Esquemas preconceptuales ejecutables

Executable pre-conceptual schemas

Carlos Mario Zapata J,[1] Ph.D Gloria Lucía Giraldo G.[2] Ph.D & Santiago Londoño V.[3] Est.
1. Líder del grupo en Lenguajes Computacionales
2. Profesora Asociada
3. Est. Ing. de Sistemas e Informática
cmzapata@unal.edu.co; glgiraldog@unal.edu.co; slondonov@unal.edu.co

Recibido para revisión 29 de julio de 2010, aceptado 03 de enero de 2011, versión final 09 de febrero de 2011

***Resumen*— Los esquemas preconceptuales son herramientas que se emplean para representar gráfica y computacionalmente el dominio de una situación o problema. Se concibieron de modo tal que personas de cualquier tipo pueden comprender la información que describen. Sin embargo, esa información no es completamente transparente para el interesado, pues tiene el reflejo de la estructura del problema que se quiere solucionar, pero no la funcionalidad del mismo. En otras palabras, el esquema preconceptual, que es el esqueleto del software, no presenta la especificación de las diferentes operaciones que se realizan dentro del dominio, pues simplemente se nombra la interacción de los actores con los objetos. Por ello, en este artículo se propone un conjunto de consideraciones que permiten realizar la edición de esquemas preconceptuales para la visualización de instancias del sistema y el comportamiento de las operaciones que allí aparecen, estableciendo la capacidad de ejecutar dichos esquemas. Este trabajo emplea analogías con Charger, una herramienta para la animación de grafos conceptuales, y con el diagrama de objetos de UML, buscando las similitudes que permitan hacer ejecutables los esquemas preconceptuales.**

***Palabras Clave*— Diseño de Bases de Datos; Normalización de Bases de Datos Relacionales; Sistema Informático.**

***Abstract*—Pre-conceptual schemas are tools used for graphically and computationally representing the domain of a problem. They were conceived in such a way that everyone can understand the information they describe. Nevertheless, such information is not completely clear to the stakeholder, because it reflexes the problem structure, instead of its functionality. In other words, a pre-conceptual schema, which is the software "backbone", does not specify in detail the different operations or procedures carried out within the domain, because it only refers to the interaction among participants and objects. By this reason, in this paper we propose a set of considerations in order to allow editing pre-conceptual schemas, so that they can depict system instances and operational behavior. Our aim is to finally establish the capacity to execute such schemas. In this work, we use analogies with Charger (a tool for animating conceptual graphs) and the UML object diagram, trying to gather similarities with them, useful for making executable the pre-conceptual schemas.**

***Keywords*— Pre-conceptual Schemas, Object Diagram, Conceptual Graphs, Instances, Operations.**

I. INTRODUCCIÓN

Un esquema preconceptual es una representación de un dominio específico y emplea una simbología que los conocedores de ese dominio pueden validar [1]. También, es un diagrama que permite comunicación entre personas con conocimientos técnicos en modelado y los conocedores del dominio que se está modelando, es decir, los interesados.

Los esquemas conceptuales se suelen utilizar para el modelado de los diferentes dominios de aplicación [2], con el fin de elaborar posteriormente una aplicación de software a partir de ellos. Uno de los principales lenguajes para la elaboración de esquemas conceptuales es el Lenguaje Unificado de Modelado (UML por sus siglas en inglés), el cual es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. Si bien desde hace tiempo existen propuestas para generar esquemas conceptuales desde subconjuntos del lenguaje natural [3], el surgimiento de UML motivó la aparición de muchos trabajos que buscan obtener diagramas UML de manera automática o semiautomática a partir de diferentes fuentes.

En un proceso de desarrollo de software, una de las tareas más relevantes para el éxito de un proyecto de diseño e implementación de un sistema informático, es garantizar, de una forma apropiada, la representación y el modelado de los requisitos del usuario. Esta labor se realiza con una serie de diagramas UML que luego el interesado debe validar, siendo éste

Revista Avances en Sistemas e Informática, Vol.7 No.3, diciembre de 2010 - Medellín. ISSN 1657-7663

el que determina la conformidad con lo se plantea y propone nuevas directrices para la mejora de lo que será el producto final [4]. Entre los diagramas de UML se encuentra el diagrama de objetos, el cual se usa para modelar las instancias de las clases contenidas en los diagramas de clases. Un diagrama de objetos muestra un conjunto de objetos y sus relaciones en un momento concreto [5]. Los diagramas de objetos se emplean para modelar la vista de diseño de un sistema, al igual que se hace con los diagramas de clases, pero desde la perspectiva de instancias reales o prototípicas. El diagrama de objetos es una herramienta para probar y entender un problema al documentar ejemplos del dominio. También, se usa durante el análisis y el diseño para verificar la exactitud de un diagrama de clases [6]. La utilidad del diagrama de objetos se basa en el hecho de que las instancias se pueden ejemplificar, lo que obliga al analista a preguntar por ejemplos concretos de las diferentes clases que conforman el sistema. Sin embargo, la notación de los diagramas de objetos aún se orienta más al analista que al interesado, por lo cual no se suelen validar con facilidad estos diagramas durante el proceso de desarrollo de una aplicación.

