Figuras 43 a 46.

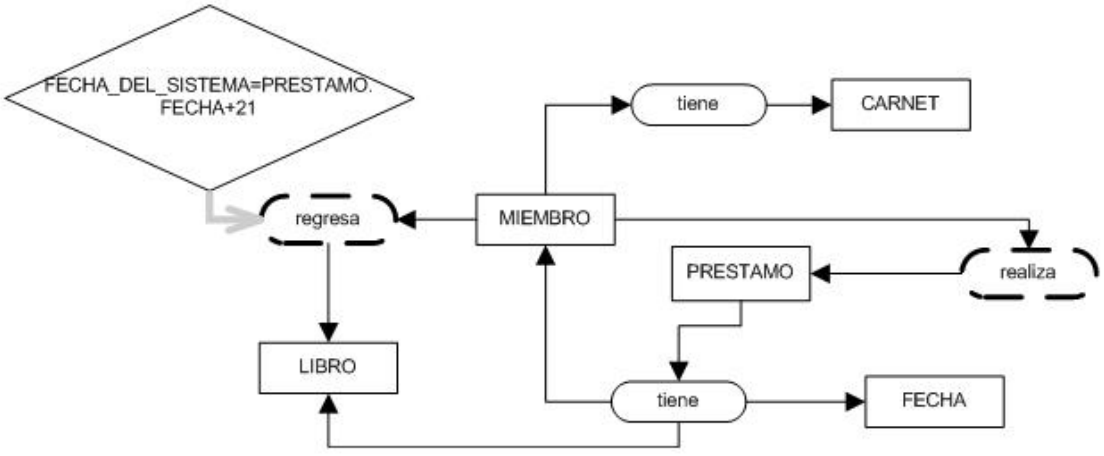


Figura 43. Esquema Preconceptual resultante del discurso en UN-Lencep basado en el ejemplo de KISS.

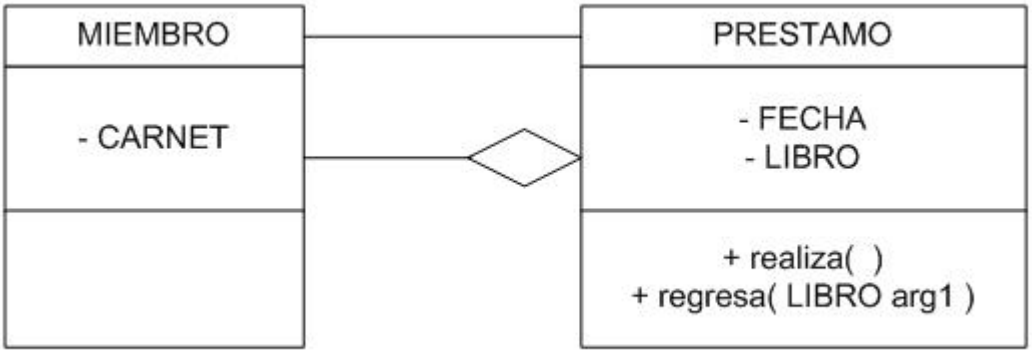


Figura 44. Diagrama de clases generado a partir del Esquema Preconceptual de la Figura 43.

Figura 45. Diagrama de comunicación generado a partir del Esquema Preconceptual de la Figura 43.

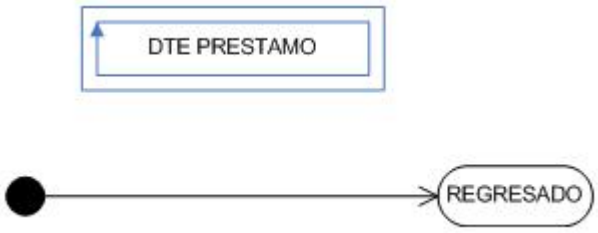


Figura 46. Diagrama de máquina de estados generado a partir del Esquema Preconceptual de la Figura 43.

Algunas observaciones en relación con este proceso son las siguientes:

- Los diagramas CPL que se generan en el proyecto KISS no pertenecen al estándar de UML. El primer diagrama parece más estructural y el segundo más dinámico, con una alta dosis este último de información en una forma de lógica de predicados. En UNC-

Diagramador es posible generar los diagramas de clases, comunicación y máquina de estados.

- Es difícil establecer cómo en este proyecto relacionan "user" con "member", porque no parece razonable desde el punto de vista del PLN, a menos que se utilice en este proyecto un diccionario de sinónimos que relacione los dos términos. Si no se dispone de información para corroborar esto, y de todos modos se presentan las frases de la manera que se expresa en el enunciado, en los modelos resultantes se pueden generar problemas de duplicidad de funciones en el software o problemas de correlación entre estos términos. En UN-Lencep, en cambio, se debió haber resuelto esta discrepancia entre los nombres, para poder escribir el discurso en este lenguaje.
- Existe información del enunciado que no se toma en cuenta para la creación de los esquemas conceptuales de CPL. Por ejemplo "with his pass" no tiene equivalencia en los diagramas CPL, en tanto que en el UN-Lencep se puede ubicar como un posesivo del concepto "miembro", lo que se interpretó como una relación "tiene" entre el miembro y el carnet.
- Las aclaraciones que se requieren para el enunciado, deberían ser realizadas mediante un proceso de diálogo que permita la validación por parte del interesado de la información que se está modelando. Es por ello que el UN-Lencep es un medio de comunicación analista-interesado, en el cual el interesado y el analista pueden usar para establecer un vocabulario común.

5.1.4. ER-Converter Tool

Omar *et al.* (2004) presentan el siguiente enunciado para ejemplificar el uso de la herramienta ER-Converter:

> *The company is organized into departments. Each department has a unique name, a unique number and a particular employee who manages the department. A department may have several locations.*

En esta herramienta, empleando un proceso semiautomático se obtiene el diagrama entidad-relación que se muestra en la Figura 47.

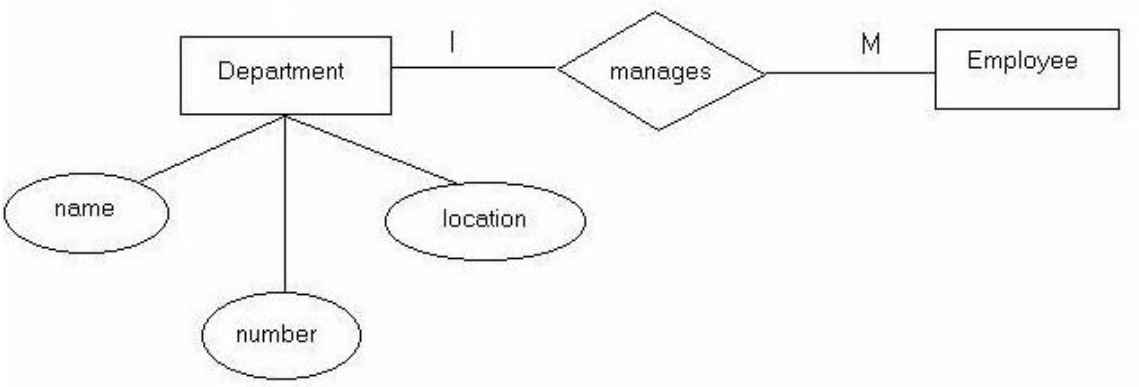


Figura 47. Diagrama entidad-relación obtenido a partir del enunciado en ER-Converter Tool.

El discurso en lenguaje controlado que es equivalente a este enunciado es:

> *El departamento posee un nombre; un numero pertenece a un departamento; el departamento posee una ubicacion; el empleado pertenece a un departamento; un empleado dirige el departamento*

El cual, traducido a UN-Lencep, se expresa así:

> *ST DEPARTAMENTO tiene NOMBRE*
> *ST DEPARTAMENTO tiene NUMERO*
> *ST DEPARTAMENTO tiene UBICACION*
> *ST DEPARTAMENTO tiene EMPLEADO*
> *RD EMPLEADO DIRIGE DEPARTAMENTO*

El Esquema Preconceptual y el diagrama de clases resultantes se pueden apreciar en las Figuras 48 y 49 respectivamente.

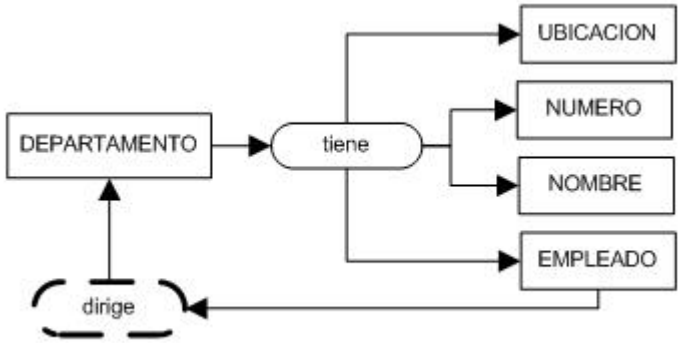


Figura 48. Esquema Preconceptual resultante del discurso en UN-Lencep basado en ER-Converter Tool.

Figura 49. Diagrama de clases generado a partir del Esquema Preconceptual de la Figura 48.

Algunas observaciones en relación con este proceso son:

- Al igual que en el ejemplo de RADD (véase sección 5.1.1), éste también es un discurso supremamente estructural y por ello sólo se obtiene el diagrama de clases.
- En el DER generado inicialmente por ER-Converter Tool, la relación "dirige" no se puede diferenciar en su funcionamiento de otras relaciones pertenecientes al diagrama. En el diagrama de clases generado en UNC-Diagramador sí aparece específicamente como una operación de la clase departamento.

- La entidad "Empleado" que aparece en el DER inicial se transforma en un atributo en el diagrama de clases generado por UNC-Diagramador porque aún es un discurso incompleto y no se tiene mayor información para deducir que se trata de una clase; se requeriría que ese concepto tuviera relaciones estructurales con otros o que se encontrase inmerso dentro de una cadena de implicaciones. Esto debería preguntarlo el analista al interesado, con el fin de precisar la información.
- El concepto "compañía" desaparece por completo del discurso en UN-Lencep. La razón para ello es que en ER-Converter Tool el concepto que le da nombre al dominio (en este caso "compañía", pero en otros casos podría ser "biblioteca" o "universidad") queda implícito en el proceso de generación de esquemas conceptuales, pero no se representa con ningún elemento de dichos esquemas.

5.1.5. NIBA

5.1.5.1. Diagramas estructurales

Mayr y Kop (2002) ejemplifican, con el siguiente enunciado, la generación automática del diagrama de clases a partir de lenguaje natural controlado:

> *(1) Publishers publish books. (2) Authors write books. (3) A book can be written by several authors. (4) An author has a unique name, a birth date and an address. (5) A book has a unique title and a prize. (6) An ISBN number identifies a book (7) Each book is published by exactly one publisher. (8) A publisher has a unique denomination and an address. (9) A publisher has employees. (10) An employee has a unique social insurance number, a name and a birth date.*

Además, en la Figura 50, se muestra el diagrama resultante mediante la aplicación del proyecto NIBA al enunciado anterior.

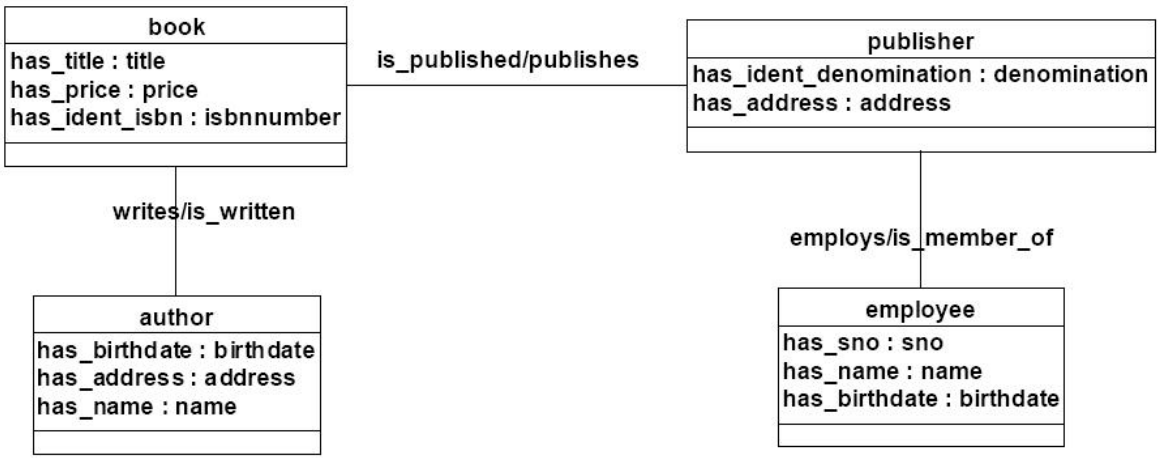


Figura 50. Diagrama de clases obtenido a partir del enunciado estructural del proyecto NIBA.

En lenguaje natural controlado, el discurso del interesado correspondiente a este enunciado se puede expresar así:

> *Una editorial publica un libro; un autor escribe el libro; un autor posee un nombre; una fecha_de_nacimiento pertenece a un autor; una direccion pertenece a un autor; un libro posee un titulo; un precio pertenece a un libro; un libro posee un isbn; una editorial posee una denominacion; una direccion pertenece a una editorial; un empleado pertenece a una editorial; un empleado posee un numero_de_seguro_social; un nombre pertenece a un empleado; una fecha_de_nacimiento pertenece a un empleado.*

La traducción a UN-Lencep de este discurso es la siguiente:

> *RD EDITORIAL PUBLICA LIBRO*
> *RD AUTOR ESCRIBE LIBRO*
> *ST AUTOR tiene NOMBRE*
> *ST AUTOR tiene FECHA_DE_NACIMIENTO*
> *ST AUTOR tiene DIRECCION*
> *ST LIBRO tiene TITULO*
> *ST LIBRO tiene PRECIO*
> *ST LIBRO tiene ISBN*

ST EDITORIAL tiene DENOMINACION
ST EDITORIAL tiene DIRECCION
ST EDITORIAL tiene EMPLEADO
ST EMPLEADO tiene NUMERO_DE_SEGURO_SOCIAL
ST EMPLEADO tiene NOMBRE
ST EMPLEADO tiene FECHA_DE_NACIMIENTO

A partir del discurso en UN-Lencep se obtiene el Esquema Preconceptual de la Figura 51 y posteriormente el diagrama de clases de la Figura 52.

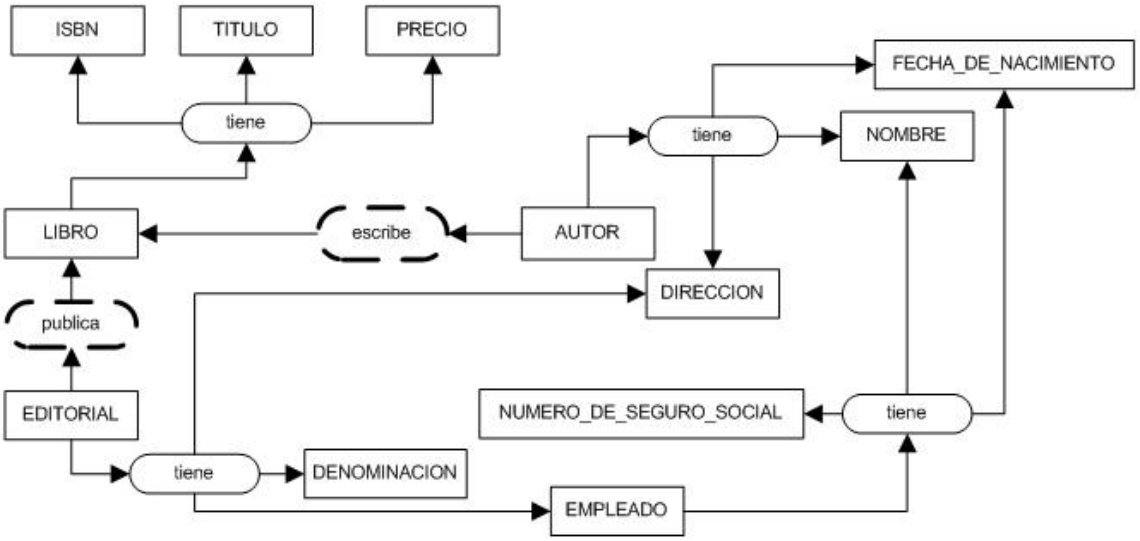


Figura 51. Esquema Preconceptual correspondiente al enunciado estructural del proyecto NIBA.

Algunas observaciones a este proceso son las siguientes:

- Por el carácter estructural del discurso, se genera únicamente el diagrama de clases. A pesar de que se habla de verbos de actividad como publica y escribe, estos no están enlazados con condicionales o implicaciones, por lo cual no se genera ningún diagrama de comunicación. Sólo basta con incluir la frase en UN-Lencep “IM Cuando autor

escribe libro, editorial publica libro" para que se generen diagramas de comunicación y máquina de estados en este enunciado.

- Las relaciones entre editorial y libro, y entre autor y libro, se especializan y se transforman en operaciones del diagrama de clases, que no eran consideradas en el caso de NIBA. Esto implica que el diagrama de clases que se obtiene mediante el entorno descrito en esta Tesis tiene un nivel mayor de refinamiento que el que se obtiene en el proyecto NIBA.
- La relación entre editorial y empleado se especializa y se convierte en una agregación, dejando de ser una asociación. Este también constituye un paso de refinamiento del diagrama de clases.

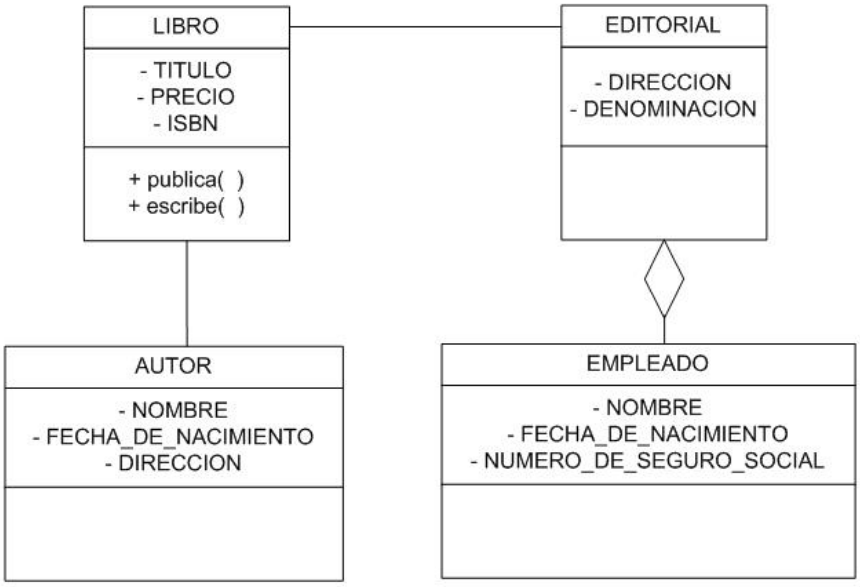


Figura 52. Diagrama de clases obtenido a partir del Esquema Preconceptual de la Figura 51.

5.1.5.2. Diagramas de Comportamiento

Kop y Mayr (2002) plantean el siguiente enunciado para ilustrar la generación automática de diagramas de comportamiento, particularmente los diagramas de actividades y máquina de estados:

(1) The order comes in. (2) The order department checks each article on the order. (3) If each article is available on stock, then the order department relates it to the order. (4) Then, if the stock quantity of an article is lower than the minimal stock quantity, the stock clerk must order this article. (5) If the order comes in, also the bookkeeping department checks the payment. (6) If the payment is authorized and we have all articles in stock, then the order department releases the order. (7) If payment is authorized but there are not all articles in stock, then the order department must put the order to the pending orders. (8) If the payment is not authorized, then the order department must reject the order.

Los diagramas que se generan a partir de este enunciado se pueden apreciar en las Figuras 53 y 54.

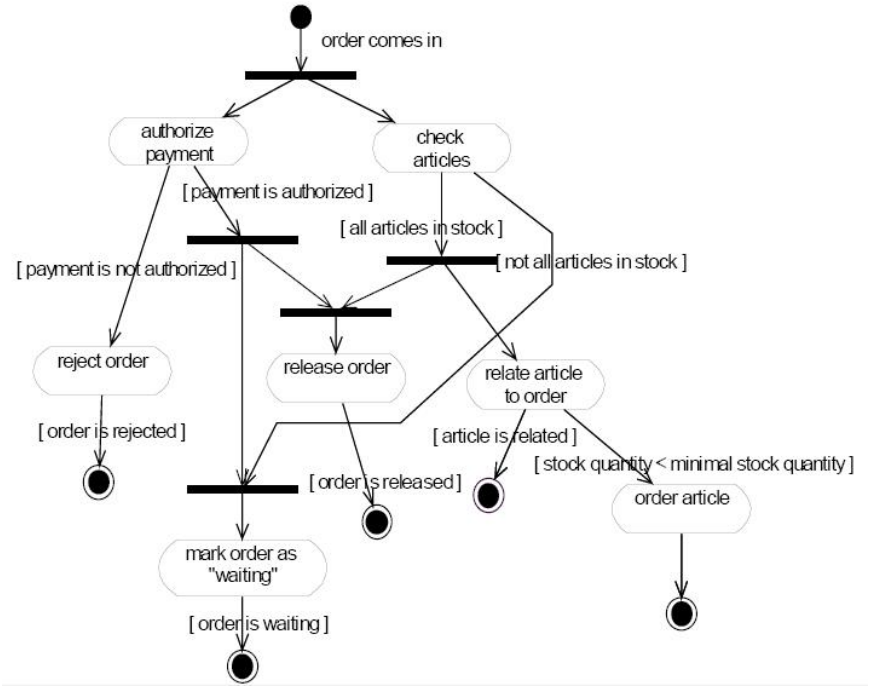


Figura 53. Diagrama de actividades que se genera a partir del enunciado comportamental del proyecto NIBA.

