Ingeniare. Revista Chilena de Ingeniería
ISSN: 0718-3291
facing@uta.cl
Universidad de Tarapacá
Chile

Zapata, Carlos M.; Giraldo, Gloria L.; Mesa, Jhon E.
UNA PROPUESTA DE METAONTOLOGÍA PARA LA EDUCCIÓN DE REQUISITOS
Ingeniare. Revista Chilena de Ingeniería, vol. 18, núm. 1, 2010, pp. 26-37
Universidad de Tarapacá
Arica, Chile

Ingeniare. Revista chilena de ingeniería, vol. 18 Nº 1, 2010, pp. 26-37

UNA PROPUESTA DE METAONTOLOGÍA PARA LA EDUCCIÓN DE REQUISITOS

A PROPOSAL OF META-ONTOLOGY FOR REQUIREMENTS ELICITATION

Carlos M. Zapata[1] Gloria L. Giraldo[1] Jhon E. Mesa[1]

Recibido 18 de julio de 2008, aceptado 4 de noviembre de 2009
Received: July 18, 2008 Accepted: November 4, 2009

RESUMEN

Las ontologías, hoy en día, juegan un papel importante en algunas áreas del saber, en especial en las ciencias de la computación. Actualmente, se viene incorporando su uso en la ingeniería de requisitos, para apoyar las tareas de educción de requisitos y, de esta manera, obtener un completo levantamiento de la información del dominio del problema. Sin embargo, las ontologías que se usan en ingeniería de requisitos son limitadas, en la medida en que están circunscritas a un dominio particular. En este artículo se propone la construcción de una metaontología para la educción de requisitos, de forma incremental e independiente del dominio del problema. Así, el conocimiento incorporado en la ontología se puede aprovechar en dominios diferentes. La implementación de la metaontología se hizo en la herramienta Protégé™, para aprovechar las capacidades que ésta ofrece en la construcción de ontologías.

Palabras clave: Ontología, metaontología, educción de requisitos, Protégé™.

ABSTRACT

Nowadays, ontologies play a crucial role in some knowledge areas, especially in computer science. Currently, ontologies have been used in requirements engineering, in order to support requirement elicitation tasks and, consequently, to obtain a complete definition of the problem domain information. However, requirement engineering ontologies exhibit a drawback: they can be only used into the problem domain for which they were defined. We propose, in this paper, the development of a requirement elicitation meta-ontology with the purpose of acquiring the information of problem domain. This information is incremental and problem-domain-independent. Thus, the information acquired in a problem domain can be used into another. The meta-ontology was implemented in the Protégé™ tool, as a way to take advantage of the capabilities of this tool for ontology development.

Keywords. Ontology, meta-ontology, requirements elicitation, Protégé™.

INTRODUCCIÓN

Hoy en día, las ontologías juegan un papel importante en algunas áreas del saber, en especial en las ciencias de la computación. Inicialmente, su uso se restringía a la inteligencia artificial pero, debido a su utilidad y generalidad, se superaron estas fronteras para utilizarlas en campos como la medicina, el desarrollo de sitios Web y la educción de requisitos, entre otros [1-2].

El término *ontología* se refiere a la clasificación de los conceptos de algún dominio. Esta clasificación permite establecer un conocimiento común en el dominio, en búsqueda de un lenguaje unificado que permita la comunicación dentro de este dominio [2-3].

Una ontología se compone de conceptos que se relacionan jerárquicamente en forma de árbol. Una ontología parte de una raíz que se va desglosando en ramas hasta llegar a las hojas, que son instancias de los conceptos de la ontología en un dominio particular. Cuando se trata de relaciones de generalización, la lectura de los conceptos de una ontología se hace desde las hojas hacia la raíz, utilizando el verbo "es un". Este verbo se utiliza entre un subconcepto y su superconcepto inmediato en la jerarquía [3], tal y como se muestra en la Figura 1, que incluye un ejemplo de una ontología de bebidas. El término *metaontología* [2] se refiere a una ontología general, cuyo uso se permite en cualquier dominio, sin importar la naturaleza de éste.

[1] Grupo de Lenguajes Computacionales. Universidad Nacional de Colombia. Carrera 80 Nº 65-223. Medellín, Colombia. E-mail: cmzapata@unal.edu.co; glgiraldog@unal.edu.co; jemesa@unal.edu.co

Por otra parte, la ingeniería de requisitos es la primera etapa en la construcción de software y es una de las más cruciales. En esta etapa se captura la información relevante del dominio (conocido como el dominio del problema) y se levantan los requisitos que debe cumplir el software. Esta etapa, a su vez, se divide en varias subetapas, entre las cuales se encuentra la educción de requisitos. Esta subetapa comprende el proceso donde los interesados descubren, articulan y entienden los requisitos del software [4].

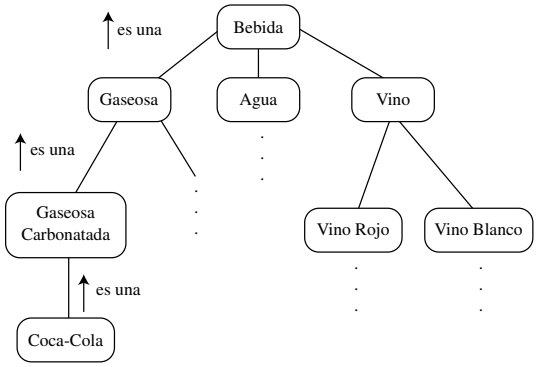

Figura 1. Ejemplo de una ontología.

Desde el surgimiento de la ingeniería de requisitos, se vienen desarrollando varias técnicas para recoger, lo mejor posible, la información relevante del dominio del problema. Una de las técnicas más reconocidas en la actualidad la constituye el uso de las ontologías. En la educción de requisitos, las ontologías permiten establecer un conocimiento común entre los interesados y el ingeniero de requisitos lo cual, a su vez, permite una mejor comunicación entre las partes, buscando una recopilación lo más completa posible de los requisitos del software [2].

