Desde esquemas preconceptuales hacia OO-Method

From pre-conceptual schemas to OO-Method

Carlos Mario Zapata Jaramillo,[1] Luz Marcela Ruiz Carmona[1], Oscar Pastor[2]*

[1]Grupo de Investigación en Lenguajes Computacionales, Escuela de Sistemas, Universidad Nacional de Colombia, sede Medellín, Carrera 80 N.° 65-223 Bloque M8A Oficina 310. Medellín, Colombia

[2]Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Camino de Vera s/n; 46022 Valencia, España

(Recibido el 23 de mayo de 2009. Aceptado el 18 de mayo de 2010)

Resumen

OO-Method (OOM), es un método basado en modelos orientados a objetos para la generación automática de código. Para la construcción de un producto de software empleando OOM, se debe partir desde los diagramas que propone este método. Esto requiere un gran esfuerzo por parte del analista quien, de manera subjetiva, debe elaborar dichos diagramas. Por otra parte, los Esquemas preconceptuales son diagramas previos a los esquemas conceptuales que permiten representar el conocimiento, presentan una sintaxis cercana a los lenguajes controlados y permiten obtener, automáticamente, diagramas de UML. En este artículo, aprovechando las ventajas anotadas para los Esquemas Preconceptuales, se propone un conjunto de reglas para la obtención de los diagramas de OOM a partir de dichos esquemas. Esta propuesta se ejemplifica con un caso de estudio.

----- *Palabras clave:* Esquemas preconceptuales, UML, OO-Method, lenguaje controlado, reglas de conversión

Abstract

OO-Method is an object-oriented method for automated code generation. In order to develop a software application by using OO-Method, the analyst must subjectively create the diagrams included in OO-Method. In this task the analyst must invest too much effort. On the other hand, Pre-conceptual Schemas are controlled-language-based diagrams for knowledge representation. Also, Pre-conceptual Schemas are previous diagrams to conceptual schemas and they are designed for automatically obtaining UML diagrams. In this paper,

* Autor de correspondencia: teléfono: + 57 + 4 + 425 53 50, fax: + 57 + 4 + 425 53 65, correo electrónico: cmzapata@unalmed.edu.co. (C.M. Zapata)

we take advantage of Pre-conceptual Schemas and we propose a set of rules for obtaining OO-Method diagrams from these Schemas. We also exemplify the proposal with a case study

----- *Keywords:* Pre-conceptual schemas, UML, OO-Method, controlled language, conversion rules

Introducción

OO-Method (*Object-Oriented Method* o, para abreviar, OOM) [1] es un método para especificar, mediante modelos orientados a objetos, el dominio de un sistema y así obtener, de manera automática, el código ejecutable de un producto de software. Los requisitos de los interesados se deben representar en un modelo conceptual, procurando que se haga en una fase temprana del desarrollo para asegurar la calidad del producto de software. Para obtener una aplicación empleando OOM, se debe partir de diagramas conceptuales orientados a objetos que siguen un estándar muy similar al de UML (*Unified Modeling Language*, el lenguaje más empleado en la actualidad para modelado de aplicaciones de software) [2]. Este lenguaje, UML, proporciona los elementos necesarios para modelar un sistema. Además, la representación gráfica de UML es precisa y finita, lo que permite el manejo adecuado de tales elementos para complementar los modelos necesarios del dominio del problema.

La captura e interpretación de los requisitos de los interesados para obtener un modelo conceptual requiere un gran esfuerzo por parte del analista, quien debe valerse de diferentes técnicas como lluvias de ideas, entrevistas, investigación histórica del sistema *etc.* [3]. Además, cuando el analista termina de realizar toda la fase de captura de información, por lo general, reduce su contacto con el interesado y debe representar todo el conocimiento adquirido, de forma subjetiva y sin una técnica establecida, en diagramas técnicos, como los que propone OOM. En este punto, el analista queda sin herramientas para poder validar la información adquirida en la fase de captura de requisitos, ya que los interesados no suelen comprender suficientemente los esquemas conceptuales, por ser un lenguaje técnico que se aleja del lenguaje natural.

Por otro lado, se tienen los Esquemas preconceptuales (EP) [4], los cuales son una representación gráfica del conocimiento, en una especie de lenguaje controlado, que puede ser un discurso sobre el sistema en construcción. En la lectura de los EP, se puede notar su cercanía con el lenguaje natural. Debido a esto, se puede obtener gráficamente un esquema intermedio entre los EP y los esquemas conceptuales de UML, que suministra un canal de comunicación entre el analista y el interesado, con la intención de validar los requisitos que el analista captura e interpreta. Los EP, se diseñaron de forma tal que, a partir de ellos, se pueden obtener 4 diagramas de UML 2.0 (Diagrama de Clases, Máquina de Estados, Diagrama de Comunicación y Diagrama de Casos de Uso) [4,5] y el Diagrama de Objetivos de KAOS (*Knowledge acquisition automated specification*, otro estándar de modelado pero basado en metas) [6].