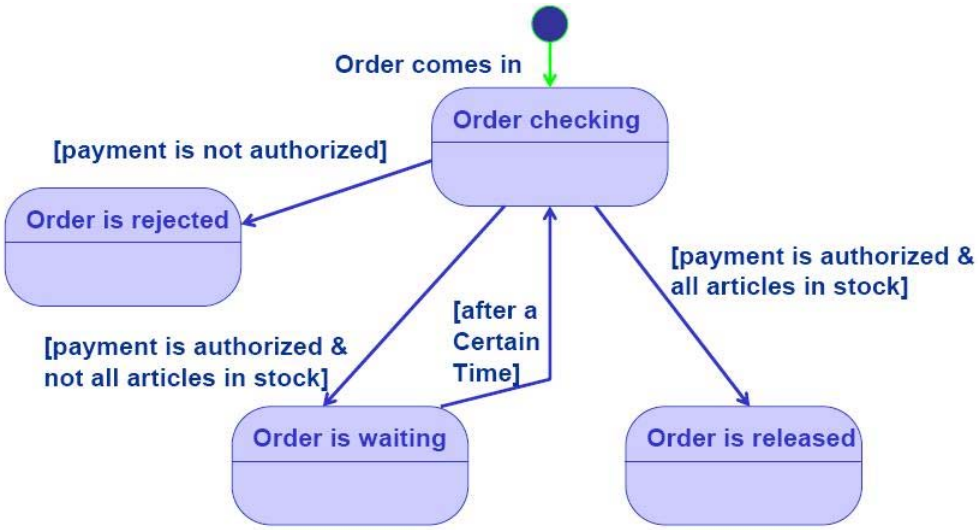


Figura 54. Diagrama de máquina de estados que se genera a partir del enunciado comportamental del proyecto NIBA.

Un discurso en lenguaje controlado que puede representar el enunciado presentado es el siguiente:

> *Un articulo es parte de una orden; cuando el asistente recibe la orden entonces el departamento_de_pedidos verifica el articulo; el inventario contiene articulo; si inventario.articulo='disponible', departamento_de_pedidos relaciona articulo; si inventario.cantidad<inventario.cantidad_minima, asistente pide articulo; cuando el asistente recibe la orden, el auxiliar_contable verifica el pago; si pago.estado='autorizado' y inventario.articulo='disponible', departamento_de_pedidos relaciona articulo; si pago.estado='autorizado' y inventario.articulo='no_disponible', entonces el departamento_de_pedidos retiene la orden; si pago.estado='no_autorizado' entonces el departamento_de_pedidos rechaza la orden; la cantidad es una parte de un inventario; la cantidad_minima esta incluida en un inventario; el pago incluye un estado; un pago contiene una orden.*

El discurso en UN-Lencep que se obtiene automáticamente en UNC-Diagramador, tomando como base el discurso anterior, es:

ST ORDEN tiene ARTICULO
IM Cuando ASISTENTE RECIBE ORDEN entonces
DEPARTAMENTO_DE_PEDIDOS VERIFICA ARTICULO
ST INVENTARIO tiene ARTICULO
CO Si INVENTARIO.ARTICULO='DISPONIBLE',
DEPARTAMENTO_DE_PEDIDOS RELACIONA ARTICULO
CO Si INVENTARIO.CANTIDAD<INVENTARIO.CANTIDAD_MINIMA,
ASISTENTE PIDE ARTICULO
IM Cuando ASISTENTE RECIBE ORDEN entonces
AUXILIAR_CONTABLE VERIFICA PAGO
CO Si PAGO.ESTADO='AUTORIZADO' AND
INVENTARIO.ARTICULO='DISPONIBLE',
DEPARTAMENTO_DE_PEDIDOS LIBERA ORDEN
CO Si PAGO.ESTADO='AUTORIZADO' AND
INVENTARIO.ARTICULO='NO_DISPONIBLE',
DEPARTAMENTO_DE_PEDIDOS RETIENE ORDEN
CO Si PAGO.ESTADO='NO_AUTORIZADO',
DEPARTAMENTO_DE_PEDIDOS RECHAZA ORDEN
ST INVENTARIO tiene CANTIDAD
ST INVENTARIO tiene CANTIDAD_MINIMA
ST PAGO tiene ESTADO
ST PAGO tiene ORDEN

En la Figura 55 se muestra el Esquema Preconceptual resultante del discurso en UN-Lencep. En las Figuras 56 a 58 se muestran respectivamente los diagramas de clases, comunicación y máquina de estados resultantes en UNC-Diagramador.

Entre los diagramas que resultan de los diferentes procesos se pueden notar las siguientes diferencias:

- En el enunciado propuesto por el proyecto NIBA se omiten los actores de ciertas acciones. Por ejemplo, “la orden llega” impide saber quién ejecuta la acción de “hacer llegar la orden”; en UNC-Diagramador esa frase se reemplaza por “El asistente recibe la orden”. En NIBA esta frase se está mapeando como un evento o una precondición, pero realmente podría estar encubriendo una acción que podría tener repercusión en el diagrama de máquina de estados.

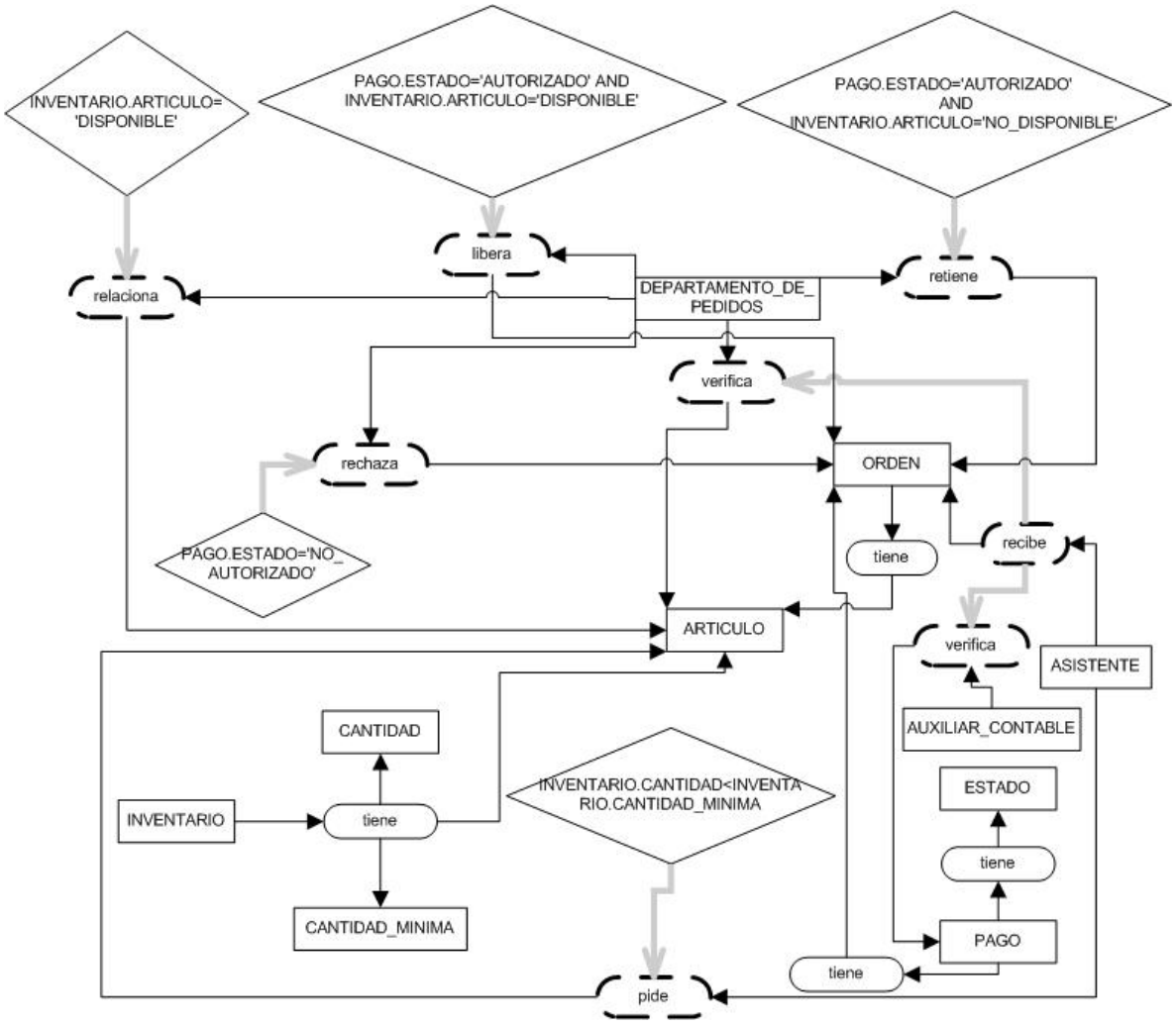


Figura 55. Esquema Preconceptual correspondiente al discurso en UN-Lencep para el enunciado comportamental del proyecto NIBA.

- Se sabe que el proyecto NIBA es capaz de definir el diagrama de clases correspondiente a un enunciado. Sin embargo, en este caso no se presenta este diagrama en la literatura. Los que sí se presentan en los diferentes artículos son los diagramas de actividades y de máquina de estados, los cuales presentan errores de consistencia, como los siguientes (los cuales se pueden verificar al comparar las Figuras 52 y 53):

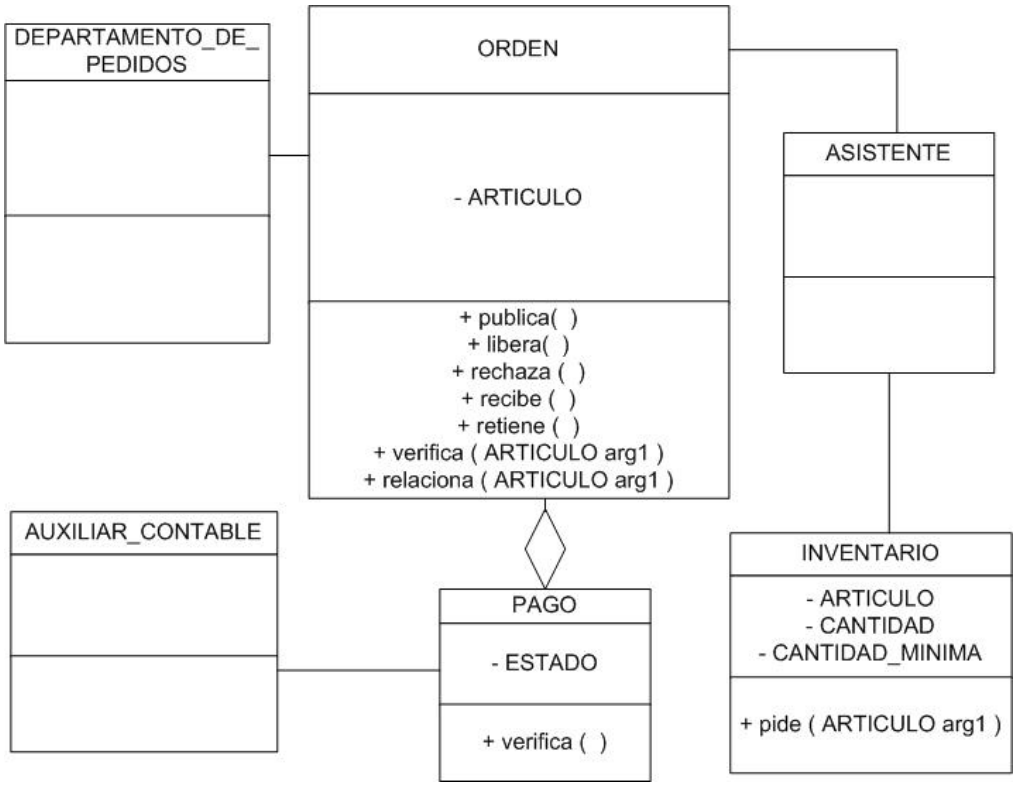


Figura 56. Diagrama de clases resultante del Esquema Preconceptual de la Figura 55.

- No se genera el estado "relacionada" en la orden, después de que se relacionan todos los artículos en la orden.
- No se generan diagramas de máquina de estados para "pago" e "inventario", correspondientes a clases del diagrama de clases que también poseen operaciones.
- No es claro de donde sale la condición de guarda "después de cierto tiempo", la cual no se define explícitamente en el enunciado y tampoco se genera en el diagrama de actividades.
- En el enunciado se habla de "colocar la orden en órdenes pendientes", en tanto que ambos diagramas establecen que "la orden está en espera". No es claro desde el punto de vista del Procesamiento del Lenguaje Natural cómo se podría generar esta equivalencia.

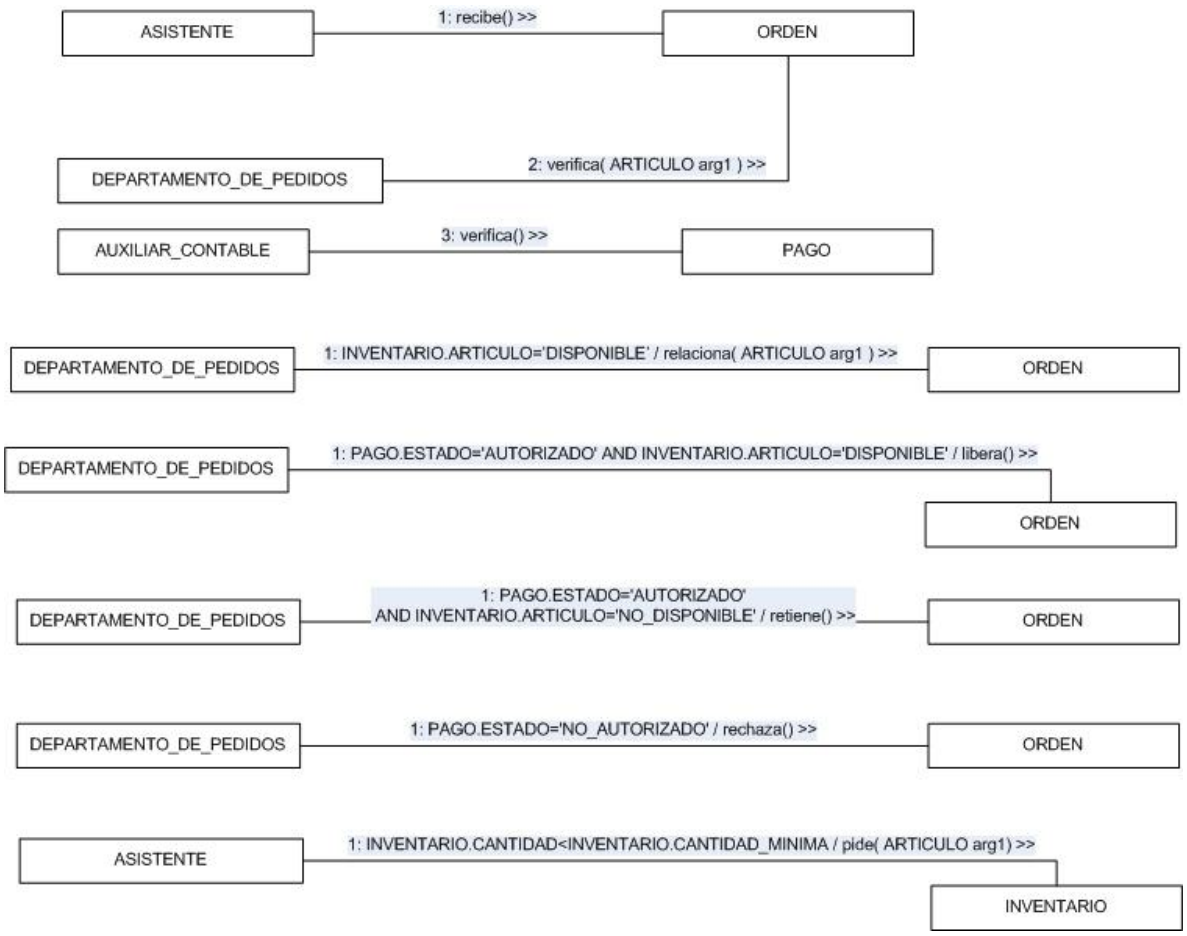


Figura 57. Diagramas de comunicación resultantes a partir del Esquema Preconceptual de la Figura 55.

- En el enunciado no es claro de qué manera se enlazan las diferentes acciones para generar el diagrama de actividades. De la forma como se definió el enunciado, da la impresión de que son diferentes transacciones las que se realizan en el mundo, lo que generaría múltiples diagramas de actividades. En el caso del UNC-Diagramador, se generan seis (6) diagramas de comunicación diferentes, los cuales son completamente consistentes con los diagramas de máquina de estados que se generan.

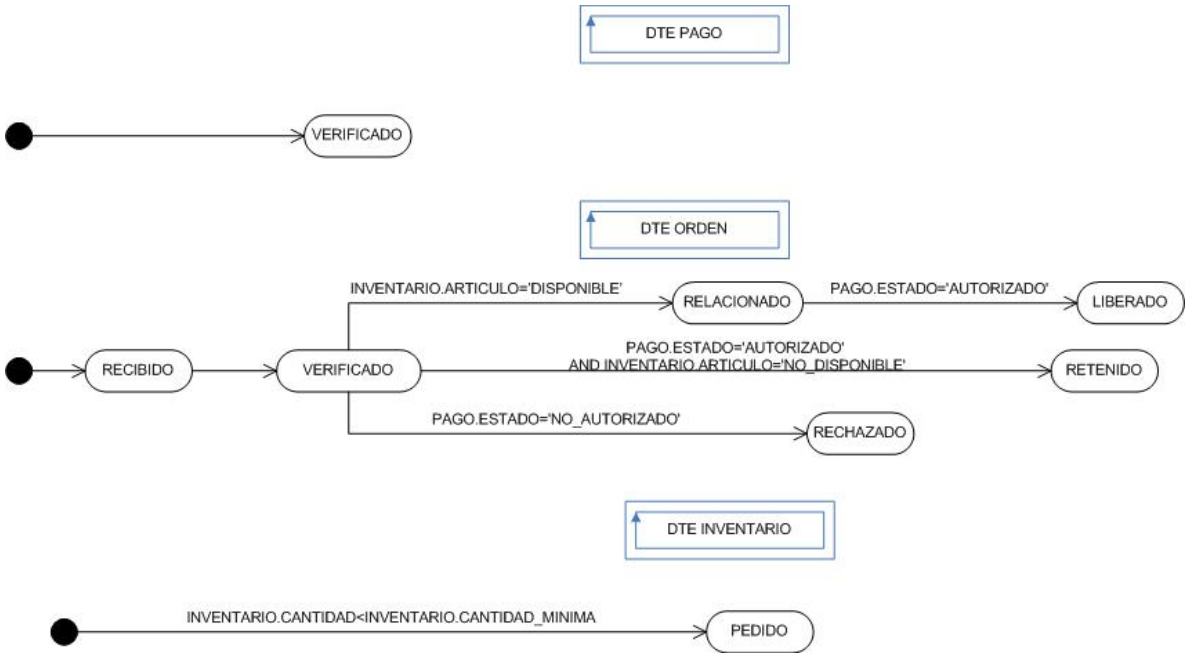


Figura 58. Diagramas de máquina de estados resultantes a partir del Esquema Preconceptual de la Figura 55.

- El carácter comportamental del discurso impide apreciar que existe una relación entre orden y pago que no se hace explícita en el discurso. En el UNC-Diagramador esa relación se declara explícitamente, de manera que no queden aisladas algunas de las clases en el diagrama de clases que se genera. Es por ello que el discurso estructural debe coexistir con el comportamental o dinámico que se establecía en el numeral 5.1.2., con el fin de complementar la información que comparten varios diagramas de UML simultáneamente. Algo similar se aprecia en el hecho de que se están usando precondiciones que no se ligan con elementos del diagrama de clases; por ejemplo, en el discurso comportamental de NIBA se declara “el pago es autorizado”, pero no se establece quién lo autoriza ni de qué manera el diagrama de clases va a atender esta parte del enunciado. En el UNC-Diagramador esa frase se sustituye por la condición “pago.estado=‘autorizado’”, la cual se complementa con la frase estructural “pago tiene estado”.

5.1.6. Díaz *et al.*

En Díaz *et al.* (2004) se presenta un enunciado para la generación automática del diagrama de secuencias, que se presenta a continuación:

> *Caso de Uso Venta de Artículos*
> *Actor Principal Cajero*
> *Resumen Permite registrar la información de la venta de uno o más artículos.*
> *Venta de Artículos INICIA CUANDO el Cajero inicia la sesión de una venta.*
> *1. El Sistema activa una nueva sesión de una venta.*
> *2. El Sistema solicita el código de barras y la cantidad vendida de un artículo.*
> *PARA un artículo*
> *3. El Cajero introduce el código de barras y la cantidad vendida del artículo.*
> *4. El Sistema obtiene la descripción y el precio por unidad del artículo.*
> *5. El Sistema calcula el monto de la venta del artículo.*
> *6. El Sistema calcula el subtotal de la venta.*
> *7. El Sistema muestra el subtotal de la venta.*
> *8. El Sistema registra la venta del artículo.*
> *8.1. El Sistema actualiza la cantidad en existencia del artículo.*
> *9. El Sistema pregunta si la venta ha concluido.*
> *FinPARA*
> *10. El Cajero finaliza la sesión de la venta.*
> *11. El Sistema calcula el monto total de la venta.*
> *12. El Sistema muestra el monto total de la venta.*
> *13. El Sistema registra la fecha y monto total de la venta.*
> *14. El Sistema imprime el comprobante de la venta.*
> *FIN Venta de Artículos*

Este enunciado está escrito en un lenguaje controlado que permite la escritura de especificaciones textuales de los casos de uso, además de posibilitar la elaboración del diagrama de secuencias que se muestra en la Figura 59.

