Revista ACTIVA, ISSN 2027-8101. Núm 3, enero-junio 2012, pp. 57-66
Tecnológico de Antioquia, Medellín (Colombia)

Reglas para la conversión de esquemas preconceptuales en diagramas de estructura compuesta

Rules for transforming preconceptual schemas into composite structure diagrams

PhD, Carlos Mario Zapata Jaramillo
Profesor Asociado Escuela de Sistemas.
Facultad de Minas, Universidad Nacional de Colombia,
sede Medellín, cmzapata@unal.edu.co

MSc, Roberto Antonio Manjarrés Betancur
Profesor Asistente, Facultad de Ingenierías
Politécnico Colombiano Jaime Isaza Cadavid
ramanjar@elpoli.edu.co

María Angélica Cano Lara
Ingeniera de Sistemas e Informática
Universidad Nacional de Colombia
macano@unal.edu.co

Recibido: 19 de febrero 2012
Aprobado: 18 de marzo 2012

Resumen

El diagrama de estructura compuesta de UML 2.0 permite la descripción de la estructura interna de una clase, facilitando la representación de sus interacciones. Es posible partir del diagrama de clases para llegar al de estructura compuesta, pero se parte de las transformaciones que sugiere el analista y no del discurso del interesado. Por otro lado, existe una tendencia hacia la generación automática de artefactos (especialmente diagramas de UML) a partir de descripciones en lenguaje controlado, que facilita la consistencia entre los diagramas generados. Siguiendo esta tendencia, en este artículo se propone la obtención automática del diagrama de estructura compuesta partiendo de los denominados esquemas preconceptuales, que son mecanismos para la representación del conocimiento.

Palabras clave: Consistencia entre diagramas, diagrama de estructura compuesta, diagrama UML 2.0, esquema preconceptual, reglas de conversión.

Abstract

UML 2.0 composite structure diagrams allow to describe the internal structure of a class, thereby facilitating the class interaction representation. A class diagram can be used for obtaining a composite structure diagram, but in this case the departing point would be the changes suggested by the analyst rather than the stakeholder discourse. On the other hand, a trend toward the automatic generation of artifacts (mostly UML diagrams) from controlled language descriptions is arising. Such trend is making the consistency among diagrams generated easier. Following this trend, we propose in this paper the automated generation of the composite structure diagram from the so-called pre-conceptual schemas, which are devices for knowledge representation.

Keywords: Composite structure diagram, consistency among diagrams, pre-conceptual schema, transformation rules, UML 2.0 diagram

1. Introducción

Al iniciar el proceso de desarrollo de una aplicación de software se realiza la recolección de información, la cual se interpreta y se representa mediante modelos del dominio, utilizando un estándar de modelado, usualmente el *unified modeling language (UML)* (Zapata & Arango, 2007; OMG, 2007). En un proyecto de software es necesario generar diferentes artefactos, algunos de ellos derivados completa o parcialmente de otros artefactos. En el mundo de los objetos se encuentran, normalmente, objetos que se componen de más objetos (Orozco, 2007). Con el surgimiento de UML 2.0 se propuso un diagrama que permite modelar dicha información y que se puede derivar del diagrama de clases. Dicho diagrama, denominado estructura compuesta, muestra las relaciones de composición entre los objetos, clasificadores y sus partes.

Los modelos en ingeniería de software se suelen elaborar en herramientas CASE (*computer-aided software engineering*), las cuales permiten el trazado y la edición de diferentes tipos de diagramas (incluida la gama de diagramas UML). La experiencia e interpretación del analista son fundamentales para elaborar dichos diagramas. Siguiendo esta tendencia, la herramienta CASE UNC-Diagramador sirve de apoyo a los analistas en el proceso de elaboración de diagramas reduciendo la subjetividad y mejorando la consistencia. Dicha herramienta se desarrolló en un ambiente para la obtención automática de algunos diagramas de UML 2.0. La herramienta se basa en un lenguaje controlado (UN-Lencep), en un mecanismo de representación del conocimiento (los esquemas preconceptuales) y en un grupo de reglas para la traducción del lenguaje controlado a tres diagramas de UML 2.0 (clases, comunicación y máquina de estados; Zapata *et al.*, 2006).