Si bien el proceso de educción de requisitos mejora con el uso de las ontologías, todavía subsisten algunos problemas. Primero, la definición de una ontología para cada dominio, en concreto, sólo sirve para ese dominio en particular; ello implica que no es posible migrar la ontología a otros dominios diferentes. Segundo, una ontología no se puede incrementar con la información de varios dominios independientes entre sí [2].

Como una contribución a la superación de estas limitaciones, el objetivo de este artículo apunta a construir una metaontología para la educción de requisitos que sea independiente del dominio del problema y permita recopilar la información relevante del dominio a partir de un diálogo controlado. Además, la metaontología propuesta debe ser general y abierta a su crecimiento, de tal manera que permita modelar, guardar y utilizar la información de varios dominios con base en los metaconceptos de la ontología general y no simplemente permitir la conformación de instancias independientes de la misma.

Para cumplir con el objetivo propuesto, este artículo se organiza de la siguiente manera: en la siguiente sección se presenta un marco teórico, con la definición de ingeniería de requisitos y la descripción de la herramienta Protégé™; luego, se presenta un análisis de los antecedentes en ontologías para la educción de requisitos; después, se presenta la propuesta de la metaontología con un caso de estudio; finalmente, en la última sección se exponen las conclusiones.

MARCO TEÓRICO

Ingeniería de Requisitos

La ingeniería de requisitos es una de las etapas más cruciales en un proyecto de software y comprende la definición de requisitos y la elaboración del modelo conceptual del sistema. En la definición de requisitos se consideran cuatro procesos fundamentales: educción, análisis, especificación y validación de requisitos. La educción de requisitos es el proceso en donde los interesados descubren, articulan y entienden sus requisitos. El análisis de requisitos es el proceso en el cual el ingeniero de requisitos detalla y organiza los requisitos de los interesados, realiza evaluaciones para encontrar conflictos o inconsistencias e identifica requisitos faltantes. La especificación de requisitos corresponde a la formulación precisa de los requisitos de los interesados, utilizando una notación y una forma de documentar que se puede basar en el lenguaje natural, en expresiones simbólicas y en expresiones gráficas. Finalmente, la validación de los requisitos es el proceso por el cual los interesados aseguran que la especificación de los requisitos y la documentación elaborada es consistente, correcta y completa. La ejecución de estos procesos no es estrictamente secuencial ni independiente, sino que estos procesos se pueden interrelacionar y realizar repetidamente [4].

Los ingenieros de requisitos utilizan varias técnicas para la definición de requisitos del software. Entre estas técnicas está la entrevista, que elaboran en conjunto el ingeniero de requisitos y los interesados. Para elaborar una entrevista es necesario, primero, identificar los posibles candidatos a entrevistar. Estos pueden ser personas involucradas en el proceso al cual se piensa adaptar una solución informática, los interesados en el desarrollo del sistema, y/o futuros usuarios del software en construcción. Después de identificar los candidatos, se prepara la entrevista, precisando las preguntas que se pueden utilizar, el lugar donde se realizará la entrevista, las herramientas que se

utilizarán, etc. Finalmente, cuando la entrevista culmina, se extraen de ella los requisitos de los entrevistados y se elabora la documentación pertinente [4].

En este artículo se utiliza un enfoque de entrevistas para obtener la información del dominio del problema y articular éste a la metaontología que se propone. Jurafsky y Martin [5] recopilan un conjunto de modelos de diálogo automáticos en estaciones de trenes, los cuales se usan para elaborar el prototipo de entrevista de una educción de requisitos. Estos modelos sirven de base para la definición de la entrevista en lenguaje controlado que se incluye en este artículo. Nótese que las entrevistas recopiladas por Jurafsky y Martin [5] pertenecen a un dominio plenamente identificable, en tanto que, en este artículo, se presenta la estructura de un diálogo que no se afecta con los cambios de dominio.

Protégé™

Es una herramienta de modelado para la construcción de ontologías desarrollada en *Stanford Medical Informatics*. Inicialmente, esta herramienta se concibió para usarla en dominios relacionados con la medicina pero, debido a su aceptación, se viene empleando en otros campos del saber [6].

La arquitectura de Protégé™ se divide en dos partes: modelo y vista de usuario. La primera parte define un metamodelo compuesto por clases, atributos, restricciones e instancias [6]. Las clases son los conceptos del dominio del discurso (*e.g.* Universidad), los atributos describen propiedades de las clases (*e.g.* Nombre_universidad), las instancias proporcionan particularizaciones de las clases (*e.g.* Universidad_Nacional_de_Colombia) y las restricciones proporcionan información adicional para poder realizar las relaciones entre instancias [6]. El metamodelo, en sí, es una ontología que representan clases, atributos, etc.

La parte de vista de usuario es el espacio en la herramienta de modelado que permite la definición de las clases, atributos, restricciones e instancias de un dominio en particular. Además de las anteriores características de la vista de usuario, Protégé™ tiene la capacidad de generar interfaces gráficas, a partir de la ontología del dominio, que permiten al usuario instanciar la ontología en casos particulares (*e.g.* una metaontología de educción de requisitos) [6].

Otras funcionalidades que posee Protégé™ son los diferentes medios para exportar la ontología, tales como: XML, RDFS, OWL y bases de datos relacionales, entre otros. Además, desde la versión 2000, Protégé™ incorpora el manejo de *plug-ins* que permiten a los programadores externos ampliar las funcionalidades de la herramienta. Igualmente, dada la arquitectura de implementación de Protégé™, existen diversas formas para acceder a ésta por medio de lenguajes de programación [7].

Gracias a las anteriores características, Protégé™ es una alternativa viable para la construcción de ontologías independientes del universo del discurso aunque, por lo general, se emplea para construir ontologías dependientes del dominio de aplicación. En efecto, esta herramienta se utiliza en este artículo para implementar la propuesta de la metaontología de educción de requisitos.

ANTECEDENTES EN ONTOLOGÍAS DE EDUCCIÓN DE REQUISITOS

Algunos trabajos en Ontologías de Educción de Requisitos se describen a continuación:

LEL: Léxico Extendido del Lenguaje

Leite y Franco [8], definen un metamodelo cuya finalidad es proporcionar una ayuda en la educción de requisitos. Su función es conocer el vocabulario del universo del discurso (UdeD) y su semántica antes de abordar el problema a resolver mediante la ingeniería de requisitos [9-10].