Por lo general, es el analista quien, de manera subjetiva, elabora los diagramas de OOM a partir de descripciones en lenguaje natural o controlado [7-10], con los errores de interpretación o representación que ello acarrea. Ahora, debido a la similitud que existe entre los diagramas conceptuales de OOM y UML, en este artículo se parte de la hipótesis de que es posible generar los diagramas conceptuales de OOM (Clases, Transición de Estados e Interacción de Objetos), mediante un conjunto de Reglas de Transformación, desde los EP.

Este artículo posee la siguiente estructura: en la sección 2, se presenta el marco teórico, que sirve de base al método de transformación que se pretende definir; en la sección 3, se muestran

los antecedentes y usos de los EP y de OOM; en la sección 4, se presenta la propuesta de solución, donde se especifica la definición de las Reglas de Transformación; en la sección 5, se ejemplifican las Reglas de Transformación por medio de un caso de estudio; en la sección 6 se presentan la conclusiones y, finalmente, en la sección 7, se muestra el trabajo futuro que se deriva de la presentación de las Reglas de Transformación.

Object-Oriented Method (OOM)

OOM [1] es un método Orientado a Objetos para la producción automática de software que se basa, principalmente, en ofrecer nociones de modelado que soporten modelos conceptuales Orientados a Objetos. Debido a esto, OOM propone la realización de diagramas que envuelvan la parte dinámica y estática, para así obtener un esquema conceptual que contenga la representación correspondiente a un producto de software completo.

OOM propone la realización del Modelo Objetual (conformado por el Diagrama de Clases) y del Modelo Funcional (conformado por el Diagrama de Transición de Estados y el Diagrama de Interacción de Objetos). El Diagrama de Clases, que representa la estructura estática del sistema, se conforma con los elementos que se pueden apreciar en la figura 1 y que se describen seguidamente.

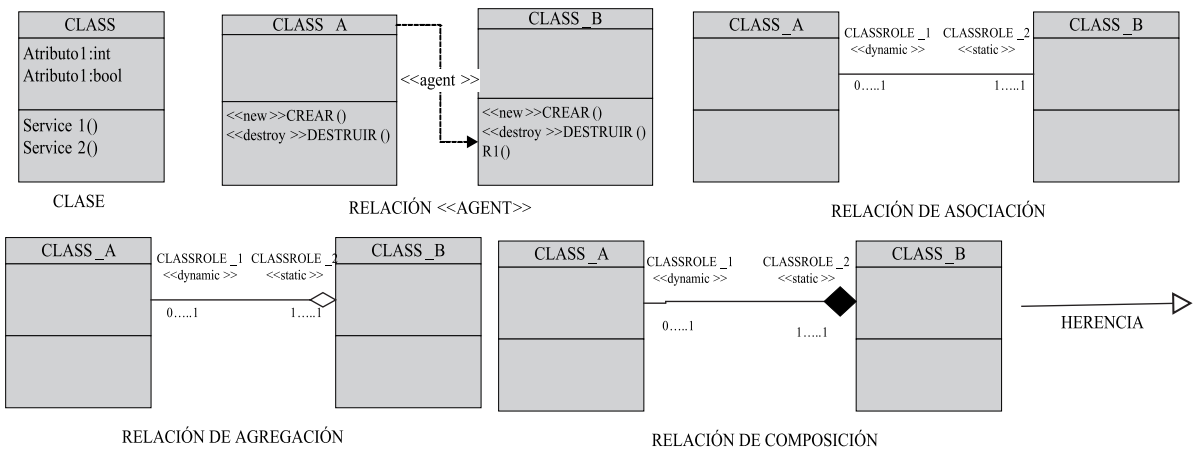


Figura 1 Simbología del Diagrama de Clases de OO-Method

Clase: Una clase contiene: un encabezado o nombre de la clase, que se ubica en el primer cajón de la clase; un conjunto de atributos, ubicados en el segundo cajón de la clase, incluyendo el tipo de dato y el indicador <<id>> (que establece si dicho atributo representa la clave primaria de dicha clase); por último, en el tercer cajón de la clase, se ubican los servicios, los cuales representan las acciones que se pueden ejecutar sobre las instancias de dicha clase y que pueden tener argumentos. Toda clase, en OOM, siempre debe tener el servicio de <<new>> y <<destroy>> los cuales indican la creación y eliminación de objetos pertenecientes a la clase.

