

Dyna  
Universidad Nacional de Colombia  
dyna@unalmed.edu.co  
ISSN (Versión impresa): 0012-7353  
COLOMBIA

2007

Carlos M. Zapata J. / Paula A. Tamayo O. / Fernando Arango I.  
CONVERSIÓN DE ESQUEMAS PRECONCEPTUALES A DIAGRAMA DE CASOS  
DE USO EMPLEANDO ATOM3  
*Dyna*, noviembre, año/vol. 74, número 153  
Universidad Nacional de Colombia  
Medellín, Colombia  
pp. 237-251

Red de Revistas Científicas de América Latina y el Caribe, España y Portugal

Universidad Autónoma del Estado de México

<http://redalyc.uaemex.mx>



# CONVERSIÓN DE ESQUEMAS PRECONCEPTUALES A DIAGRAMA DE CASOS DE USO EMPLEANDO AToM<sup>3</sup>

## CONVERSION OF PRE-CONCEPTUAL SCHEMA INTO USE CASE DIAGRAMS BY USING AToM<sup>3</sup>

CARLOS M. ZAPATA J

*Grupo de Investigación en Ingeniería de Software. Universidad Nacional de Colombia. cmzapata@unal.edu.co*

PAULA A. TAMAYO O

*Grupo de Investigación en Ingeniería de Software. Escuela de Sistemas. Universidad Nacional de Colombia*

FERNANDO ARANGO I

*Grupo de Investigación en Ingeniería de Software. Escuela de Sistemas. Universidad Nacional de Colombia*

Recibido para revisar febrero 26 de 2007, aceptado julio 27 de 2007, versión final agosto 08 de 2007

**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 semiautomática del diagrama de casos de uso desde descripciones en lenguajes naturales o controlados. Sin embargo, estos esfuerzos no han sido suficientes porque algunos parten de un lenguaje controlado 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 controlado del dominio del problema: los denominados esquemas preconceptuales. Además, se realiza la implementación de estas heurísticas en la herramienta metaCASE AToM<sup>3</sup> y se ejemplifica con un caso de estudio.

**PALABRAS CLAVE:** Metamodelamiento, diagrama de casos de uso, esquemas preconceptuales..

**ABSTRACT:** Use case diagram describes user-software interactions. Work in automated or semi-automated generation of use case diagram from natural or controlled languages have been done. However, this work has not been enough, due to the fact that some of it uses a solution-driven controlled language, and the solution does not exist in the first stages of software development life cycle; other works require higher analyst intervention in order to generate the use case diagram, and this kind of intervention is not suited for automating this process; finally, special relationships (<<include>>, <<extends>>, and <<inheritance>>) are not still identified. We define, in this paper, a heuristic-rule-based method for identifying actors, use cases, and special relationships of use case diagram. We use as a source place a representation in a problem-domain controlled language: the so-called pre-conceptual schemas. Furthermore, we implement these heuristic rules in the AToM<sup>3</sup> metaCASE tool, and we exemplify them in a case study.

**KEYWORDS:** Metamodeling, use case diagram, pre-conceptual schemas.

### 1. INTRODUCCIÓN

El diagrama de casos de uso representa las principales interacciones entre los usuarios y el sistema, mostrando las distintas operaciones y

cómo se relacionan con su entorno (OMG, 2002). Así, este diagrama se encarga de mostrar la funcionalidad futura de una aplicación de software.

Algunos investigadores como Ben Achour (1998, 1999), Pastor *et al.* (2003, 2004), Shishkov *et al.* (2002), Liu y Dong (2003) y Liu *et al.* (2004), han abordado el tema de la obtención automática o semiautomática del diagrama de casos de uso; este tema tiene gran importancia en la Ingeniería de Requisitos, puesto que, si se puede acortar el tiempo en la elaboración de estos diagramas, la aplicación de software podría conceptualizarse en un tiempo menor. En el trabajo de Ben Achour (1998, 1999) el punto de partida es lenguaje natural y en el resto de los trabajos, es lenguaje controlado, pero todos buscan garantizar que los requisitos del interesado se reflejan en el sistema obtenido. Sin embargo, aún subsisten problemas, tales como:

- El lenguaje de partida de algunos trabajos es un lenguaje controlado que debe mencionar la funcionalidad de la aplicación de software, la cual no existe en las etapas iniciales del ciclo de vida del software.
- Existe un obstáculo para la automatización en el hecho de que el analista completa el diagrama de manera subjetiva en la mayoría de los trabajos.
- No se identifican todos los elementos que componen el diagrama de casos de uso, en especial, las relaciones `<<include>>`, `<<extends>>` e `<<inheritance>>`.
- La mayoría de estos trabajos son realizados para el idioma Inglés, el cual presenta grandes diferencias con el Español.

En este artículo se define un método basado en reglas heurísticas, que permite obtener de forma automática el diagrama de casos de uso a partir de los denominados esquemas preconceptuales (Zapata *et al.*, 2006a y 2006c). Estos esquemas constituyen una representación en lenguaje controlado del dominio del problema, debido a que representan gráficamente los diferentes elementos del discurso del interesado, pero no necesariamente se refieren a la aplicación de software que soluciona los problemas del interesado, sino más bien se puede referir a los conceptos del dominio. Además, se realiza la implementación de estas reglas en la herramienta metaCASE AToM<sup>3</sup>® y se ejemplifica con un caso de estudio.

Este artículo está organizado de la siguiente manera: en la Sección 2 se realiza una presentación de los conceptos fundamentales relacionados con el diagrama de casos de uso, el metamodelamiento, los esquemas preconceptuales y la herramienta metaCASE AToM<sup>3</sup>®; en la Sección 3 se hace un análisis crítico de los trabajos en obtención del diagrama de casos de uso a partir de diferentes representaciones en lenguaje natural o controlado; las reglas de conversión entre el esquema preconceptual y el diagrama de casos de uso son presentadas en la Sección 4, así como también se plantea un caso de estudio para examinar los resultados obtenidos. Por último, en las Secciones 5 y 6 se presentan las conclusiones y el trabajo futuro respectivamente.