El proceso de construcción del LEL, esbozado en la Figura 2, comienza con la captura del UdeD. Esta captura se logra revisando documentos técnicos que puedan existir sobre el dominio y realizando con los usuarios entrevistas en donde se enfatizan las frases o expresiones que más repiten.

Una vez elaborado el UdeD con la información recolectada (que se presenta en lenguaje natural), se obtienen los símbolos (palabras o frases) relevantes del dominio, los cuales se clasifican en cuatro tipos: Sujeto, Objeto, Verbo y Estado. Además, cada símbolo se identifica con un nombre y una descripción. Finalmente, los símbolos obtenidos se verifican y se validan con los usuarios [9-10].

La ventaja del LEL consiste en la facilidad que brinda al ingeniero para obtener los requisitos de los interesados. Esta herramienta sirve, además, para la elaboración de escenarios, otra de las técnicas aceptadas para realizar la educción de requisitos [9].

Una dificultad con el uso de esta herramienta se percibe en la ambigüedad de los símbolos obtenidos en la construcción del UdeD, utilizando el lenguaje natural. A partir de éste se puede generar una incorrecta detección de símbolos y algún grado de "borrosidad"

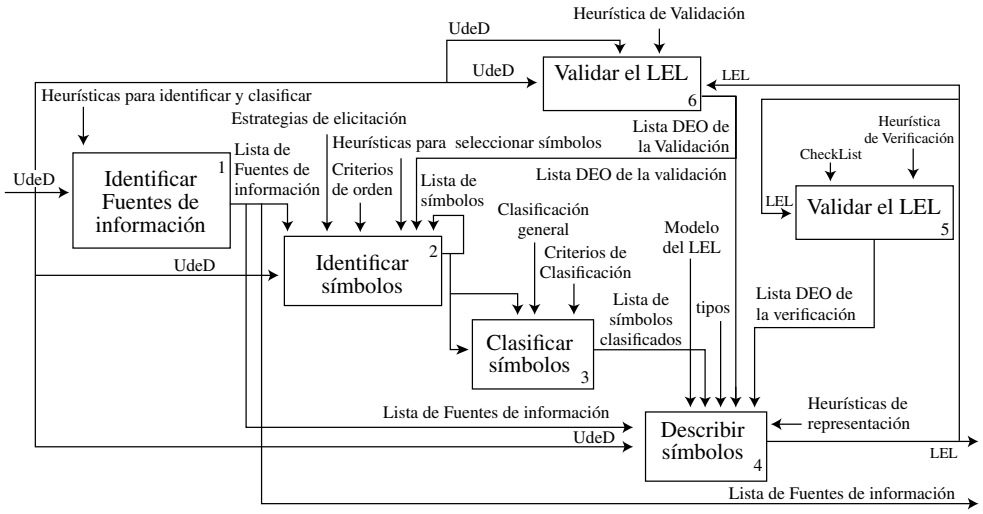

Figura 2. Proceso de construcción del LEL [10].

en la información suministrada por los usuarios. Este problema, origina demoras en la verificación y validación realizadas por el LEL [10].

Otra desventaja de la propuesta de LEL radica en la imposibilidad del uso del LEL como un metamodelo abierto al crecimiento, incorporando la información de cada dominio. Tal como se concibe el LEL, en cada dominio en el cual se aplique, se debe construir un nuevo metamodelo a partir de la información de este dominio particular. En consecuencia, la información de un dominio específico no se puede complementar y relacionar con información de otros dominios, cobijados por metaconceptos. Además, la organización de la información y la complejidad semántica del lenguaje natural hacen aún más difícil el uso del LEL con fines computacionales en el proceso de educción de requisitos.

IEEE *Recommended Practice for Software Requirements Specifications*

El *Institute of Electrical and Electronics Engineers* (IEEE) define un conjunto de especificaciones para obtener los requisitos de una aplicación de software que está en proceso de construcción [11]. Este conjunto de especificaciones se plasma en una plantilla que el ingeniero de requisitos llena a partir del dominio del software, usando el lenguaje natural. Básicamente, la plantilla incluye una introducción al software a desarrollar, una descripción del sistema con las funciones de éste, las características de los usuarios, algunas restricciones y la especificación de los requisitos del software en construcción.

La ventaja de usar esta especificación radica en que ésta es una opción para documentar los requisitos del software a desarrollar que, si se usa según los lineamientos del estándar, permite que los interesados validen el software en construcción.

La desventaja de la especificación está en la poca diferenciación que se hace del dominio del problema y el dominio de la solución. Estos dos dominios se combinan en la especificación y no existe un mecanismo para establecer el alcance y la pertinencia del uno y del otro. Se usan modelos de texto para describirlos y esto puede traer problemas de ambigüedad en la descripción que se realice. Además, la especificación hace más énfasis en qué tareas (funcionalidades) debe realizar el software por construir y no sobre qué dominio y con qué requisitos deben interactuar los agentes.

Ontology Based Requirements Analysis: Lightweight Semantic Processing Approach

Kaiya y Saeki [2] proponen un método para el análisis de los requisitos de software basado en técnicas de ontologías de dominio. El objetivo es validar los requisitos del software utilizando un conjunto de reglas y la ontología del dominio (representada por medio de diagramas de clases del Lenguaje Unificado de Modelado UML).

El método consiste en el uso de una metaontología para la validación de la educción de requisitos. La metaontología se compone de los conceptos del dominio (función,

objeto, ambiente, restricción y cualidad) y los tipos de relaciones entre éstos (*e.g.* sinónimo, antónimo, requiere, etc.) [2]. En la Figura 3 se encuentra la estructura de la metaontología propuesta [2].

A partir de esta metaontología y una ontología del dominio, se aplica un conjunto de reglas derivadas de la metaontología y, utilizando mecanismos de inferencia, se validan y completan los requisitos del software en la perspectiva de la completitud de requisitos [12].