Siguiendo una filosofía similar a la que emplea el diagrama de objetos, pero de forma interactiva, dos herramientas se emplean para "animar" las especificaciones incluidas en algunos modelos: MAST [7] y Charger [8]. La primera se emplea para "predecir" el comportamiento de sistemas de tiempo real, con base en las entradas que se pueden realizar para sus cálculos. La segunda se encarga de realizar operaciones que se encuentran definidas en los grafos conceptuales, que son diagramas que ayudan en la representación del conocimiento en cualquier área. Estas simulaciones son específicas del dominio de los sistemas en tiempo real, en el primer caso, y de la sintaxis propia de los grafos conceptuales, en el segundo caso. Ninguna de ellas sirve para el desarrollo de aplicaciones de software en general, como sí es el caso de los esquemas preconceptuales, los cuales, sin embargo, aún no permiten la "animación" de sus especificaciones.

Por las razones anteriores, el objetivo de este artículo es proponer un conjunto de consideraciones que permitan identificar las operaciones y hacer interactivo y ágil el proceso de diseño de un producto de software partiendo de los esquemas preconceptuales. Se pretende no sólo mostrar el "esqueleto" de una aplicación, sino la funcionalidad del mundo que se está modelando. Así, la ejemplificación que se puede realizar de los conceptos en los esquemas preconceptuales y la descripción de las operaciones son los fundamentos que posibilitan el surgimiento de los esquemas preconceptuales ejecutables.

Este artículo se organiza de la siguiente manera: en la sección 2 se presenta el marco teórico que incluye una descripción de los principales elementos que componen los esquemas preconceptuales y los diagramas de objetos UML; en la sección 3 se presentan los antecedentes, MAST y Charger, resaltando sus principales aportes y desventajas; en la sección 4 se presentan generalidades para la implementación de instancias y su aplicación en los esquemas preconceptuales; en la sección 5 se presenta el caso de estudio; finalmente, en la sección 6 se presentan las principales conclusiones obtenidas y el trabajo futuro que se deriva de esta propuesta.

II. MARCO TEÓRICO

2.1 Elementos del esquema preconceptual

En el contexto de la descripción del mundo que hace el interesado, durante el desarrollo de una aplicación, los esquemas preconceptuales juegan un papel importante, pues utilizan una notación que los interesados pueden comprender y validar (véase la Figura 1), además de permitir la expresión de los diferentes elementos del discurso del interesado [9].

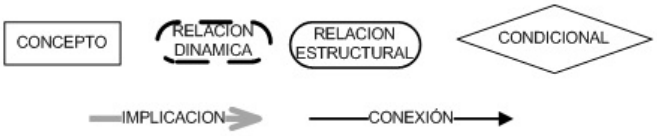

Figura 1. Simbología Básica de los Esquemas Preconceptuales

Los esquemas preconceptuales se conciben como una especificación semiformal que cualquier persona puede usar y entender [10], algo que es importante dentro de la relación analista-interesado para lograr un acuerdo en relación con lo que se quiere solucionar. Pese a que estos esquemas permiten expresar, en un lenguaje controlado, los conceptos, relaciones y restricciones que pueden estar presentes en el mundo del interesado, todavía no es posible mostrar de una manera transparente la funcionalidad de las diferentes operaciones que se ejecutan dentro del mundo que se está modelando (y que se representan con las relaciones dinámicas). A partir de esto, el interesado tan sólo responde a lo que el analista pregunte, logrando describir roles, objetos, restricciones, condiciones, implicaciones y actividades, entre otras, dentro del mundo que se está modelando, y permitiendo definir, específicamente, las diferentes representaciones de las acciones que allí se presentan. [11]. Esta etapa de validación con el interesado, es de vital importancia para el avance de una aplicación de software.

2.2 Elementos del diagrama de objetos

Los diagramas de objetos de UML se utilizan durante el proceso de Análisis y Diseño de los sistemas informáticos [6]. En general, un diagrama de objetos es como una ejemplificación de los elementos de una clase en un momento dado, lo cual permite visualizar, especificar, construir y documentar la existencia de ciertas instancias y las relaciones entre ellas.

El diagrama de objetos se puede considerar un caso especial del diagrama de clases, en el que se muestran los valores específicos que toman los atributos de las clases en un momento

particular y utilizan para su representación un subconjunto de los elementos de los diagramas de clases. Una diferencia con los diagramas de clases es que en el compartimiento superior va el nombre del objeto con la siguiente notación: "Nombre de objeto: Nombre de clase". Esta notación se muestra en la Figura 2.

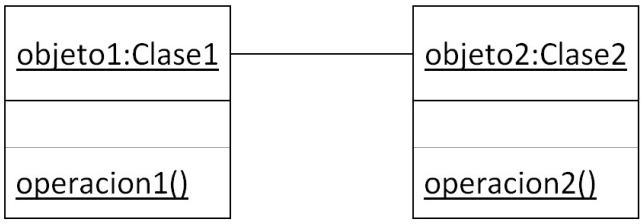

Figura 2. Uso de los campos del diagrama de objetos
También, se pueden obviar los compartimientos donde van los atributos y las operaciones (véase la Figura 3)