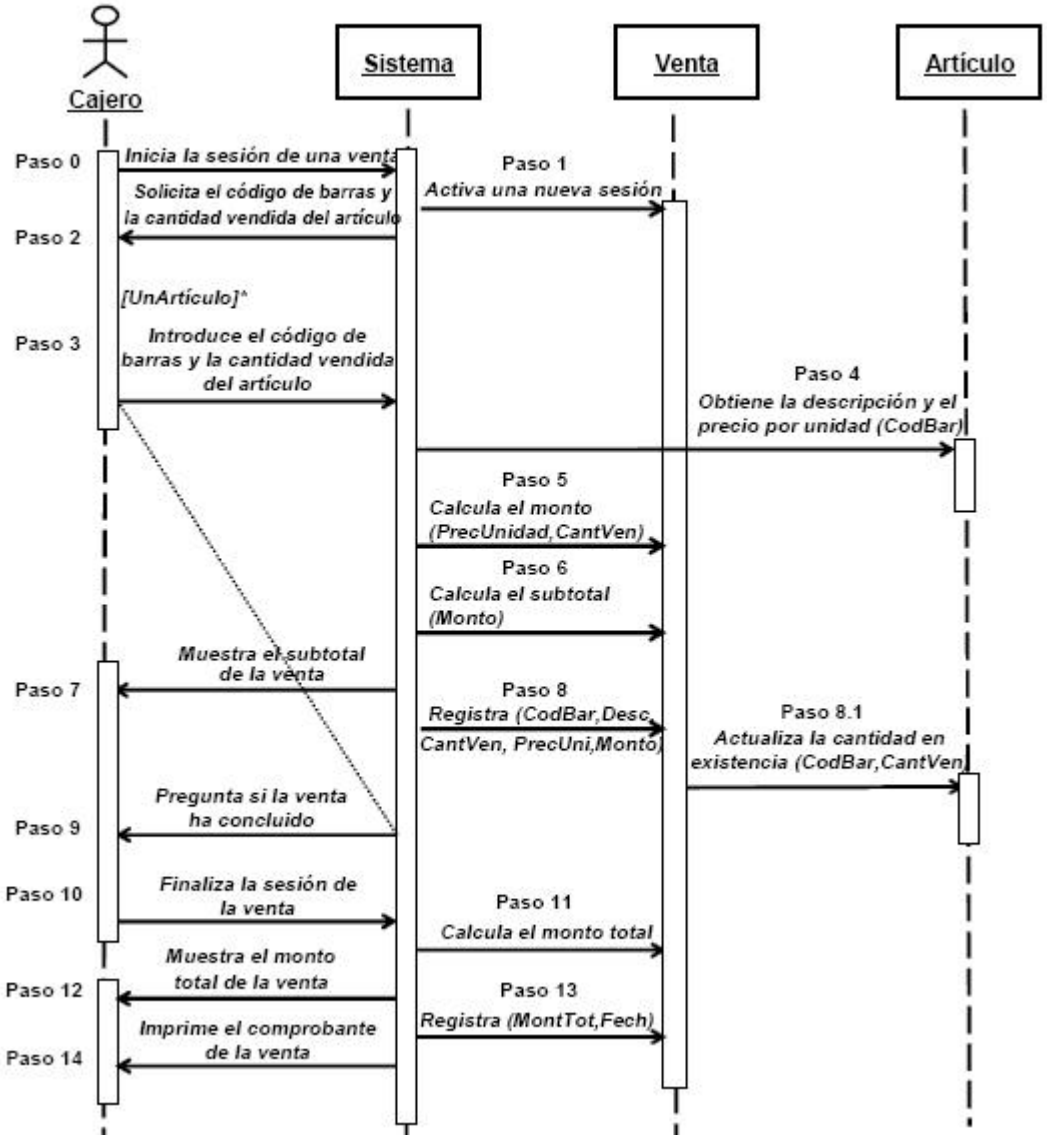


Figura 59. Diagrama de secuencias generado a partir del enunciado presentado por Díaz *et al.* (2004).

En el caso del UNC-Diagramador este lenguaje debe ser complementado con algunas frases de tipo estructural que presenten las relaciones entre los diferentes objetos del dominio. Tal discurso en lenguaje controlado tiene la siguiente apariencia:

Cuando el sistema activa una nueva_sesion entonces el sistema solicita el codigo_de_barras; venta tiene nueva_sesion; cuando el sistema solicita el codigo_de_barras, el sistema solicita la cantidad_vendida; un articulo posee codigo_de_barras; una venta posee articulo y cantidad_vendida; cuando el sistema solicita la cantidad_vendida, el sistema obtiene la descripcion; cuando el sistema obtiene la descripcion, el sistema obtiene el precio_por_unidad; un articulo posee descripcion y precio_por_unidad; cuando el sistema obtiene el precio_por_unidad, el sistema calcula el monto; el monto pertenece a una venta; el subtotal pertenece a la venta; cuando el sistema calcula el monto, el sistema muestra el subtotal; cuando el sistema muestra el subtotal, el sistema registra la venta; cuando el sistema registra la venta, el sistema actualiza la cantidad_en_existencia; la cantidad_en_existencia pertenece a un articulo; si venta.estado=concluido, entonces el cajero finaliza la nueva_sesion; cuando el sistema finaliza la nueva_sesion, el sistema registra la fecha; cuando el sistema registra la fecha, el sistema registra el monto_total; una venta posee fecha y monto_total; cuando el sistema registra el monto_total, el sistema imprime el comprobante; una venta posee comprobante; un estado pertenece a una venta

El discurso en UN-Lencep correspondiente al anterior discurso en lenguaje controlado es:

IM Cuando SISTEMA ACTIVA NUEVA_SESION entonces SISTEMA SOLICITA CODIGO_DE_BARRAS
ST VENTA tiene NUEVA_SESION
IM Cuando SISTEMA SOLICITA CODIGO_DE_BARRAS entonces SISTEMA SOLICITA CANTIDAD_VENDIDA
ST ARTICULO tiene CODIGO_DE_BARRAS
ST VENTA tiene ARTICULO CANTIDAD_VENDIDA
IM Cuando SISTEMA SOLICITA CANTIDAD_VENDIDA entonces SISTEMA OBTIENE DESCRIPCION
IM Cuando SISTEMA OBTIENE DESCRIPCION entonces SISTEMA OBTIENE PRECIO_POR_UNIDAD
ST ARTICULO tiene DESCRIPCION PRECIO_POR_UNIDAD
IM Cuando SISTEMA OBTIENE PRECIO_POR_UNIDAD entonces SISTEMA CALCULA MONTO
ST VENTA tiene MONTO
ST VENTA tiene SUBTOTAL
IM Cuando SISTEMA CALCULA MONTO entonces SISTEMA MUESTRA SUBTOTAL

IM Cuando SISTEMA MUESTRA SUBTOTAL entonces SISTEMA REGISTRA VENTA
IM Cuando SISTEMA REGISTRA VENTA entonces SISTEMA ACTUALIZA CANTIDAD_EN_EXISTENCIA
ST ARTICULO tiene CANTIDAD_EN_EXISTENCIA
CO Si VENTA.ESTADO=CONCLUIDO, CAJERO FINALIZA NUEVA_SESION
IM Cuando SISTEMA FINALIZA NUEVA_SESION entonces SISTEMA REGISTRA FECHA
IM Cuando SISTEMA REGISTRA FECHA entonces SISTEMA REGISTRA MONTO_TOTAL
ST VENTA tiene FECHA MONTO_TOTAL
IM Cuando SISTEMA REGISTRA MONTO_TOTAL entonces SISTEMA IMPRIME COMPROBANTE
ST VENTA tiene COMPROBANTE
ST VENTA tiene ESTADO

Con el anterior discurso, en el UNC-Diagramador se puede generar el Esquema Preconceptual que se presenta en la Figura 60. Los diagramas resultantes de clases, comunicación y máquina de estados se presentan en las Figuras 61 a 63.

Una comparación entre ambos entornos presenta las siguientes observaciones:

- El enunciado presentado por Díaz *et al.* (2004) es una descripción del sistema futuro. Por ello, las responsabilidades propias de los diferentes actores se diluyen y no se pueden observar en los diferentes diagramas, pues se esconden tras el rol de "sistema". Cabe anotar que para llegar a un enunciado como el que se presenta, es necesario realizar una buena parte de las fases de definición y análisis del ciclo de vida de una aplicación de software.
- El entorno presentado por Díaz *et al.* (2004) únicamente genera el diagrama de secuencias. En el caso del UNC-Diagramador se generan los tres diagramas (clases, comunicación y máquina de estados), pero el discurso debe ser complementado con la parte estructural para ello. Por ejemplo, se adicionan las frases que contienen el verbo "tiene", las cuales posibilitan la identificación y el mapeo de clases y atributos.

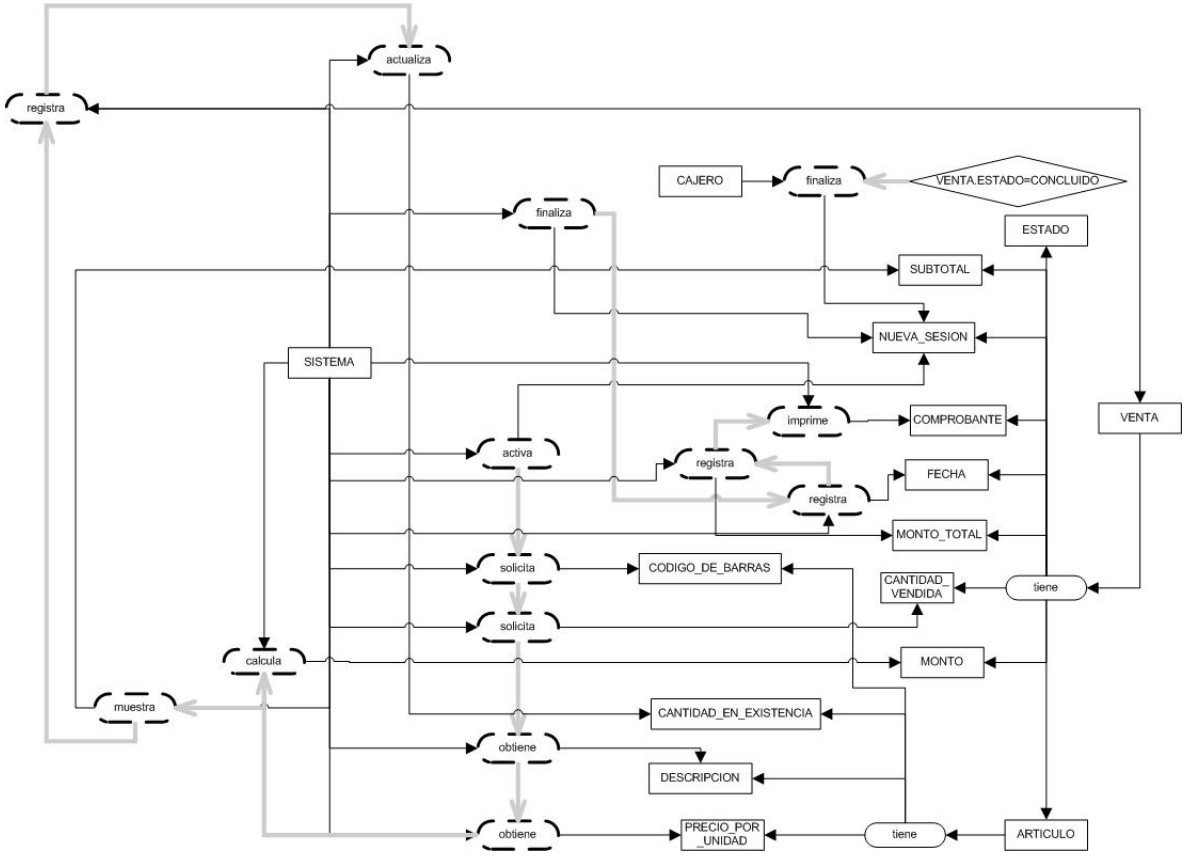


Figura 60. Esquema Preconceptual correspondiente al UN-Lencep del enunciado presentado por Díaz *et al.* (2004).

- Por la forma como está redactado el discurso, en el UNC-Diagramador las operaciones se reparten entre las clases "venta" y "artículo". El "cajero" como clase no posee ningún tipo de operación. Sin embargo, están llegando a él varios mensajes en el diagrama de secuencias presentado por Díaz *et al*. sin que se pueda establecer una diferenciación sobre los casos en que se asignan las operaciones al actor "cajero"; por ejemplo, no es claro por qué en la frase "El Sistema calcula el subtotal de la venta" el objeto que posee la operación "calcula" es "venta", en tanto que en la frase "El Sistema muestra el subtotal de la venta" el objeto que posee la operación "muestra" es "cajero".

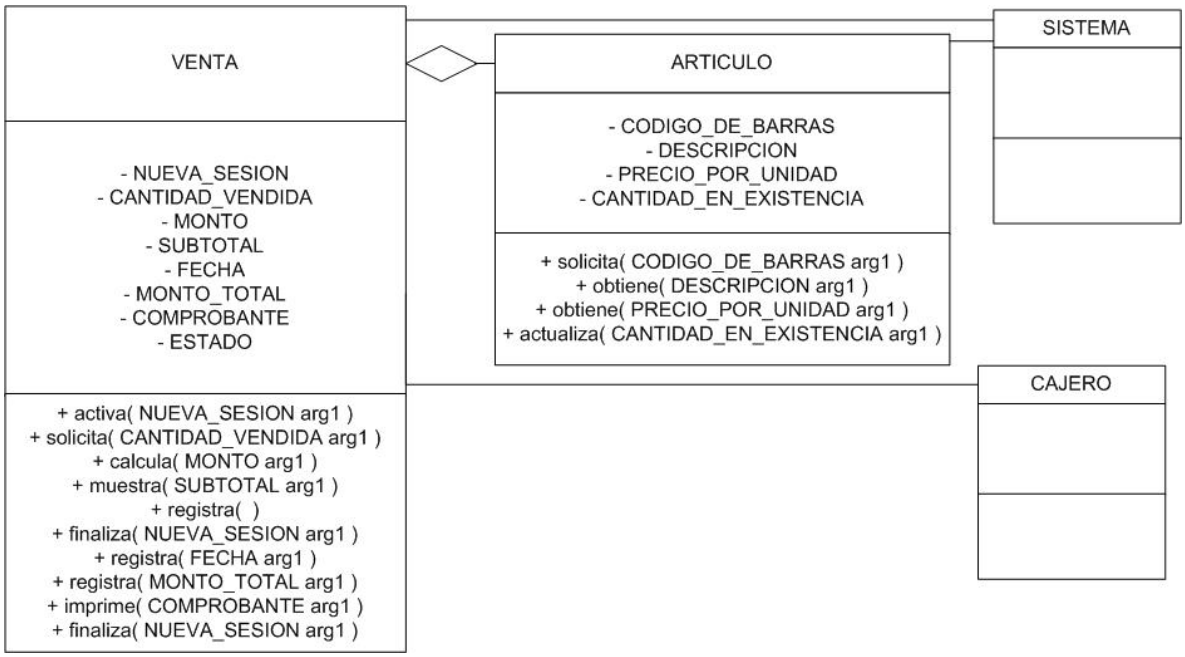


Figura 61. Diagrama de clases obtenido a partir del Esquema Preconceptual de la Figura 60.

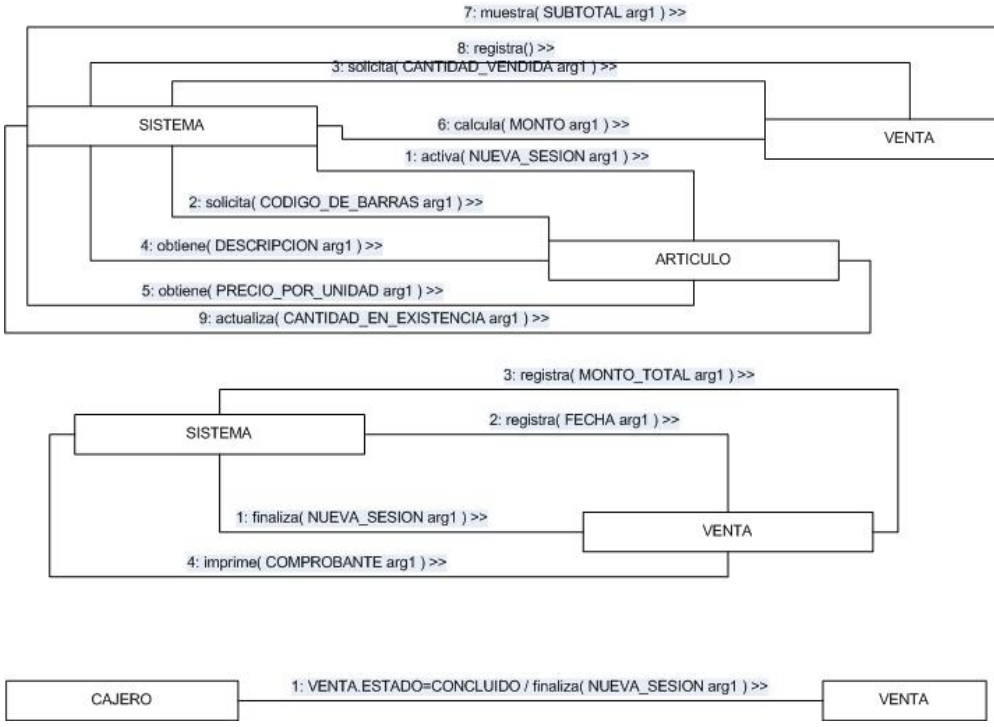


Figura 62. Diagramas de comunicación obtenidos a partir del Esquema Preconceptual de la Figura 60.

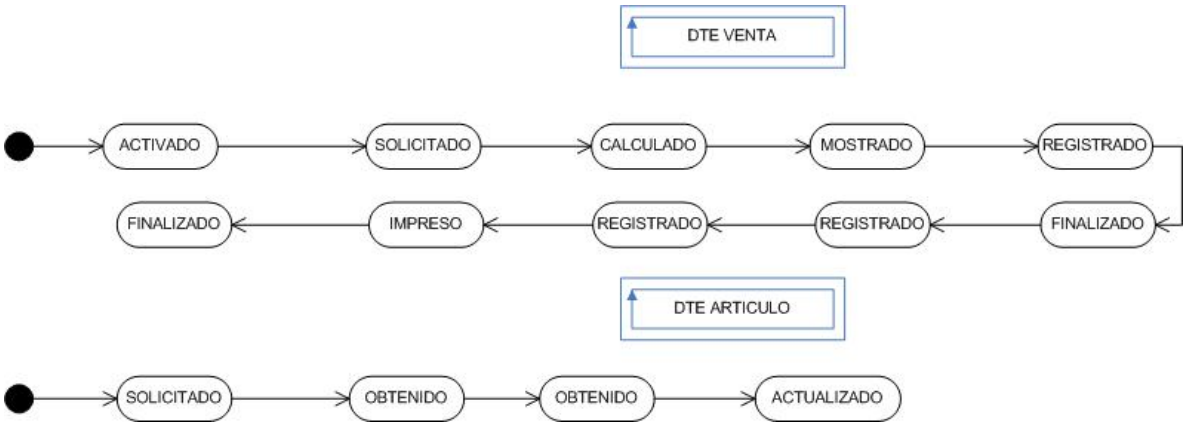


Figura 63. Diagramas de máquina de estados obtenidos a partir del Esquema Preconceptual de la Figura 60.

- En el diagrama de secuencias presentado por Díaz *et al.* (2004) la transacción es única (lo que daría la impresión de ser un caso de uso único), en tanto que en el UNC-Diagramador existen al menos tres transacciones diferentes: el cálculo y registro de los detalles de la venta, la totalización e impresión de la venta como tal y la finalización de la sesión. Si bien es una diferencia apreciable entre ambos entornos, se puede ligar con el diferente nivel de granularidad que se puede presentar en la definición de los casos de uso correspondientes a una aplicación.
- El entorno presentado por Díaz *et al.* (2004) permite identificar los ciclos, en tanto que el UNC-Diagramador no tiene en el UN-Lencep una sintaxis apropiada para la representación de los ciclos.

5.2. Un experimento para medir la cantidad de errores cometidos al elaborar manualmente diagramas de UML

- Objetivo: Se busca establecer el comportamiento de los analistas al elaborar manualmente algunos diagramas de UML (partiendo desde un enunciado definido), tomando en consideración diferentes tiempos de elaboración de diagramas, diferentes

cantidades de diagramas y diferentes grados de capacitación. Se pretende medir la cantidad de errores de calidad (en términos de Consistencia, Corrección y Completitud) que cometen los analistas en estas condiciones.