La ventaja de esta propuesta se centra en el uso de una metaontología basada en la educción de requisitos y la existencia de un conjunto de reglas de inferencia sobre las que se realiza la validación de los requisitos del software. Además, aunque los requisitos de software se capturan, en esta solución, en lenguaje natural, no es necesario disponer de pesados procesadores semánticos para analizarlos [2].

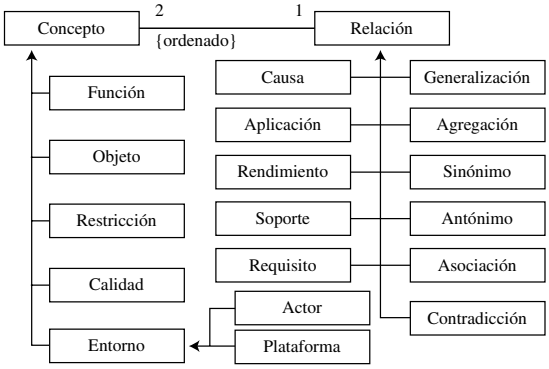

Figura 3. Metaontología para educción de requisitos [2].

La desventaja de la solución radica en que se debe disponer de una ontología del dominio ya construida y se deben determinar los requisitos iniciales del software, sin que la solución describa método alguno para obtenerlos. Además, la propuesta se orienta a la solución del problema y no al análisis del problema sobre el cual se construyen los requisitos del software.

PROPUESTA DE SOLUCIÓN: METAONTOLOGÍA PARA LA EDUCCIÓN DE REQUISITOS

Al igual que en Kaiya y Saeki [2], la propuesta de este artículo consiste en la definición de una metaontología para la educción de requisitos que permite construir la información relevante de un dominio cualquiera. La diferencia es que la ontología de este artículo es mucho más detallada y se estructura de forma que las instancias se puedan obtener a partir de un diálogo en lenguaje natural. Todos los elementos de la propuesta se definen y especifican a continuación.

Estructura general de la metaontología de educción de requisitos

Los conceptos que se identificaron en la metaontología se obtuvieron de los trabajos de Leite y Franco [8], IEEE Std 830-1998 [11] y Kaiya y Saeki [2]. Los conceptos identificados son "Objeto", "Actor", "Restricción", "Función Actor", "Actividad" y "Organización". Todos estos conceptos se refieren a aspectos de la educción de requisitos. En la Figura 4 se muestra la estructura general de la metaontología para la educción de requisitos.

Figura 4. Propuesta de metaontología para la educción de requisitos. Construcción propia de los autores.

A continuación, se detallan las propiedades de cada concepto de la metaontología. De ahora en adelante, estos conceptos se llamarán "clases" para evitar confusiones al lector.

Superclase "Concepto"

La clase *concepto* es una clase abstracta que engloba varios subconceptos y de la cual no es posible crear una instancia. Esta clase se refiere, en una educción de requisitos, a los Actores y sus funciones, las restricciones y los objetos de un dominio particular.

Clase "Objeto"

La clase *objeto* se refiere a todos los conceptos que puedan existir en un dominio particular. Las características (también llamadas *slots*) de esta clase son *Nombre_Objeto*

y *Característica_Objeto*. La primera guarda el nombre del objeto en el dominio particular, mientras que la segunda relaciona a nivel de instancia una o varias instancias de tipo *objeto*. En la Figura 5 se muestra la definición de esta clase en Protégé™.

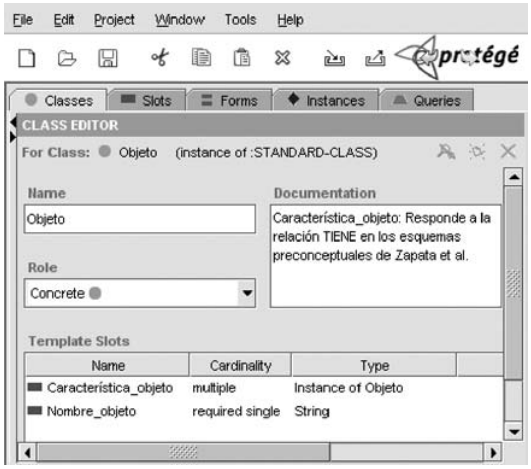

Figura 5. *Slots* de la clase *objeto*. Construcción propia de los autores.

Es importante resaltar que las instancias de objetos que surjan de un dominio particular, deben ser únicas en la metaontología. En ningún momento puede haber dos instancias con diferente *Nombre_objeto* que hagan alusión al mismo objeto.

Clase "Actor"

La clase *actor* incluye a todos los actores que tengan algún tipo de rol relevante en el dominio particular. Las características de esta clase son *Característica_actor*, *Función_actor*, *Rol_actor* y *Rol_adicional*. La primera relaciona, a nivel de instancia, una o varias características que pueda tener el actor del tipo *objeto*. La segunda relaciona, a nivel de instancia (de la clase *Función_actor*), las diferentes funciones que realiza el actor en el dominio particular, de las cuales es necesario que realice por lo menos una. La tercera guarda el tipo de rol que el actor tiene en el dominio particular. Finalmente, la cuarta relaciona, a nivel de instancia y con otras instancias de tipo *actor*, otros tipos de roles que el actor pueda tener. En la Figura 6 se muestra la definición de esta clase en Protégé™.

Superclase "Restricción"

La clase *restricción* es una clase abstracta que engloba varios subconceptos, por lo que no es posible crear una instancia de ella. Esta clase se refiere a las restricciones que puedan existir sobre las funciones que realizan los actores. Los *slots* de esta clase son *Función_actor_implicada* y *Nombre_restricción*. El primero relaciona, a nivel de instancia, la función del actor sobre la cual recae la restricción. El segundo, guarda el nombre de la restricción (atributo opcional). En la Figura 7 se muestra la definición de esta clase en Protégé™.