## 2. MARCO TEÓRICO

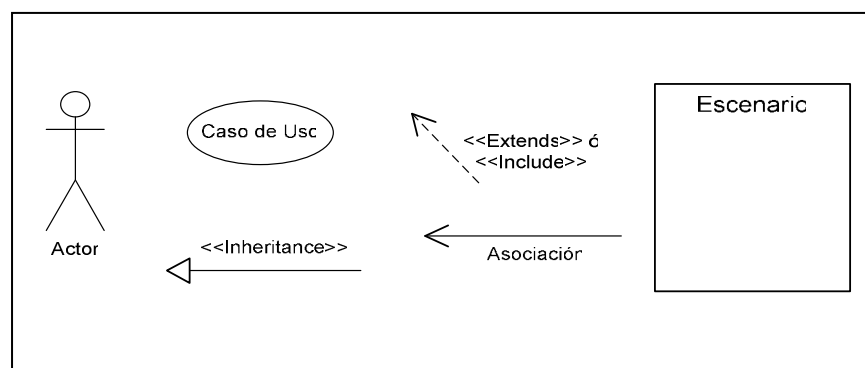
### 2.1 Diagrama de Casos de Uso

En la especificación de la superestructura del Unified Modeling Language UML (OMG, 2002) el diagrama de casos de uso se define como el “diagrama que muestra las relaciones entre los actores y el sujeto (sistema) y los casos de uso. El modelo de casos de uso describe los requisitos funcionales del sistema en términos de las secuencias de acciones, incluyendo las variantes que el sistema u otra entidad puede realizar interactuando con actores del sistema”. En OMG (2002) y Fowler (2004) se presentan los siguientes elementos de la especificación del diagrama de casos de uso:

- Casos de uso: son las especificaciones de un conjunto de acciones realizadas por el actor sobre el sistema; es decir, son los pasos que describen de principio a fin un proceso.
- Actores: son los roles que los usuarios desempeñan respecto del sistema y que emplean los casos de uso. Los actores pueden ser usuarios humanos u otros sistemas, que se comunican con el sistema que se requiere.
- Relaciones: son interacciones que se pueden presentar entre casos de uso, entre actores y casos de uso y entre los actores. Las relaciones pueden ser de cuatro tipos:

- Asociación: se establece entre los actores y casos de uso.
- <<include>>: se presenta cuando una instancia del caso de uso origen incluye también el comportamiento descrito por el caso de uso destino; es decir, un caso de uso incluido describe un objetivo de bajo nivel de un caso de uso base (Cockburn, 2000).
- <<extends>>: ocurre cuando el caso de uso origen extiende el comportamiento del caso de uso destino; en otras palabras, el caso de uso a extender invoca el caso de uso base bajo ciertas condiciones (Cockburn, 2000).
- <<inheritance>>: se presenta cuando un caso de uso origen hereda la especificación del caso de uso destino y posiblemente la modifica y/o amplía. Este tipo de relación también se presenta entre los actores.
- Escenario: es una secuencia específica de acciones que ilustra un comportamiento. Los escenarios se usan para ilustrar una interacción o la ejecución de una instancia de un caso de uso.

La representación gráfica de los elementos del diagrama de casos de uso se puede apreciar en la Figura 1. Las principales ventajas de utilizar el



**Figura 1.** Elementos del diagrama de casos de uso.

**Figure 1.** Elements of the use case diagram.

El metamodelamiento, utilizado en la conversión de modelos, presenta las siguientes ventajas:

- Produce formalismos personalizados creando ambientes propios de metamodelamiento.

diagrama de casos de uso, según Firesmith (2002) son:

- La captura de los requisitos desde el punto de vista del usuario, lo cual permite el correcto desarrollo del sistema.
- La utilización de los casos de uso para elicitar y documentar los requisitos funcionales que se recolectan en la fase de definición del ciclo de vida del software. Estos requisitos también se verifican y validan mediante los casos de uso.
- El manejo de la complejidad en sistemas robustos, donde se descompone el problema en funciones más simples.

## 2.2 Metamodelamiento

Según De Lara y Vangheluwe (2003), el metamodelamiento proporciona los formalismos apropiados para describir los diferentes aspectos o componentes de los modelos que representan sistemas lógicos y físicos. Este proceso se realiza mediante la descripción de las clases, atributos y relaciones que conforman el modelo. Los metamodelos se describen gráficamente mediante metaformalismos (notaciones de modelamiento de alto nivel). El diagrama de clases de UML y el diagrama entidad-relación son algunos de los formalismos que se utilizan.

- Sirve de base para el análisis de modelos, debido a que facilita la documentación y especificación de los mismos.

- Permite diseñar nuevos formalismos mediante la modificación parcial o total del metamodelo.

Las herramientas que se emplean para el metamodelamiento se denominan metaCASE; algunas de las más conocidas son:

- DOME® (Domain Modeling Environment): es una herramienta gratuita creada por Honeywell Technology Center. El formalismo que utiliza es una modificación del diagrama de clases de UML a nivel gráfico, que se especifica empleando código en los lenguajes Alter y Smalltalk (DOME, 2007a y 2007b).
- MetaEdit®: es un metaCASE comercial desarrollado por la compañía MetaCASE Consulting. Su formalismo es el diagrama entidad-relación (Smolander *et al.*, 1991).
- AToM<sup>3</sup>®: es una herramienta gratuita desarrollada por Juan F. de Lara, que permite la creación, edición, transformación, simulación y optimización de metamodelos y modelos. Su formalismo es el diagrama entidad-relación y genera código Python (De Lara y Vangheluwe, 2002).

### 2.3 Esquemas Preconceptuales

Según Zapata *et al.* (2006b), los esquemas preconceptuales son diagramas que permiten la representación de la terminología de un dominio para facilitar su traducción posterior a diferentes esquemas conceptuales. En la Figura 2 se muestran los diferentes elementos de los esquemas preconceptuales, cuya descripción es la siguiente (Zapata *et al.*, 2006a, 2006b y 2006c):

- Conceptos: es un sustantivo o un sintagma nominal del discurso del interesado.
- Relación estructural: es una relación permanente entre los conceptos. Está asociada con los verbos “es” y “tiene”.
- Relación dinámica: está asociada con los verbos de actividad definidos por Vendler (1967).
- Implicación: sirve para unir relaciones dinámicas o para unir condicionales con

relaciones dinámicas, estableciendo entre ellas una relación causa-efecto.