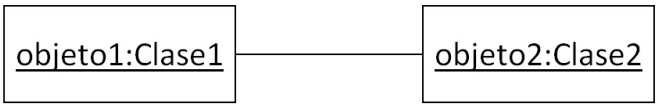

Figura 3. Caso de diagrama de objetos sin atributos ni operaciones

III. ANTECEDENTES

Desde el punto de vista de la calidad de una aplicación de software en desarrollo, varios trabajos señalan la importancia de la adecuada educción de las características dinámicas de las aplicaciones. Por ejemplo, en [12] se presenta una suite complementaria de medidas que se concentra en la complejidad de un esquema conceptual visto desde una perspectiva dinámica. Asimismo, el trabajo presentado en [13] señala que la definición de métricas para diagramas que capturen los aspectos dinámicos del software Orientado a Objetos es un área relevante, pero un tanto descuidada en el ámbito de la medición del software. En [14] y [15] se hace mayor énfasis en el análisis y descomposición de las tareas en subtareas de menor dificultad, que es la forma habitual en que los seres humanos resuelven los problemas una vez se define el modelo que los explica. Estos proyectos coinciden en señalar las dificultades que se presentan a la hora de comprender la parte dinámica de las aplicaciones. Si bien el proceso de descomposición ayuda (lo que sustenta la utilización del diagrama de objetos para reducir la complejidad), se requieren otras estrategias como las que se describen en esta sección y que dan fundamento al trabajo que se propone en este artículo sobre los esquemas preconceptuales ejecutables.

En el análisis de la animación de las especificaciones de una aplicación, un primer antecedente lo constituye el diagrama de objetos, pues representa una imagen estática del valor de los atributos de las diferentes clases en un momento dado [16]. Sin embargo, para el uso de este diagrama en la animación de especificaciones, existen dos limitaciones fundamentales: la sintaxis del diagrama se define casi exclusivamente para los analistas (dificultando la labor de validación que podrían realizar los interesados) y no existen herramientas que puedan tomar los valores de los atributos para generar resultados completos de operaciones definidas en el diagrama de objetos.

Una comunidad académica dedicada al estudio de los sistemas en tiempo real en España [7] creó MAST (Modeling and Analysis Suite for Real-Time Applications) como una forma de evaluar nuevos algoritmos de análisis de planificación y técnicas de diseño de sistemas en tiempo real. La metodología que se propone, denominada UML-MAST, acuerda las diferencias entre la visión del diseñador para sistemas en tiempo real y para sistemas orientados a objetos. Así, se define un nivel de abstracción adecuado para los elementos de modelado del comportamiento en tiempo real, que permite formularlos con una estructura paralela a la arquitectura lógica del sistema y vincularlos con ésta. UML-MAST está aún en fase experimental, tanto por depuración como porque parte de las herramientas de análisis que se procuran integrar están en fase de desarrollo. Sin embargo, esta herramienta operativa del modelado permite analizar los aspectos más relevantes de un sistema en tiempo real. Su principal desventaja es el ámbito reducido para el cual se aplica la solución, pues se trata específicamente de sistemas en tiempo real. Ello implica un conocimiento previo amplio para poder definir los eventos de este tipo de sistemas de forma detallada, antes de ejemplificar el funcionamiento de cada uno con la aplicación. Esto, sin embargo, no se podría hacer para el desarrollo de software en general, puesto que las temáticas podrían variar sustancialmente entre aplicación y aplicación.

Otra opción de animación de especificaciones la constituye el Charger, que es una aplicación para editar grafos conceptuales, posibilitando su traducción a diferentes lenguajes de representación del conocimiento [11]. Dentro de los procesos de edición de dichos grafos, el Charger emplea algunos símbolos asociados con operaciones, con una semántica definida, y con los cuales se puede realizar desde operaciones algebraicas sencillas hasta ciertas lecturas a tablas que podrían representar bases de datos. El Charger soporta operaciones, generalmente, usando las técnicas descritas en el libro original de Sowa [17], entre las que se cuentan las siguientes:

Copy, dbfind, lookup, divide, displaybar, equal, exp, greaterequal, greaterthan, lessequal, lessthan, minus, multiply, notequal, time, pulse, stokequote y plus. Casi todas son de orden lógico y aritmético. La simbología de conceptos y operaciones que ofrece Charger se observa en la figura 4:

Figura 4: Simbología de conceptos y funciones del Charger.

Revista Avances en Sistemas e Informática, Vol.7 No.3, diciembre de 2010 - Medellín. ISSN 1657-7663

Un grafo en el ambiente de Charger es cualquier colección de conceptos, relaciones, actores y etiquetas de tipo, unidas con flechas apropiadas, relaciones de coreferente, flechas de entrada/salida o flechas de super/subtipo. El Charger, igualmente, facilita la construcción de jerarquías de tipos y su almacenamiento como parte de otro grafo. En el caso de Charger, la principal desventaja la constituye su lejanía con la notación estándar de UML, pues no existen equivalencias que puedan conducir a los diagramas de UML, como sí se puede hacer a partir de los esquemas preconceptuales.

IV. ESQUEMAS PRECONCEPTUALES EJECUTABLES: IDENTIFICACIÓN E IMPLEMENTACIÓN DE OPERACIONES DESDE LA ESPECIFICACIÓN

En este numeral se presenta, como parte de la solución propuesta, una nueva terminología que hace uso de información basada en ejemplos, para incorporarla en los esquemas preconceptuales. Esta solución se apoya en algunas características del diagrama de objetos de UML (que son comunes a la representación de los valores en los grafos de Sowa) y en las consideraciones que se definen más adelante. La idea es familiarizar al interesado con lo que sería un boceto de su aplicación, permitiéndole ver el estado actual y las condiciones que se cumplen para realizar una acción (representada en los esquemas preconceptuales por medio de una relación dinámica), tomando como base los valores de los atributos que se ligan con cada concepto.