- Principios de calidad de los resultados: En relación con el modelamiento de piezas de software, se suelen examinar tres principios de calidad: Consistencia, Completitud y Corrección (Zowghi y Gervasi, 2002), los cuales se definen de la siguiente manera:
    - Consistencia: Se define como la carencia de contradicciones en la especificación. Para los diagramas de UML, la consistencia se verifica con la representación de un mismo concepto o relación por medio de diferentes primitivas conceptuales en los diferentes diagramas.
    - Completitud: Se define como el vínculo de una especificación con lo que se puede considerar "cierto" en un dominio particular, o como la cantidad de conceptos propios de un área que se pueden identificar en los diagramas.
    - Corrección: Se define como el uso adecuado de los símbolos que están disponibles para la elaboración de un diagrama particular.
- Hipótesis: Se definen las siguientes hipótesis para el experimento:
    - Hipótesis nula: Los analistas no cometen errores de consistencia, completitud y/o corrección en la construcción de diagramas de UML, tomando como punto de partida un enunciado en lenguaje natural.
    - Hipótesis alternativa: Sin importar el tiempo que empleen en la elaboración de diagramas, la cantidad de diagramas que se realicen, ni el grado de capacitación que posean los analistas, cometerán errores de consistencia, completitud y/o corrección cuando elaboren manualmente un conjunto de diagramas a partir de un enunciado en lenguaje natural.
- Sujetos experimentales: En este experimento se emplearon estudiantes universitarios de Ingeniería de Sistemas de tercer y noveno semestre, con conocimientos variables sobre UML.

- Material Experimental: Para la realización del experimento fue necesario diseñar dos plegables que incluían información sobre la elaboración de diagramas UML; el primero de ellos incluía los diagramas de comunicación y máquina de estados y el segundo, además de estos diagramas, incluía el diagrama de clases. La información contenida en los plegables se puede apreciar en el Anexo 1.
- Tareas Experimentales: Cada sujeto debía elaborar dos o tres diagramas, según se le hubiera repartido uno u otro plegable, en un tiempo también variable. Además, cada sujeto debía llenar una encuesta donde se preguntaba sobre el grado de dificultad de elaboración de la tarea correspondiente.
- Procedimiento: El experimento se realizó en diferentes sesiones para cada uno de los grupos de sujetos, a saber:
    - Inicialmente, se tomaron 10 estudiantes de noveno semestre de Ingeniería de Sistemas e Informática (los cuales tenían una experiencia de un año en la construcción de diagramas de UML) y a ellos se les entregó el plegable que incluía los tres diagramas, el cual estudiaron durante quince (15) minutos, pudiendo realizar cuantas preguntas quisieran para aclarar el contenido. Terminado ese tiempo, se les entregó un enunciado correspondiente a la descripción de una situación académica, el cual se incluye en el Anexo 2, con el fin de que durante 50 minutos realizaran los tres diagramas solicitados (clases, comunicación y máquina de estados).
    - En la segunda sesión, se tomó un grupo de estudiantes de tercer semestre de Ingeniería de Sistemas e Informática (los cuales tenían una experiencia de seis meses en la construcción de diagramas de UML) y se dividió el grupo en cuatro subgrupos de manera aleatoria. A dos de los subgrupos se les entregó el plegable de dos diagramas y a los otros dos se les entregó el plegable de tres diagramas. A los cuatro grupos se les permitió estudiar el plegable durante quince (15) minutos, con la posibilidad de realizar las preguntas necesarias para aclarar el contenido. Posteriormente, se les entregó el enunciado correspondiente a la

descripción de una situación académica y se les solicitó realizar los diagramas de UML que modelaban esa situación; el tiempo para el desarrollo de esa labor fue de 25 minutos para un grupo de estudiantes con dos diagramas y para un grupo de estudiantes con tres diagramas. Para los dos grupos restantes, el tiempo fue de 50 minutos.

Antes de finalizar cada una de las sesiones se entregó a los participantes una encuesta para evaluar la dificultad para elaborar los diferentes diagramas.

- Factores: Se tomaron en consideración los siguientes tres factores para la conformación de los bloques:
  - Tiempo de elaboración de diagramas: Se toman tiempos de 25 y 50 minutos.
  - Número de diagramas elaborados: Se propone la elaboración de dos y tres diagramas.
  - Grado de capacitación de los sujetos experimentales: Se toman 0, 6 y 12 meses de capacitación.
- Variables: Para la evaluación de calidad de los diagramas elaborados, se tomaron en consideración las siguientes variables, que corresponden a errores típicos que cometen los analistas en la elaboración de diagramas:
  - Número de elementos faltantes (completitud): A partir del enunciado, se elaboran los diagramas de clases, comunicación y máquina de estados que se pueden generar, que se incluyen en el Anexo 2, y que sirven como base de evaluación para los diagramas elaborados por los sujetos experimentales. A los elementos totales de los diagramas que sirven de base se les restan los elementos correctamente identificados por cada sujeto experimental, para obtener el número de elementos faltantes. Para manejar la variabilidad de los diagramas base (dado que un mismo enunciado puede generar un número variable de diagramas del mismo tipo), se toma en consideración el hecho de que algunos elementos se pueden representar de manera diferente; por ejemplo, algunos

elementos podrían ser considerados clases o atributos, siendo igualmente válidos los resultados.

- Número de elementos sobrantes (completitud): Los elementos que no se encuentren presentes en los diagramas base (y, consecuentemente, que no se encuentran declarados dentro del enunciado) se anotan para conformar los elementos sobrantes de cada uno de los diagramas. Esta es una tendencia de los analistas a completar los diagramas sin tomar en consideración la opinión de los interesados.
- Número de errores de consistencia: En los diagramas elaborados por los sujetos experimentales se verifican las siguientes reglas de consistencia:
  - La clase de un objeto al que llega un mensaje en el diagrama de comunicación debe ser una clase del diagrama de clases.
  - Las clases de dos objetos que intervienen como emisor y receptor de un mensaje en el diagrama de comunicación deben tener algún tipo de relación (generalización, agregación o asociación) en el diagrama de clases.
  - Todo mensaje del diagrama de comunicación debe estar asociado con una operación del diagrama de clases. La clase del objeto de llegada en el diagrama de comunicación debe ser la clase que posee la operación en el diagrama de clases.
  - El nombre del entorno de un diagrama de máquina de estados debe corresponder al nombre de una clase del diagrama de clases.
  - Cada estado de un diagrama de máquina de estados se origina a partir de una operación de la clase a la cual pertenece el diagrama de máquina de estados.
  - Todos los estados de los diagramas de estados que se generen deben corresponder a mensajes del diagrama de comunicación. El nombre del

entorno en el cual se encuentre el estado debe corresponder a la clase del objeto de llegada del mensaje en el diagrama de comunicación.

- Una condición de guarda que esté presente en el diagrama de máquina de estados debe estar representada como una condición de guarda del diagrama de comunicación y viceversa.
- Todo mensaje del diagrama de comunicación debe estar asociado con un estado del diagrama de máquina de estados. La clase del objeto de llegada del mensaje deberá corresponder con el nombre del entorno del diagrama de máquina de estados.

Cuando se solicita la construcción de únicamente dos diagramas (comunicación y máquina de estados), sólo se deben analizar las tres últimas reglas de consistencia.

- Número de errores de corrección: Los errores de corrección se generan por el uso inadecuado de los símbolos disponibles para los diferentes diagramas. El cálculo se hace totalizando los errores cometidos por cada uno de los sujetos experimentales en este aspecto.

- Resultados Experimentales: la Tabla 4 recoge los resultados de la aplicación del experimento a cinco grupos de sujetos experimentales. Se aprecia en dicha Tabla que el promedio de errores es similar para los grupos con tres diagramas y para los grupos con dos diagramas. Los errores de consistencia fueron siempre los más bajos excepto para el grupo 1, conformado por los sujetos más experimentados en el uso de UML, los cuales cometieron el número menor de errores en corrección. Para los grupos 1, 2 y 3, a quienes se encomendó la tarea de realizar tres diagramas, se nota una tendencia a reconocer pocos elementos de los diagramas, pero también a "inventar" pocos elementos inexistentes en los diagramas. En los grupos restantes, con dos diagramas por realizar, los errores cometidos en el reconocimiento de elementos presentes en el enunciado y en la tendencia a "inventar" elementos ausentes en el mismo fueron más balanceados. En general, todos los grupos cometieron errores de algún tipo, lo que

suministra argumentos para negar la hipótesis nula y con ello entregar un poco más de credibilidad a la hipótesis alternativa.

Tabla 4. Resultados de la aplicación del experimento a cinco grupos de sujetos experimentales.

Descripción	Grupo 1	Grupo 2	Grupo 3	Grupo 4	Grupo 5
Tiempo de elaboración (Minutos)	50	50	25	50	25
Número de Diagramas	3	3	3	2	2
Experiencia (Meses)	12	6	6	6	6
Promedio faltantes completitud	31,4	35,3	46,2	9,3	17,3
Promedio sobrantes completitud	12,2	14,1	8,1	18,7	16,6
Promedio errores consistencia	7,5	6,5	5,3	5,1	4,1
Promedio errores corrección	4,9	13,1	8,6	7,1	9,4
Promedio total errores	56,0	69,0	68,2	40,2	47,5

Este experimento muestra la necesidad de emplear herramientas y métodos como los descritos en esta Tesis, con el fin de minimizar los errores que se cometen al elaborar los diagramas de UML. En el caso del entorno conformado por el UN-Lencep y el UNC-Diagramador los errores se subsanan de la siguiente manera:

- Errores de completitud faltantes y sobrantes: Si bien aún no existe un método para obtener el discurso en UN-Lencep a partir de un enunciado (así este enunciado se escriba en lenguaje controlado), sí se puede garantizar que, siempre y cuando el analista y el interesado traten de construir el discurso en UN-Lencep respetando al máximo los términos del enunciado, no se crearán conceptos o relaciones que no se establezcan explícitamente en el discurso en UN-Lencep. Además, como las reglas de generación de Esquemas Preconceptuales a partir de UN-Lencep y las reglas para la generación de esquemas conceptuales de UML a partir de Esquemas Preconceptuales se han

definido de manera unívoca, se garantiza que se crearán los elementos de los diagramas de UML que se encuentren presentes en el discurso en UN-Lencep. De esta manera, se atacan los errores de completitud por falta de identificación de elementos y por exceso de creación de elementos inexistentes en el discurso.

- Errores de consistencia: Por el proceso mismo de generación de los diagramas de UML a partir de Esquemas Preconceptuales, se garantiza que por lo menos las reglas de consistencia que se establecieron en esta Sección se tomarán en consideración en la elaboración de los diagramas de manera automática, liberando al analista de la responsabilidad en la verificación de la consistencia entre los diagramas de UML resultantes.
- Errores de corrección: El uso de una herramienta que, como UNC-Diagramador, emplea la simbología de UML, elimina la gran mayoría de errores que se cometen en la realización de los diagramas. Además, a diferencia de las herramientas CASE convencionales, en el caso de UNC-Diagramador la generación de los diagramas (por lo menos los iniciales que se derivan del discurso) es automática, lo que minimiza la posibilidad de cometer errores de corrección al elaborar los diagramas.

- Amenazas a la validez de los resultados: En la realización de los experimentos se procuró incentivar la cooperación de los sujetos experimentales (estudiantes en todos los casos) con incentivos académicos como el mejoramiento de alguna de las notas de los respectivos cursos, con el fin de eludir amenazas relacionadas con la falta de motivación. Se procuró también que las tareas a realizar fueran cortas para evitar el cansancio de los participantes y se les permitió conservar el plegable durante la realización del experimento, con el fin de que no fuera un asunto memorístico sino de análisis el que condujera los resultados del experimento. Pese a lo anterior, la principal amenaza a la validez de los resultados la constituye el proceso mismo de elicitación de requisitos, que es altamente complejo, y para el cual aún no existe un conjunto de reglas predefinidas que guíen su elaboración; por ejemplo, un aspecto que se notó en la

realización de los experimentos fue el hecho de que, a pesar de contar con los plegables para completar las tareas experimentales, algunos de los participantes se apartaban de la notación estándar, o cometían errores que se encontraban explícitos al interior de los plegables. Una de las posibles explicaciones a estos tipos de comportamientos en los participantes radica precisamente en el hecho de que podrían no entender el proceso que debían realizar, por la alta complejidad que presenta.

6. ESTRATEGIAS DE DIVULGACIÓN DE LOS RESULTADOS

Esta Tesis ha generado un conjunto de resultados directos e indirectos, que se han traducido en formación de estudiantes de Maestría y pregrado, proyectos de investigación y publicaciones en revistas y congresos a nivel nacional e internacional. El resumen general de productos se aprecia en la Tabla 5. El detalle de los diferentes productos se puede consultar en las secciones 6.1. a 6.9.

Tabla 5. Resumen General de Productos de divulgación de resultados.

Producto	**Cantidad**	**Estado**
Tesis de Maestría	1	Culminada y Aprobada. Con mención "Meritoria"
Tesis de Maestría	2	Culminadas y Aprobadas
Tesis de Maestría	1	En proceso
SUBTOTAL TESIS DE MAESTRÍA	**4**	
Trabajos Dirigidos de Grado	3	Culminados y Aprobados
SUBTOTAL TRABAJOS DIRIGIDOS DE GRADO	**3**	
Proyectos de Investigación	2	
SUBTOTAL PROYECTOS DE INVESTIGACIÓN	**2**	
Artículo en revista indexada internacional A1	1	Publicado
Artículo en revista indexada internacional A1	1	En evaluación
Artículos en revista indexada nacional B	1	Publicado
Artículos en revista indexada nacional B	1	Aprobado para publicación
Artículos en revista indexada nacional B	3	En evaluación
Artículos en revista indexada nacional C	4	Publicado
Artículos en revista indexada nacional C	1	Aprobado para publicación
Artículos en revista indexada nacional C	2	En evaluación
SUBTOTAL ARTÍCULOS EN REVISTAS INDEXADAS	**14**	
Artículos en revista no indexada internacional	1	Publicado
Artículos en revista no indexada nacional	2	Publicado
SUBTOTAL ARTÍCULOS EN REVISTAS NO INDEXADAS	**3**	
Ponencias en congresos internacionales	4	Presentadas y publicadas
Ponencias en congresos nacionales	3	Presentadas y publicadas
SUBTOTAL PONENCIAS EN CONGRESOS	**7**	
TOTAL PRODUCTOS	**33**	

6.1. Tesis de Maestría

Título:	"MÉTODO PARA EL RECONOCIMIENTO SEMIAUTOMÁTICO DE OPERACIONES DEL DIAGRAMA DE CLASES A PARTIR DE GRAFOS CONCEPTUALES"
Autor:	Aldrin Fredy Jaramillo
Tutores:	Carlos Mario Zapata J. Fernando Arango Isaza
Estado:	Culminada y Aprobada, con mención de "Meritorio"

Resumen:

Esta tesis pertenece a la línea de automatización de procesos de Ingeniería de requisitos (IR). La automatización de procesos de IR cobra relevancia debido a que la mayoría de los defectos del software se originan en esta actividad y se producen por problemas ocasionados por la intervención del analista y su interacción con los interesados (clientes y usuarios). En este contexto son diferentes las propuestas que buscan la automatización del proceso de modelamiento conceptual de un dominio a través del diagrama de clases. Ahora, aunque dichas propuestas reconocen automática o semiautomáticamente elementos del diagrama de clases como: Clases, asociaciones y atributos, no reconocen las operaciones presentes en las descripciones de los sistemas (escenarios). Teniendo en cuenta lo anterior, esta tesis busca resolver la pregunta de investigación: ¿Es posible plantear un método para el reconocimiento semiautomático de operaciones del diagrama de clases a partir de grafos conceptuales? Con lo cual se busca contribuir al proceso de automatización de la etapa de modelamiento conceptual y en última instancia al mejoramiento de la calidad de los artefactos de software construidos.

El método propuesto está fundamentado en teoría de grafos conceptuales y teorías lingüísticas, que permiten, a partir de los elementos presentes en los grafos conceptuales en combinación con las características de los verbos y las frases, reconocer de manera semiautomática las operaciones. Con el fin de validar el método se desarrolló una herramienta denominada UN-Mapper la cual, una vez sometida al análisis de siete

escenarios con el fin de reconocer sus operaciones, presentó una completitud del 73.07% y una precisión del 63.33%.

Las contribuciones de la investigación son del orden conceptual y metodológico; además, exceden el alcance inicial de la investigación (reconocimiento semiautomático de operaciones), y están dadas por la definición de las características aspectuales de una operación, el planteamiento de heurísticas originales para la identificación de operaciones, clases, atributos y asociaciones del diagrama de clases y la conformación del método que permite el reconocimiento semiautomático de operaciones. Entre los trabajos futuros se tienen: Plantear un método de identificación automática del sentido de los verbos, extender la propuesta al reconocimiento de elementos de otros diagramas UML como actividades y casos de uso y el desarrollo de un componente para la generación automática de grafos conceptuales de Sowa a partir de escenarios.

Título:	"REFINAMIENTO DEL DIAGRAMA DE CLASES DE UML BASADO EN LENGUAJE NATURAL"
Autora:	Betsy Mary Estrada Perea
Tutores:	Carlos Mario Zapata J. Fernando Arango Isaza
Estado:	Culminada y Aprobada

Resumen:

Durante el proceso de elicitación de requisitos se presentan problemas de comunicación entre analistas e interesados que suelen ocasionar pérdidas de requisitos funcionales. Estas pérdidas se aminoran mediante el refinamiento de los esquemas conceptuales, en particular el diagrama de clases de UML.

Existen algunos acercamientos al refinamiento del diagrama de clases, pero que no realizan ciclos de interacción con el interesado; otros enfoques realizan refinamiento interactivo del diagrama entidad-relación, un diagrama que no posee toda la información contenida en el diagrama de clases.

En este trabajo se realiza el refinamiento del diagrama de clases de UML mediante la interacción con el interesado. Para ello, se proponen reglas de completitud que se disparan en lenguaje natural y se emplea un corpus de diagramas de clases para complementar el conocimiento del analista en un determinado dominio. El análisis de completitud propuesto se ilustra con un prototipo en la herramienta UNC-Diagramador y se ejemplifica con un caso de estudio.

Palabras Clave: Refinamiento, Reglas de Completitud, Diagrama de Clases de UML.

Título:	"ESPECIFICACIÓN FORMAL EN OCL DE REGLAS DE CONSISTENCIA ENTRE EL MODELO DE CLASES Y DE CASOS DE USO DE UML"
Autor:	Guillermo González Calderón
Tutores:	Carlos Mario Zapata J. Fernando Arango Isaza
Estado:	Culminada y Aprobada

Resumen:

En el ciclo de vida del software, durante las fases de definición y análisis se realiza una especificación de los requisitos. Para ello, es necesario realizar un proceso de captura de las necesidades y expectativas de los interesados, que se traduce posteriormente en un conjunto de modelos que representan tanto el problema como su solución. Por lo general, la mayoría de esos modelos se expresan en el Lenguaje Unificado de Modelamiento (UML), propuesto por el Grupo de Gestión de Objetos (Object Management Group, OMG). UML define un conjunto de artefactos que permiten especificar los requisitos del software, los cuales deberían guardar consistencia cuando se traten del mismo modelo, ya que están definidos bajo las reglas de buena formación (Well-Formedness Rules, WFR), las cuales se encuentran definidas dentro de la especificación de UML en el Lenguaje de Restricciones de Objetos (Object Constraint Language, OCL). La consistencia interna de cada artefacto está por tanto definida en la especificación de UML y algunas de las herramientas CASE (Computer-Aided Software Engineering) utilizan esas reglas intramodelo para validar este tipo de consistencia. Sin embargo la consistencia entre diferentes artefactos no se encuentra

definida en la especificación de UML y poco se ha trabajado con este tipo de consistencia; además, los trabajos que se han realizado en este tema aún poseen problemas:

- No se suele definir de manera formal la consistencia entre los diferentes artefactos.
- Por lo general se estudia sólo la consistencia entre alguno de los artefactos y el código resultante.
- Algunos establecen reglas formales de consistencia, pero únicamente a nivel de intramodelo.

Entre los artefactos más utilizados para especificar una pieza de software se encuentran el diagrama de clases y el diagrama de casos de uso, los cuales suministran dos perspectivas diferentes del desarrollo de software: por un lado el diagrama de clases, de tipo estructural, muestra los objetos del mundo y sus relaciones; por otro lado, el diagrama de casos de uso, de tipo comportamental, se concentra en las funciones que realizan los actores del mundo para lograr un resultado en el software. Estos dos puntos de vista deberían ser complementarios y, por ello, tener información común, susceptible de ser sometida a un análisis de consistencia.

En esta Tesis se propone un método para verificar la consistencia entre el diagrama de clases y el diagrama de casos de uso de UML de una manera formal. Dicho proceso se lleva a cabo evaluando una serie de reglas definidas en OCL que se deben cumplir para garantizar que la información brindada por dichos modelos sea consistente. Como se reconoce la participación de los dos diagramas en la elaboración de las Interfaces Gráficas de Usuario (Graphical User Interfaces, GUI), se define adicionalmente la consistencia con este artefacto. Las reglas se implementaron en XQuery, utilizando como diagramas de entrada representaciones en XML generadas con la herramienta CASE ArgoUML® (en el caso del diagrama de clases y de casos de uso) y con Microsoft Visio® (en el caso de las GUI). Finalmente, se muestra un caso de estudio donde se aplican estas reglas y se

muestran los posibles errores y advertencias que se tienen entre los elementos de tales artefactos.