Oliver y Luukkala (2006) muestran la relación entre el diagrama de clases y el de estructura compuesta, pero la transformación que plantean parte de las transformaciones del analista y no del discurso del interesado. Röhl (2008) propone un esfuerzo para la descripción interna del diagrama de estructura compuesta empleando un lenguaje formal, pero limita su ámbito a los modelos de eventos discretos. Desde la formalización se puede comprender la conformación del diagrama de estructura compuesta, pero sólo en el dominio de los modelos de eventos discretos. Ober y Dragomir (2010; 2011), empleando el denominado perfil OMEGA2, realizan un análisis detallado del diagrama de estructura compuesta y formalizan un conjunto de reglas de consistencia en el lenguaje OCL (*object constraint language*) que rigen su comportamiento. Ya sea que el punto de partida de elaboración del diagrama de estructura compuesta sea el diagrama de clases, un lenguaje formal o el lenguaje OCL, la representación inicial se encuentra cercana al analista, dificultando la validación que podría realizar un interesado sobre su comportamiento, pues estos lenguajes requieren mucho entrenamiento para su uso. Además, la propuesta de Röhl (2008) se aplica únicamente al dominio de los sistemas en tiempo real.

Atendiendo a estas limitaciones, en este artículo se propone, diseña y desarrolla un grupo de reglas para la generación automática de diagramas de estructura compuesta (DEC), apoyados en el diagrama de clases, a partir de esquemas preconceptuales (EP). Con la generación automática se mantiene la consistencia, al pasar del lenguaje controlado al EP y finalmente al diagrama requerido.

Este artículo se estructura de la siguiente forma: en la sección 2 se describen los diagramas relacionados, el esquema preconceptual y, de forma detallada, el diagrama de estructura compuesta; en la sección 3 se realiza una breve descripción de los antecedentes en la conversión de diagramas; en la sección 4 se presenta el grupo de reglas para el proceso de transformación de EP a DEC; en la sección 5 se describen ejemplos de aplicación de las reglas. Por último, se entregan las conclusiones y algunas perspectivas de trabajo futuro.

2. Descripción del esquema preconceptual y del diagrama de estructura compuesta de UML

En UML 2.0 hay 13 tipos diferentes de diagramas, diferenciados en tres grandes grupos: los diagramas de estructura, que hacen énfasis en los elementos que deben existir en el sistema modelado; los diagramas de comportamiento, que ponen el énfasis en lo que debe suceder; los diagramas de interacción, que constituyen un subtipo de los diagramas de comportamiento, que se basan en el flujo de control y de datos (OMG, 2007).

2.1 Esquema preconceptual

Es una representación intermedia entre las especificaciones textuales en lenguaje natural (modelos verbales o documentos de la organización) y los diferentes esquemas conceptuales que permiten el modelado de una aplicación de software. Su sintaxis básica se muestra en la figura 1.

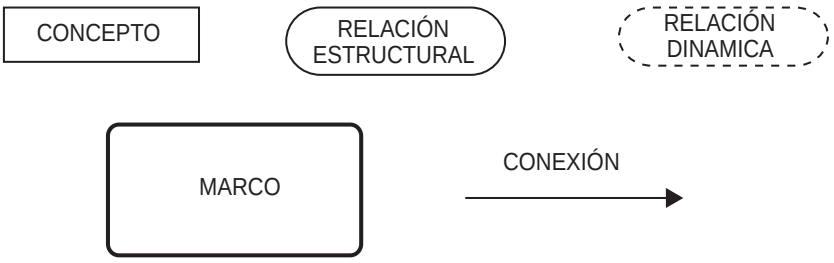


Figure 1. Sintaxis básica de los esquemas preconceptuales (adaptado de Zapata *et al.,* 2006)

En los conceptos se escriben sustantivos simples o uniones de dos sustantivos con la preposición "de". En las relaciones estructurales sólo se permiten los verbos "es" y "tiene". Los verbos que expresen actividades se representan con relaciones dinámicas. Las implicaciones representan una secuencia de actividades, indicando que una relación dinámica es precondición de otra determinada relación dinámica. Finalmente, el marco agrupa los elementos que conforman una parte o un esquema preconceptual completo.

2.2. Diagrama de estructura compuesta

Muestra la estructura interna de una clase, las colaboraciones que esta estructura hace posibles (véase la figura 2), la configuración y la relación de las partes que, juntas, realizan el comportamiento del clasificador contenido (OMG, 2011).

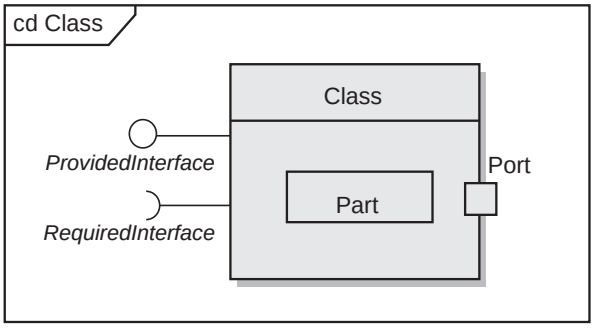


Figure 2. Elementos del diagrama de estructura compuesta (EA, 2008)

En un diagrama de estructura compuesta se encuentran los elementos que se muestran en la figura 3 y que se describen seguidamente.