El proceso de educción de requisitos presenta, entre otros, los siguientes inconvenientes:

- No hay una comunicación efectiva entre los analistas y los interesados, pues se comparte información pero no se ejemplifican los valores de los conceptos.
- El interesado no tiene una idea clara de lo que necesita, por lo cual responde los interrogantes de forma rápida y sin comprender la importancia de lo que debería responder.
- Cuando las sesiones son muy largas, el interesado tiende a entrar en contradicciones con lo que ya dijo, pues se dedica a hablar in abstracto.

Un método como el que se pretende establecer con los esquemas preconceptuales ejecutables podría favorecer el proceso dado que:

- Se disminuye la incertidumbre, pues la ejemplificación provee un terreno común de comunicación práctica entre el analista y el interesado, logrando esclarecer mejor las ideas.
- Se generan acuerdos entre analistas e interesados, pues los esquemas preconceptuales permiten la representación de la información, posibilitando la solución de discrepancias.
- La ejemplificación permite ver información que está oculta en la mente del interesado. Además, permite que el interesado aclare los conceptos de su mundo pues ya habla en términos prácticos de aquello que lo rodea.
- Se pueden simular mejor las interacciones e, incluso, validar los resultados de esas interacciones que ocurren con los actores del proceso.

Una modificación en la terminología del esquema preconceptual no es suficiente para hacerlos ejecutables, pues pueden surgir todavía problemas de entendimiento, claridad y falta de información sobre las actividades que se realizan para clarificar la representación de un dominio. Así, se proponen varias consideraciones para la implementación de relaciones dinámicas dentro de los esquemas preconceptuales, con el fin de posibilitar la visualización de instancias del sistema. Con ello, se busca identificar posibles fallas que se pueden presentar a la hora de la entrega del producto, mediante la validación con el interesado de esas instancias. Las consideraciones que se presentan son las siguientes:

1. Se debe generar el esquema preconceptual correspondiente partiendo del modelo verbal.
2. Se debe seleccionar y analizar cada una de las relaciones dinámicas.
3. Para cada relación dinámica obtenida en el paso anterior, se establece el sentido de utilización del verbo. Uno de los principales problemas a que se enfrenta el analista es la descripción de la relación dinámica, pues el verbo allí contenido puede tener varios sentidos dependiendo de lo que se pretende. Por ejemplo, el verbo “buscar” puede tener el sentido de investigar (si lo que se requiere es información) o examinar (si lo que se quiere es encontrar información, por ejemplo, en una base de datos). Como resultado, se logra obtener un subconjunto de sentidos, los cuales se exponen al interesado en un orden que determina el analista, quien subjetivamente elegirá según su criterio el orden en el que se presentarán.
4. Se debe hacer una recolección de información basada en ejemplos de los diferentes conceptos que se albergan en el esquema preconceptual, para almacenarlos en tablas.
5. Para cada relación dinámica se debe analizar, dentro del esquema preconceptual, qué conceptos se afectarían o se tendrían en cuenta para realizar completamente esta relación dinámica. Es importante que, a la hora de analizar estas relaciones dinámicas, se busque desglosar en lo más mínimo el verbo, es decir, definir uno o varios verbos con operaciones atómicas (esto se debe a que, para el caso de los esquemas preconceptuales, una relación dinámica no puede llamar a otra mientras se está ejecutando).
6. Luego de obtener esta información e incorporarla como posibles valores dentro de la tabla, se debe hacer un estudio sobre los posibles resultados de las operaciones que se realizan con las relaciones dinámicas, es decir, observar detalladamente de qué manera se puede implementar la

función que se desea para relacionar los conceptos que la función afecta en su estado.

7. Después de capturar toda la información necesaria para que la relación dinámica funcione como se desea, existe la posibilidad de hacer funcional la operación albergada en la relación dinámica, es decir, transmitir las especificaciones necesarias para definir la funcionalidad de la operación.
8. Terminado el proceso con una de las relaciones dinámicas, se realiza un proceso de igual manera para las restantes. Teniendo terminado esto, se podrá plasmar en una aplicación estas especificaciones y ver el comportamiento del sistema que se está modelando.

Para una mayor facilidad de lectura del esquema preconceptual basado en ejemplos, la simbología que se debe presentar se define de la siguiente manera:

Los conceptos pueden ser de dos tipos:

- Conceptos-clase: Son los conceptos que contienen en su interior atributos y que se pueden instanciar. Para estos, no varía la especificación de los esquemas preconceptuales originales, simplemente se escribe el concepto sin acompañarlo de otros elementos, como se presenta en la Figura 5. Se reconocen porque de ellos parten relaciones estructurales de tipo "tiene".

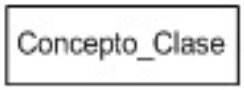

Figura 5. Representación de un concepto_clase

- Conceptos-atributo: Son los conceptos hoja o atómicos dentro de un esquema preconceptual y son atributos de un concepto-clase. Los conceptos-atributo se especifican de la siguiente forma: primero se coloca el nombre del atributo y, seguido del símbolo igual ("="), el valor que toma, entre comillas cuando es un valor alfanumérico y sin comillas cuando es un valor numérico, como se puede observar en la figura 6. Se reconocen porque de ellos no se desprende ninguna relación estructural.