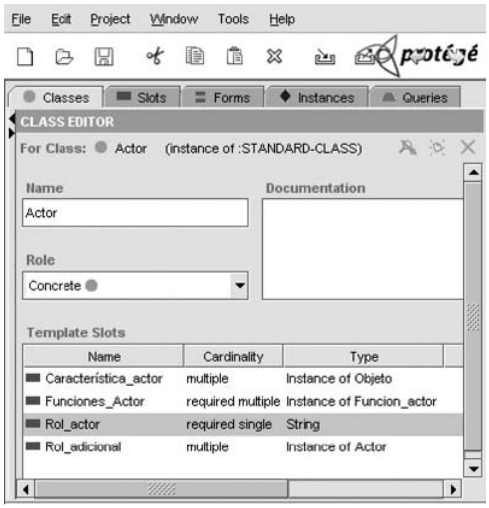

Figura 6. *Slots* de la clase *actor*. Construcción propia de los autores.

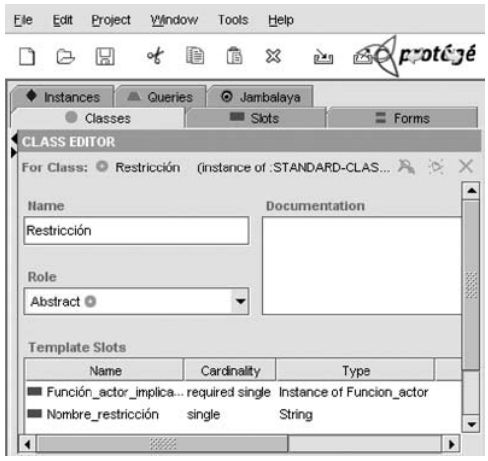

Figura 7. *Slots* de la superclase *restricción*. Construcción propia de los autores.

Clase "Restricción_matemática"

La clase *restricción_matemática* se refiere a todas las restricciones que sean expresiones matemáticas del tipo "Concepto"-"Operador_matemático"-"Valor". Los *slots* de esta clase son *Concepto_implicado* que relaciona, a nivel de instancia, el concepto que hace parte de la restricción matemática, *Operador_matemático* que guarda el operador utilizado en la restricción, en este caso "=", "<", ">", "<="

o ">=" y *Valor_restricción*, que guarda el valor con el cual se compara, usando el operador, la instancia de la clase *concepto*. Los *slots Función_actor_implicación* y *Nombre_restricción*, son atributos heredados de la clase *restricción*. En la Figura 8 se muestra la definición de esta clase en Protégé™.

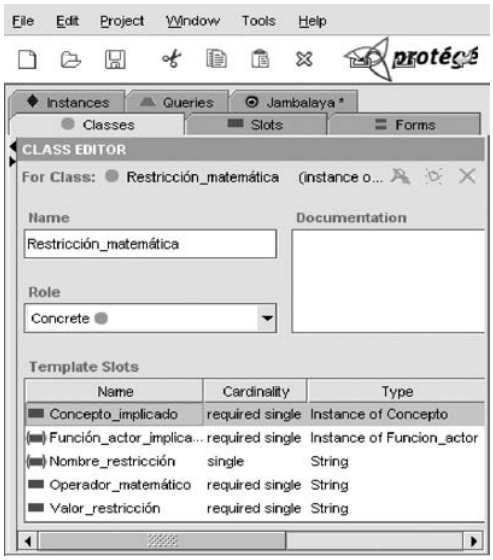

Figura 8. *Slots* de la clase *restricción_matemática*. Construcción propia de los autores.

Clase "Función_actor"

La clase *función_actor* incluye todas las funciones que realicen los actores en un dominio particular. Estas funciones se representan en tríadas "Actor"-"verbo_acción"-"Objeto". Los *slots* de esta clase son *Actor_asociado*, que relaciona, a nivel de instancia (de la clase *actor*), el actor que realiza la función, *Objeto_asociado*, que relaciona, a nivel de instancia (de la clase *objeto*), el objeto sobre el cual el actor realiza la acción y *Verbo_acción*, que guarda el verbo de la acción que realiza el actor sobre el objeto. En la Figura 9 se muestra la definición de esta clase en Protégé™.

Clase "Actividad"

La clase *actividad* se refiere a todas las actividades que se realicen en dominio particular. En general, una actividad es el conjunto de varias funciones de actores. Los *slots* de esta clase son *Funciones_propias_actividad*, que relaciona, a nivel de instancia (de la clase *función_actor*), las funciones de los actores que se realicen en la actividad, las cuales deben tener, por lo menos, una instancia de *función_actor* asociada y *Nombre_actividad*, que guarda el nombre de la actividad. En la Figura 10 se muestra la definición de esta clase en Protégé™.

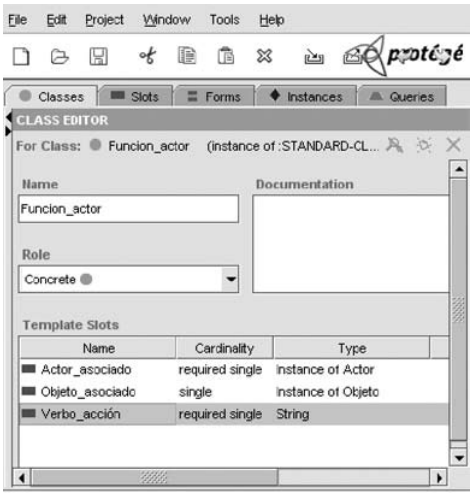

Figura 9. *Slots* de la clase *función_actor*. Construcción propia de los autores.

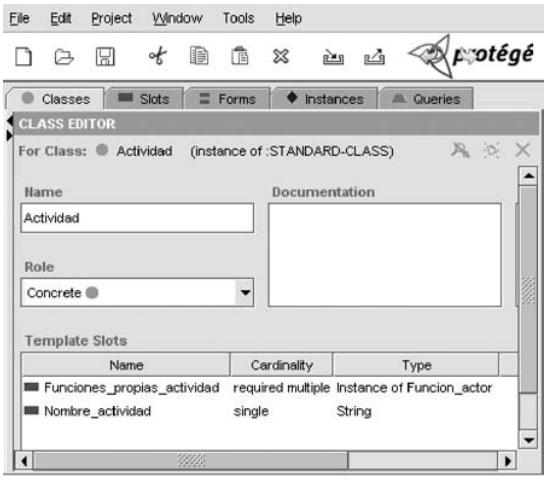

Figura 10. *Slots* de la clase *actividad.* Construcción propia de los autores.