Relaciones entre clases: Permiten mostrar las interacciones que se llevan a cabo entre los objetos del sistema.

Relación <<agent>>: Muestra qué objetos activan servicios que otras clases ofrecen.

Relación de Asociación: Muestra el vínculo que puede existir entre un par de objetos de un sistema e incluye la cardinalidad, la representación de roles y la temporalidad, la cual indica si la relación es estática (sólo para una instancia de la clase) o dinámica (para cualquier instancia de la clase).

Relación de Agregación: Existe entre un par de objetos del sistema, donde las acciones de un objeto son parte de las acciones de otro objeto; la representación gráfica es un diamante en blanco, el cual toca la clase que representa el objeto componente.

Relación de composición: Es un caso especial de la relación de Agregación, donde la relación que existe entre ambos objetos relacionados es una relación "FUERTE". Así, cada instancia del objeto compuesto debe existir. Gráficamente, se representa con un diamante negro.

Relación de herencia: se representa la derivación que se puede dar entre diferentes clases, es decir, la relación padre-hijo entre objetos de un sistema. Gráficamente se representa con una flecha que apunta a la clase padre.

El Modelo Dinámico, representa la parte dinámica de un determinado sistema (relaciones entre objetos, comunicación entre objetos) y se basa del Modelo Estático. Por lo tanto, éste se debe construir cuando se cuente con el modelo dinámico.

Diagrama de Transición de Estados: Los elementos que se pueden apreciar en la figura 2 conforman este diagrama y se detallan seguidamente.

Figura 2 Simbología del diagrama de Transición de estados de OO-Method

Estado: El cual puede ser de los siguientes tipos:

Estado de precreación: Representa la creación o generación de un determinado objeto.

Estado de destrucción: Representa la finalización del ciclo de vida de un determinado objeto.

Estado simple: Representa las transiciones posibles que se pueden dar en un determinado objeto, mostrando la activación de los servicios.

Transiciones: Representan el cambio de estado de un objeto y muestran cómo se cambia de una situación a otra debido a la activación de un servicio. En la transición, se puede dar a conocer qué tipos de objetos pueden activar ciertos servicios.

Diagrama de Interacción de Objetos: Con él, se busca mantener un control de la comunicación y la interacción del sistema. Su simbología, se puede apreciar en la figura 3.

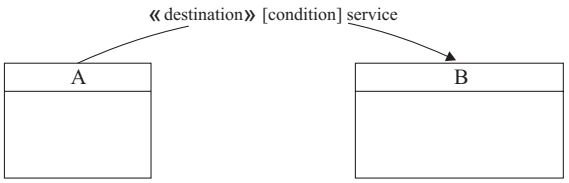


Figura 3 Simbología del diagrama de Interacción de Objetos de OO-Method

<<Destination>> se refiere a la entidad que impulsa la realización del servicio. Acá se indica quiénes pueden ejecutar dicha condición: "Self", cuando el servicio lo ejecuta el objeto que dispara la condición, "Object", cuando un solo objeto ejecuta el servicio y cumple la condición, "Class", cuando todas las instancias de la clase ejecutan el servicio, sin necesidad de que la clase active la condición y "For All", cuando un conjunto de instancias de la clase ejecuta el servicio.

<condition> es una fórmula bien formada cuyo resultado puede ser falso o verdadero, la cual envuelve constantes, funciones y atributos de la definición de la clase. Cada instancia de la definición de la clase que cumpla con la fórmula bien formada, puede activar la condición.

<service> es la consecuencia que se debe llevar a cabo como resultado del cumplimiento de la condición.

Esquemas preconceptuales

Los Esquemas Preconceptuales (EP) [4] son representaciones gráficas que permiten expresar un discurso, lluvia de ideas, modelo verbal y demás ideas que el analista interprete a partir de la información que obtiene del interesado, en relación con el dominio en el cual se envuelve el sistema que se desea modelar.