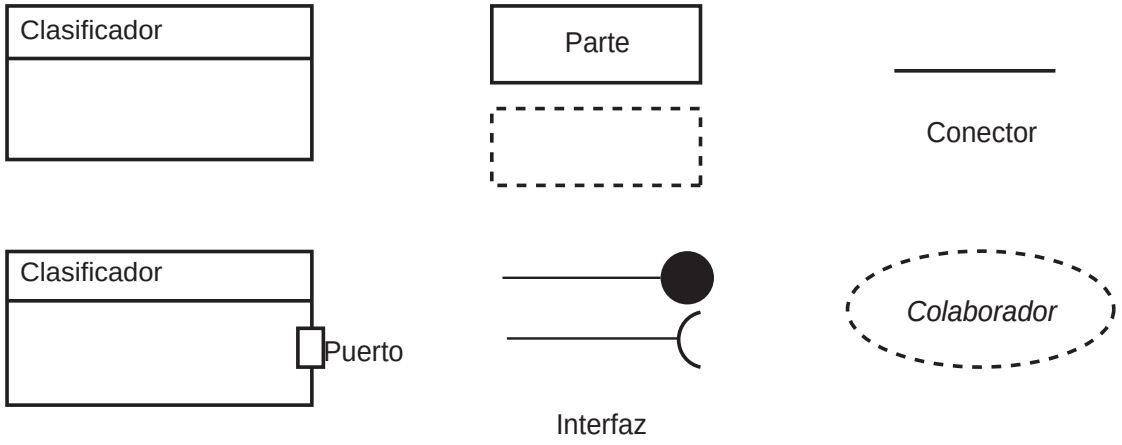


Figure 3. Elementos del diagrama de estructura compuesta

Clasificador estructurado: Representa una clase, frecuentemente una clase abstracta, cuyo comportamiento se puede describir completa o parcialmente mediante interacciones entre partes. Se reconoce como una clase, pues representa uno o más objetos y refleja su estructura y comportamiento en el sistema. Se ve como una plantilla desde la cual se crean las instancias actualmente en ejecución.

Parte: Se muestra con un rectángulo e indica los objetos que conforman el objeto principal. Las partes son instancias en tiempo de ejecución de clases o interfaces. La multiplicidad se puede especificar, para una parte, usando la notación [x{...y}]. Se permite una parte con una línea sólida dentro de una clase, lo que significa, en un diagrama de clases, que la clase contenedor tiene una relación de composición con dicho elemento. Alternativamente, si el contorno es con línea discontinua, esto indica que la parte rodeada refiere y utiliza la propiedad, pero sin una composición explícita.

Conector: Ilustra un enlace de comunicación entre partes para llegar al propósito de la estructura. Un conector se representa con una línea que une una combinación de partes, puertas y clasificadores estructurados. Los conectores pueden tener multiplicidad. Se distinguen dos tipos, el conector de montaje que une dos partes interiores o puertos. Similar a una relación de asociación, un conector de montaje demuestra que una parte de los compuestos se conecta y suministra servicios que la otra requiere. Finalmente, el conector *delegar* define el ensamble interno de los puertos e interfaces externos de un componente. Al usar un conector delegar se conectan los trabajos internos del sistema con el mundo exterior, mediante una delegación de las conexiones de las interfaces externas. Se muestra con una punta de flecha abierta en el extremo de la línea de conexión.

Puertos e interfaces: Un puerto define un punto de interacción entre una instancia del clasificador y su entorno, o entre el comportamiento del clasificador y sus partes internas. Las puertas pueden, opcionalmente, especificar los servicios que proveen y los servicios requeridos de otras partes del sistema. Se representan con pequeños cuadros.

3. Antecedentes

En un diagrama de estructura compuesta se encuentran características y finalidades comunes con otros diagramas de UML. En el diagrama de componentes se representa la dependencia y la relación entre componentes, la vista estática y dinámica de un sistema. En un diagrama de paquetes se divide un sistema por funcionalidad, permitiendo claridad interna en los paquetes y acoplamiento externo entre ellos. En el diagrama de comunicación se modela la interacción entre los objetos. En el diagrama de clases y en el de objetos se describe la estructura del sistema, los atributos y las relaciones (Fowler, 2003). Cada una de las funcionalidades mencionadas se puede representar en el diagrama de estructura compuesta.