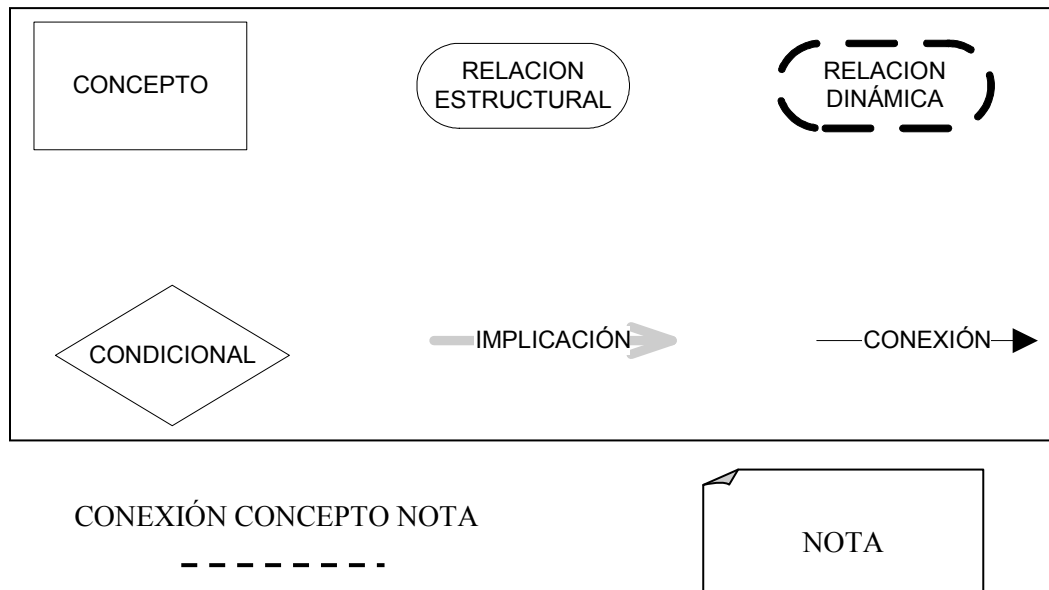
- Condicional: es una relación de causalidad que indica las restricciones que los conceptos deben cumplir.
- Nota: permite especificar los diferentes valores que puedan ser asignados a los sintagmas nominales o conceptos. Este es un elemento nuevo que se viene trabajando en el grupo de Ingeniería de Software para representar las posibles instancias de un determinado concepto del mundo.
- Conexión: permite enlazar los conceptos con las relaciones y las relaciones con los conceptos. Un tipo especial de conexión permite enlazar los conceptos y las notas.

Las principales ventajas de utilizar Esquemas Preconceptuales en la definición de los requisitos son:

- Se obtienen fácilmente desde un lenguaje controlado, denominado UN-Lencep (Zapata *et al.*, 2006b), por lo cual su discurso no presenta ambigüedades.
- Agrupan características de los diferentes tipos de diagramas de UML.

### 2.4 ATOM<sup>3</sup>®

AToM<sup>3</sup>® es una herramienta usada para describir formalismos de cualquier tipo de esquema conceptual; esta herramienta MetaCASE que permite la creación, edición, transformación, simulación y optimización de metamodelos y modelos, y además la generación de código en lenguaje Python (De Lara y Vangheluwe, 2002). Además, AToM<sup>3</sup>® utiliza técnicas de reescritura gráfica y gramática de grafos para realizar las transformaciones entre formalismos.



**Figura 2.** Elementos de los esquemas preconceptuales.  
**Figure 2.** Elements of Pre-conceptual schemas.

En AToM<sup>3</sup>®, los metamodelos son creados y editados en un ambiente que emplea como formalismo gráfico el diagrama entidad-relación (Véase la Figura 3). La herramienta MetaCASE AToM<sup>3</sup> ® tiene la siguiente estructura de modelamiento:

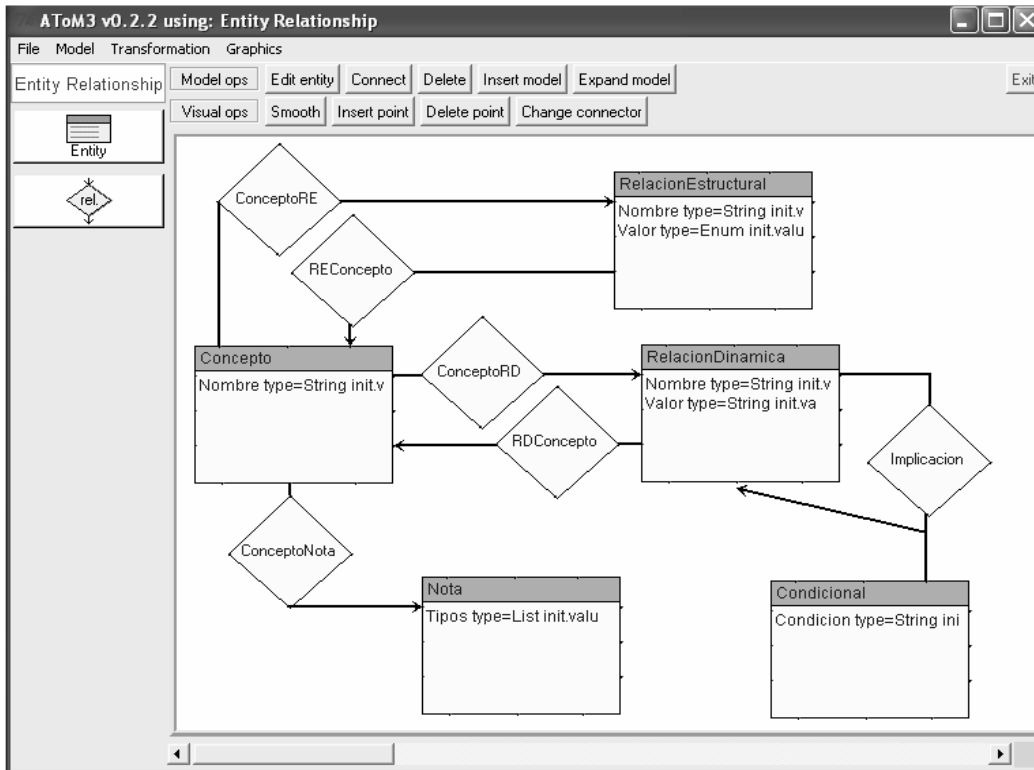
- Metametanivel: metamodelo del diagrama entidad-relación.
- Meta nivel: ambiente de modelamiento con el diagrama entidad-relación.
- Nivel modelo: ambiente de modelamiento con formalismos creados por el usuario.
- Modelo: varias instancias de modelos.

La Gramática de Grafos en AToM<sup>3</sup>® permite expresar restricciones y describir las transformaciones entre grafos de manera gráfica. La Gramática de Grafos está compuesta por reglas que mapean desde el lado izquierdo (LHS) al lado derecho (RHS), condiciones y acciones. En las condiciones, se define cuándo una regla puede ser aplicada; en las acciones, se especifica lo que se va a realizar y pueden tener asociadas una prioridad. El LHS puede ser especificado en un formalismo diferente al RHS, y en este caso se trataría de una transformación

entre diagramas, o pueden estar especificados en el mismo formalismo, y en ese caso se podría tratar del refinamiento de un mismo diagrama. Algunos de los trabajos que realizan la conversión entre modelos utilizando la herramienta AToM<sup>3</sup>® son:

Muñoz *et al.* (2004) proponen un enfoque metodológico para la transformación de un modelo de relaciones de asociación, agregación y composición de UML a un modelo de diseño de un lenguaje de programación genérico orientado a objetos. En este trabajo, se utiliza la gramática de grafos para realizar una descripción precisa y operativa de las transformaciones entre modelos, debido a que poseen una base formal sólida y una sintaxis gráfica que permite la especificación visual y operativa de transformaciones.