La simbología de los EP le permite al lector, al momento de su lectura, obtener fácilmente el discurso que conforma el sistema. Además, la simbología es sencilla y la lectura de los EP se puede realizar iniciando en cualquier nodo del mismo. La simbología de los EP se muestra en la figura 4. Los elementos se detallan seguidamente:

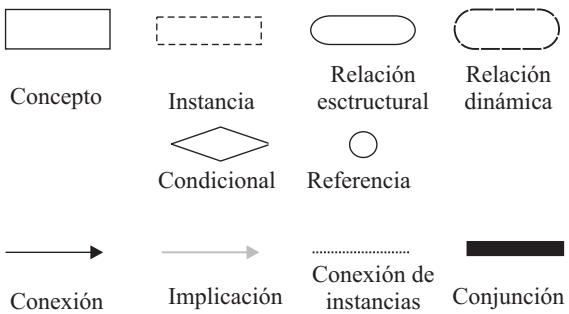


Figura 4 Simbología del Esquema Preconceptual

Concepto: Representa un sustantivo o frase nominal.

Instancia: Indica un posible valor de un concepto.

Relación Estructural: Representa el tipo de relación que es constante en el tiempo; de las relaciones estructurales existentes, se emplean comúnmente: "ES" y "TIENE".

Relación Dinámica: Son verbos de acción que tienen una durabilidad fija en el tiempo, por ejemplo "corre", "juega", *etc.*

Condicional: Expresa condiciones o valores que deben tener los conceptos para que se pueda llevar a cabo una relación.

Referencia: Es un vínculo que ayuda a unir los elementos del EP, especialmente cuando éste toma un tamaño considerable y los elementos que se desean relacionar están muy distantes.

Conexión: Permite la unión de los diferentes elementos del EP.

Implicación: Define una relación causa-efecto entre relaciones dinámicas o entre condicionales y relaciones dinámicas.

Conexión de Instancias: Línea que une un concepto con sus valores de instancias.

Conjunción: Representa el "Y" y el "O" lógico que se puede presentar especialmente en las relaciones dinámicas.

A partir de los Esquemas Preconceptuales se pueden obtener 4 diagramas de UML 2.0 (Clases, Comunicación, Máquina de Estados y Casos de Uso), los cuales representan la estructura, comportamiento e interacción de una aplicación de software [4,5]. Además, se puede generar el diagrama de objetivos que se presenta en la metodología KAOS. Esta última característica abre un campo de acción interesante para los esquemas preconceptuales, pues se podría ligar el discurso del interesado con las metas y objetivos de alto nivel de la organización.

El uso de OO-Method para modelado de sistemas

En la literatura especializada, OOM se suele emplear para la generación automática de código ejecutable a partir de modelos, ya sea en aplicaciones convencionales de software [7-10] o aplicaciones web [11]. Otro uso identificado de OOM, es el cálculo de puntos de función a partir de sus modelos [12]. OOM, sirve de base para el entorno *OlivaNova Model Execution* (ONME), que comercializa el desarrollo de aplicaciones de software a nivel mundial a partir de modelos [13].

En cualquiera de los usos identificados, se arguye que los esfuerzos en desarrollo de software se deben concentrar en la etapa de modelado conceptual en la que, a partir de los requisitos capturados de las entrevistas con los interesados, es el analista quien debe construir los diagramas correspondientes. Los procesos convencionales de modelado requieren conocimientos técnicos especializados que suelen tener los analistas, pero en los cuales la validación se dificulta pues, como afirman Haumer [14], "los modelos conceptuales son difíciles de entender para alguien que no se involucra en su proceso de definición".

Si bien, los "modelos ejecutables" que plantea OOM constituyen una evolución en el proceso de desarrollo, pues el analista puede dejar los detalles de implementación en manos del convertidor

a código desde los diagramas de OOM, aún no se puede garantizar que los interesados puedan validar los diagramas de OOM. De allí, la propuesta de este artículo para representar dichos diagramas en esquemas preconceptuales, que poseen un equivalente en lenguaje controlado que el interesado puede fácilmente entender y validar. Así, se podría representar el discurso del interesado en EP y, con las reglas de conversión adecuadas, obtener los diagramas de OOM.

Metodología

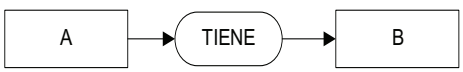
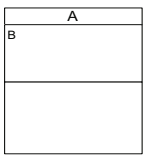
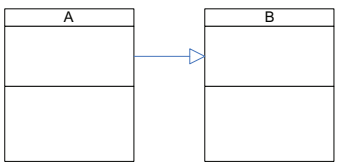
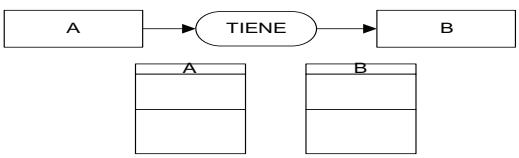
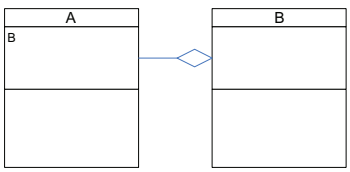
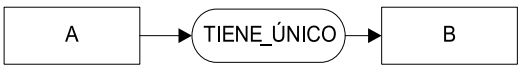
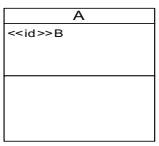
Propuesta de solución: uso de EP para generar los diagramas de OOM