En UML 2.0 se introdujo este diagrama para complementar los artefactos existentes, lo que desencadenó una serie de interpretaciones para los modeladores, que no siempre van en consonancia con el uso previsto de la estructura ni del lenguaje con el que se creó. Oliver y Luukkala (2006) describen la estructura básica y las relaciones en contraste con el diagrama de clases, para luego definir una serie de posibles usos. Este trabajo permite ayudar a entender zonas de ambigüedad, extensión e incoherencia (Fowler, 2003). Sin embargo, estas definiciones planteadas se basan en diagramas existentes, es decir, quedan a disposición de interpretaciones particulares y la consistencia pasa a ser un asunto subjetivo.

En cuanto a la manera como se podría elaborar el diagrama de estructura compuesta, Oliver y Luukkala (2006) plantean que se podría obtener desde una representación basada en el diagrama de clases, mostrando las equivalencias entre ambos diagramas. Röhl (2008) formaliza el diagrama de estructura compuesta para analizar su sintaxis y sus reglas de buena formación, pero dicha formalización se circunscribe a un tipo especial de modelos: los de eventos discretos. Otra formalización, que emplea el lenguaje OCL (2010; 2011), define lo que ellos denominan el perfil OMEGA2, que permite evaluar también la consistencia y la buena construcción del diagrama de estructura compuesta. Partiendo de la base de que es necesario integrar la información consignada en un diagrama de estructura compuesta con la que arrojan los demás diagramas de un modelo y que dicha información sea consistente y completa para representar el discurso de un interesado, el diagrama de clases (como lenguaje semiformal) y los lenguajes formales como el OCL presentan inconvenientes para la validación de la información que puede realizar un interesado. Si bien las propuestas de Dragomir y Ober (2010; 2011) y de Röhl (2008) validan la consistencia interna de la información consignada en el diagrama de estructura compuesta, las reglas no incluyen la consistencia con los demás diagramas que conforman un modelo. Además, la propuesta de Röhl (2008) se aplica únicamente a un dominio restringido.

Los diagramas deben partir de un discurso del interesado que se realiza al inicio del proceso de análisis. Los analistas deben crear los diagramas con la ayuda de las herramientas CASE (Zapata *et al.,* 2006a). Las herramientas CASE disponen de un conjunto de restricciones gráficas que permiten chequear las reglas del diseño y analizar la lógica del diagrama. Entre las funciones principales de estas herramientas se encuentran el apoyo a la elaboración de los diferentes diagramas (especialmente en el trazado y edición) y la generación parcial de código a partir de estos (IBM, 2011). Sin embargo, estas herramientas no constituyen un avance sustancial en la obtención automática de los diagramas UML, puesto que dejan la interpretación del discurso del interesado completamente en manos del analista (Zapata & Arango, 2005).

Para atender este problema, se ha venido desarrollando la herramienta CASE UNC-Diagramador, que permite obtener diferentes artefactos de UML a partir de un lenguaje controlado, en una tendencia similar a la arquitectura orientada por modelos (Sedillo, 2007). Inicialmente, se tiene un archivo con expresiones en UN-Lencep con el cual se generan los esquemas preconceptuales, que sirven como base para el desarrollo de los diferentes diagramas de UML. Inicialmente se implementaron las reglas para la elaboración

del diagrama de clases, el de comunicación y el de máquina de estados (Zapata *et al.,* 2006). Al incluir reglas que incluyan la construcción del diagrama de estructura compuesta, se podría garantizar la consistencia con otros diagramas y no sólo la consistencia interna. Además, los esquemas preconceptuales son de dominio general, pues se pueden construir para cualquier discurso de los interesados.

4. Reglas de conversión entre esquemas preconceptuales y diagramas de estructura compuesta

Partiendo de los elementos que se especifican en la sección 2 sobre los esquemas preconceptuales y el diagrama de estructura compuesta, es posible definir un conjunto de reglas, con el fin de establecer la transformación entre el esquema y este diagrama. En la tabla 1 se muestran las reglas propuestas en el presente artículo para lograr la conversión. En la primera columna se encuentra la representación del esquema preconceptual y en la columna DEC se ve el resultado de la transformación. Algunas reglas poseen una tercera columna que corresponde al equivalente al diagrama de clases (nótese que estos elementos parten, también, del esquema preconceptual empleando las reglas que definen Zapata *et al.,* 2006). En el esquema preconceptual, donde se identifica 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. A es un clasificador.

En un diagrama de estructura compuesta se puede encontrar un clasificador dentro de otro, como se esquematiza en la figura 4.