De Lara *et al.* (2003) realizan la conversión del modelo del diagrama de estados a redes de Petri, para realizar el análisis de sistemas híbridos; los autores se basan en el metamodelamiento y en la Gramática de Grafos para definir formal y visualmente las manipulaciones del modelo.



**Figura 3.** Metamodelo de los esquemas preconceptuales en el formalismo entidad-relación en AToM<sup>3</sup>®.  
**Figure 3.** Pre-conceptual schemas metamodel by using the AToM<sup>3</sup>® entity-relationship formalism.

Zapata y Alvarez (2005) establecen una transformación entre el diagrama de procesos y el diagrama de casos de uso utilizando la gramática de grafos y complementándola con la implementación de restricciones en código Python.

Los tres trabajos anteriores son indicios de las capacidades de la Gramática de Grafos para permitir la definición de transformaciones entre modelos, uno de los mecanismos que se requiere en el contexto de este artículo para la obtención del diagrama de casos de uso (incluyendo sus relaciones especiales) a partir de los esquemas preconceptuales.

### 3. OBTENCIÓN DEL DIAGRAMA DE CASOS DE USO A PARTIR DE DIFERENTES REPRESENTACIONES: ESTADO DEL ARTE

En la generación automática o semiautomática del diagrama de casos de uso a partir de las

especificaciones de los requisitos de los interesados, se han realizado diversos trabajos que tienen como punto de partida la descripción de los requisitos en lenguajes naturales o controlados orientados a la solución. Esos trabajos se listan a continuación.

Ben Achour (1998 y 1999), el cual está basado en un diálogo interesado-analista, que se enfoca en identificar o especificar los elementos principales de la descripción de los casos de uso. Para ello, define un conjunto de reglas de clarificación, análisis, completitud, mapeo e integración del discurso, y finalmente obtiene algunos de los actores y los casos de uso. Ninguna de las relaciones especiales <<include>>, <<extends>> o <<inheritance>> se identifica en este trabajo.

De manera similar, Pastor *et al.* (2003 y 2004) proponen una serie de reglas que, a partir de la descripción textual de los casos de uso, que es una especificación funcional en lenguaje controlado, permite obtener el diagrama de

secuencias de UML 1.4. Por medio de estas reglas, es posible extender el esquema definido para obtener los elementos esenciales del diagrama de casos de uso, como lo son los actores y los casos de uso, pero estos trabajos se enfocaron únicamente en el trazado del diagrama de secuencias. Las relaciones especiales <<include>> y <<extends>> deben ser declaradas explícitamente en el lenguaje controlado que sirve de base para la conversión.

Dietz (1999) y Shishkov *et al.* (2002) toman como punto de partida una especificación detallada a partir de la especificación verbal de los requisitos. La especificación detallada utilizada en este trabajo es determinada por el analista y está orientada a la solución del problema; es decir, existe una transformación, realizada por el analista, de los requisitos especificados por el interesado, antes de su utilización para la generación del diagrama. Con este punto de partida, derivan algunos componentes del diagrama de casos de uso como son los actores, los casos de uso, las relaciones de asociación y las relaciones <<include>>; para ello, utilizan el análisis de responsabilidades, el análisis de las denominadas protonormas, el análisis de disparadores y las normas de especificación; el uso de estos elementos requiere la intervención del analista. Además, la relación <<extends>> y la relación <<inheritance>> no son identificadas en estos trabajos.

Liu y Dong (2003) y Liu *et al.* (2004) identifican algunos elementos del diagrama de casos de uso y los utiliza como resultado intermedio para identificar los elementos del diagrama de clases. Las principales falencias de este proceso radican en que no se identifican todos los actores, los casos de uso y las relaciones especiales <<include>>, <<extends>> e <<inheritance>>. Además, se necesita la intervención del analista para validar y completar el modelo.

Por último, Zapata y Alvarez (2005) realizan la conversión de diagramas de procesos en diagramas de casos de uso, mediante la definición e implementación de reglas de consistencia entre los dos modelos. Sin embargo, en el diagrama de casos de uso obtenido no se

identifican las relaciones especiales de este diagrama.

Si bien se ha trabajado en la generación automática del diagrama de casos de uso, existen aún problemas remanentes que se pueden sintetizar así:

- Se suele utilizar como punto de partida un discurso en lenguaje natural o controlado que describe el “sistema”. Tal es el caso de las especificaciones textuales de los casos de uso o las especificaciones detalladas que se emplean en dos de los trabajos. Si bien este punto de partida permite la obtención del diagrama, un discurso tal sólo se puede obtener cuando ha transcurrido gran parte de la etapa de análisis del problema que se está analizando, e incluso requiere que ya se haya definido una solución al mismo. Si se procura la obtención temprana del diagrama, el punto de partida debería ser una descripción del dominio del problema y no de su solución.
- Casi todos los trabajos requieren una alta participación del analista para tomar las decisiones pertinentes a la generación del diagrama de casos de uso. Tomando en consideración que un trabajo como los planteados se realiza para liberar al analista de trabajos en los que lo podría sustituir de manera adecuada la máquina, sería deseable que las decisiones que debiera tomar para el trazado del diagrama de casos de uso fueran mínimas, o incluso nulas. Esto haría que el papel del analista se especializara hacia una mejor comprensión del mundo, en lugar de tener que conceptualizar los diagramas y además trazarlos de forma asistida en las herramientas CASE actuales.
- Ninguno de los trabajos presentados realiza la identificación completa de los elementos del diagrama de casos de uso. En especial, las relaciones especiales <<include>>, <<extends>> e <<inheritance>>, correspondientes a la superestructura de UML 2.0 (OMG, 2002), son las que menos se identifican.

Tomando como base estos problemas, se propone en este artículo una solución que, de manera automática, realice la conversión de una representación del discurso del interesado en los diagramas de casos de uso correspondientes,

incluyendo todos sus elementos. En la Sección siguiente se presenta esa solución.

#### 4. UNA PROPUESTA PARA LA CONVERSIÓN DE ESQUEMAS PRECONCEPTUALES EN DIAGRAMAS DE CASOS DE USO

La obtención automática del diagrama de casos de uso requiere, como punto de partida, una representación del dominio del problema; para esta propuesta, esa representación se realiza mediante los esquemas preconceptuales, que se describieron en la Sección 2.3. La lectura de estos esquemas se realiza mediante triadas de información, que son conjuntos de tres elementos así: concepto-relación-concepto. Si la relación es estructural, la triada será estructural, y si la relación es dinámica, la triada será dinámica. El concepto que precede a la relación se denomina concepto origen y el que se ubica después de la relación se denomina concepto destino.

##### 4.1 Reglas de Conversión

Para llevar a cabo la conversión de esquemas preconceptuales a diagramas de caso de uso, esta propuesta define las nueve reglas que se describen seguidamente.

###### Regla 1: Actor

En una triada dinámica, el concepto origen es un actor.