Se proponen, en este artículo, 12 reglas de transformación de EP a OOM, consignadas en la tabla 1. La relación estructural TIENE_ÚNICO, incluida en la regla 4, no es una relación nativa de los esquemas preconceptuales y se define en esta propuesta para representar los identificadores únicos de las clases, información necesaria para el diagrama de clases de OOM. Nótese, adicionalmente, que las reglas emplean de manera recursiva los resultados, pues algunas reglas incluyen no sólo esquemas preconceptuales, sino diagramas de OOM.

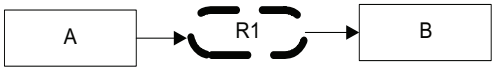
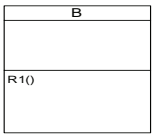
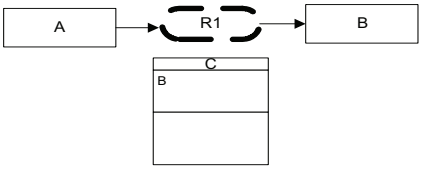
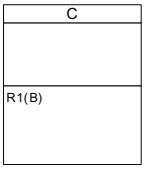
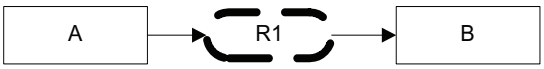
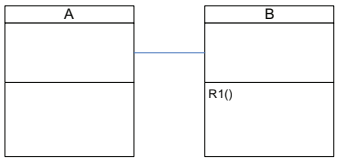
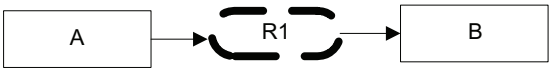
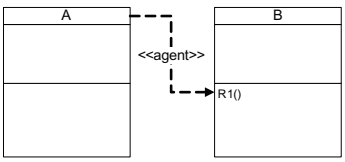
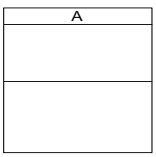
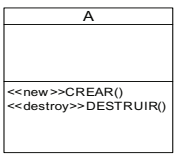
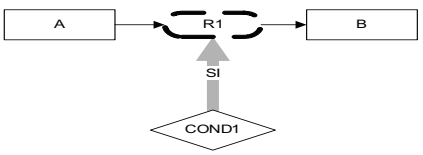
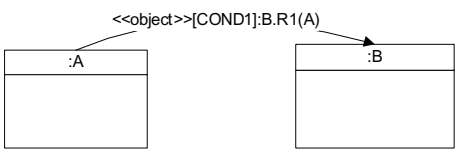
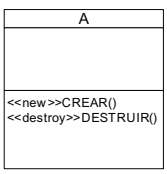
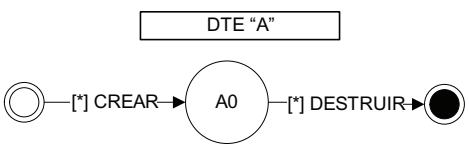
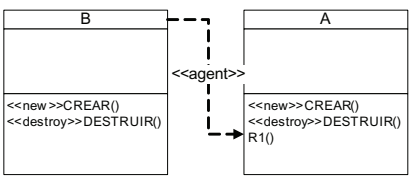
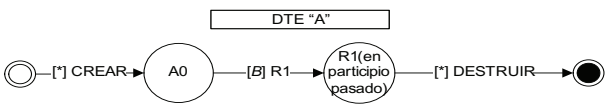
Estudio de caso: guardería de mascotas

En la figura 5, se presenta el EP correspondiente a una Guardería de Mascotas. Como se establece en [4], existen equivalentes textuales para cualquier EP, que el interesado está en capacidad de validar. Para el ejemplo, algunas de esas frases, expresadas en el lenguaje controlado UN-Lencep, son: "cuando el veterinario entrega la mascota, el cliente recoge la mascota", "un cliente tiene un único nombre", "un spa es un servicio", *etc.*