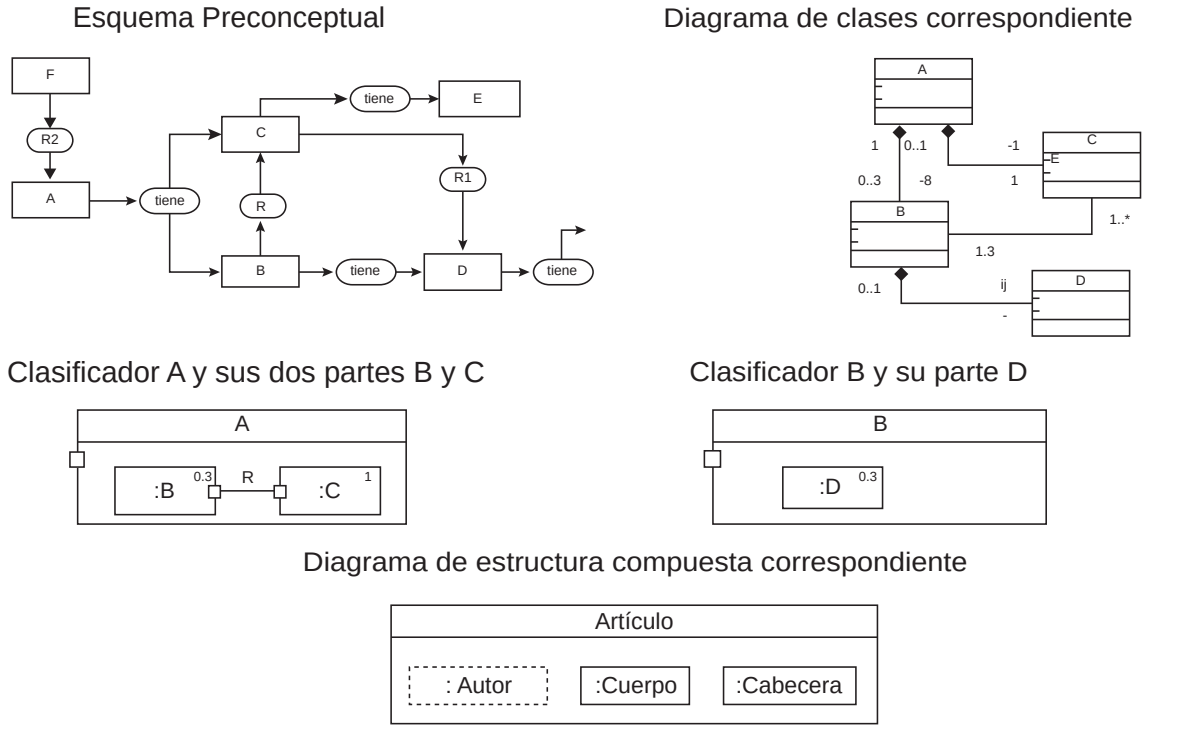
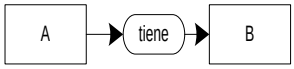
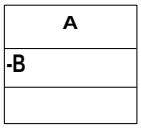
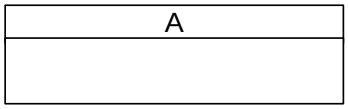
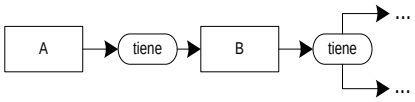
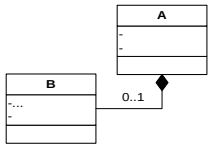
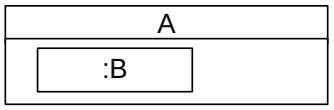
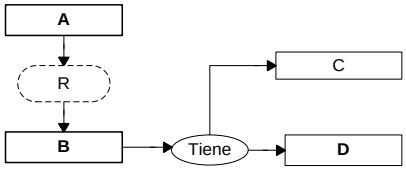
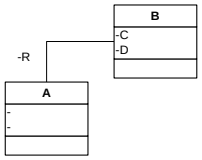
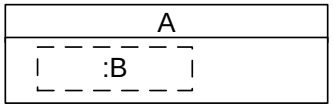
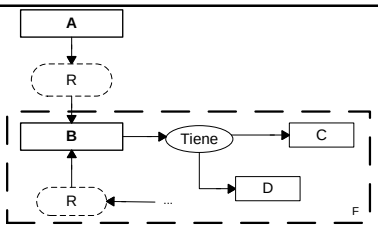
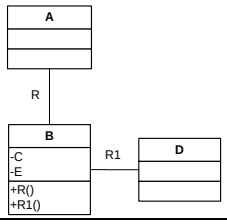
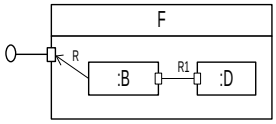
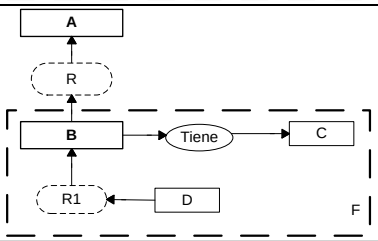
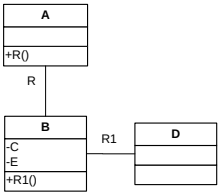
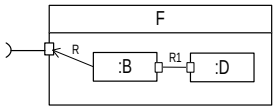
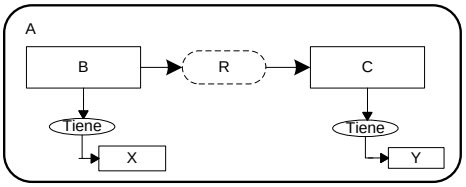
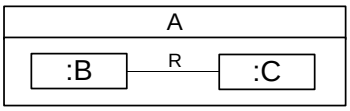
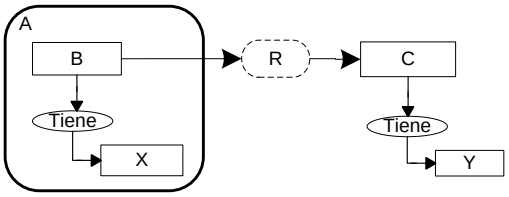
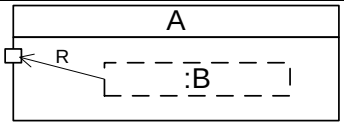