###### Regla 2: Caso de uso

Dada una triada dinámica, el nombre de los casos de uso se obtiene concatenando la relación dinámica y el concepto destino.

###### Regla 3: Asociación o relación entre el actor y el caso de uso.

El actor y el caso de uso detectados a partir de una misma triada dinámica tienen una relación de asociación entre ellos.

###### Regla 4: Relación <<include>>

Dadas dos triadas dinámicas 1 y 2, si de la relación dinámica de la triada 2 se inicia una

relación de implicación a la relación dinámica de la triada 1, entonces existe una relación <<include>>, formada de la siguiente manera:

- El caso de uso base es el que se identifica en la triada 1.
- El caso de uso incluido es el que se identifica en la triada 2.

###### Regla 5: Herencia entre actores

Si se tiene una triada estructural y una triada dinámica en las que el concepto destino de a triada estructural sea el mismo concepto origen de la triada dinámica, entonces existe una relación de herencia entre actores definida de la siguiente manera:

- El Actor hijo es el concepto que está en ambas triadas.
- El Actor padre está dado por el concepto origen de la relación estructural.

###### Regla 6: Herencia entre casos de uso

Si se tienen las triadas Concepto1-Relación Dinámica1-Concepto2 y Concepto2-Relación Estructural-Concepto3, entonces existe una relación de herencia entre casos de uso definida de la siguiente manera:

- El caso de uso hijo esta dado por la relación dinámica1, el concepto2 y el concepto3.
- El caso de uso padre esta dado por la relación dinámica1 y concepto2.

###### Regla 7: Otra herencia entre casos de uso

Si se tiene una triada Concepto1-Relación dinámica1-Concepto2 y se tiene una construcción de la forma Concepto2-Nota1 entonces existe una relación de herencia entre casos de uso definida así:

- El caso de uso padre está dado por la Relación dinámica1 y el Concepto2.
- Los casos de uso hijos están determinados por la Relación dinámica1, el Concepto2 y cada línea o elemento de la nota.

###### Regla 8: Relación <<extends>>

Cuando un concepto tiene conexiones salientes a varias relaciones dinámicas que poseen el mismo nombre, entonces existe una relación <<extends>>, formada así:

- El caso de uso base está dado por el nombre de la relación dinámica.

- Cada caso de uso extendido se forma con la relación dinámica y el concepto que tiene asociado

### Regla 9: Escenarios

Si existen tres o más triadas dinámicas unidas por implicaciones, entonces se obtiene un escenario, donde los casos de uso del escenario se determinan aplicando las reglas anteriores, con excepción de la Regla 4, que no generaría un camino de relaciones <<include>>.

## 4.2 Implementación de las Reglas en AToM<sup>3</sup>®

La implementación de las reglas anteriores en AToM<sup>3</sup>®, para obtener automáticamente la conversión entre los dos modelos consta de los siguientes pasos:

- Se definen, mediante el diagrama entidad-relación disponible en el entorno de AToM<sup>3</sup>®, los metamodelos del esquema preconceptual y el diagrama de casos de uso, los cuales se pueden

observar en las Figuras 3 y 4. A cada entidad de cada uno de los metamodelos se le puede asignar una representación gráfica propia; en el caso de las relaciones estructurales, sólo aceptan las palabras clave ES y TIENE, en tanto que las relaciones dinámicas aceptan cualquier tipo de verbo.

- Se crea la secuencia de aplicación de las reglas de transformación utilizando las reglas definidas en la Sección 4.1. Para ello, se emplea el entorno de AToM<sup>3</sup>® y se define para cada regla una prioridad, tal como se muestra en la Figura 5; en esta Figura, la secuencia de transformaciones se denomina “EPToDCU” y está compuesta por cada una de las reglas descritas, en tanto que la prioridad de cada regla es el número que acompaña al nombre de la regla. Además de estas reglas, se implementaron 10 reglas más que permiten eliminar cada uno de los elementos del esquema preconceptual; estas reglas son necesarias puesto que, en el proceso de transformación, los dos metamodelos están activos en la herramienta y deben quedar al final únicamente elementos del diagrama de casos de uso. Cada una de las reglas está compuesta por 4 elementos: LHS, RHS, *Condition* y *Action*.

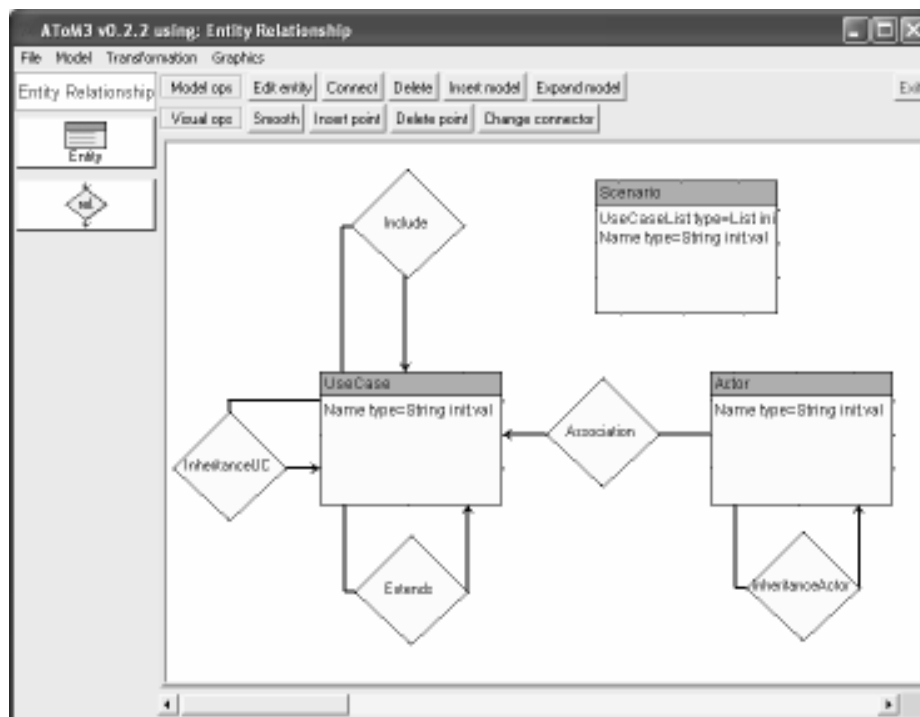
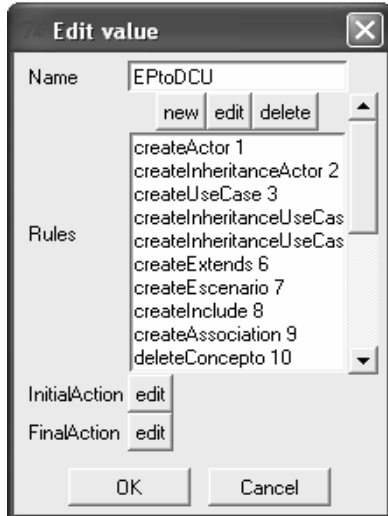


Figura 4. Metamodelo del Diagrama de casos de uso en AToM<sup>3</sup>.