Tabla 1 Reglas de transformación de EP en diagramas de OOM

No	*Precondición*	*Resultado*
1		
2		
3		
4		

Cont. Tabla 1

No	Precondición	Resultado
5		
6		
7		
8		
9		
10		
11		
12		

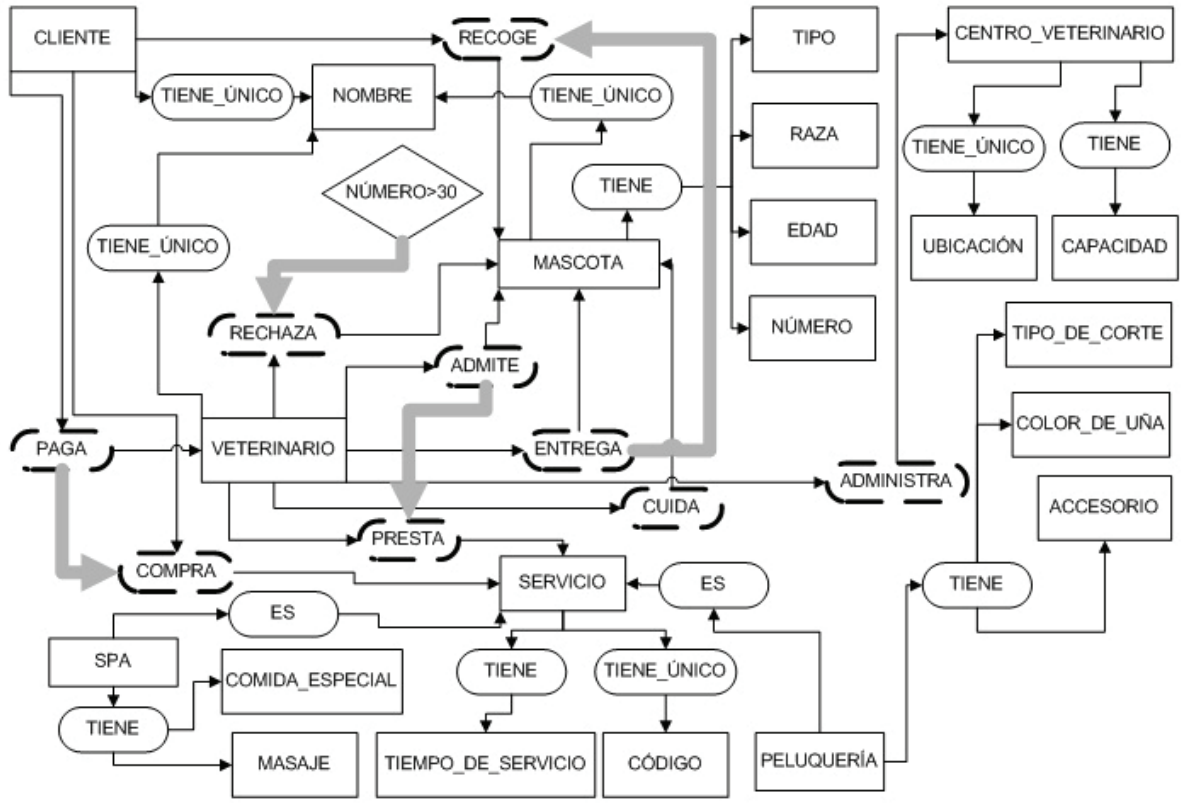


Figura 5 Esquema Preconceptual correspondiente a una guardería de mascotas

Mediante la aplicación sistemática de las reglas de la tabla 1 sobre el EP de la figura 5, se obtienen los diagramas de OOM que se muestran en las figuras 6, 7 y 8. Por ejemplo, para el Diagrama de interacción de objetos de la figura 6, se aplica la regla 1 para determinar que veterinario y mascota son clases y, posteriormente, se aplica la regla 10 que construye la interacción entre los objetos definidos.

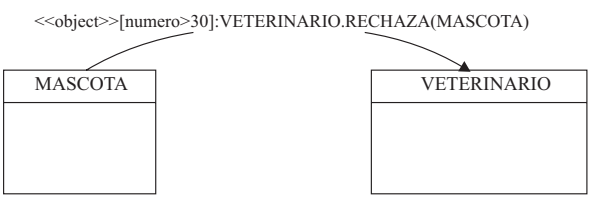


Figura 6 Diagrama de Interacción de objetos obtenido a partir del EP de la figura 5

Para el Diagrama de clases de la figura 7, la relación estructural especial "TIENE_ÚNICO", que se define en este artículo como un elemento adicional a la sintaxis básica de los Esquemas Preconceptuales, se emplea para determinar los identificadores de las diferentes clases, que se marcan con el símbolo <<id>>, empleando la regla 4.