Figure 4. Clasificador dentro de un clasificador

Tabla 1. Reglas para la obtención del diagrama de estructura compuesta

Esquema Preconceptual	Diagrama de Clases	Diagrama Estructura Compuesta
		
		
		 B es un objeto anónimo.
		
		

Esquema preconceptual	Diagrama estructura compuesta
	 El marco es un concepto que define una agrupación de elementos, el nombre del marco debe ser un concepto.
	 El conector delegar permite poner a disposición o requerir las operaciones internas o externas

5. Casos de estudio

Con la herramienta que actualmente se está desarrollado, UNC-Diagramador, se ingresa el discurso del interesado escrito en el lenguaje controlado UN-Lencep. Posteriormente, se realiza la traducción automática del esquema preconceptual, generando los diagramas de clases, comunicación y máquina de estados (Zapata *et al.*, 2006). En las figuras 5 y 6 se muestra la conversión correspondiente al DEC.

A continuación se ejemplifica con dos discursos bajo la especificación de UN-Lencep, el correspondiente esquema preconceptual y lo que sería el diagrama de estructura compuesta que se genera en UNC-Diagramador.

Clasificador y partes:

Un artículo tiene cabecera y cuerpo
Cabecera tiene introducción y resumen
Cuerpo tiene antecedentes, propuesta y conclusiones.
Autor tiene nombre.
Un autor escribe el artículo.

Puertos y conectores:

La ventana tiene nombre, área de trabajo, barra de herramientas, una barra lateral y una inferior.
Las barras lateral e inferior mueven el área de trabajo.
El área de trabajo tiene tamaño.
La barra de herramientas tiene comando.
Uno de los comandos cierra la ventana.