Figure 4. Use cases diagram metamodel in AToM<sup>3</sup>.

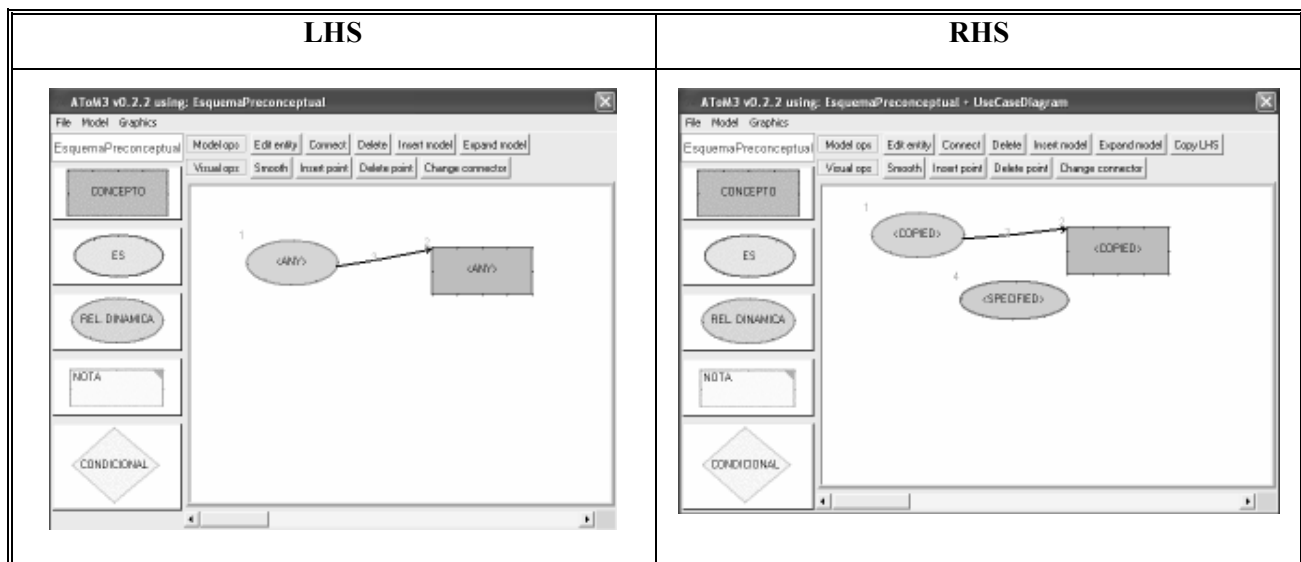


**Figura 5.** Creación de la transformación “EPtoDCU”.  
**Figure 5.** Creation of the transformation “EPtoDCU”.

Por ejemplo, para la regla *createUseCase* la implementación gráfica del LHS y el RHS se puede observar en la Tabla 1; el elemento *Condition* es el siguiente fragmento de código Python:

```
dName = self.getMatched(graphID,
self.LHS.nodeWithLabel(1)).Valor.toString()
cName = self.getMatched(graphID,
self.LHS.nodeWithLabel(2)).Nombre.toString()
name = dName + "" + cName
return not self.graphRewritingSystem.existsUseCase(name)
```

**Tabla 1.** Elementos de la regla *createUseCase*.  
**Table 1.** Elements of the rule *createUseCase*.



que establece la concatenación de los elementos para conformar el nombre del caso de uso y retornar finalmente ese valor en la imagen del caso de uso. Ahora, el elemento *Action* es el siguiente fragmento de código Python:

```
useCase = self.getMatched(graphID,
self.RHS.nodeWithLabel(4))
Name = useCase.Name.toString()
Self.graphRewritingSystem.addUseCase(name, useCase)
```

que finaliza con la creación del caso de uso; éste incluye el nombre y la imagen gráfica del caso de uso. Con las demás reglas, se procede de forma similar, para obtener los elementos restantes del diagrama de casos de uso. Finalmente, se aplican las reglas de borrado de los elementos del esquema preconceptual, con el fin de que queden únicamente en pantalla los elementos correspondientes al diagrama de casos de uso.

- Se procede a la creación del esquema preconceptual en el entorno suministrado por el ATOM<sup>3</sup>® para ese fin. En ese caso, se debe mantener activo el metamodelo correspondiente.
- Se activa la transformación y aparece el diagrama de casos de uso, que es el resultado de la aplicación de las reglas que se definieron.

### 4.3 Caso de Estudio

A continuación, se presenta un caso de estudio correspondiente al manejo de una biblioteca. El discurso del interesado, expresado en UN-Lencep, es el siguiente:

*socio\_de\_la\_biblioteca reserva revista*  
*socio\_de\_la\_biblioteca renueva prestamo*  
*socio\_de\_la\_biblioteca reserva libro*  
*socio\_investigador es socio\_de\_la\_biblioteca*  
*cuando socio\_de\_la\_biblioteca presta libro, entonces bibliotecario valida socio*  
*bibliotecario actualiza catalogo*  
*socio\_investigador recomienda libro*

El esquema preconceptual correspondiente a este discurso se muestra en la Figura 6. En la Tabla 2 se muestra paso a paso la aplicación de las reglas de transformación para el caso de estudio; en dicha Tabla se presenta la porción del esquema preconceptual, las reglas aplicadas y la porción resultante del diagrama de casos de uso.

Finalmente, en la Figura 7 se puede observar el diagrama de casos de uso obtenido al ejecutar la transformación.

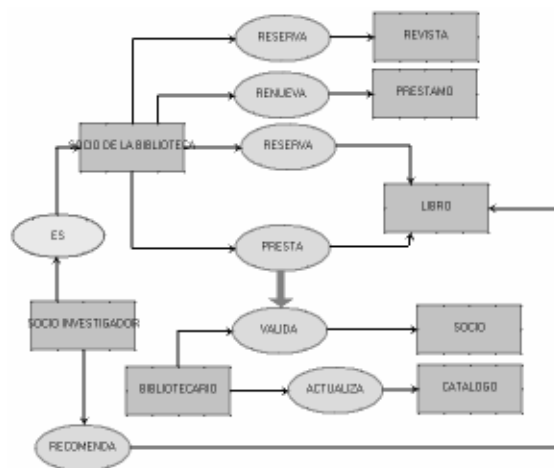






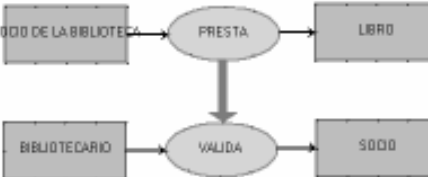
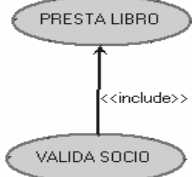


Figura 6. Esquema preconceptual del caso de estudio.