Figura 6. Representación de un concepto_atributo

Es importante resaltar que, para cada concepto-clase se presenta una tabla, que alberga en las cabeceras de las columnas los conceptos atributos que lo componen. En caso de que un concepto clase contenga otro concepto clase, se omite este concepto clase y sólo se escriben los conceptos atributo (véase la Tabla 1, en la sección siguiente, a modo de ejemplo). Para los conceptos atributo, se presenta una lista, la cual contendrá el nombre del concepto como cabecera. Esto se realiza con el objetivo de tener más interactividad con el usuario y procurando para tener un aspecto más claro y amigable para el analista, teniendo en cuenta que la información que allí se describe puede ser intrincada o demasiado robusta.

Las relaciones dinámicas y el resto de elementos del esquema preconceptual, no varían en su simbología. Al seleccionar una relación dinámica, lo que se debe hacer es resaltar, dentro del esquema preconceptual, los conceptos que se afectan únicamente con dicha relación dinámica, pidiendo información al usuario para poder realizar la operación correspondiente.

V. CASO DE ESTUDIO

En el proceso de ejemplificación del caso de estudio, se detalla un ejercicio planteado desde su especificación en lenguaje natural, que se presenta a continuación:

Para cada pregunta, el Estudiante, que posee una identificación y un nombre, presenta una Respuesta que pertenece a la Solución y que tiene una descripción. Si la Descripción de la Respuesta coincide con la Respuesta correcta a la Pregunta, se asigna a la Respuesta una Calificación igual al Porcentaje de la Pregunta. Luego, se suman las Calificaciones a cada Respuesta de la Solución y se multiplica esa suma por la nota máxima del Examen y se divide entre cien para obtener la nota de la Solución. Cada pregunta tiene un número de pregunta y un enunciado. El profesor es el encargado de calificar el examen.

El objetivo del ejercicio es aplicar las consideraciones presentadas en la sección anterior y demostrar que se puede comprobar la funcionalidad del dominio modelado en un esquema preconceptual ejecutable (basado en ejemplos) y hacer entendible para el usuario la funcionalidad del software a desarrollar.

A partir del modelo verbal y modelando su contenido, se logró obtener el esquema preconceptual que se presenta en la figura 7.

Continuando con el estudio, se presentan dos relaciones dinámicas, estudiante presenta solución y profesor califica examen. Para la primera de ellas, con base en el análisis que se realizó tomando posibles interpretaciones del verbo, se lograron obtener los siguientes sentidos que podía tomar "presenta":

- Proponer a una persona para una dignidad o cargo.
- Dirigir y comentar un espectáculo o un programa de radio o televisión.
- Dar a conocer a una persona indicando su nombre.
- Poner algo de manera que se pueda observar o juzgar con detenimiento.

El analista logró proponer la serie de posibles sentidos, logrando definir con el interesado la última de las opciones. El paso a seguir fue contemplar una serie de ejemplos de los conceptos y almacenarlos en la tabla para el concepto-clase, tal como se observa en la Tabla 1. Nótese que "enunciado", "respuesta correcta" y "descripción" son listas de valores que pueden pertenecer o no al concepto-clase "pregunta".

Revista Avances en Sistemas e Informática, Vol.7 No.3, diciembre de 2010 - Medellín. ISSN 1657-7663