Título:	"ELABORACIÓN DE DIAGRAMAS DE CASO DE USO A PARTIR DE MODELOS VERBALES"
Autora:	Paula Andrea Tamayo
Tutores:	Carlos Mario Zapata J. Fernando Arango Isaza
Estado:	En proceso

Resumen:

A partir de la crisis de software que se presentó en los años 60s, nace una nueva ciencia denominada "Ingeniería de Software". El principal objetivo de esta ciencia es aplicar los conceptos y principios de la ingeniería en el desarrollo e implementación de sistemas informáticos. Para el desarrollo de estos sistemas se emplean diversos modelos conceptuales, como por ejemplo los Diagramas de Casos de Uso, que permiten capturar los requisitos funcionales del sistema desde el punto de vista del interesado.

Sin embargo, la Ingeniería de Software trajo consigo problemas que se presentan sobre todo en la interacción entre los interesados y el analista; para ser más específicos, los problemas principales que se presentan son: la comunicación insuficiente entre el interesado y el analista y la incompatibilidad entre los lenguajes propios del dominio del interesado y el técnico del analista.

Esta propuesta aborda la obtención del diagrama de casos de uso a partir del lenguaje natural, de modo que mediante el análisis de la especificación del dominio del interesado expresada en lenguaje natural se pueda obtener cada uno de los elementos del diagrama de casos de uso; permitiendo resolver los problemas mencionados anteriormente.

PALABRAS CLAVE: Diagramas de Casos de uso, Lenguaje Natural, Lenguaje Unificado de Modelamiento, Especificación de Requisitos.

6.2. Trabajos Dirigidos de Grado

Título:	"EL JUEGO DE LA CONSISTENCIA UNA ESTRATEGIA DIDACTICA PARA LA INGENIERIA DEL SOFTWARE"
Autora:	Mary Inés Duarte Herrera
Tutor:	Carlos Mario Zapata J.
Estado:	Culminado y Aprobado

Resumen:

La Ingeniería del Software es un área de la informática. Esta área como tal brinda métodos y técnicas para desarrollar y mantener software de calidad que resuelva todo tipo de problemas de la vida diaria. Actualmente se le considera como una de las áreas principales de la ingeniería y de las ciencias de computación, ya que aborda todas las fases del ciclo de vida del desarrollo de cualquier tipo de sistema de información. Su aplicación se puede notar en áreas tan diversas como la banca, la investigación científica, el control de tráfico, el mundo del derecho e Internet, entre muchas otras.

Zelkowitz ha definido la Ingeniería del Software como el estudio de los principios y las metodologías que se requieren, no sólo para el desarrollo, sino también para el mantenimiento de sistemas de software; al mismo tiempo, Boehm afirma que es la aplicación práctica del conocimiento científico, en el diseño y construcción de programas computarizados, que incluyen la documentación asociada requerida para su desarrollo, operación y mantenimiento. Esto es lo que comúnmente se conoce como producción o desarrollo de software. Lo que se pretende al poner en práctica la Ingeniería del Software, es obtener programas computarizados que sean funcionales, estables, confiables y seguros, y que dichas herramientas, operen en maquinas reales. Así, la Ingeniería del Software utiliza el modelamiento como herramienta para la solución de problemas de la vida diaria.

El modelamiento es el conjunto de abstracciones que se construyen para entender un problema antes de implementar una solución. Todas las abstracciones son un subconjunto de realidades seleccionadas para un propósito en particular. En Ingeniería del Software el

modelamiento se apoya en las técnicas de modelamiento objetual, y más recientemente en UML, que se basan en tres clases de diagramas: estáticos, dinámicos y comportamentales. Los diagramas estáticos representan estructura de un sistema en términos de objetos y relaciones de correspondencia en el mundo real. Los diagramas dinámicos describen el control de la estructura de un sistema en términos de eventos y estados. Los diagramas comportamentales describen la composición estructural de un sistema en términos de valores y funciones.

Estas clases de modelamiento se pueden usar conjuntamente, pero para ello deben contar con consistencia, es decir con coherencia entre modelos. La consistencia pretende que los requerimientos de una especificación no se contradigan entre sí.

Ahora, Zapata y Awad afirman que los ingenieros de software deben tener aptitudes de tipo administrativo, que poco se cultivan en la enseñanza convencional. Hoy día la enseñanza no se basa sólo en la impartida por el maestro, sino que se vuelca hacia el estudiante como elemento central de la clase. Es por ello que ahora se emplean los juegos como herramienta pedagógica.

En este trabajo se define “El juego de la Consistencia”, una herramienta didáctica, para la enseñanza de Ingeniería del Software.

Palabras clave: Esquemas preconceptuales, diagrama de clases, diagrama de secuencias, diagrama de casos de uso, consistencia, juegos.

Título:	“DESARROLLO DE UN PROTOTIPO DE DESAMBIGUADOR SINTÁCTICO PARA EL LENGUAJE NATURAL EN ESPAÑOL”
Autores:	Karla Cristina Palomino Z. Roberto José Rosero
Tutor:	Carlos Mario Zapata J.
Estado:	Culminado y Aprobado

Resumen:

El procesamiento del lenguaje Natural (PLN) investiga y formula mecanismos computacionales que permiten la comunicación hombre-máquina. Conceptualmente, un sistema de PLN se divide en tres procesos principales: análisis morfológico, sintáctico y semántico. En cada uno de estos procesos es factible que se presenten múltiples interpretaciones de una misma palabra o frase, según sea el proceso que se esté llevando a cabo; estas interpretaciones dan origen al concepto de ambigüedad. Para resolver la ambigüedad se han propuesto métodos basados en estadística, inteligencia artificial y métodos híbridos, los cuales aún presentan dificultades como el alto consumo de recursos léxicos y computacionales y el uso de elementos pertenecientes a dominios restringidos. En este artículo se propone un método basado en reglas heurísticas para resolver la ambigüedad sintáctica de tipo coordinativo y preposicional. Se muestra igualmente la implementación del método empleando herramientas del paquete NLTK y se ejemplifica la desambiguación de dos frases.

PALABRAS CLAVE: Procesamiento del Lenguaje Natural, Análisis sintáctico, Información sintáctica y semántica, Ambigüedad sintáctica Coordinativa, Ambigüedad sintáctica preposicional, Desambiguación.

Título:	"ANALIZADOR MORFOLÓGICO DE VERBOS DEL ESPAÑOL"
Autores:	John Edison Mesa Bedoya
Tutor:	Carlos Mario Zapata J.
Estado:	Culminado y Aprobado

Resumen:

El análisis morfológico de verbos del español no es una tarea fácil gracias a sus peculiares características. Una de esas características es la gran cantidad de conjugaciones que puede poseer un verbo, lo cual dificulta la generación automática de dichas conjugaciones. Aunque en la actualidad existen propuestas de analizadores de verbos que realizan la

conjugación y lematización de estos, aún presentan problemas. En este artículo se propone un analizador morfológico que utiliza plantillas para generalizar las conjugaciones de los verbos. Con estas plantillas se pueden realizar los procesos de conjugación y lematización sin tener que almacenar todas las posibles conjugaciones de un verbo de manera independiente. La implementación del analizador morfológico de verbos se hizo en el lenguaje de programación Python para aprovechar las capacidades que este ofrece este lenguaje en la manipulación de cadenas de texto.

6.3. Proyectos de Investigación

Este trabajo ha motivado la aparición de dos (2) proyectos de investigación: "Construcción Automática de Esquemas Conceptuales a partir de Lenguaje Natural", financiado por la DIME y "Definición de un Esquema Preconceptual para la Obtención Automática de Esquemas Conceptuales de UML", financiado por DINAIN y administrado por la DIME.

6.4. Artículos en revistas indexadas internacionales

Título:	"Pre-conceptual Schema: A Conceptual-Graph-Like Knowledge Representation for Requirements Elicitation" (Zapata *et al.*, 2006a)
Autores:	Carlos Mario Zapata J. Alexander Gelbukh Fernando Arango I.
Revista:	Lecture Notes in Computer Sciences, Springer, CATEGORIA A1
Estado:	Publicado.

Resumen:

A simple representation framework for ontological knowledge with dynamic and deontic characteristics is presented. It represents structural relationships (is-a, part/whole), dynamic relationships (actions such as register, pay, etc.), and conditional relationships (if-then-else). As a case study, we apply our representation language to the task of requirements elicitation in software engineering. We show how our pre-conceptual schemas can be

obtained from controlled natural language discourse and how these diagrams can be then converted into standard UML diagrams. Thus our representation framework is shown to be a useful intermediate step for obtaining UML diagrams from natural language discourse.

Título:	"El Juego de la Consistencia: Una Estrategia Didáctica para la Ingeniería de Software"
Autores:	Carlos Mario Zapata J. Mary Inês Duarte
Revista:	Revista Técnica de Ingeniería, Universidad del Zulia, CATEGORIA A1
Estado:	En evaluación.

Resumen:

La enseñanza de la Ingeniería de Software se ha abordado tradicionalmente con métodos convencionales como clases magistrales y proyectos prácticos. Estas estrategias, si bien han permitido la formación de profesionales en esta área durante las últimas cuatro décadas, se han centrado en el docente como fuente de aprendizaje y dejan de lado estrategias en las cuales los alumnos son el centro del aprendizaje, tales como los juegos, estudios de casos y simulaciones. En este artículo se propone "El Juego de la Consistencia", como una manera de lograr el aprendizaje de los alumnos desde una experiencia lúdica; el juego trabaja el concepto de consistencia entre diagramas de UML (Unified Modeling Language), el estándar más aceptado para el modelado de requisitos en Ingeniería de Software.

Palabras Claves: Consistencia, Juegos, UML, Ingeniería de Software.

6.5. Artículos en revistas indexadas nacionales

Título:	"Los Modelos Verbales y su utilización en la elaboración de Esquemas Conceptuales: Una revisión crítica" (Zapata y Arango, 2005)
Autores:	Carlos Mario Zapata J. Fernando Arango Isaza
Revista:	Revista Universidad Eafit CATEGORIA C
Estado:	Publicado.

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 del Software

Título:	"Una propuesta para el Reconocimiento Semiautomático de Operaciones utilizando un enfoque lingüístico" (Jaramillo *et al.*, 2005)
Autores:	Aldrin Fredy Jaramillo Carlos Mario Zapata J. Fernando Arango Isaza
Revista:	Revista Ingeniería Universidad de Antioquia CATEGORIA C
Estado:	Publicado.

Resumen:

En el contexto de automatización de los procesos de desarrollo de software el reconocimiento automático de las operaciones de las clases a partir de las descripciones textuales de un sistema es un tema de investigación que permanece abierto. En este artículo se presenta una propuesta que aborda este problema, la cual está basada en Grafos Conceptuales de Sowa y un tratamiento lingüístico de sus componentes que se fundamenta en la identificación de la clasificación del verbo tomando como base tres características: telicidad, dinamicidad y durabilidad. Esta propuesta es uno de los resultados obtenidos en la investigación de Maestría: "Método para el reconocimiento semiautomático de

operaciones a partir de grafos conceptuales de Sowa" adelantada en la Universidad Nacional de Colombia bajo la tutoría del Grupo de Investigación en Informática UN-INFO.

Palabras clave: Ingeniería de requisitos, reconocimiento automático de operaciones, diagrama de clases, Procesamiento de Lenguaje Natural (PLN), grafos de Sowa, generación automática de diagramas UML.

Título:	"Una propuesta para mejorar la completitud de requisitos utilizando un enfoque lingüístico" (Zapata *et al.*, 2006d)
Autores:	Carlos Mario Zapata J. Aldrin Fredy Jaramillo Fernando Arango I.
Revista:	Ingeniería y Desarrollo, Universidad del Norte, CATEGORIA C
Estado:	Publicado.

Resumen:

La calidad de los productos de software está estrechamente relacionada con la calidad de los requisitos especificados desde las primeras etapas del proceso de desarrollo. En este artículo se presenta una propuesta para mejorar la calidad, en cuanto a completitud, de especificaciones de requisitos escritas en un subconjunto del español denominado español restringido. Esta propuesta es uno de los resultados obtenidos en la investigación de Maestría: "Método para el reconocimiento semiautomático de operaciones a partir de esquemas preconceptuales" adelantada en la Universidad Nacional de Colombia bajo la tutoría del Grupo de Investigación en Informática UN-INFO.

Palabras claves: Ingeniería de requisitos, calidad de requerimientos, completitud de requerimientos, Procesamiento de Lenguaje Natural (PLN), gramática de casos, análisis automático de completitud de requisitos.

Título:	"El Experimento Mago de Oz y sus Aplicaciones: Una Mirada Retrospectiva" (Zapata y Carmona, 2007)
Autores:	Carlos Mario Zapata J. Nicolás Carmona
Revista:	Dyna, Facultad de Minas, Universidad Nacional de Colombia, CATEGORIA B
Estado:	Publicado.

Resumen:

En la Lingüística Computacional los corpus han sido uno de los recursos más utilizados para estudiar los diferentes fenómenos lingüísticos. La recopilación de corpus en los cuales se presenta interacción entre el humano y el computador presenta como problema fundamental los errores que se pueden inducir por parte del humano, cuyo lenguaje se simplifica en presencia del computador. Para realizar la recopilación de estos corpus se han empleado los Experimentos Mago de Oz (MDO). En este artículo se muestra una mirada retrospectiva del Experimento Mago de Oz, clasificando sus principales aplicaciones.

Palabras Clave: Corpus, Mago de Oz, Lingüística Computacional, Reconocimiento del habla.

Título:	"Un ambiente para la Obtención automática de Diagramas UML a partir de un Lenguaje controlado" (Zapata y Arango, pendiente publicación).
Autores:	Carlos Mario Zapata J. Fernando Arango I.
Revista:	Dyna, Facultad de Minas, Universidad Nacional de Colombia, CATEGORIA B
Estado:	Aprobado para publicación

Resumen:

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

Diagramador, en la cual se presenta adicionalmente un caso de estudio. Con esta implementación se apoya la labor de conceptualización de los modelos por parte de los analistas y se mejora la consistencia, puesto que los modelos resultantes se elaboran a partir del mismo texto en lenguaje controlado.

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

Título:	"Un método para el refinamiento interactivo del diagrama de clases de UML" (Zapata *et al.*, en evaluación b).
Autores:	Carlos Mario Zapata J. Betsy Mary Estrada P. Fernando Arango I.
Revista:	Dyna, Facultad de Minas, Universidad Nacional de Colombia, CATEGORIA B
Estado:	En evaluación

Resumen:

Durante el proceso de elicitación de requisitos se presentan problemas de comunicación entre analistas e interesados que suelen ocasionar pérdidas de requisitos funcionales. Estas pérdidas se aminoran mediante el refinamiento de los esquemas conceptuales, en particular el diagrama de clases de UML. Existen algunos acercamientos al refinamiento del diagrama de clases, pero que no realizan ciclos de interacción con el interesado; otros enfoques realizan refinamiento interactivo del diagrama entidad-relación, un diagrama que no posee toda la información contenida en el diagrama de clases. En este artículo se realiza el refinamiento del diagrama de clases de UML mediante la interacción con el interesado. Para ello, se proponen reglas de completitud que se disparan en lenguaje natural y se emplea un corpus de diagramas de clases para complementar el conocimiento del analista en un determinado dominio. El análisis de completitud propuesto se ilustra con un prototipo en la herramienta UNC-Diagramador y se ejemplifica con un caso de estudio.

Palabras Clave: Refinamiento, reglas de completitud, diagrama de clases de UML, corpus de diagramas.

Título:	"Conversión de Esquemas Preconceptuales a diagramas de casos de uso empleando AToM^3" (Zapata *et al.*, en evaluación e).
Autores:	Carlos Mario Zapata J. Paula Andrea Tamayo O. Fernando Arango I.
Revista:	Dyna, Facultad de Minas, Universidad Nacional de Colombia, CATEGORIA B
Estado:	En evaluación

Resumen:

El diagrama de Casos de Uso describe las interacciones entre un usuario y una pieza de software. Se han realizado algunos trabajos que buscan la generación automática o semi-automática del diagrama de Casos de Uso desde descripciones en lenguajes naturales o restringidos. Sin embargo, estos esfuerzos no han sido suficientes porque algunos parten de un lenguaje restringido orientado a la solución, la cual no existe en las etapas iniciales del ciclo de vida del software; otros trabajos requieren una alta intervención del analista para la generación del diagrama, lo cual es altamente inconveniente si se trata de automatizar el proceso; finalmente, no se identifican todos los elementos del diagrama de Casos de Uso, en particular las relaciones especiales (include, extends e inheritance). En este artículo se define un método basado en reglas heurísticas que permite identificar los actores, los casos de uso y las relaciones especiales del Diagrama de Casos de Uso, tomando como punto de partida una representación en lenguaje restringido del dominio del problema: los denominados Esquemas Preconceptuales. Además, se realiza la implementación de estas heurísticas en la herramienta MetaCASE AToM3 y se ejemplifica con un caso de estudio.

PALABRAS CLAVE: Metamodelamiento, Diagrama de Casos de Uso, Esquemas Preconceptuales.

Título:	"Un método para la desambiguación sintáctica de tipo coordinativo y preposicional" (Zapata *et al.*, en evaluación c).
Autores:	Carlos Mario Zapata J. Karla Cristina Palomino Roberto José Rosero
Revista:	Dyna, Facultad de Minas, Universidad Nacional de Colombia, CATEGORIA B
Estado:	En evaluación

Resumen:

El procesamiento del lenguaje Natural (PLN) investiga y formula mecanismos computacionales que permiten la comunicación hombre-máquina. Conceptualmente, un sistema de PLN se divide en tres procesos principales: análisis morfológico, sintáctico y semántico. En cada uno de estos procesos es factible que se presenten múltiples interpretaciones de una misma palabra o frase, según sea el proceso que se esté llevando a cabo; estas interpretaciones dan origen al concepto de ambigüedad. Para resolver la ambigüedad se han propuesto métodos basados en estadística, inteligencia artificial y métodos híbridos, los cuales aún presentan dificultades como el alto consumo de recursos léxicos y computacionales y el uso de elementos pertenecientes a dominios restringidos. En este artículo se propone un método basado en reglas heurísticas para resolver la ambigüedad sintáctica de tipo coordinativo y preposicional. Se muestra igualmente la implementación del método empleando herramientas del paquete NLTK y se ejemplifica la desambiguación de dos frases.

PALABRAS CLAVE: Procesamiento del Lenguaje Natural, Análisis sintáctico, Información sintáctica y semántica, Ambigüedad sintáctica Coordinativa, Ambigüedad sintáctica preposicional, Desambiguación.

Título:	"Reglas de Conversión entre el Diagrama de Clases y los Grafos Conceptuales de Sowa" (Zapata *et al.*, 2006e).
Autores:	Carlos Mario Zapata J. Betsy Mary Estrada P. Guillermo González C.
Revista:	Revista Ingenierías Universidad de Medellín, CATEGORIA C
Estado:	Publicado.

Resumen:

La conversión entre modelos de un nivel de abstracción inferior a otro de nivel de abstracción superior facilita la comunicación entre los involucrados en un proceso de desarrollo de software. Los grafos conceptuales son diagramas que presentan la información modelada de una manera semiformal, y pueden llegar a ser comprensibles tanto por el humano como por el computador. El diagrama de clases, en cambio, presenta las clases, atributos, operaciones y relaciones principales de un sistema en un lenguaje propio de los expertos en modelamiento de productos de software. En este artículo se propone un conjunto de reglas de conversión para traducir el diagrama de clases (más detallado y, en consecuencia, de bajo nivel de abstracción) en una forma más comprensible al interesado (y de más alto nivel de abstracción) como lo son los grafos conceptuales de Sowa.

PALABRAS CLAVE: Reglas de conversión, Grafos conceptuales de Sowa, Diagrama de clases.

Título:	"UNC-Analista: hacia la captura de un corpus de requisitos a partir de la aplicación del experimento Mago de Oz" (Zapata *et al.*, pendiente publicación b).
Autores:	Carlos Mario Zapata J. Carolina Palacio López Natalí Olaya Morales
Revista:	Revista EIA, Escuela de Ingeniería de Antioquia, CATEGORIA C
Estado:	Aprobado para publicación.

Resumen:

La elicitación de requisitos es una de las etapas más importantes en el proceso de desarrollo del software. Una buena comprensión de los requisitos puede conducir a mejores productos de software que satisfagan las necesidades de los interesados. Sin embargo, el proceso de captura de requisitos se torna difícil para el analista debido, en gran parte, al carácter presencial que tienen las reuniones que se realizan con tal fin y a la dificultad que presentan algunas personas para expresar sus ideas de forma clara. En este artículo se presenta UNC-

Analista, una propuesta de experimento Mago de Oz enfocado al diseño de un sistema de diálogo controlado que posibilite la labor del analista durante el proceso de elicitación de requisitos. Con este sistema será posible capturar un corpus de requisitos, que servirá como base para la construcción futura de un sistema automático para la elicitación de requisitos.

PALABRAS CLAVE: requisitos; software; corpus; experimento Mago de Oz; sistemas de diálogo.

Título:	"Analizador Sintáctico de Lenguaje Natural con Reglas Editables" (Zapata y Hernández, en evaluación).
Autores:	Carlos Mario Zapata J. Juan Carlos Hernández
Revista:	Avances en Sistemas e Informática, Escuela de Sistemas, Universidad Nacional de Colombia, CATEGORIA C
Estado:	En evaluación.

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 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. 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.

Palabras clave: lenguaje natural, análisis sintáctico, gramática de contexto li-bre, ambigüedad, constituyentes, diagramas UML.