Figure 6. Pre-conceptual Schema of the case study.

Tabla 2. Aplicación de las reglas para el caso de estudio.  
 Table 2. Rules application to the case study

ESQUEMA PRECONCEPTUAL	REGLA APLICADA	DIAGRAMA DE CASOS DE USO
	1, 2 y 3	
	1, 2 y 3	
	1, 2 y 3	
	1, 2 y 3	
	1, 2 y 3	
	1, 2 y 3	

**Tabla 2.** Aplicación de las reglas para el caso de estudio (continuación).  
**Table 2.** Rules application to the case study

ESQUEMA PRECONCEPTUAL	REGLA APLICADA	DIAGRAMA DE CASOS DE USO
	1, 2 y 3	
	1 y 5	
	4	
	8	

## 5. CONCLUSIONES

En el ámbito de la elicitación de requisitos, vienen cobrando fuerza los intentos por automatizar la elaboración de los diferentes esquemas conceptuales. En este artículo se abordó la problemática del trazado automático del diagrama de casos de uso y se identificaron tres problemas aún no completamente resueltos: se suele partir de representaciones de la solución y no de representaciones del problema, se requiere aún una alta participación del analista en el proceso y no se identifican todos los elementos del diagrama de casos de uso.

En este artículo se propuso un método que parte de la representación del discurso del interesado en esquemas preconceptuales y luego los traduce de manera automática a diagramas de casos de uso. Los tres problemas anotados se solucionan con este método, dado que se admite que el discurso del interesado no se relacione con la aplicación de software (sino que más bien se refiera a los términos del dominio), el proceso es

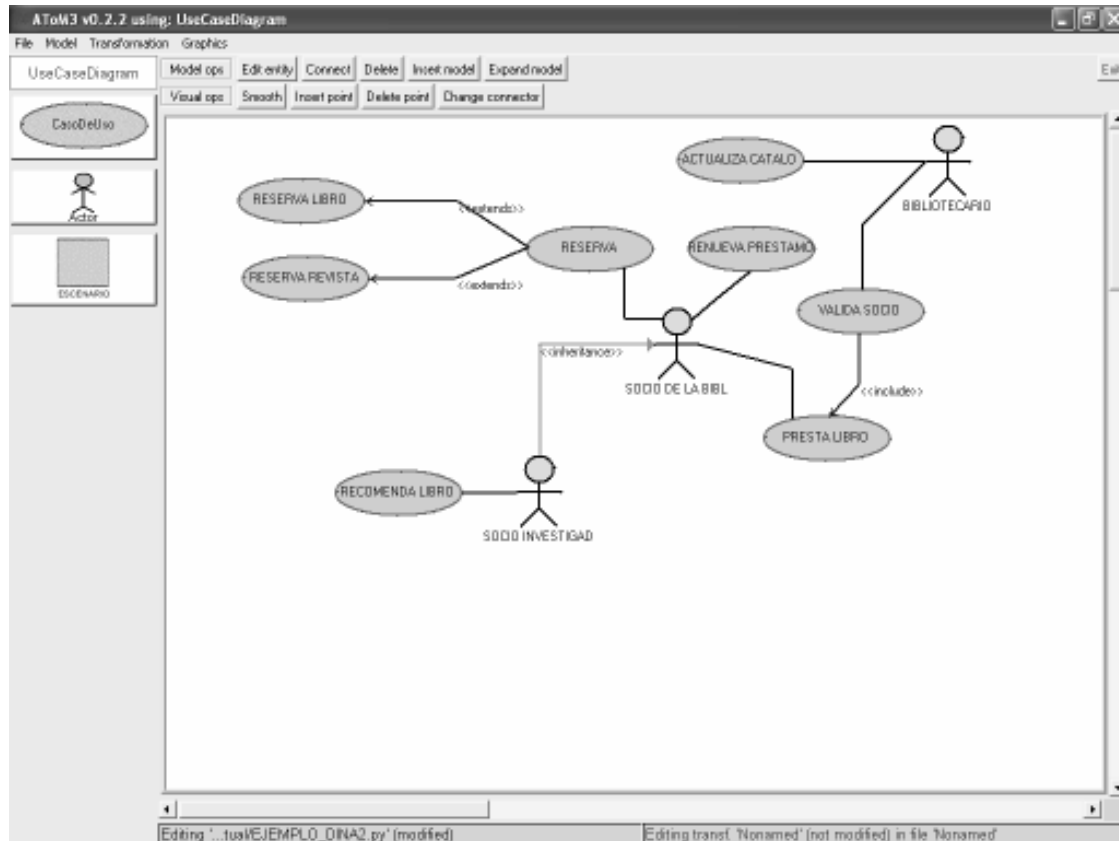
completamente automático y se identifican completamente las relaciones que existen entre actores y casos de uso. Para la solución mediante esquemas preconceptuales se debió complementar la sintaxis básica de dichos esquemas con dos elementos nuevos (la nota y la conexión de la nota con el concepto al que pertenece), con el fin de poder identificar una forma de herencia entre casos de uso.

La implementación en AToM<sup>3</sup>® presentó las mismas dificultades que se han planteado en otros artículos del grupo: la Gramática de Grafos fue insuficiente para especificar de manera completa las reglas de transformación y se debió recurrir nuevamente al lenguaje Python para complementarlas. Pese a esto, la implementación mostró ser adecuada para las necesidades de transformación y se pudieron hacer los ensayos correspondientes a las diferentes reglas.

De esta manera, se pudo comprobar que efectivamente el diagrama de casos de uso se puede obtener de manera automática (sin

intervenciones por parte del analista en el proceso de transformación) a partir de un esquema preconceptual que represente el dominio de un problema particular y no así su solución. Las reglas permiten hallar todos los

elementos de dichos diagramas, incluyendo las relaciones especiales <<include>>, <<extends>> e <<inheritance>>, para las cuales existían pocas soluciones en el estado del arte



**Figura 7.** Diagrama de Casos de uso destino.

**Figure 7.** Target Use Case Diagram.

## 6. TRABAJO FUTURO

Para este trabajo en particular, se generaron nuevas preguntas a partir de la propuesta realizada y su implementación, que pueden motivar la continuación de los esfuerzos de esta línea:

- Se podría comprobar si es posible obtener el último de los diagramas comportamentales que aún no se obtiene automáticamente de los Esquemas Preconceptuales: el diagrama de actividades.
- Igualmente, se podría comprobar si es posible obtener a partir de estos Esquemas el diagrama de secuencias de la versión 2.0 de UML, cuya sintaxis ya lo aleja sustancialmente del diagrama

de comunicación, que ya es obtenible desde Esquemas Preconceptuales.