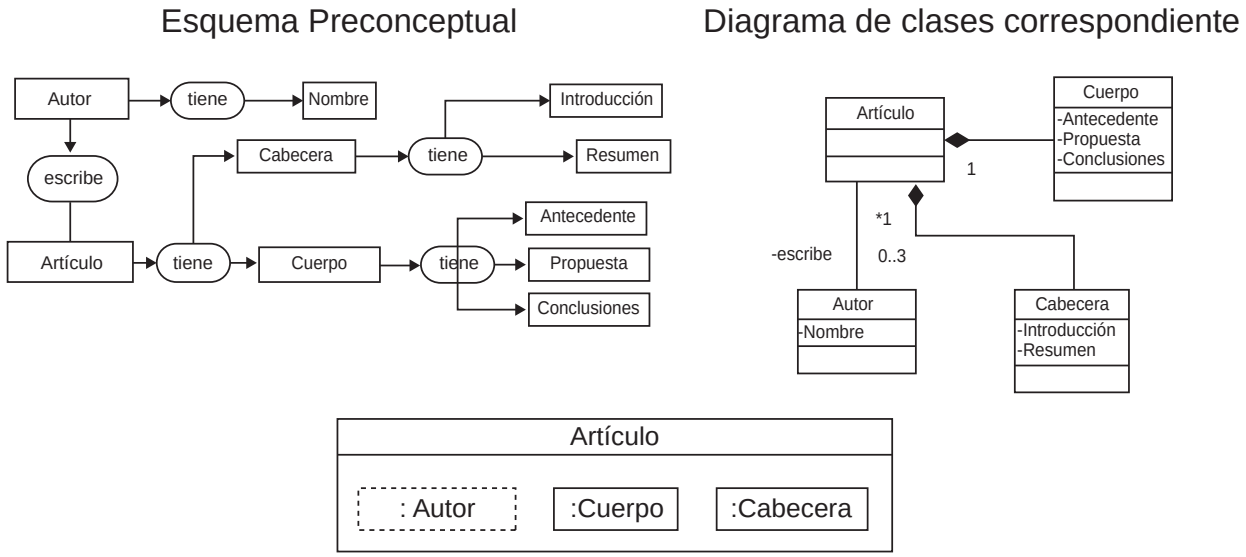


Figure 5. Conversión caso 1

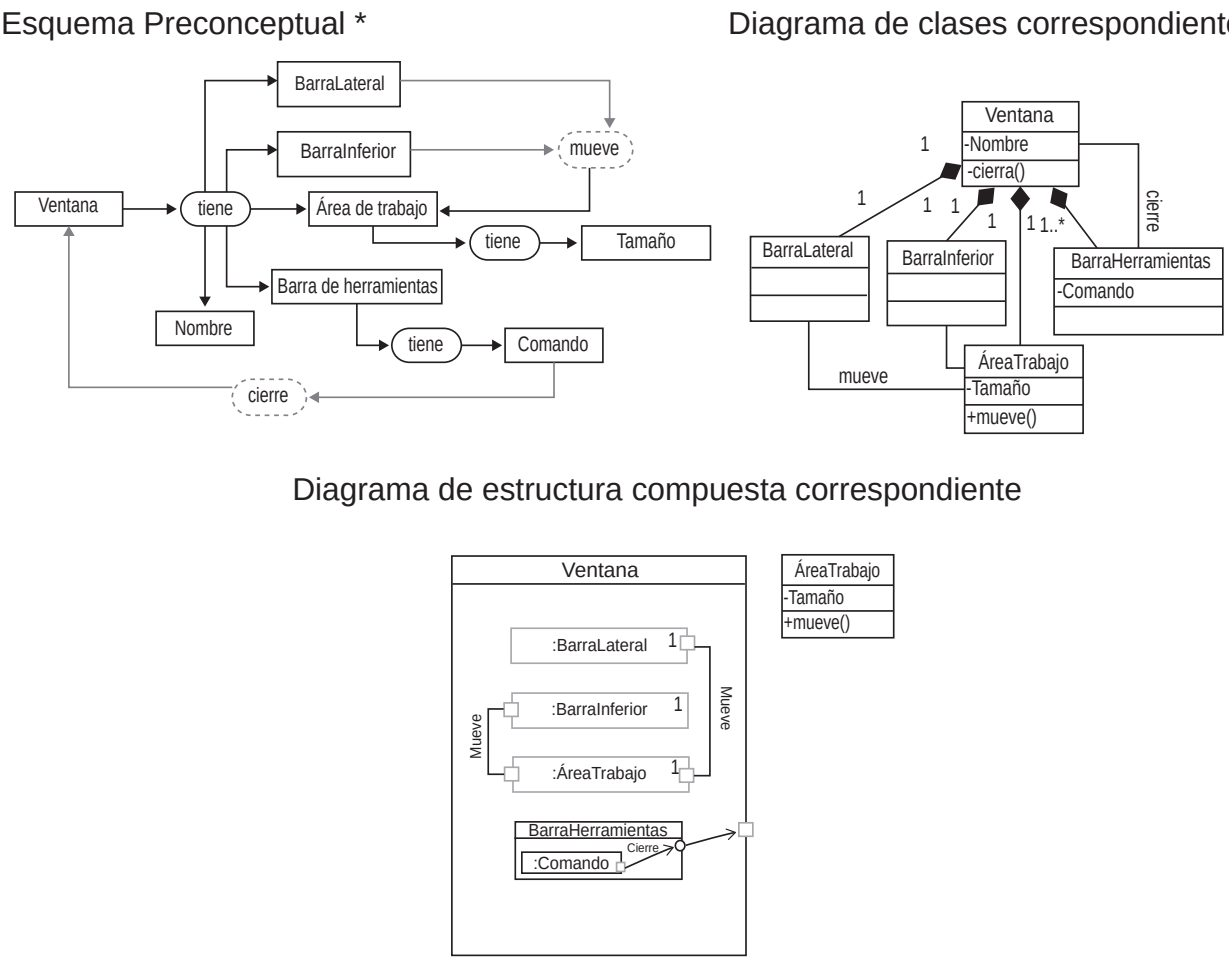


Figure 6. Conversión caso 2

*El esquema preconceptual solo permite relaciones dinámicas antecedidas de un actor. Para este tipo de problemas se levanta esta restricción, de modo que se pueden tener conceptos inanimados ejecutando ciertas acciones.