Título:	"Una propuesta para la Asistencia al Proceso de Interpretación de textos utilizando técnicas de Procesamiento del Lenguaje Natural e Ingeniería de Software" (Jaramillo *et al.*, en evaluación).
Autores:	Aldrin Fredy Jaramillo Carlos Mario Zapata J. Fernando Arango I.
Revista:	Avances en Sistemas e Informática, Escuela de Sistemas, Universidad Nacional de Colombia, CATEGORIA C
Estado:	En evaluación.

Resumen:

En este artículo se presenta una propuesta para la asistencia al proceso de interpretación de textos. La propuesta está basada en la generación automática, a partir del texto, de un esquema conceptual utilizado en ingeniería de software llamado modelo Entidad-Relación (ER). Además, se muestra la utilidad del modelo ER en el proceso de interpretación de textos, así como las técnicas de Procesamiento de lenguaje natural y de Ingeniería de Software que se utilizan para su derivación automática. Los resultados obtenidos muestran cómo el modelo ER puede ser una valiosa herramienta de apoyo al proceso de interpretación gracias a las inferencias que de manera automática se realizan a través de él. Este trabajo es uno de los resultados obtenidos en la investigación de Maestría: "Método de generación automática de diagramas de clases a partir de esquemas preconceptuales" culminada en la Universidad Nacional de Colombia bajo la tutoría del Grupo de Investigación en Ingeniería de Software.

Palabras claves: Interpretación de textos, Procesamiento del Lenguaje Natural, esquema conceptual, Lingüística Computacional.

6.6. Artículos en Revistas no Indexadas Internacionales

Título:	"Pre-conceptual Schema: a UML isomorphism for automatic obtaining of UML conceptual schemas" (Zapata *et al.*, 2006b).
Autores:	Carlos Mario Zapata J., Fernando Arango I., Alexander Gelbukh
Revista:	Research in Computing Science, Instituto Politécnico Nacional México, Sin CATEGORÍA para ese momento.
Estado:	Publicado.

Resumen:

Software development methodologies improve model quality and conceptual schemas are representations of the universe of discourse for development purposes. In modeling, UML had become a de-facto standard, and obtaining it from natural language is gaining importance nowadays. In this paper we present a proposal for improving some drawbacks from the previous works on this area; we call it Preconceptual Schema, because it's an intermediate stage between natural language and UML conceptual Schemas. Finally, we show a case study for rules applying.

6.7. Artículos en Revistas no Indexadas Nacionales

Título:	"Análisis de un caso de estudio en KCPM para la generación de diagramas de clases" (Zapata *et al.*, 2005).
Autores:	Carlos Mario Zapata J. Aldrin Fredy Jaramillo Fernando Arango I.
Revista:	Avances en Sistemas e Informática, Escuela de Sistemas, Universidad Nacional de Colombia, Sin CATEGORÍA para ese momento.
Estado:	Publicado.

Resumen:

Son pocas e incipientes las propuestas que abordan el problema de generación semiautomática de esquemas conceptuales (en especial el diagrama de clases) a partir de modelos verbales escritos en español. Sin embargo, para otros idiomas como inglés, francés y alemán, entre otros, existe una mayor experiencia (Denger et al., 2001). En este artículo se analiza un caso de estudio basado en el método KCPM (Klagenfurt Conceptual Predesign Model) para la generación del diagrama de clases, se discute su aplicabilidad al lenguaje español y se proponen mejoras que permitan obtener mayor completitud de los diagramas generados y un mecanismo de asistencia al usuario que retroalimente la manera de generación de los esquemas.

Palabras Clave: Diagrama de Clases, KCPM, generación semiautomática de esquemas conceptuales.

Título:	"Analizador de Completitud de Requisitos escritos en Español restringido" (Navarro *et al.*, 2005).
Autores:	Javier Navarro Ricardo Orozco Aldrin Fredy Jaramillo F. Carlos Mario Zapata J.
Revista:	Avances en Sistemas e Informática, Escuela de Sistemas, Universidad Nacional de Colombia, Sin CATEGORÍA para ese momento.
Estado:	Publicado.

Resumen:

En este artículo se propone una herramienta para el análisis de completitud de requisitos escritos en Español restringido. La herramienta implementa un método para mejorar la completitud de los requisitos utilizando un enfoque lingüístico, basado en los casos semánticos asociados al sentido de un verbo. Se presenta también el proceso seguido en la evaluación de la efectividad de la herramienta al analizar la completitud de varios casos de estudio utilizando la precisión como indicador cuantitativo de la efectividad. Los resultados preliminares obtenidos confirman las hipótesis y muestran que la herramienta servirá de ayuda en la escritura de especificaciones de requisitos completas y el mejoramiento de la calidad del software que se desarrolla a partir de ellas.

Palabras clave: Calidad de requisitos, especificación de requisitos, completitud de requisitos.

6.8. Ponencias en Congresos internacionales

Título:	"Método Semiautomático para la Identificación de Operaciones a partir de Grafos Conceptuales" (Zapata *et al.*, 2006f).
Autores:	Carlos Mario Zapata J., Aldrin Fredy Jaramillo, Fernando Arango I.
Evento:	9° Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes de Software.
Lugar:	Buenos Aires, Argentina.
Fecha:	Abril de 2006.

Resumen

Desde sus inicios, las herramientas CASE han asistido a los analistas en el trazado y edición de diferentes tipos de diagramas. Actualmente existe una tendencia para asistir a los analistas en la concepción de algunos diagramas UML, como el diagrama de clases. En los trabajos correspondientes a esta tendencia, el reconocimiento de las operaciones del diagrama de clases sigue siendo un problema abierto. En este artículo se presenta un método para la identificación de las operaciones del diagrama de clases partiendo de Grafos Conceptuales y empleando para ello un enfoque lingüístico.

Palabras clave: Diagrama de Clases UML, Operaciones, Grafos Conceptuales, Estructuras Léxicas Conceptuales.

Título:	"UN-Lencep: Obtención Automática de Diagramas UML a partir de un Lenguaje Controlado" (Zapata *et al.*, 2006c).
Autores:	Carlos Mario Zapata J. Alexander Gelbukh Fernando Arango I.
Evento:	VII Encuentro Nacional de Computación ENC'06.
Lugar:	San Luis Potosí, México.
Fecha:	Septiembre de 2006.

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 Natura, para obtener una especificación; dicha especificación suele estar conformada por un conjunto de diagramas (generalmente 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 Pre-conceptuales, que se

utilizan en la automatización del proceso de elaboración de diagramas UML. Adicionalmente, se muestra un ejemplo de la aplicación de UN-Lencep.

Título:	"Elicitación de Requisitos empleando UN-Lencep y Esquemas Preconceptuales" (Zapata y Arango, 2007).
Autores:	Carlos Mario Zapata J. Fernando Arango I.
Evento:	VI Jornadas Iberoamericanas de Ingeniería del Software e Ingeniería del Conocimiento.
Lugar:	Lima, Perú.
Fecha:	Enero de 2007.

Resumen:

Con la Elicitación de Requisitos se pretende la captura de las necesidades de los interesados y su traducción en especificaciones del software por construir. Las técnicas tradicionales de elicitación de requisitos tienen una alta incidencia del analista y son difíciles de automatizar, debido a la subjetividad del proceso. Por ello, desde mediados de los noventa se viene presentando una tendencia hacia la automatización del proceso de elicitación de requisitos, especialmente para especificaciones en diagramas de UML; en estos trabajos aún subsisten problemas, pues no se obtienen todos los tipos de diagramas de UML y se presentan problemas de consistencia entre los diagramas resultantes. En este artículo se presenta una técnica de elicitación basada en el lenguaje controlado UN-Lencep y los denominados Esquemas Preconceptuales, como una forma de solución a los problemas remanentes. Esta técnica se ejemplifica con un caso de estudio.

Título:	"Tutorial: Uso de Esquemas Preconceptuales para la generación automática de Diagramas de Clases, Comunicación y Máquina de Estados" (Zapata, 2007).
Autor:	Carlos Mario Zapata J.
Evento:	VI Jornadas Iberoamericanas de Ingeniería del Software e Ingeniería del Conocimiento.
Lugar:	Lima, Perú.
Fecha:	Enero de 2007.

Resumen:

El desarrollo de una pieza de software suele iniciar con un conjunto de entrevistas entre interesados y analistas, con el fin de descubrir las necesidades y expectativas de los

interesados y poderlas plasmar en los esquemas conceptuales que representan la solución. Este proceso, sin embargo, está plagado de inconvenientes que surgen por los problemas de comunicación que se originan por las diferencias en las especialidades de los protagonistas del desarrollo de la pieza de software: los interesados, cuyo énfasis fundamental es el dominio del problema, y los analistas, cuyos conocimientos se centran en las herramientas de modelamiento.

Sin importar cuál sea el método de desarrollo que se emplee para la pieza de software, es necesario conciliar estas dos posiciones complementarias, pero cuyos lenguajes son altamente disímiles. Los interesados, por ejemplo, se suelen expresar en lenguaje natural o en documentos de la organización para la que trabajan, en tanto que los analistas se suelen expresar en lenguajes de modelamiento, de los cuales el UML es el principal representante actualmente.

En la Escuela de Sistemas de la Universidad Nacional de Colombia, sede Medellín, se ha propuesto una forma de captura de la información concerniente al dominio del problema: los Esquemas Preconceptuales. Con estos esquemas, es posible realizar dos funciones que se suelen realizar de forma manual en el desarrollo de software:

- La representación del dominio del problema en un lenguaje gráfico entendible por los interesados y, consecuentemente, posible de validar por ellos mismos.
- La conversión de esa representación del dominio en tres de los diagramas pertenecientes a UML: clases, comunicación y máquina de estados.

En cuanto a la primera función, es posible que la construcción de esa representación del dominio se realice de manera conjunta entre los analistas y los interesados, con el fin de que se incorporen los conceptos del dominio por parte del interesado y se concilien con la visión técnica del analista, lo cual permite un establecimiento temprano de los requisitos del software y una validación de los mismos por parte de los interesados. En lo relativo a la

segunda función, la cual suele ser realizada en forma manual o con asistencia de las herramientas CASE únicamente para efectos de trazado de los diagramas de UML, los esquemas preconceptuales suministran apoyo en la concepción de los diagramas mismos, procurando la automatización de tareas normalmente desarrolladas por los analistas.

En este tutorial se pretende enseñar la sintaxis básica de los esquemas preconceptuales y las bases teóricas de su transformación en los tres diagramas mencionados del estándar UML 2.0. Posteriormente, se construirán algunos de casos de prueba de manera conjunta con los participantes en el tutorial y se mostrará el funcionamiento del prototipo de la herramienta UNC-Diagramador para la elaboración de esquemas preconceptuales y su conversión a los diagramas de UML.

Es importante anotar que el público para este tutorial puede ser diverso, variando desde interesados en el desarrollo de una pieza de software (incluyendo todas las gamas y niveles: profesionales, técnicos, auxiliares, etc.) hasta analistas y desarrolladores. Igualmente, este tutorial se convierte en una herramienta didáctica que puede ser empleada para entender fundamentos de modelamiento de software, por lo cual es importante la asistencia de estudiantes de diversos dominios del conocimiento.

6.9. Ponencias en Congresos nacionales

Título:	"GSC Mapper Mapeador de grafos de Sowa a diagramas de clases" (Jiménez *et al.*, 2005).
Autores:	Sergio Andrés Jiménez M., Giovany Villegas Gómez, Aldrin Fredy Jaramillo F., Carlos Mario Zapata J., Fernando Arango I.
Evento:	Encuentro de Investigación sobre Tecnologías de Información Aplicadas a la Solución de Problemas EITI2005.
Lugar:	Medellín, Colombia.
Fecha:	Octubre de 2005.

Resumen:

En el proceso de desarrollo de software, el reconocimiento automático de operaciones de clases a partir de los requerimientos del sistema, es un tema de investigación complejo que

se ha venido desarrollando por varios autores en diferentes proyectos de investigación y aún permanece abierto. En este artículo se presenta una propuesta a partir de grafos conceptuales de Sowa, que permite obtener lo que sería una primera versión de un diagrama de clases. Además, presenta los resultados obtenidos, siguiendo un caso de estudio específico, en la tesis de grado: ”GSC Mapper - Mapeador de grafos de Sowa a diagramas de clases” adelantada en la Universidad de Antioquia.

Palabras clave: Ingeniería de requisitos, reconocimiento automático de operaciones, diagrama de clases, Procesamiento de Lenguaje Natural (PLN), grafos de Sowa, generación automática de diagramas UML.

Título:	“UNC–Diagrammer: A CASE Tool for automatic Transformation of Pre-conceptual Schemas into UML Diagrams” (Zapata *et al.*, 2007).
Autores:	Carlos Mario Zapata J., Alexander Gelbukh, Fernando Arango I.
Evento:	II Congreso Colombiano de Computación.
Lugar:	Bogotá, Colombia.
Fecha:	Abril de 2007.

Resumen:

.

En el proceso de desarrollo del software las herramientas CASE pueden suministrar ayuda a los analistas en el trazado de diferentes tipos de diagramas, en especial los de UML. Sin embargo, las herramientas CASE tradicionales no pueden ayudarle a los analistas a comprender el discurso de los interesados, lo cual constituye un proceso previo en la construcción de los diagramas. Como consecuencia, el Procesamiento del Lenguaje Natural ha propuesto un nuevo tipo de herramientas CASE, que incluyen interpretación del lenguaje natural y generación automática de diagramas UML. En este artículo se propone una herramienta CASE para la interpretación de discursos, denominada UNC–Diagramador, que usa los denominados Esquemas Preconceptuales para la transformación automática del discurso a tres diagramas de UML 2.0. Finalmente, se muestra el uso de UNC–Diagramador por medio de un ejemplo.

PALABRAS CLAVE: CASE tools; Pre-conceptual Schemas, UML.

Título:	"Definición de un Esquema Preconceptual para la Obtención automática de Esquemas Conceptuales de UML" (Zapata y Arango, 2007b).
Autores:	Carlos Mario Zapata J., Fernando Arango I.
Evento:	II Encuentro de la Investigación en Postgrados
Lugar:	Bogotá, Colombia.
Fecha:	Marzo de 2007.

Resumen:

Las herramientas CASE convencionales permiten dibujar y modificar modelos, que posibilitan la solución a un problema específico mediante una aplicación informática. Sin embargo, estas herramientas no suministran ayudas concretas en el proceso de idear los modelos, es decir, la estructura lógica de las ideas plasmadas en la sintaxis de los diferentes diagramas que representan. A este respecto, actualmente se vienen adelantando varios proyectos que buscan la obtención automática de esquemas conceptuales a partir de lenguajes controlados o gráficos. Si bien se han logrado algunos resultados importantes en esta tendencia, todavía existen problemas: la gama de diagramas con los que se trabaja aún es reducida y los diagramas obtenidos suelen presentar aún problemas de consistencia. En este artículo se definen los Esquemas Preconceptuales, que son grafos unificadores de las características estructurales y dinámicas de los esquemas conceptuales, y que por ello permiten la generación de diagramas de UML 2.0 a partir de un conjunto de reglas de transformación. Se muestra, también, la manera de obtener los Esquemas Preconceptuales desde un lenguaje controlado (denominado UN-Lencep). La elaboración y conversión de los Esquemas Preconceptuales en tres diagramas de UML 2.0 se ejemplifican en la herramienta CASE UNC-Diagramador, actualmente en desarrollo en la Escuela de Sistemas de la Universidad Nacional de Colombia.

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

7. CONCLUSIONES Y TRABAJO FUTURO

El proceso de elicitación de requisitos del software presenta problemas de comunicación entre los interesados (expertos en el dominio, pero que se suelen expresar en lenguaje natural) y los analistas (expertos en modelamiento, y por ello se expresan en lenguajes técnicos). Por esta razón, los analistas tienen que elaborar los modelos que representan el problema y su solución de manera aislada, y sin la posibilidad de que el interesado realice una validación de dichos modelos. Como una forma de solución, en esta Tesis se ha propuesto, diseñado e implementado un entorno que incluye los siguientes aportes:

- La definición de UN-Lencep, un lenguaje controlado en su estructura (es decir, el tipo de frases que se pueden utilizar en su sintaxis), pero que se puede utilizar en cualquier dominio del conocimiento.
- La definición de los Esquemas Preconceptuales, una representación gráfica del dominio planteado en UN-Lencep, y de los cuales se pueden extraer tres tipos de diagramas correspondientes a UML 2.0.
- Un conjunto de reglas de transformación de los discursos en UN-Lencep a Esquemas Preconceptuales y de allí a tres tipos de diagramas de UML 2.0 (clases, comunicación y máquina de estados).
- El prototipo de una herramienta CASE, denominada UNC-Diagramador, que emplea el entorno definido para generar los diagramas UML a partir de un discurso en UN-Lencep. Este prototipo se elaboró tomando en consideración las capacidades gráficas del Microsoft Visio®, para evitar la construcción desde cero de un editor gráfico que hubiera dificultado la realización del proyecto; además, con las posibilidades del conectividad a través del lenguaje XMI en que se pueden almacenar los diagramas, fue posible realizar la integración de este entorno gráfico con .NET mediante el lenguaje C#.

Los resultados anotados buscan la reducción en el tiempo de elaboración de los diferentes diagramas, que es uno de los factores que dificultan el desarrollo de productos de software.

Además, desde el punto de vista de los analistas que utilicen este entorno, se pretende el mejoramiento de la consistencia entre los diferentes diagramas, que en la actualidad se debe realizar manualmente por parte de los analistas, aunque con algo de ayuda por parte de las herramientas CASE de la actualidad; en UNC-Diagramador la consistencia entre los diferentes diagramas está garantizada, por provenir de un mismo discurso en UN-Lencep, y se realiza de manera automática sin que el analista tenga que preocuparse por ello, dado que las reglas de transformación se han programado en dicha herramienta.

Un tercer propósito que se busca con este entorno es la creación de un lenguaje que permita la comunicación entre analistas e interesados, de forma tal que se puedan validar las necesidades y expectativas de los interesados desde el principio, sin tener que esperar a que el ciclo de vida del software avance hasta el trazado de diagramas y artefactos que tengan mejores posibilidades de comprensión por parte del interesado. A este respecto, se espera que el discurso en UN-Lencep sea creado de manera conjunta entre el interesado y el analista, aportando el primero su conocimiento del mundo y el segundo su conocimiento de UML traducido al UN-Lencep, para que los discursos que se generen tengan validez desde el punto de vista del dominio y de los modelos que de allí se deriven; se debe aclarar que no se pretende sustituir el trabajo del analista en el ciclo de vida del software, sino, por el contrario, hacer mucho más especializada su función, para ocuparse de labores que sean más acordes con su papel de experto en modelamiento, sin malgastar el tiempo del proyecto en actividades repetitivas que puedan ser elaboradas de manera automática por la máquina.

La principal dificultad por la que tuvo que atravesar esta Tesis fue la gran complejidad que posee el lenguaje natural, especialmente en el idioma español. El UN-Lencep, que hace parte de la solución planteada, es un subconjunto del lenguaje natural, que es entendible por

parte de interesados sin ningún tipo de entrenamiento en modelamiento, pero que presenta limitaciones de tipo estructural que restringen la utilización de las frases. Además, la expresión de ciertos elementos del mundo se dificulta cuando se usa el UN-Lencep, debido a que no maneja algunos tipos de palabras de común ocurrencia, como es el caso de los adjetivos y los adverbios.

Algunas de las propuestas de Trabajo Futuro se basan en estas limitaciones, y se listan a continuación:

- Complementación del UN-Lencep para incluir adjetivos, adverbios y determinantes. En especial, los determinantes podrían suministrar información valiosa para agregar cardinalidades a los diagramas de clases, un elemento importante de dichos diagramas y para el que en la actualidad no existen reglas de conversión. Los adjetivos, a su vez, podrían entregar algunas pistas para la identificación de otros atributos en los diagramas de clases, que se pueden también reflejar en otros elementos como parámetros de los mensajes del diagrama de comunicación o complementos a los estados del diagrama de máquina de estados. Estas modificaciones deberían también ser representadas en los Esquemas Preconceptuales.
- Exploración en UN-Lencep de incorporación de ciertas formas de ambigüedad y su resolución. Por ejemplo, el uso de anáforas podría darle mayor legibilidad al discurso, haciendo más fácil la transición de los analistas hacia el uso de esta herramienta.
- Generación, a partir de los Esquemas Preconceptuales, de reglas para otros diagramas de UML y la complementación de la sintaxis de los existentes. Se han mencionado la cardinalidades como elementos necesarios para las etapas posteriores de refinamiento del diagrama de clases, pero también sería necesario incorporar los nombres de las relaciones, los roles de las relaciones de asociación y algunos estereotipos que contribuyan a clarificar los elementos del diagrama. Además, en los diagramas de máquina de estados es necesario considerar las acciones que tienen lugar al entrar (“on

entry"), al salir ("on exit") o durante ("do") la permanencia de un objeto en un estado. Otros diagramas factibles de generar a partir de los discursos en UN-Lencep podrían ser los de casos de uso, secuencias y actividades. Otros diagramas diferentes de UML serían susceptibles de generar; en particular, el diagrama de procesos, que contiene la información combinada de los diagramas estructurales y dinámicos de UML podría ser una opción interesante para la continuación de este trabajo. El diagrama de objetivos, por su importancia en la justificación de la aparición de una pieza de software en función de los objetivos de la organización, podría ser susceptible de generar automáticamente desde el discurso en lenguaje natural, aunque podría ser necesario la adición de nuevos elementos que permitan representar los objetivos a partir del discurso.