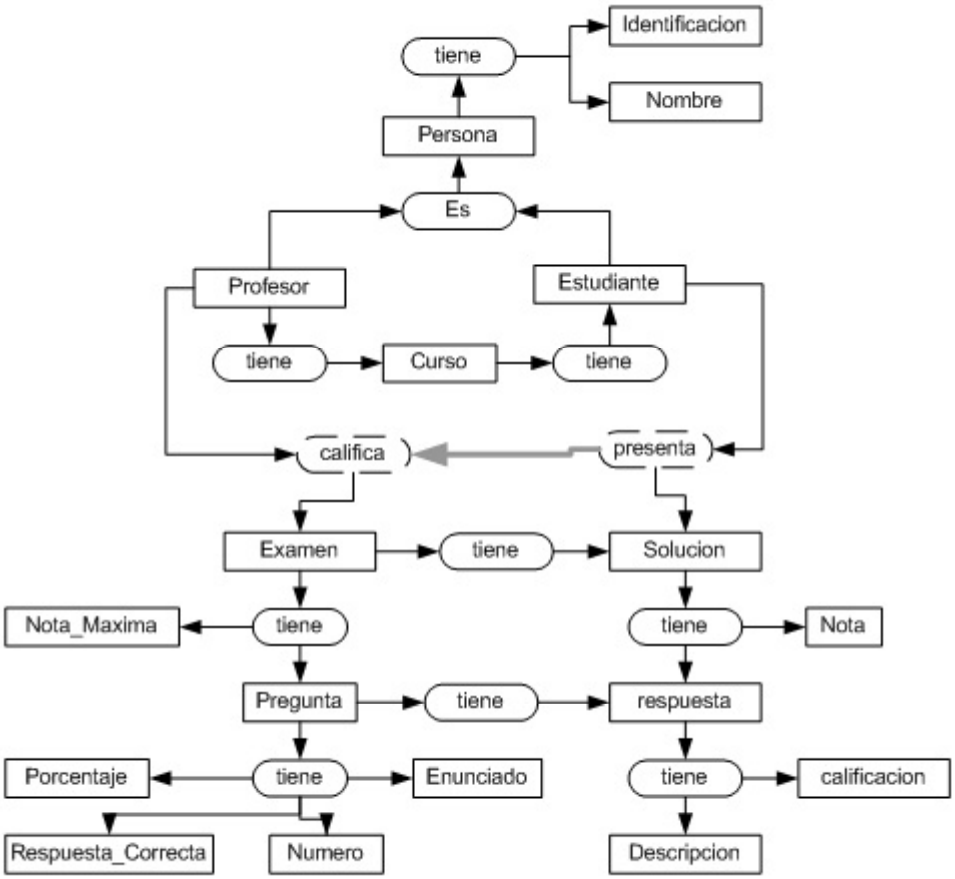

Figura 7. Esquema preconceptual correspondiente al modelo verbal

Tabla 1. Correspondiente al concepto-clase "pregunta".

NUMERO	PORCENTAJE	ENUNCIADO	RESPUESTA CORRECTA	DESCRIPCION	CALIFICACION
1	20	¿Qué es UML?	Unified Modeling Language	Un Modelo Lógico	0
2	25	¿Cómo se representa una relación dinámica?	Verbo de acción.	Verbo de acción.	25
3	25	¿Qué es NAT?	Network Address Translation	Network Address Translation	25
4	20	¿Qué es IP?	Internet Protocol	Internet Process	0
5	10	¿Qué es DHCP?	Dynamic Host Configuration Protocol	Dynamic Host Configuration Protocol	10

Luego de llenar la tabla con los valores de los diferentes conceptos_atributo (excepto "descripción", que se llena en el siguiente paso del método), detallar qué conceptos se modifican con esta operación fue el siguiente paso, en el cual se identificó que sólo el concepto descripción era necesario para realizar esta operación. Para esto, el analista notó que el estudiante sólo puede escribir la descripción de la respuesta. Así, se puede definir que esta operación sólo incluye este concepto y no necesita operaciones lógicas ni de ningún tipo, simplemente inserción en la tabla de valores. Por ejemplo, para la pregunta identificada con el número 1, el estudiante cuyo nombre es "Carlos" y que tiene como identificación "8741034", para un examen del curso escribe, en el concepto-atributo "descripción" (que pertenece al concepto-clase "respuesta") el valor "Un

Modelo Lógico". Igualmente, debe llenar las descripciones para las preguntas 2 a 5.

De similar manera, se realizó el estudio para la otra relación dinámica, Profesor califica examen, en la cual se toman en consideración varios conceptos_clase, incluyendo examen y todos sus conceptos_atributo. Para esto, el analista estudió que, para cada pregunta del examen, se debe tener una descripción de la respuesta que otorga el estudiante en la solución, para así poder realizar una comparación entre la respuesta correcta de la pregunta y la descripción de la respuesta y, así, calcular la nota de la solución teniendo en cuenta la nota máxima que un estudiante puede obtener. Por ejemplo, en el caso de la pregunta identificada con el número 1, se tenían previamente los valores de "porcentaje", "enunciado" y "respuesta correcta". Cuando el estudiante ingresó el valor de "descripción", ya fue posible realizar el cálculo del concepto_atributo "calificación", realizando la comparación entre la descripción y la respuesta correcta. En este caso, el valor asignado es cero (0), ya que las respuestas no coinciden. Con un proceso similar se realiza el cálculo para las demás preguntas, para obtener las calificaciones que se muestran en la Tabla 1. En la Figura 8 se presenta una posición intermedia del cálculo que se realiza sobre el esquema preconceptual, aplicando la simbología mencionada anteriormente a la pregunta identificada con el número 1. Nótese que se trata de un conjunto de valores que se ingresan y calculan en un instante específico, en el cual se encuentra activa una determinada relación dinámica. De esta manera, se pueden encontrar las similitudes entre el esquema preconceptual ejecutable y el diagrama de objetos. Además, la combinación del esquema gráfico y los valores consignados en las tablas que los acompañan permite lograr una animación completa del cálculo que encierra una determinada relación dinámica.

Por simplicidad, en el caso de estudio de esta propuesta se supuso el ingreso de información en varios de los conceptos_atributo que se incluyen en el esquema preconceptual. Ese esquema podría ser mucho más complejo, por ejemplo empleando descripciones predefinidas para cada uno de los valores que se desee ingresar. En ese caso, el funcionamiento sería análogo y la única diferencia sería la adición de otros conceptos_clase y conceptos_atributo para describir la información por seleccionar. Por ejemplo, la respuesta podría tener un código, al igual que la respuesta correcta, de modo que sólo fuera necesario comparar los dos códigos para saber si se puede asignar el porcentaje. La adición de este tipo de elementos dependerá del detalle con el que aborden la comunicación el analista y el interesado, pero se espera que, con la ejemplificación, se puedan detectar incompletitudes de este tipo.