Clase "Organización"

La clase *organización* se refiere a la información de la organización que opera en el dominio particular. Los *slots* de esta clase son *Actividad_económica*, *Actividades* y *Nombre_organización*. El primero guarda la actividad económica de la organización, como pesca, caza, agricultura, comercio, etc. y sirve para restringir el espacio de búsqueda cuando en la metaontología hay información de más de un dominio particular. El segundo relaciona, a nivel de instancia (de la clase *actividad*), las actividades que se lleven a cabo en la organización. El tercero guarda el nombre de la organización que la diferencia de otras organizaciones similares. En la Figura 11 se muestra la definición de esta clase en Protégé™.

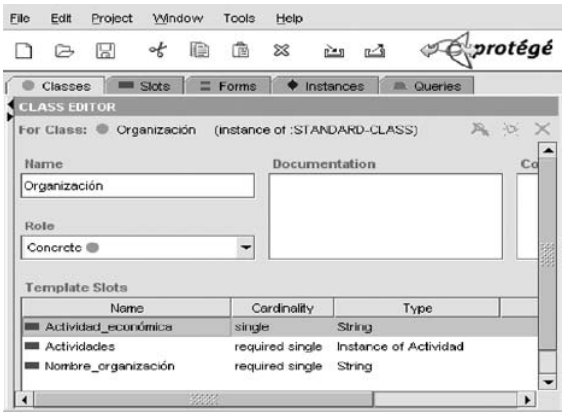

Figura 11. *Slots* de la clase *organización*. Construcción propia de los autores.

CASO DE ESTUDIO

Para mostrar el uso de la metaontología, a continuación se presenta un modelo (simulado) de diálogo de educción de requisitos en una panadería. El diálogo se elaboró de forma similar a como Jurafsky y Martin [5] definen diálogos por turnos, tipo Hombre-máquina. Se definió un diálogo simplificado en un subconjunto del lenguaje natural, con lo cual se evitan problemas semánticos que aún son objeto de estudio en este campo. De esta manera, el caso se centra en el objeto del trabajo, el cual es la traducción del diálogo a la metaontología.

{INICIO}
El siguiente es un diálogo entre un analista y un agente interesado, para capturar los requisitos de una aplicación de software.
***ANALISTA**: Buenos días. El objetivo de esta entrevista es aclarar la información concerniente al dominio del problema que vamos a trabajar. Por favor responda de la forma más concreta posible.*
***INTERESADO**: De acuerdo. Comencemos.*
***ANALISTA**: ¿Cuál es el nombre de esta organización?*
***INTERESADO**: "El buen pan".*
***ANALISTA**: ¿En qué actividad económica se cataloga esta organización?*
***INTERESADO**: "Alimentos"*
***ANALISTA**: Elabore, por favor, una lista de los actores.*
***INTERESADO**: PANADERO, ADMINISTRADOR y VENDEDORA.*
***ANALISTA**: ¿Quiénes pueden ser PANADERO?*
***INTERESADO**: NADIE.*
***ANALISTA**: ¿Quiénes pueden ser ADMINISTRADOR?*
***INTERESADO**: NADIE.*
***ANALISTA**: ¿Quiénes pueden ser VENDEDORA?*
***INTERESADO**: ADMINISTRADOR.*
***ANALISTA**: ¿Puede mencionar características del PANADERO?*
***INTERESADO**: NOMBRE, APELLIDO, CÉDULA, TELÉFONO, SUELDO y FECHA_DE_NACIMIENTO.*
***ANALISTA**: ¿Algunas de estas características que acaba de mencionar poseen a su vez otras características internas?*
***INTERESADO**: SÍ, FECHA_DE_NACIMIENTO.*
***ANALISTA**: Puede mencionar características de la FECHA_DE_NACIMIENTO?*
***INTERESADO**: DÍA, MES y AÑO.*
***ANALISTA**: ¿Algunas de estas características que acaba de mencionar posee a su vez otras características internas?*
***INTERESADO**: NO.*
***ANALISTA**: ¿Los otros actores que usted mencionó anteriormente poseen las mismas características del PANADERO?*
***INTERESADO**: SÍ.*
***ANALISTA**: Elabore una lista de las funciones del PANADERO.*
***INTERESADO**: Lee ORDEN, Elabora PAN, Hornea PAN y Saca PAN.*
***ANALISTA**: ¿Puede mencionar las características de la ORDEN?*
***INTERESADO**: NOMBRE_CLIENTE, FECHA_PEDIDO, DIRECCIÓN_CLIENTE, TELÉFONO, CANTIDAD_PAN y ESTADO_ORDEN.*
***ANALISTA**: ¿Algunas de estas características que acaba de mencionar posee a su vez otras características internas?*
***INTERESADO**: SÍ, FECHA_PEDIDO.*
***ANALISTA**: ¿Puede mencionar características del FECHA_ PEDIDO?*
***INTERESADO**: DÍA, MES y AÑO.*
***ANALISTA**: ¿Puede mencionar las características del PAN?*
***INTERESADO**:CÓDIGO_PAN, CANTIDAD_PAN_ELABORADO y FECHA_DE_ELABORACIÓN.*
***ANALISTA**: ¿Algunas de estas características que acaba de mencionar posee a su vez otras características internas?*
***INTERESADO**: SÍ, FECHA_DE_ELABORACIÓN.*
***ANALISTA**: ¿Puede mencionar características de FECHA_DE_ELABORACIÓN?*
***INTERESADO**: DÍA, MES y AÑO.*
***ANALISTA**: Elabore una lista de las funciones del ADMINISTRADOR.*
***INTERESADO**: Actualiza INVENTARIO y Cancela ORDEN.*
***ANALISTA**: Elabore una lista de las características del INVENTARIO.*