- Definición de reglas para el chequeo de la completitud en UNC-Diagramador; para ello, podría valerse de bases de conocimientos que complementaran la labor del analista. En particular, el trabajo iniciado para el diagrama de clases, que tiene como insumo principal para el análisis el denominado "corpus de diagramas" es una fuente interesante de trabajo futuro, porque se puede generalizar hacia el conjunto de diagramas que se generan desde UNC-Diagramador.
- Creación de nuevos juegos para simular el proceso de conversión desde Esquemas Preconceptuales hasta diagramas de UML. La estrategia seguida en uno de los Trabajos Dirigidos de Grado (que se ligaron a esta Tesis) con "El Juego de la Consistencia" para la deducción, por parte de los participantes, de las reglas de transformación entre los Esquemas Preconceptuales y los diferentes diagramas de UML ha demostrado ser exitosa, y suministra elementos para que los futuros Ingenieros de Software tengan precaución en la elaboración de los diagramas y sus interconexiones. También, se pueden crear otros juegos que fomenten el uso de UN-Lencep.
- En algunos de los Trabajos Dirigidos de Grado se trabajó también en la elaboración de herramientas para PLN en español. Este es un aspecto importante que se fomentó durante el desarrollo de esta Tesis y que sería interesante continuar. El procesamiento

de Lenguaje Natural en español requiere gran cantidad de herramientas que realicen análisis morfológico, sintáctico, semántico y pragmático, y que sirva para apoyar las diferentes iniciativas para generar herramientas para el procesamiento automático de textos que asistan a los analistas en su función. Una herramienta que desarrolle las diferentes etapas de este tipo de procesamiento podría dar lugar a nuevas investigaciones en este campo, ligadas con la Ingeniería de Requisitos y la Ingeniería de Software en general, y podría dar lugar a la creación de nuevos grupos de investigación en el país, especialmente en temas que poco se trabajan en la actualidad como la Lingüística Computacional.

- Planteamiento de nuevos proyectos de investigación en torno a temas relacionados con la elicitación de requisitos de software. En particular, el estudio del diálogo que se realiza entre el analista y el interesado, y que es una de las principales fuentes de recolección de información y de refinamiento de la misma, podría ser un tema fundamental para cerrar el ciclo de generación de esquemas conceptuales desde UN-Lencep, puesto que podría suministrar herramientas para obtener el discurso en lenguaje controlado de manera interactiva; a este respecto, ya se han iniciado trabajos como el relacionado con el Experimento Mago de Oz que permitirán el estudio de la interacción analista-interesado, con miras a determinar las principales características de esa interacción y su posterior incorporación en una herramienta automática de elicitación de requisitos. Otro tema que podría ser de interés es el reconocimiento temprano de aspectos a partir del interesado, particularmente trabajando en el área conocida actualmente como Aspect-Oriented Requirements Engineering; este tema adquiere importancia por su relación con los requisitos no funcionales del software, puesto que en esta Tesis se abordó sólo la elicitación de requisitos funcionales ligados con los diferentes puntos de vista de los interesados. Un tercer tema susceptible de propuestas de proyectos de investigación es la optimización de los esquemas conceptuales que se pueden obtener a partir de discursos en UN-Lencep; a este respecto, la forma misma en que se elabora el discurso puede plantear diferentes alternativas de modelamiento que

se podrían estudiar para que el programa pueda, en un futuro, sugerir la mejor alternativa.

- Realización de nuevos experimentos que conduzcan a demostrar las bondades del entorno desarrollado. El campo experimental es necesario para validar la certeza de las teorías aquí planteadas; se podría igualmente determinar si el UNC-Diagramador podría tener buenas posibilidades de apoyar industrialmente la producción de software de calidad. Este tipo de pruebas podría suministrar validez experimental al entorno definido de manera teórica en este proyecto.
- Búsqueda de interoperabilidad con otras herramientas CASE para garantizar transformación a código. Si el proceso de generación de esquemas conceptuales se regula para que adquiera la calidad necesaria, la integración con otras herramientas CASE podría suministrar la posibilidad de una generación completa a código a partir del discurso en UN-Lencep.

REFERENCIAS

AECMA. (1995). AECMA Simplified English: A Guide for the Preparation of Aircraft Maintenance Documentation in the International Aerosace Maintenance Language. AECMA, Brussels.

Albrecht, A., Buchholz, E., Düsterhöft, A. y Thalheim, B. (1998). An Informal and Efficient Approach for Obtaining Semantic Constraints Using Sample Data and Natural Language Processing. En: Lecture Notes In Computer Science, Vol. 1358. Pp. 1–28.

Allen, J. (1995). Natural Language Understanding. The Benjamin/Cummings Publishing Company, Inc., Redwood City.

Blackburn, P. y Bos, J. (2003). Computational Semantics. En: Theoria, Volumen 18, No. 1. Pp. 27-45.

Booch, G., Rumbaugh, J. y Jacobson, I. (1998). The Unified Modeling Language User Guide. Addison Wesley.

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.

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

Burg, J. y Van de Riet, R. (1996). Analyzing informal requirements specifications: a first step towards conceptual modeling. En: van de Riet, R. P., J. F. M. Burg and A. J. van der Vos, eds., Proceedings of the 2nd International Workshop on Applications of Natural Language to Information Systems, Amsterdam. Pp. 15–27.

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.

Cooper, K. (2001). Stimulus Response Requirements Specification Notation: An Empirically Evaluated Requirements Specification Notation. Ph.D. Thesis, University of British Columbia.

Copestake, A. (2002). Implementing Typed Feature Structure Grammars. CSLI Publications.

Copestake, A., Flickinger, D., Pollard, C. y Sag, I. (2001). Minimal Recursion Semantics: An Introduction. En: Language and Computation, Vol. 1, No. 3. Pp. 1-47.

Copestake, A., Flickinger, D., Malouf, R., Riehemann, S. y Sag, I. (1995). Translation using Minimal Recursion Semantics. En: Proceedings of the Sixth International Conference on Theoretical and Methodological Issues in Machine Translation, Leuven.

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

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. 2004.

Díaz, I., Losavio, F., Matteo, A. y Pastor, O. (2004b). A Specification Pattern for Use Cases. En: Information & Management, Vol. 41, No. 8, pp. 961–975.

Dorr, B. J. (2001). Computational Lexicon. Copyright (c) University of Maryland.

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

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.

Fowler, M. (2004). UML Distilled: A brief guide to the Standard Object Modeling Language. Addison-Wesley, tercera edición.

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

Gibbs, W. (1994). Software's chronic crisis. En: Scientific American. Pp. 86-101.

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

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.

Haumer, P., Jarke, M., Pohl, K. y Weidenhaupt, K. (2000). Improving reviews of conceptual models by extended traceability to captured system usage. En: Interacting with Computers 13, 77-95.

Hausser, R. (1999). Foundations of Computational Linguistics. En: Human-Computer communication in Natural Language, Springer-Verlag, Berlin.

Heidegger. M. (1976). Protokoll zu einem Seminar über den Vortrag "Zeit und Sein". En: Zur Sache des Denkens, Tübingen, Germany.

Hutchins, J. (2003). Machine Translation: General Overview. En: The Oxford Handbook of Computational Linguistics, editado por Ruslan Mitkov, Oxford University Press, New York. Pp. 501-511.

Jaramillo, A. F., Zapata, C. M. y Arango, F. (2005). Análisis de un Caso de Estudio en KCPM para la Generación de diagramas de Clases. En: Avances en Sistemas e Informática, Vol. 2, No. 1, pp. 27-39.

Jaramillo, A. F., Zapata, C. M., Arango, F. (2005b). Una propuesta para el Reconocimiento Semiautomático de Operaciones utilizando un enfoque lingüístico. En: Revista Ingeniería Universidad de Antioquia, No. 34, pp. 42-51.

Jiménez, S., Villegas, G., Jaramillo, A. F., Zapata, C. M. y Arango, F. (2005). GSC Mapper Mapeador de grafos de Sowa a diagramas de clases. En: Memorias del Encuentro de Investigación sobre Tecnologías de Información Aplicadas a la Solución de Problemas EITI2005, pp. 53-56.

Juristo, N., Morant, J. y Moreno, A. (1999). A formal approach for generating oo specifications from natural language. En: Journal of Systems and Software, Vol. 48, No. 2. Pp. 139–153.

Juristo, N., Moreno, A. (2000) Introductory paper: Reflections on Conceptual Modelling. Data & Knowledge Engineering 33(2). Pp. 103-117.

KIF. (2006). Knowledge Interchange Format. Draft proposed American National Standard (dpANS) NCITS.T2/98-004. Disponible en línea en http://logic.stanford.edu/kif/dpans.html [Citado 19 de Noviembre de 2006]

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. 5p.

Lappin, S. Semantics. (2003). En: The Oxford Handbook of Computational Linguistics, editado por Ruslan Mitkov, Oxford University Press, New York, Pp. 91-111.

Leffingwell, D. y Widrig, D. (1999). Managing Software Requirements. Addison-Wesley.

Leite, J. (1987). A survey on requirements analysis, Advanced Software Engineering Project. En: Technical Report RTP-071, Department of Information and Computer Science, University of California at Irvine.

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.

Mel'čuk, I. y Žolkovskij, A. (1970). Towards a functioning 'meaning-text' model of language. En: Linguistics, No. 57, pp. 10–47.

Mel'čuk, I. y Polguère, A. (1987). A formal lexicon in the meaning-text theory (or how to do lexica with words). En: Computational Linguistics, No. 13, Vol 3-4, pp. 261–275.

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.

Mitkov, R. (2003). The Oxford Handbook of Computational Linguistics. Oxford University Press, New York.

Moreno, A., Juristo, N. y Van de Riet, R. (2000). Formal justification in object-oriented modelling: A linguistic approach. En: Data & Knowledge Engineering, Vol. 33, pp. 2 –47.

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

Navarro, J., Orozco, R., Jaramillo, A. F. y Zapata, C. M. (2005). Analizador de Completitud de Requisitos escritos en Español restringido. En: Avances en Sistemas e Informática, Vol. 2, No. 1, pp. 19-25.

Omar, N., Hanna, P. y McKevitt, P. (2004). Heuristics-based entity-relationship modeling through natural language processing. En: Proceedings of the 15th Irish Conference on Artificial Intelligence and Cognitive Science (AICS-04), Lorraine McGinty and Brian Crean (Eds.), Castlebar, pp. 302–313.

OMG. (2006). UML 2.0 Superestructure Specification. Disponible en línea en http://www.omg.org/uml/ . [Citado: 19 de Noviembre de 2006].

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

Peirce, C. S. (1931-58). Collected Papers of C.S. Peirce. En: Charles Hartshorne and Paul Weiss, ed., vols. 1-6, A. W. Burks, ed., vols. 7-8, Harvard, Cambridge.

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

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

Pollard, C. y Sag, I. (1994). Head-driven phrase structure grammar. CSLI Publications y Chicago University Press, Chicago.

Pressman, R. (2005). Software Engineering: a practitioner's approach, 6th ed. McGraw Hill, New York.

Pylyshyn, Z. (2001). Visual indexes, preconceptual objects, and situated vision. En: Cognition, Vol. 80, pp. 127–158.

Richards, D., Boettger, K. y Aguilera, O. (2002). A Controlled Language to Assist Conversion of Use Case Descriptions into Concept Lattices. En: Proceedings of AI 2002, Canberra, LNCS 2557.

Sommerville, I. (2001). Software engineering (6th. ed). Addison-Wesley Longman, Boston.

Sowa, J. F. (1984). Conceptual Structures: Information Processing in Mind and Machine, Addison-Wesley Publishing Co., Reading, MA.

Vendler Z. (1957). Verbs and Times. En: Linguistics in Philosophy. P97-121. Ithaca. NY. CUP.

Zapata, C. M. y Arango, F. (2005). Los Modelos Verbales en Lenguaje Natural y su utilización en la elaboración de esquemas conceptuales para el desarrollo de software: Una revisión crítica. En: Revista Universidad EAFIT, Vol. 41, No. 137, pp. 77-95.

Zapata, C. M., Jaramillo, A. F. y Arango, F. (2005). Análisis de un Caso de Estudio en KCPM para la Generación de diagramas de Clases. En: Avances en Sistemas e Informática, Vol. 2, No. 1, pp. 27-39.

Zapata, C. M., Gelbukh, A. y Arango, F. (2006a). Pre-conceptual Schemas: a Conceptual-Graph-like Knowledge Representation for Requirements Elicitation. En: Lecture Notes in Computer Sciences, Vol. 4293, pp. 17-27.

Zapata, C. M., Arango, F. y Gelbukh, A. (2006b). Pre-conceptual Schemas: a UML isomorphism for automatically obtaining UML diagrams. En: Research in Computing Sciences, Vol. 19, pp. 3-13.

Zapata, C. M., Gelbukh, A. y Arango, F. (2006c). UN–Lencep: Obtención Automática de Diagramas UML a partir de un Lenguaje Controlado. En: Taller de Tecnologías del Lenguaje Humano del Encuentro Nacional de Computación, México.

Zapata, C. M., Jaramillo, A. F., Arango, F. (2006d). Una propuesta para mejorar la completitud de requisitos utilizando un enfoque lingüístico. En: Ingeniería y Desarrollo, No. 19, pp. 1-16.

Zapata, C. M., Estrada, B. y González, G. (2006e). Reglas de Conversión entre el Diagrama de Clases y los Grafos Conceptuales de Sowa. En: Revista Ingenierías Universidad de Medellín, Vol. 5, No. 9, pp. 111-122.

Zapata, C. M., Jaramillo, A. F. y Arango, F. (2006f). Método Semiautomático para la Identificación de Operaciones a partir de Grafos Conceptuales. En: Memorias del 9° Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes de Software, Buenos Aires, pp. 213-225.

Zapata, C. M. y Carmona, N. (2007). El experimento Mago de Oz y sus aplicaciones: Una mirada retrospectiva. En: Dyna, No. 151, pp. 125-135.

Zapata, C. M. y Arango, F. Un ambiente para la Obtención automática de Diagramas UML a partir de un Lenguaje controlado. En: Dyna, pendiente publicación.

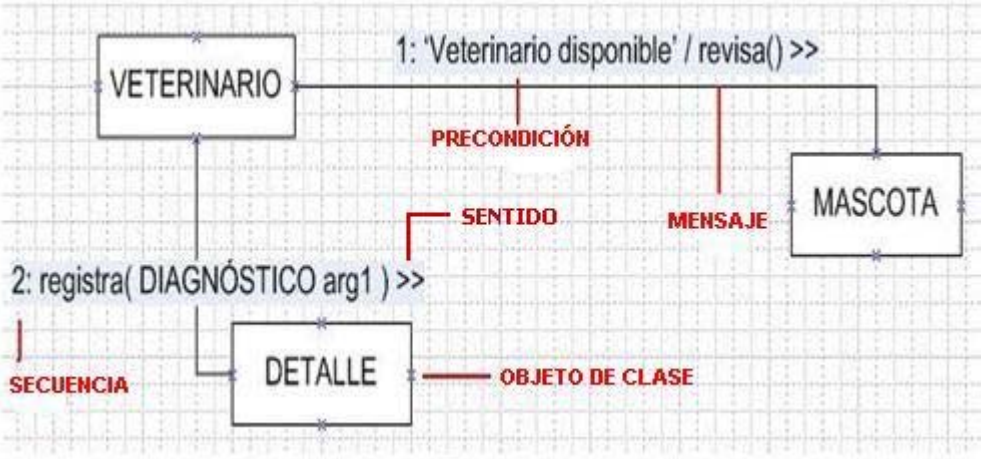
Zowghi, D. y Gervasi, V. (2002). The Three Cs of Requirements: Consistency, Completeness and Correctness. En: Proceedings of 8th International Workshop on Requirements Engineering: Foundation for Software Quality, (REFSQ'02), Essen, pp. 155-164.

ANEXO 1: PLEGABLES PARA LA REALIZACIÓN DEL EXPERIMENTO

A.1.1. Plegable de dos diagramas

Diagrama de Comunicación

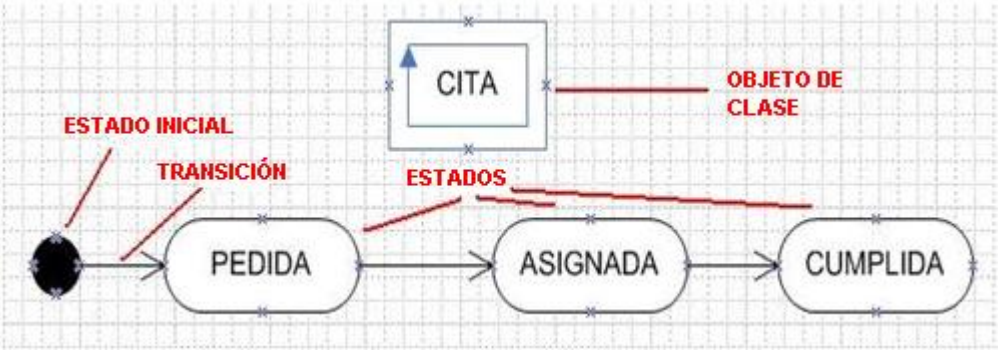
Se usa para indicar el orden en el que interactúan las clases (paso de mensajes), además de las condiciones necesarias para que las interacciones se den.



Objeto de clase: Elementos que se comunican en el diagrama.
Precondición: lo que se debe cumplir para que se de un mensaje.
Mensaje: Acción que le envía un objeto de clase a otro.
Secuencia: Número que identifica el orden en que se dan los mensajes.
Sentido: Define qué objeto emite y cual recibe el mensaje.

Diagrama de Máquina de Estados

Se usa para indicar las propiedades que tienen los objetos (clases), en un instante dado, particularmente después de que estos hayan realizado una acción (operación), se debe hacer una por objeto.



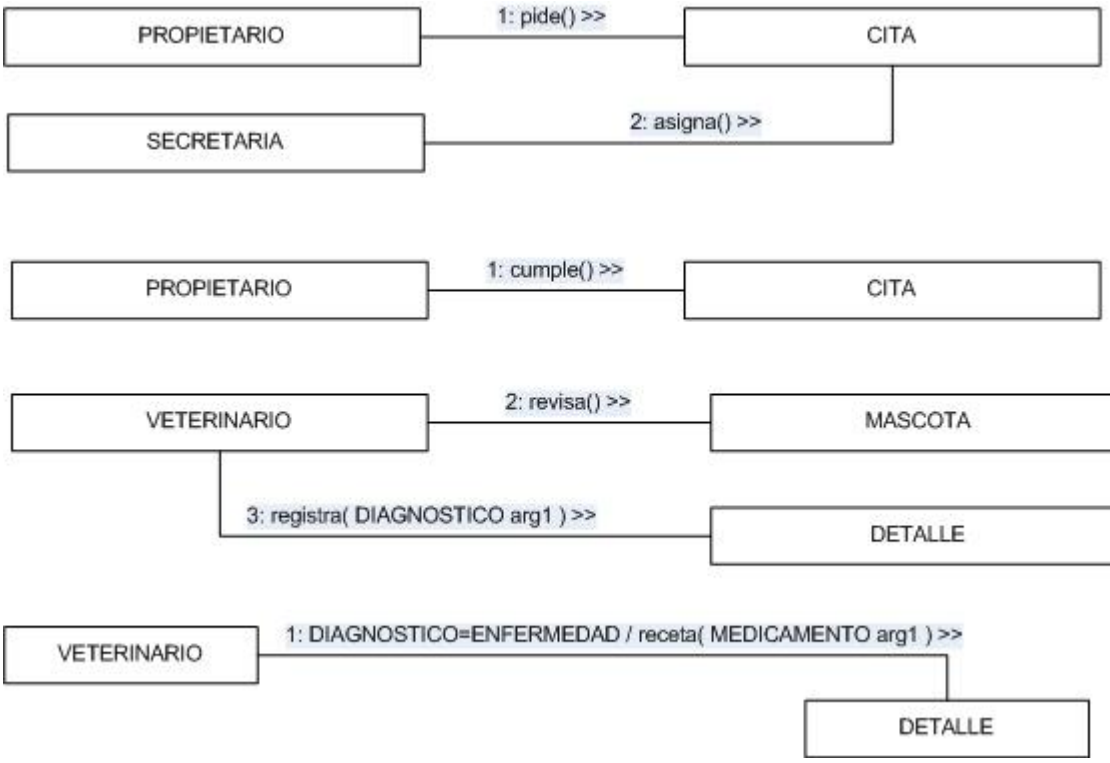
Estado: Cómo se encuentra un objeto en un momento dado; por convención, al principio de cada máquina de estados siempre habrá un estado inicial.