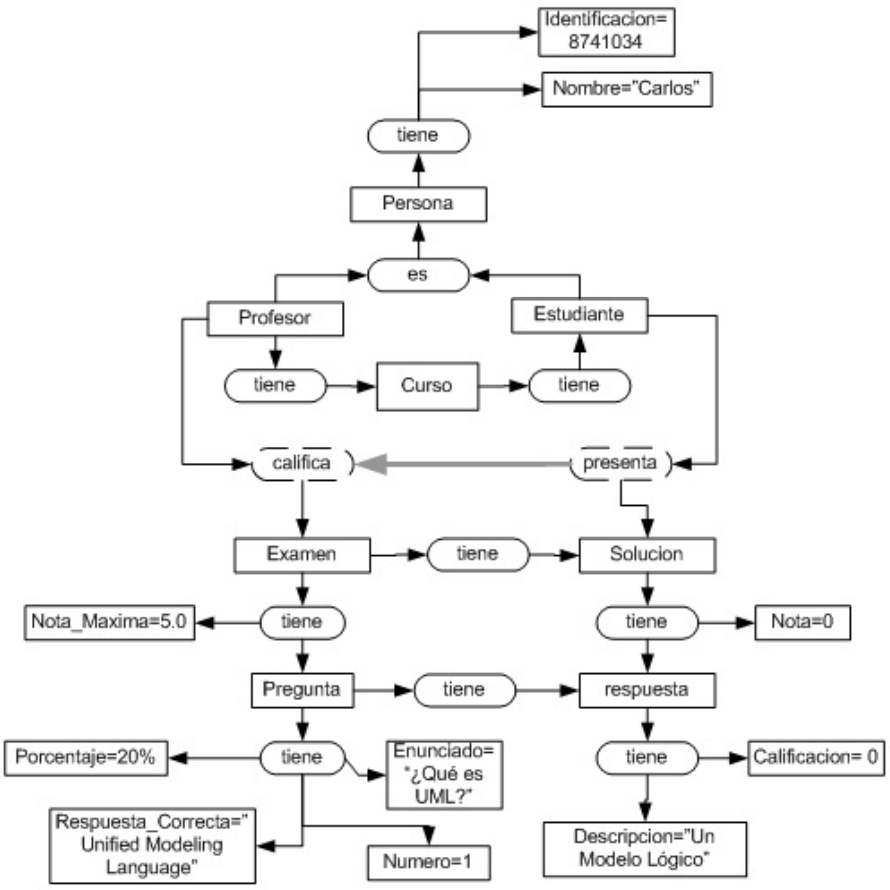

Figura 8. Esquema preconceptual con la simbología "concepto=valor".

VI. CONCLUSIONES Y TRABAJO FUTURO

Se presentó en este artículo una propuesta de modificación de los esquemas preconceptuales, de forma tal que se puedan considerar "ejecutables". La propuesta incluye una serie de consideraciones para la identificación adecuada de operaciones asociadas a relaciones dinámicas a partir de esquemas preconceptuales y una nueva notación que toma en cuenta la diferenciación entre conceptos según se ubiquen en un esquema preconceptual (conceptos-clases y conceptos-atributos). Esta propuesta se ejemplificó con un caso de estudio.

Las consideraciones presentadas definen un derrotero para la definición de las diferentes operaciones que se pueden implementar dentro de un producto de software, partiendo de las acciones que realizan los diferentes actores del dominio que se está modelando. Este derrotero se puede emplear en el proceso de educción de requisitos y permite solucionar algunos de los principales problemas que se asocian con esta fase del desarrollo de una aplicación.

Cabe subrayar que estas consideraciones son independientes del dominio de aplicación, permitiendo solucionar limitaciones de otros proyectos similares, para los cuales era necesario conocer profundamente el dominio de aplicación. La modificación de la notación de los conceptos dentro de los esquemas preconceptuales, fue de gran ayuda para la especificación de las consideraciones, pues la descripción que el analista presenta al interesado, permite expresar de una manera transparente la funcionalidad del sistema a partir de lo que sucede con las acciones que allí aparecen. La aplicación de esta propuesta depende básicamente de la información que tenga el interesado, por lo cual es fundamental la generación de acuerdos en relación con el dominio de estudio; sin embargo, el uso de esquemas preconceptuales facilita esta interacción analista-interesado, suministrando una terminología común que ambos pueden discutir y negociar. Lo que añade esta propuesta es la orientación hacia la ejemplificación de los conceptos, en busca de realizar un análisis intensivo de la funcionalidad que poseen las relaciones dinámicas del mundo del interesado.

Como trabajo futuro, se hace necesaria la implementación de una herramienta que permita mostrar al interesado el funcionamiento del producto en construcción de una forma clara teniendo en cuenta las consideraciones nombradas en la sección 4, evitando así, obviar o dejar desapercibidos aspectos importantes en las fases de especificación, diseño e implementación. También, sería importante la construcción de esquemas preconceptuales a partir de lenguaje natural, incorporando las funciones que el interesado desea y definiendo su especificación de manera automática. La tercera línea de investigación necesaria en este caso es la realización de un estudio relacionado con los tipos de datos que se pueden emplear en la definición y análisis de una aplicación de software, con el fin de "predecir" cierta información que el interesado no revela, en ocasiones, por desconocimiento, en relación con la apariencia final que tendrán los datos en un ejemplo de su dominio.

AGRADECIMIENTOS

Este trabajo se financió parcialmente con fondos de la Vicerrectoría de Investigación de la Universidad Nacional de Colombia, mediante el proyecto de investigación "TRANSFORMACIÓN SEMIAUTOMÁTICA DE LOS ESQUEMAS CONCEPTUALES, GENERADOS EN UNC-DIAGRAMADOR, EN PROTOTIPOS FUNCIONALES".