- Falta por establecer si existen reglas adicionales que puedan generar variaciones sobre los elementos identificados en el diagrama de Casos de Uso.
- Se debería verificar también si se pueden establecer reglas para generar algún tipo de código de las interfaces gráficas de usuario a partir de los resultados de este artículo.
- Finalmente, un área en la que se podría verificar la aplicación de un método similar al propuesto por este artículo, podría ser el área de los requisitos no funcionales; se podría estudiar si tendencias actuales como los aspectos podrían ser modeladas en Esquemas Preconceptuales o si pueden existir reglas de transformación que

posibiliten su modelamiento desde fases tempranas del ciclo de vida del software.

## 7. AGRADECIMIENTOS

Este artículo se realizó en el marco de los siguientes proyectos de investigación: “construcción automática de esquemas conceptuales a partir de lenguaje natural”, financiado por la dime y “definición de un esquema preconceptual para la obtención automática de esquemas conceptuales de uml”, financiado por dinain y administrado por la DIME.

## REFERENCIAS

- [1] ATOM<sup>3</sup> ®. MSDL – ATOM<sup>3</sup> ®. Página Web de la herramienta ATOM<sup>3</sup> ®. Available: <http://atom3.cs.mcgill.ca/>. [Citado Febrero 22 de 2007]
- [2] BEN ACHOUR, C (1998). Guiding the construction of textual case use specifications. *Data & Knowledge Engineering Journal*, vol 25 No. 1-2. p. 125–160.
- [3] BEN ACHOUR, C. (1999). *Extraction des besoins par analyse of scenarios textuels*. Tesis de Doctorado de la Universidad de Paris.
- [4] COCKBURN, A. (2000). *Writing Effective Use Cases*. Addison-Wesley Pub Co.
- [5] DE LARA, J., y VANGHELUWE, H. (2002). ATOM<sup>3</sup>: A tool for Multi-Formalism and Meta-Modelling. *Proceedings of the Fifth International Conference on Fundamental Approaches to Software Engineering*. 174–188.
- [6] DE LARA, J., VANGHELUWE, H y ALFONSECA, M. (2003). Using Meta-Modelling and Graph-Grammars to Create Modelling Environments. *Electronic Notes in Theoretical Computer Science*, Vol. 72, No. 3.
- [7] DE LARA, J., GUERRA, E. y VANGHELUWE, H. (2003). *Meta-Modelling, Graph Transformation and Model Checking for the Analysis of Hybrid Systems*. Juan de Lara, Esther Guerra and Hans Vangheluwe. AGTIVE'2003 (Applications of Graph Transformation with Industrial Relevance), Charlottesville, USA. *Lecture Notes in Computer Science 3062*. Springer.
- [8] DIETZ, J. (1999). ‘Understanding and Modelling Business Processes with DEMO’. *Proc: 18th International Conference on Conceptual Modeling*. Paris.
- [9] DOME Users’ Guide, available from <http://www.htc.honeywell.com/dome/support.htm#documentation>. [Citado Febrero 22 de 2007a]
- [10] DOME. What is Dome. Available: <http://www.htc.honeywell.com/dome/description.htm>. [Citado Febrero 22 de 2007b].
- [11] FIRESMITH, D. (2002). Use case: the pros and cons. Knowledge Systems Corporation.
- [12] FOWLER, M. (2004). *UML Distilled: A brief guide to the Standard Object Modeling Language*. Addison-Wesley, Reading.
- [13] JACOBSON, I. (1999). “UML y patrones. Introducción al análisis y diseño orientado a objetos”. Prentice Hall.
- [14] JACOBSON, I., BOOCH, G. y RUMBAUGH, J. (2001). *El Proceso Unificado de Desarrollo de Software*. Addison Wesley.
- [15] LIU, D. (2003). *Automating Transition From Use Cases to Class Model*. Msc Thesis, University of Calgary. Capítulo 7.
- [16] LIU, D. SUBRAMANIAM, K. EBERLEIN, A. y FAR, B. (2004). *Natural Language Requirements Analysis and Class Model Generation Using UCDA*. IEA/AIE, LNAI 3029. p. 295–304.
- [17] MUÑOZ J. (2004). *Una Gramática de Grafos para la Transformación de Relaciones de Asociación desde Modelos de Análisis hacia Modelos de Diseño*. Technical report, DSIC-UPV.

- [18] OBJECT MANAGEMENT GROUP, Inc. (OMG). (2002) Unified Modeling Language: Superstructure 2.0. Draft Adopted Specification. p. 511 - 528.
- [19] PASTOR O, DIAZ I, MORENO L. (2003). Traducción de casos de uso en patrones de interacción de instancias: una aproximación lingüística. Jornadas Iberoamericanas de Ingeniería de Software e Ingeniería del Conocimiento.
- [20] PASTOR O, DIAZ I, LOSAVIO F, MATTEO A. (2004). Specification pattern of use. Information and Management. Vol 41. p. 961-975.
- [21] SHISHKOV, B., XIE, Z., LUI, K., y DIETZ, J. (2002). Using norm analysis to derive use case from business processes. 5th Workshop on Organizations semiotics. June 14-15. Delft the Netherlands.
- [22] SMOLANDER, K., LYYTINEN, K. TAHVANAINEN, V. and MARTIIN, P. (1991). Metaedit—A flexible graphical environment for methodology modeling. In Proceedings of Advanced Information Systems Engineering CAiSE'91, Lecture Notes in Computer Science, pages 168-193.
- [23] VENDLER, Z. (1967). Verbs and Times. Linguistics in philosophy, Ithaca, Cornell University Press.
- [24] ZAPATA, C, y ALVAREZ, C. (2005). Conversión de Diagramas de Procesos en Diagramas de Casos de Uso Usando AToM<sup>3</sup>. Revista Dyna. Nro 146. 103-113.
- [25] ZAPATA, C. 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. Revista EAFIT. Vol 41. No 137. 77-95.
- [26] ZAPATA, C., GELBUKH, A. y ARANGO, F. (2006a). Pre-conceptual Schema: a UML Isomorphism for Automatically Obtaining UML Conceptual Schemas. Research in Computing Science: Advances in Computer Science and Engineering, Vol. 19, 3-13.
- [27] ZAPATA, C., GELBUKH, A. y ARANGO, F. (2006b). UN-Lencep: Obtención Automática de Diagramas UML a partir de un Lenguaje Controlado. 3<sup>er</sup> Taller de Tecnologías del Lenguaje Humano. Encuentro Nacional de Ciencias de la Computación, San Luis Potosí, Septiembre.
- [28] ZAPATA, C., GELBUKH, A. y ARANGO, F. (2006c). Pre-conceptual Schema: A Conceptual-Graph-Like Knowledge Representation for Requirements Elicitation. Lecture Notes in Computer Science, Vol. 4293, 2006, 17-27.