Finalmente, para los diferentes Diagramas de transición de estados de la figura 8, se emplean las reglas 11 y 12.

Es necesario aclarar que algunos de los elementos de los diagramas de OOM no se pudieron generar a partir de los elementos de los EP. Por ejemplo, los tipos de datos de los atributos, las cardinalidades de las relaciones y los parámetros correspondientes a las operaciones del diagrama de clases no poseen equivalencias en los elementos existentes de los Esquemas Preconceptuales. Pese a ello, se obtienen diagramas completos y, sobre todo, consistentes unos con otros, una característica

deseable cuando se elaboran diagramas a partir de un mismo discurso.

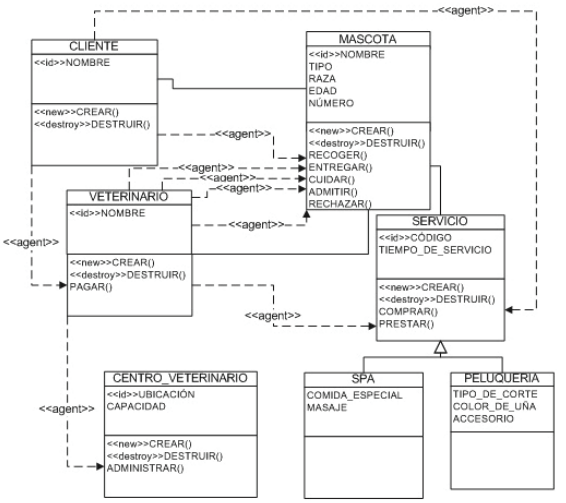


Figura 7 Diagrama de Clases obtenido a partir del EP de la figura 5

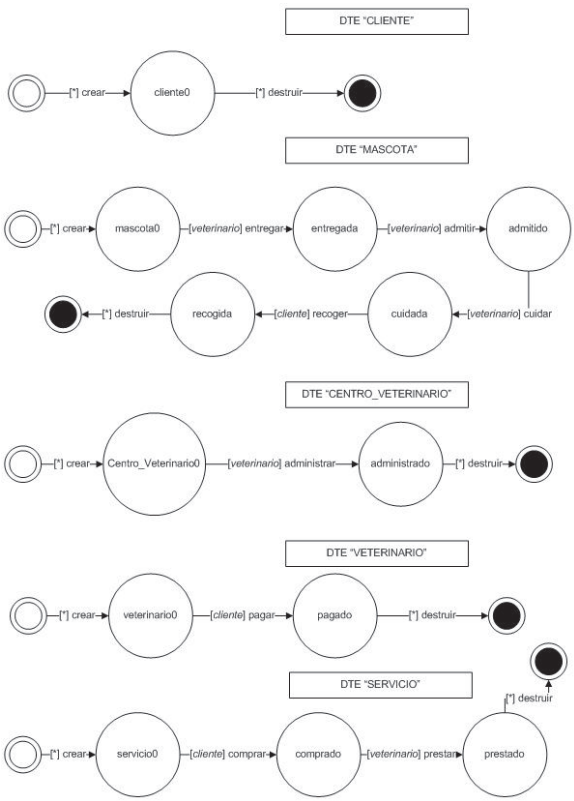


Figura 8 Diagrama de Transición de Estados obtenido a partir del EP de la figura 5

Conclusiones

Las tendencias en el desarrollo de software apuntan, cada vez más, hacia la automatización de los diferentes procesos, buscando reflejar los requisitos de los interesados, capturados en las fases tempranas del desarrollo, en la aplicación implementada. En este artículo se fusionaron dos tendencias que, aisladamente, persiguen este objetivo; por un lado, OO-Method, una propuesta para el desarrollo automático de aplicaciones ejecutables a partir de esquemas conceptuales; por otro lado, los Esquemas Preconceptuales, diagramas que procuran acercar el discurso del interesado, generalmente expresado en lenguaje natural, a los esquemas conceptuales, escritos en el lenguaje técnico de los analistas. La definición de las reglas de transformación de EP a los diagramas de OOM, acerca los lenguajes controlados a la generación automática de código, garantizando en el proceso la consistencia que, por lo general, se dificulta cuando el proceso se realiza de forma manual.

Como se pudo deducir a partir del caso de estudio mostrado, si bien los diagramas de OOM que resultan del discurso del interesado expresados en EP no son lo suficientemente completos (pues carecen de elementos como las cardinalidades y los tipos de datos de los atributos), sí reflejan la información inicial obtenida en el proceso de captura de requisitos. El interesado, adicionalmente, puede complementar y validar dicha información, puesto que la sintaxis de los EP cumple la doble función de acercarse al lenguaje natural y servir de nexo hacia los esquemas conceptuales.