***INTERESADO**: CANTIDAD_PAN_DISPONIBLE.*
***ANALISTA**: Elabore una lista de las funciones del VENDEDOR.*
***INTERESADO**: Registra ORDEN y Despacha ORDEN.*
***ANALISTA**: ¿Qué se requiere para que ocurra que ADMINISTRADOR Cancela ORDEN?*
***INTERESADO**: ESTADO_ORDEN (de la ORDEN) =“Entregada”.*
***ANALISTA**: ¿Qué se requiere para que ocurra que VENDEDOR Despacha ORDEN?*
***INTERESADO**: ESTADO_ORDEN (de la ORDEN) =“Pendiente”.*
***ANALISTA**: ¿Puede establecer una secuencia en las funciones que enunció?*
***INTERESADO**: VENDEDOR Registra ORDEN->PANADERO Lee ORDEN-> PANADERO Elabora PAN-> PANADERO Hornea PAN -> PANADERO Saca PAN -> ADMINISTRADOR Actualiza INVENTARIO->VENDEDOR Despacha ORDEN -> ADMINISTRADOR Cancela ORDEN.*
***ANALISTA**: ¿Puede establecer un nombre a la secuencia que enunció?*
***INTERESADO**: Venta de Pan.*
***ANALISTA**: Eso es todo. Muchas gracias por su tiempo.*
***INTERESADO**: A usted.*
{FIN}

A partir del diálogo, los conceptos inanimados que el interesado identificó son ORDEN, PAN e INVENTARIO con sus características. Éstos, en la metaontología, son instancias de la clase *objeto*. Dentro de las características de estos conceptos se encuentran FECHA_PEDIDO y FECHA_DE_ELABORACIÓN que, a su vez, poseen tres características internas que son DÍA, MES y AÑO. En la Figura 12, se muestra cómo quedaron instanciados estos objetos en la metaontología.

Figura 12. Instancias de la clase *objeto*. Construcción propia de los autores.

Figura 13. Instancias de la clase *actor*. Construcción propia de los autores.

Los actores que el interesado identificó son: PANADERO, VENDEDOR Y ADMINISTRADOR, con sus características. Éstos, en la metaontología, son instancias de la clase actor. En la Figura 13 se muestra cómo quedaron instanciados estos actores en la metaontología. Igualmente, en esta figura se muestran las instancias de la clase *Función_actor*, asociadas con su actor correspondiente.

Las restricciones que el interesado identificó son dos y se relacionan con el objeto ESTADO_ORDEN. Estas dos restricciones en la metaontología son instancias de la clase *restricción_matemática*. En la Figura 14 se muestra cómo quedaron instanciadas estas restricciones en la metaontología.

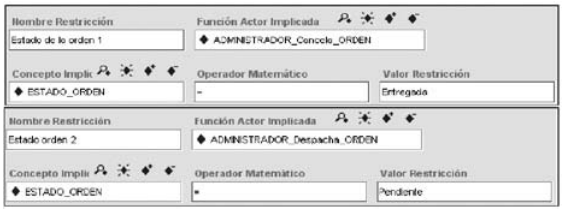

Figura 14. Instancias de la clase *restricción_matemática*. Construcción propia de los autores.

A partir de la secuencia de funciones de los actores que el interesado identificó, se obtiene una actividad dentro de la organización que es "venta de pan". Esta actividad es una instancia de la clase *actividad*. En la Figura 15 se muestra cómo quedó instanciada esta actividad en la metaontología.

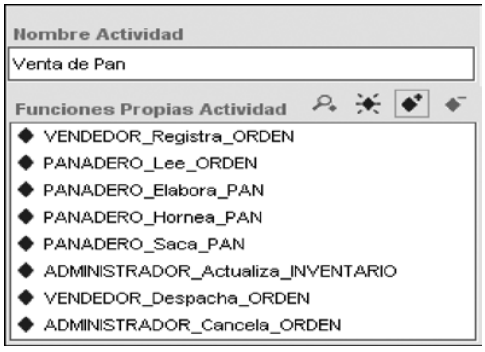

Figura 15. Instancias de la clase *actividad*. Construcción propia de los autores.

Finalmente, el interesado, al inicio del diálogo, proporcionó el nombre de la organización y el tipo de actividad económica que ejercen. Esta información es una instancia de la clase Organización. En la Figura 16 se muestra cómo quedó instanciada esta información en la metaontología.

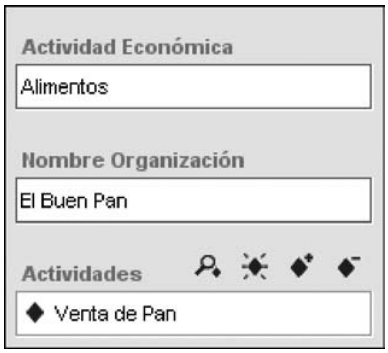

Figura 16. Instancias de la clase *organización*. Construcción propia de los autores.

En la Figura 17 se muestra un esquema de todas las instancias creadas a partir del diálogo con el interesado, asociadas con su respectiva clase de la metaontología.

CONCLUSIONES

Los conceptos de la metaontología y su organización jerárquica mostraron ser útiles y pertinentes para representar conocimientos de dominios particulares y los requisitos de los agentes que intervienen sobre los objetos portadores de dichos conocimientos. Los metaconceptos de la ontología son independientes de dominios particulares y permiten modelar los conocimientos de estos diversos dominios.

Algunos enfoques existentes utilizan una ontología como modelo del conocimiento de un dominio, pero no pueden extender los conceptos y relaciones a otros dominios, haciendo que para cada dominio se deba repetir el trabajo de modelado de su conocimiento, a partir de cero. La metaontología propuesta en este artículo brinda conceptos genéricos aprovechables en diversos dominios con los cuales se modelan los conocimientos de cada uno de ellos. De esta manera, se facilita el desarrollo de aplicaciones que resuelvan clases de problemas existentes en diversos dominios. Igualmente se posibilita la reutilización de los códigos de las aplicaciones que trabajan sobre dichos dominios.