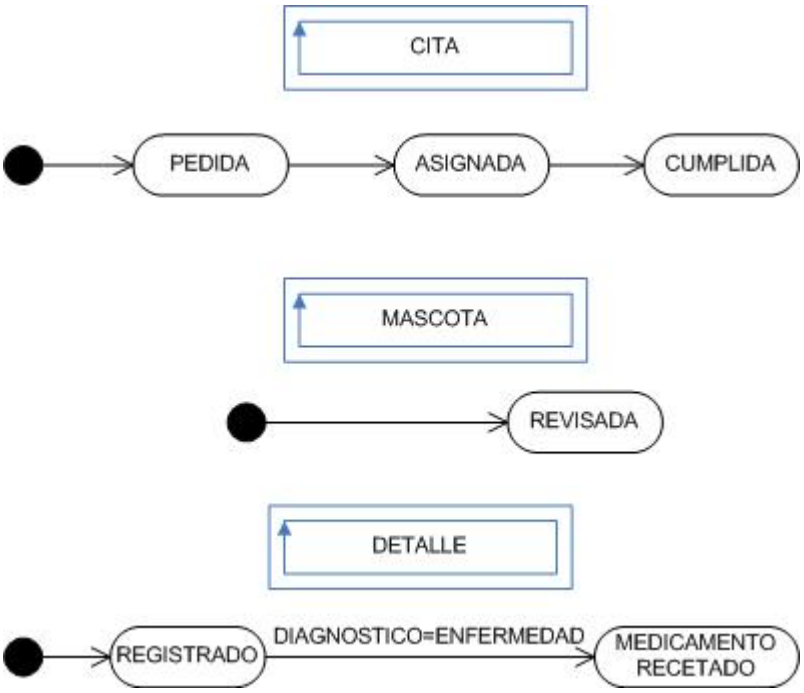
Transición: El paso de un estado a otro; indica a qué estado pasa el objeto después de terminar el actual.

Notas Relevantes

- Las condiciones de guarda presentes en el diagrama de máquina de estados deben también estar presentes en el diagrama de comunicación.
- Todos los mensajes del diagrama de comunicación deben corresponder a estados del diagrama de máquina de estados. En este caso, el nombre del diagrama de máquina de estados debe corresponder con el nombre del objeto de llegada. Igualmente, todos los estados deben corresponder a mensajes del diagrama de comunicación, con la misma limitación en los nombres.
- Los cambios de estado suelen determinarse por el orden en que se envían los mensajes en el diagrama de comunicación.

Ejemplo de UML

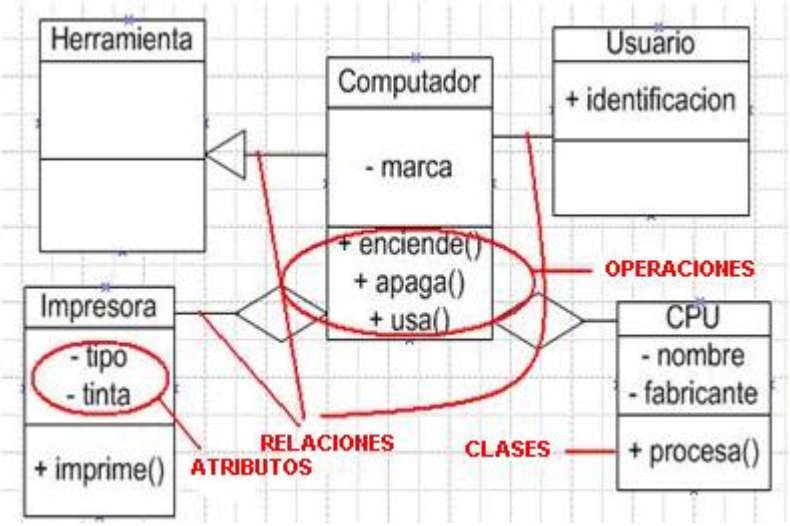

PROPIETARIO
1: pide() >>
CITA
SECRETARIA
2: asigna() >>
PROPIETARIO
1: cumple() >>
CITA
VETERINARIO
2: revisa() >>
MASCOTA
3: registra(DIAGNOSTICO arg1) >>
DETALLE
VETERINARIO
1: DIAGNOSTICO=ENFERMEDAD / receta(MEDICAMENTO arg1) >>
DETALLE


CITA
PEDIDA
ASIGNADA
CUMPLIDA
MASCOTA
REVISADA
DETALLE
REGISTRADO
DIAGNOSTICO=ENFERMEDAD
MEDICAMENTO RECETADO

A.1.2. Plegable de tres diagramas

Diagrama de Clases

Se usa para indicar cuáles son los elementos importantes del dominio que se está modelando.



Clases: Conjuntos de objetos que existen en el mundo y son importantes en el problema que se está modelando. Sus nombres son sustantivos. Una clase aislada de las demás no puede existir en el diagrama.

Atributos: Propiedades que describen una clase. Sus nombre son sustantivos.

Operaciones: Lo que se puede hacer con una clase. Sus nombres son verbos. Pueden recibir argumentos, que son los atributos de la clase. Una clase por lo general tiene atributos y operaciones. Si aparece una clase sin ellos, se debería preguntar a la persona que está describiendo el mundo.

Relaciones: Definen cómo se asocian las clases. Hay tres tipos básicos. Agregación, cuando una clase hace parte de otra (Impresora -> Computador); Generalización, cuando una clase es una especialización de otra (Herramienta -> Computador, Martillo); Asociación, es una relación que indica que dos clases se relacionan pero no por agregación ni generalización.

Suele usarse para indicar que una clase es el encargado de que la otra realice una acción, en el ejemplo, el Usuario es responsable de que la Computadora se encienda, use, o apague.

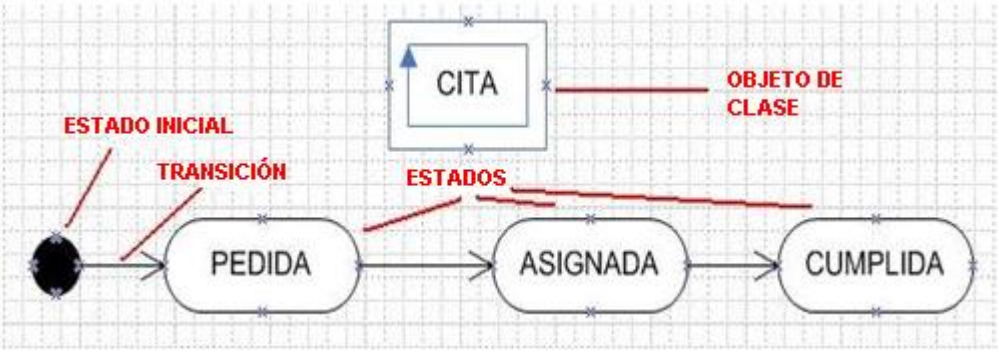
Diagrama de Comunicación

Se usa para indicar el orden en el que interactúan las clases (paso de mensajes), además de las condiciones necesarias para que las interacciones se den.

Objeto de clase: Elementos que se comunican en el diagrama.
Precondición: lo que se debe cumplir para que se de un mensaje.
Mensaje: Acción que le envía un objeto de clase a otro.
Secuencia: Número que identifica el orden en que se dan los mensajes.
Sentido: Define qué objeto emite y cual recibe el mensaje.

Diagrama de Máquina de Estados

Se usa para indicar las propiedades que tienen los objetos (clases), en un instante dado, particularmente después de que estos hayan realizado una acción (operación), se debe hacer una por objeto.



Estado: Cómo se encuentra un objeto en un momento dado; por convención, al principio de cada máquina de estados siempre habrá un estado inicial.
Transición: El paso de un estado a otro; indica a qué estado pasa el objeto después de terminar el actual.

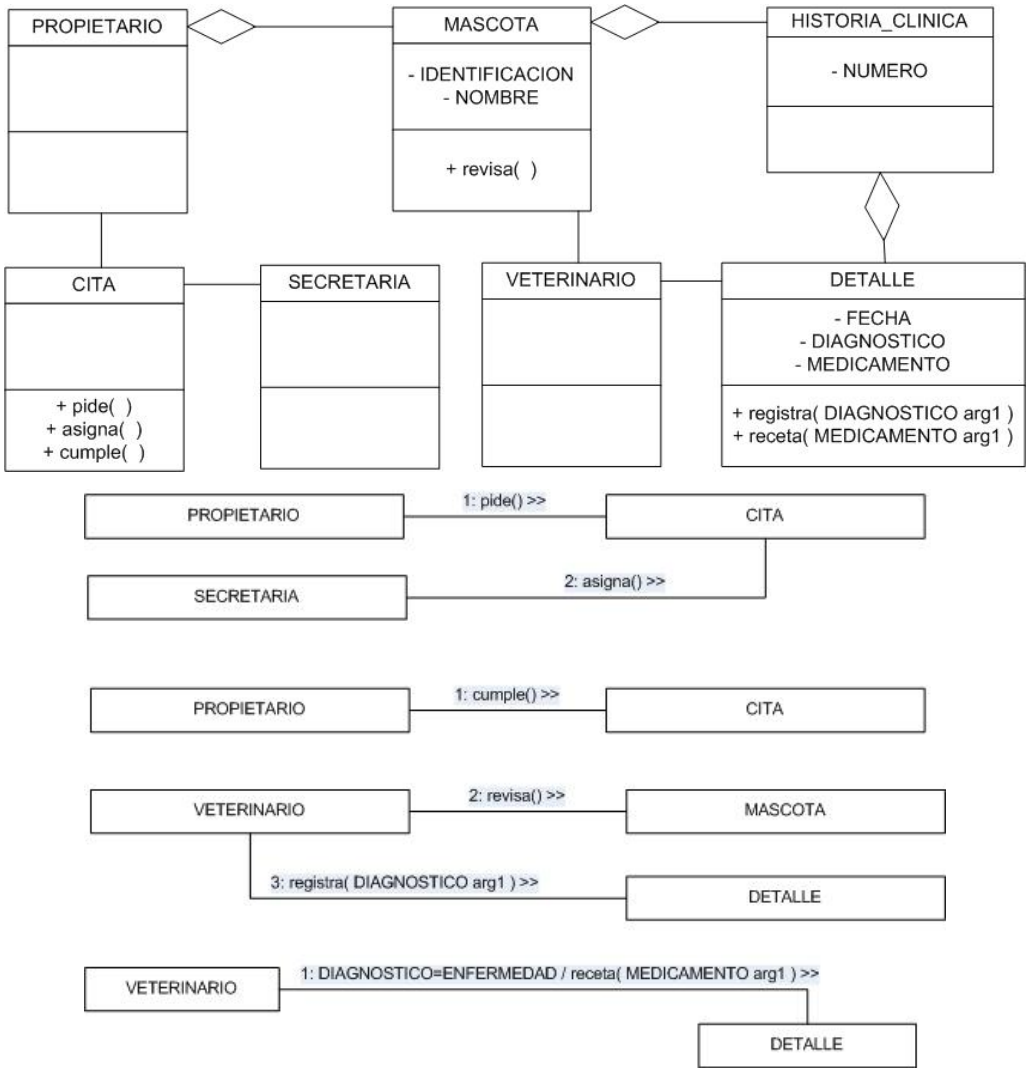
Notas Relevantes

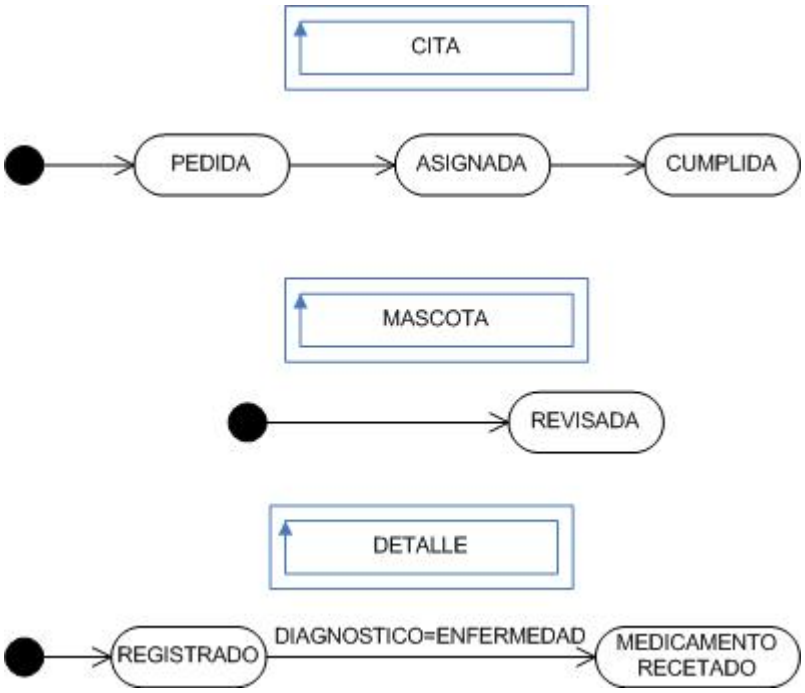
- Todo mensaje del diagrama de comunicación debe estar presente en el diagrama de clases como una operación de la clase del objeto que recibe el mensaje.
- Toda clase de objeto del diagrama de comunicación debe ser una clase del diagrama de clases.
- Para que una clase le envíe un mensaje a otra en el diagrama de comunicación, debe existir una asociación entre ellas en el diagrama de clases.
- Cada diagrama de máquina de estados debe corresponder a una clase del diagrama de clases.
- Los estados del diagrama de máquina de estados deben corresponder a operaciones del diagrama de clases (en participio pasado).
- Las condiciones de guarda presentes en el diagrama de máquina de estados deben también estar presentes en el diagrama de comunicación.
- Todos los mensajes del diagrama de comunicación deben corresponder a estados del diagrama de máquina de estados. En este caso, el nombre del diagrama de máquina de

estados debe corresponder con el nombre del objeto de llegada. Igualmente, todos los estados deben corresponder a mensajes del diagrama de comunicación, con la misma limitación en los nombres.

- Los cambios de estado suelen determinarse por el orden en que se envían los mensajes en el diagrama de comunicación.
- Una clase sin operaciones no debe aparecer en un diagrama de comunicación.
- Una clase sin operaciones no debe tener una máquina de estados asociada.

Ejemplo de UML




CITA
PEDIDA
ASIGNADA
CUMPLIDA
MASCOTA
REVISADA
DETALLE
REGISTRADO
DIAGNOSTICO=ENFERMEDAD
MEDICAMENTO RECETADO

ANEXO 2: ENUNCIADO Y DIAGRAMAS CORRESPONDIENTES AL EXPERIMENTO

A.2.1. Enunciado

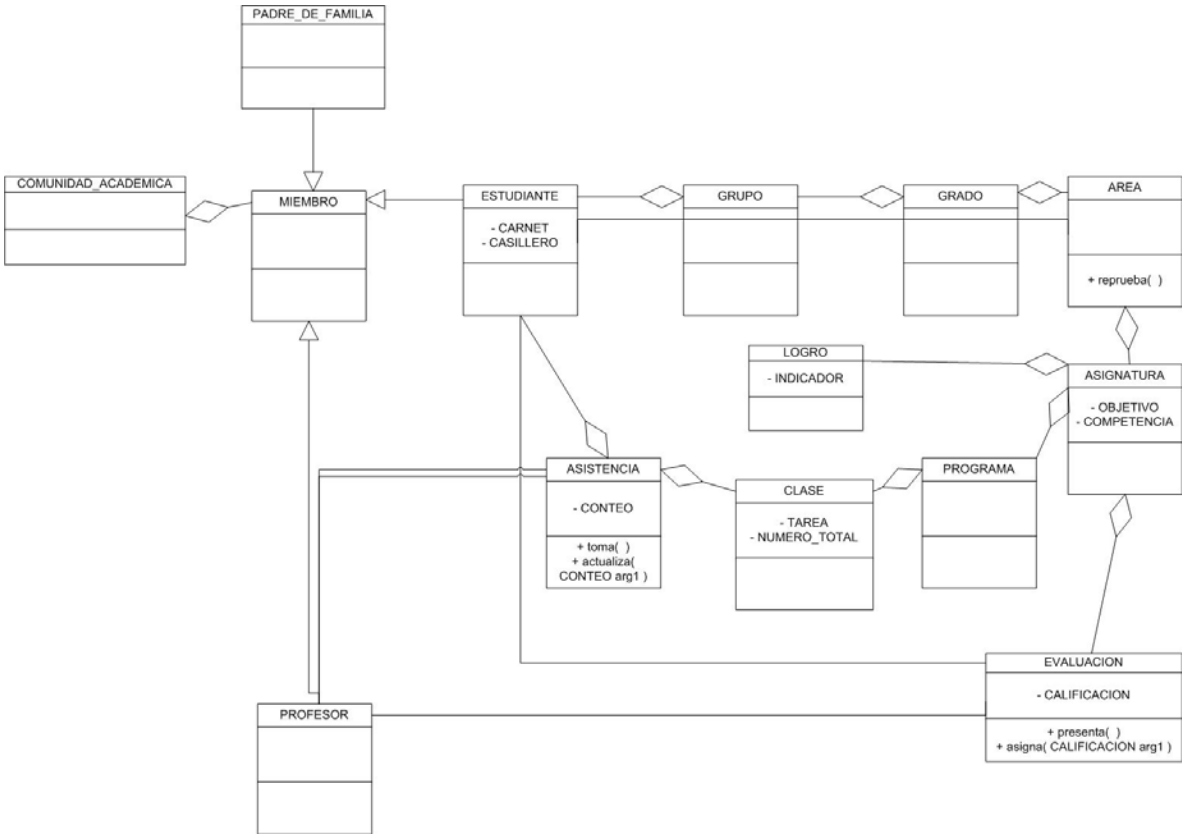
Los miembros de la comunidad académica son los estudiantes, los padres de familia y los profesores. Cada estudiante dispone de un carnet y de un casillero. Los estudiantes conforman los grupos, que a su vez conforman los grados. Un grado, a su vez, está conformado por áreas, que contienen asignaturas. Las asignaturas comprenden objetivos, programas, competencias y logros; los logros tienen asociados indicadores. Además, las asignaturas se componen de evaluaciones, que poseen calificaciones. Un conjunto de clases conforma un programa y posee un conjunto de tareas. El profesor toma la asistencia a las clases por parte de los estudiantes, con lo cual actualiza el conteo; cabe anotar que el conteo pertenece a la asistencia. A este respecto, cada programa posee un número total de clases; si el conteo de asistencias es inferior al 75% del total de clases, entonces el estudiante reprueba el área. Cuando el estudiante presenta una evaluación, el profesor le asigna una calificación a esa evaluación.

A.2.2. Versión en UN-Lencep del enunciado

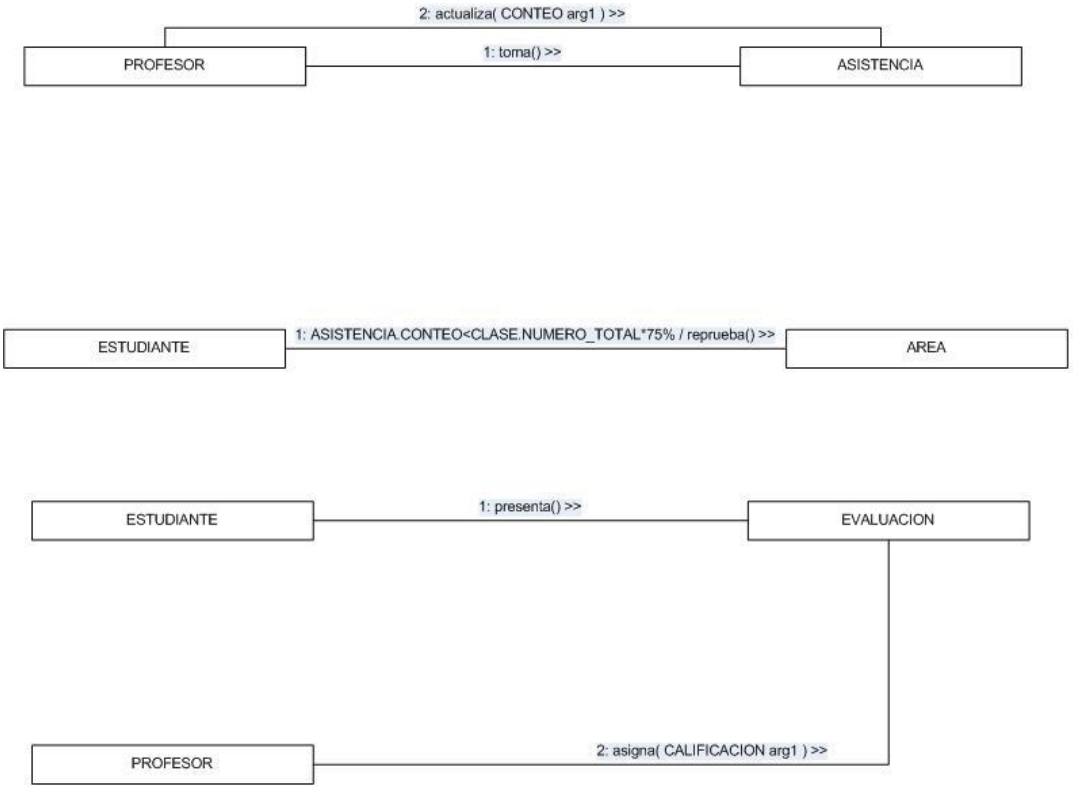
ST comunidad_academica tiene miembro
ST estudiante es miembro
ST profesor es miembro
ST padre_de_familia es miembro
ST estudiante tiene carnet casillero
ST grupo tiene estudiante
ST grado tiene grupo
ST grado tiene area
ST area tiene asignatura
ST asignatura tiene objetivo programa competencia logro evaluacion
ST evaluacion tiene calificacion
ST logro tiene indicador

ST programa tiene clase
ST clase tiene tarea
IM cuando profesor toma asistencia, profesor actualiza conteo
ST asistencia tiene clase
ST asistencia tiene estudiante
ST asistencia tiene conteo
ST clase tiene numero_total
*CO si asistencia.conteo<clases.numero_total*75/100, entonces estudiante reprueba area*
IM cuando estudiante presenta evaluacion, profesor asigna calificacion

A.2.3. Diagrama de clases



A.2.4. Diagramas de comunicación



A.2.5. Diagramas de máquina de estados