6. Conclusiones y trabajo futuro

Los analistas requieren representar, por medio de modelos, el mundo real a partir de la información que proporcionan los interesados. Los analistas deben elaborar dichos modelos con base en la experiencia y los conocimientos, tolerando la posibilidad de omitir o malinterpretar dicha información. Por tal razón se está desarrollando una herramienta para facilitar y mejorar la elaboración de los modelos. En UNC-Diagramador ya se definieron las reglas de transformación para llegar a tres diagramas de UML 2.0 (clases, comunicación, máquina de estados) (Zapata *et al.*, 2006).

En este artículo se definió un grupo de reglas de transformación para generar el diagrama de estructura compuesta. Los resultados de esta propuesta buscan reducir el tiempo de elaboración del diagrama y mejorar la consistencia que debe existir entre la especificación formal y los diferentes diagramas.

Como trabajo futuro se propone implementar las reglas ya definidas para la generación del diagrama de estructura compuesta por medio de

la herramienta CASE UNC-Diagramador, que emplea el entorno definido para generar los diagramas UML a partir de un discurso en UN-Lencep.

Un uso adicional para los diagramas de estructura compuesta es la posibilidad de reutilización. Por ejemplo, un elemento de colaboración a menudo implementa un patrón que se puede utilizar para mostrar las partes que colaboran, por ejemplo, en un caso de uso. Queda como trabajo futuro la definición de reglas para llegar a este elemento.

Referencias

[1] EA: Enterprise Architect (2008), Guía de Usuario de Enterprise Architect 7.0. Disponible: www.sparxsystems.com.ar/download/ayuda/index.html?compositestructurediagram.htm, [Consultado: 2 de noviembre de 2011].

[2] Fowler, M. (2003), UML Distilled, Addison-Wesley, 3a. ed.

[3] IBM: International Business Machines (2011). Rational Systems Developer Disponible: www.ibm.com/developerwork [Consultado el 2 de Noviembre de 2011].

[4] Ober, I., Dragomir, I. (2010), OMEGA2: A new version of the profile and the tools, Memorias del 15 Congreso Internacional IEEE sobre Ingeniería de Sistemas Informáticos Complejos, Oxford, pp. 373-378.

[5] Ober, I., Dragomir, I. (2011), Unambiguous UML Composite Structures: the OMEGA2 Experience, Lecture Notes in Computer Science, Vol. 6543, pp. 418-430.

[6] Oliver, I., Luukkala, V. (2006), On UML's Composite Structure Diagram. V Seminario sobre Análisis y Modelado de Sistemas (Workshop on System Analysis and Modelling), Kaiserslautern, Alemania, 31 de mayo-2 de junio.

[7] OMG Unified Modeling Language (2007), Superstructure, V2.1.2, OMG Available Specification without Change Bars.

[8] OMG: Object Management Group (2011). OMG Unified Modeling Language Specification. Disponible: http://www.omg.org/UML/. [Consultado: 2 de noviembre de 2011].

[9] Orozco, S. (2007), Componiendo lo descompuesto. Revista Software Gurú Conocimiento en práctica, No. 2, pp. 44- 45.

[10] Röhl, M. (2008), Definition and analysis of composition structures for discrete-event models, Memorias de la Conferencia de Simulación Invierno 2008, Miami, pp. 942-950.

[11] Sedillo, S. (2007), Desarrollo dirigido por modelos. Revista Software Gurú Conocimiento en práctica, No. 2, Abr., pp. 40- 42.

[12] 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, Revista Universidad EAFIT, Vol. 41, No. 137, pp. 77–95.

[13] Zapata, C.M y Arango, F. (2007), Un ambiente para la obtención automática de diagramas UML a partir de un lenguaje controlado. Revista Dyna, Año/Vol. 74, No. 153, pp. 223-236.

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

[15] Zapata, C.M., Gelbukh, A. y Arango, F. (2006), "UN-Lencep: Obtención automática de diagramas UML a partir de un lenguaje controlado," Avances en la Ciencia de la Computación, VII Encuentro Internacional de Computación ENC'06, ISBN 968-5733-06-6.