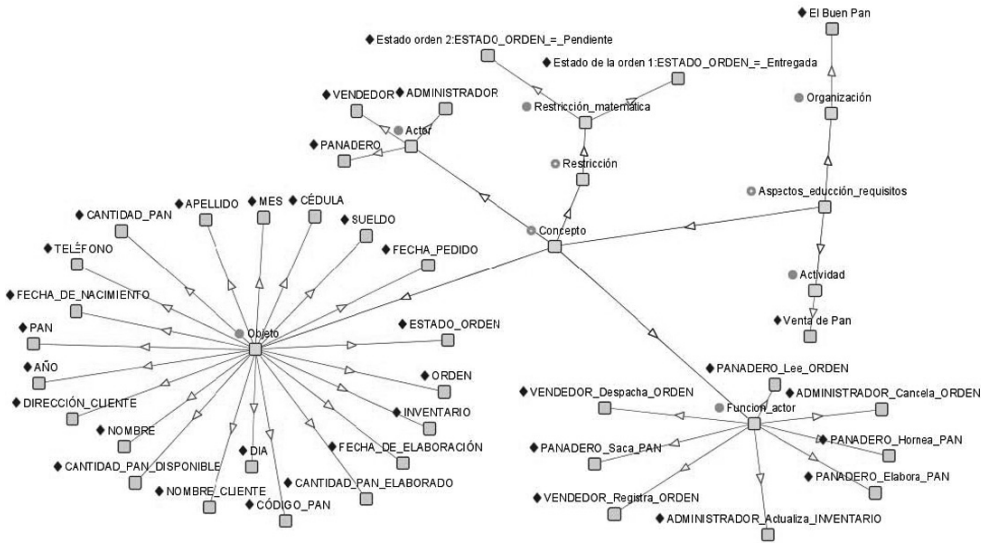

Figura 17. Estructura de la metaontología asociada con varias instancias de un dominio particular. Construcción propia de los autores.

Los conceptos de la metaontología propuesta cubren las categorías de conocimiento de los diálogos corrientes en las entrevistas para la educción de requisitos, como los presentados por Jurafsky y Martin [5]. El comportamiento de la ontología fue satisfactorio al instanciarla con los valores de los conceptos extraídos del diálogo desarrollado en el caso de estudio. Las verificaciones de la ontología, realizadas con base en los diálogos antes mencionados, muestran el carácter genérico de la metaontología, cuyos metaconceptos permiten representar conocimiento de diversos dominios.

El modelo de diálogo que se presentó en este artículo es experimental y permite mostrar la utilidad de la metaontología. Este diálogo no posee una estructura definida, ni se cuenta con un conjunto de preguntas fijas para utilizar en una entrevista de educción de requisitos. Sería interesante proporcionar al diálogo propuesto una estructura y establecer ese conjunto de preguntas en concordancia con la metaontología.

En este artículo se presentó una primera aproximación a una metaontología para la educción de requisitos. Aunque ésta recoge un conjunto de clases que son propias de la educción de requisitos, no incluye las clases *problema* ni *objetivo*, que son comunes en una entrevista. Sería deseable incorporar en la metaontología estas dos clases y definir los *slots* necesarios.

La propuesta de esta metaontología se hizo pensando en una posible automatización de una entrevista con un interesado, para levantar la información requerida en una entrevista de educción de requisitos con miras a la construcción de una aplicación de software. Un trabajo futuro consiste en desarrollar una aplicación que interactúe con un usuario, por medio de un diálogo similar al presentado en este artículo, de tal manera que, a medida que el interesado responda las preguntas, la aplicación llene la metaontología con sus respuestas.

AGRADECIMIENTOS

Este artículo se realizó en el marco del proyecto de investigación: "UN MODELO DE DIÁLOGO PARA LA GENERACIÓN AUTOMÁTICA DE ESPECIFICACIONES EN UN-LENCEP", financiado por la DIME. Los autores manifiestan su agradecimiento a la Ingeniera Luz Marcela Ruiz Carmona, quien participó en la definición de la estructura inicial de los modelos de diálogo, como el que se usa en el Caso de Estudio de este artículo.

REFERENCIAS

[1] N.F. Noy and D.L. McGuinness. "Ontology Development 101: A Guide to Creating Your First Ontology". Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880. March 2001.

[2] H. Kaiya and M. Saeki. "Ontology Based Requirements Analysis: Lightweight Semantic Processing Approach". In Proceedings of the 15th

International Conference on Quality Software, pp. 223-230. IEEE. New York, USA. 2005.

[3] M. Uschold and M. Grüninger. "Ontologies: Principles, Methods, and Applications". Knowledge Engineering Review. Vol. 11 Nº 2, pp. 93-155. 1996.

[4] S. Raghavan, G. Zelesnik, and G. Ford. "Lecture Notes on Requirements Elicitation". Educational Materials CMU/SEI-94-EM-10, Software Engineering Institute, Carnegie Mellon University. 1994.

[5] D. Jurafsky and J. Martin. "Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition". Prentice Hall, p. 1024. 2000.

[6] H. Knublauch, R. Fergerson, N. Noy and M. Musen. "The Protege OWL Plugin: An Open Development Environment for Semantic Web Application". Lecture notes in computer science, pp. 229-243. 2004.

[7] J. Gennari, M.A. Musen, R.W. Fergerson, W.E. Grosso, M. Crubézy, H. Eriksson, N.F. Noy and S. Tu. "The Evolution of Protégé: An Environment for Knowledge-Based System Development". International Journal of Human-Computer Studies, pp. 89-123. 2003.

[8] J.C.S.P. Leite e A. Franco. "O uso de Hipertexto na Elicitaçao de Linguagens da Aplicação". Anais de IV Simpósio Brasilero de Engenharia de Software, SBC, pp. 134-149. Outubro 1990.

[9] M. Ridao, J. Doorn and J.C.S.P. Leite. "Uso de patrones en la Construcción de Escenarios". III Workshop de Engenharia de Requisitos, pp. 140-157. Río de Janeiro, Brasil. 2000.

[10] G.N. Kaplan, G.D.S. Hadad, J.H. Door and J.C.S.P. Leite. "Inspección del Léxico Extendido del Lenguaje". III Workshop de Engenharia de Requisitos, pp. 70-91. Río de Janeiro, Brasil. 2000.

[11] IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications.

[12] H. Kaiya and M. Saeki. "Using Domain Ontology as Domain Knowledge for Requirements Elicitation". In Proceedings of the 14th IEEE International Requirements Engineering Conference, pp. 186-195. Minnesota, USA. 2006.