Trabajo futuro

Debido a que los Esquemas Preconceptuales no tienen definida una propuesta para representar la cardinalidad entre las clases de un Diagrama de Clases, se considera importante realizar un estudio que permita definirla, presumiblemente a partir de los cuantificadores y artículos definidos e indefinidos. Una vez se disponga de los elementos correspondientes, se podrían deducir

las reglas de transformación a diagramas de UML y OOM, a partir de los EP. En esta propuesta de solución tampoco se incluyeron los tipos de datos, los cuales hacen parte de los diagramas de OOM; es necesario incluir dicha funcionalidad en los Esquemas Preconceptuales, ya sea empleando una ontología de pares {palabra, tipoDato} o buscando la forma de representar el tipo de dato en el Esquema Preconceptual de forma gráfica.

Un trabajo adicional por realizar es la incorporación del entorno para editar los EP y las reglas de transformación de EP a diagramas de OOM en las herramientas que soportan la generación de código a partir de dichos diagramas, como es el caso de *OlivaNova Model Execution*.

Finalmente, cualquier iniciativa que se realice para acercar cada vez más los Esquemas Preconceptuales a lenguaje natural redundará, consecuentemente, en el acercamiento de los modelos de OO-Method al lenguaje natural. Un trabajo tal se acercaría paulatinamente al objetivo de generación automática de código a partir de lenguaje natural.

Referencias

1. O. Pastor, J Molina. *Model-Driven Architecture in Practice*. Ed. Springer-Verlag. Berlin. 2007. pp. 49-146.
2. OMG. OMG *Unified Modeling Language Specification*. Object Management Group. Available: http://www.omg.org/UML/. Consultada el 20 de Mayo de 2009.
3. R. Pressman. *Software Engineering: a practitioner's approach*. Ed. McGraw Hill. New York. 2005. pp. 274-281.
4. C. M. Zapata, A. Gelbukh, F. Arango. "Pre-conceptual Schema: A Conceptual-Graph-Like Knowledge Representation for Requirements Elicitation". *Lecture Notes in Computer Science*. Vol. 4293. 2006. pp.17-27.
5. C. M. Zapata, P. A. Tamayo, F. Arango. "Conversión de Esquemas Preconceptuales a diagrama de casos de uso empleando AToM3". *Revista DYNA*. Vol. 74. 2007. pp. 237-251.
6. C. M. Zapata, L. Lezcano, P. Tamayo. "Validación del Método para la Obtención Automática del Diagrama de Objetivos desde Esquemas Preconceptuales". *Revista EIA*. Vol. 8. 2007. pp. 21-35.
7. O.Pastor, J. Gómez, E. Insfrán, V. Pelechano. "The OO-Method Approach for Information Systems Modeling: From Object-Oriented Conceptual Modeling to Automated Programming". *Information Systems.* Vol. 26 .2001. pp. 507-534.
8. V. Pelechano, O. Pastor, E. Insfrán. "Automated code generation of dynamic specializations: an approach based on design patterns and formal techniques*". Data & Knowledge Engineering*. Vol. 40. 2002. pp. 315-353.
9. O. Pastor, E. Insfrán, V. Pelechano, J. Romero, J. Merseguer. "OO-METHOD: An OO Software Production Environment Combining Conventional and Formal Methods". *Lecture Notes in Computer Science*. Vol. 1250. 1997. pp. 145-158.
10. P. Molina. "User Interface Generation with OlivaNova Model Execution System". *Proceedings of the 9th International conference on Intelligent user interfaces.* Madeira. 2004. pp. 358-359.
11. O. Pastor, S. Abrahao, J. Fons. "Building e-commerce applications from object-oriented conceptual models". *ACM SIGecom Exchanges*. Vol. 2. 2001. pp. 28-36.
12. G. Giachettii, B. Marim, N. Condori-Fernandezi, J. C. Molina. "Updating OO-Method Function Points". *Proceedings of the 6th International Conference on Quality of Information and Communications Technology*. Lisboa. 2007. pp. 55-64.
13. CARE Tecnologies: "OlivaNova: The programming machine". Available: http://www.care-t.com/index.asp. Consultada el 20 de Mayo de 2009.
14. P. Haumer, M. Jarke, K. Pohl, K. Weidenhaupt. "Improving reviews of conceptual models by extended traceability to captured system usage". *Interacting with Computers*. Vol. 13. 2000. pp. 77-95.