REFERENCIAS

[1] Arango F.; Gelbukh A. y Zapata C. M., 2006. Pre-conceptual Schema: A Conceptual-Graph-Like Knowledge Representation for Requirements Elicitation. En: Lecture Notes in Computer Science, Vol. 4293, pp. 17-27.

[2] Arango F. y Zapata C. M., 2005. Los Modelos Verbales en Lenguaje Natural y su utilización en la elaboración de esquemas conceptuales para el desarrollo de software: Una revisión crítica. En: Revista Universidad EAFIT, Vol. 41, pp. 77-95.

[3] Chen P., 1983. English Sentence Structure and Entity-Relationship Diagram. En: Information Sciences. Vol. 1, pp. 127–149.

[4] Arango F.; Gelbukh A. y Zapata C. M., 2006. UN-Lencep: Obtención Automática de Diagramas UML a partir de un Lenguaje Controlado. En: Avances en la Ciencia de la Computación. Guanajuato, Mexican Society of Computer Science. pp. 254–259.

[5] Caro M. F.; David M. E.;Hernández F. M. y Toscazo R. E., 2009. Diseño de software educativo basado en competencias. En: Ciencia e Ingeniería Neogranadina, Vol. 19-1, pp. 71-98.

[6] Estrada B. M.; González G. y Zapata C. M., 2006. Reglas de conversión entre el diagrama de clases y los grafos conceptuales de Sowa. En: Revista de Ingeniería Universidad de Medellín, Vol. 5, pp. 111-122.

[7] Drake J. M.; Gonzalez M.; Gutiérrez J. J. y Palencia J. C., 2001. MAST: Modeling and Analysis Suite for real Time Applications. En: 13th EuromicroConferenceon Real-Time Systems, pp. 125-134.

[8] Delugach. H. S., 2001. CharGer: A Graphical Conceptual Graph Editor. En: Computer Science Department. University of Alabama in Huntsville, pp. 1-12.

[9] Arango F.; Tamayo P. A. y Zapata C. M., 2007. Conversión de esquemas preconceptuales a diagrama de casos de uso empleando Atom³. En: Revista Dyna, Vol.74, pp. 237-251.

[10] Arango F.; Jaramillo A. F. y Zapata C. M.; 2006. Una propuesta para mejorar la completitud de requisitos utilizando un enfoque lingüístico. En: Ingeniería & Desarrollo. Universidad del Norte. Vol. 19, pp. 1-16.

[11] Arango F. y Zapata C. M., 2007. Un ambiente para la obtención automática de diagramas UML a partir de un lenguaje controlado. En: Revista Dyna. Universidad Nacional de Colombia. Vol.74, pp. 223-236.

[12] Dedene G. y Poels G., 2000. Measures for Assessing Dynamic

Complexity Aspects of Object-Oriented Conceptual Schemes. En: Lecture Notes in ComputerScience, Vol. 1920, pp.499-512.

[13] Brito e Abreu F.;Melo W.; Sahraoui H..y Zuse H., 1999. Quantitative Approaches in Object-Oriented Software Engineering. En: ECOOP'99 Workshops. Lecture Notes in ComputerScience, Vol. 1743, pp. 326-337.

[14] Collazos C. A.; Giraldo F. D. y Giraldo W. J., 2009. Desarrollo basado en modelos de la interfaz de usuario de sistemas groupware. En: Revista Avances en Sistemas e informática, Vol.6, pp. 197-204.

[15] Garcia F.; Piattini M.; Ruiz F.;Soto J. P. y Vizcaino A., 2006. Aplicando gestión del conocimiento en el proceso de mantenimiento del software. En: Revista Iberoamericana de Inteligencia Artificial, Vol. 10. pp. 91-98.

[16] Booch, G., 2001. Lenguaje de Modelado Unificado. Editorial Adisson Wesley. Madrid.

[17] Sowa J. F. 2000. Knowledge Representation: Logical, Philosophical, and Computational Foundations. Brooks Cole Publishing Co. 512 P

Revista Avances en Sistemas e Informática, Vol.7 No.3, diciembre de 2010 - Medellín. ISSN 1657-7663

Universidad Nacional de Colombia Sede Medellín
Facultad de Minas

Escuela de Ingeniería de Sistemas

Misión

La misión de la Escuela de Ingeniería de Sistemas es fomentar y apoyar la generación o la apropiación de conocimiento, la innovación y el desarrollo tecnológico en el área de ingeniería de sistemas e informática sobre una base científica, tecnológica, ética y humanística.

Visión

La formación integral de profesionales desde el punto de vista científico, tecnológico y social que les permita adoptar, aplicar e innovar conocimiento en el campo de los sistemas e informática en sus diferentes aspectos, aportando con su organización, estructuración, gestión, planeación, modelamiento, desarrollo, procesamiento, validación, transferencia y comunicación; para lograr un desempeño profesional, investigativo y académico que contribuya al desarrollo social, económico, científico y tecnológico del país.

Escuela de Ingeniería de Sistemas
Dirección Postal:
Carrera 80 No. 65 - 223 Bloque M8A
Facultad de Minas. Medellín - Colombia
Tel: (574) 4255350 Fax: (574) 4255365
Email: esistema@unalmed.edu.co
http://pisis.unalmed.edu.co/