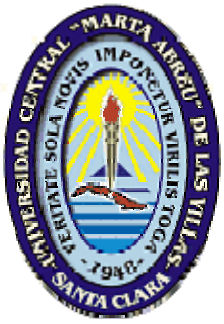
Universidad Central "Marta Abreu" de Las Villas

Facultad de Matemática, Física y Computación

TRABAJO DE DIPLOMA



Validación semántica de esquemas conceptuales Entidad Relación

Autores:

Yaise Georgina Hidalgo Alvarez
Leandro Miguel López Vilacha

Tutores: **MSc. Carlos Ernesto García González**
Dra. C. Luisa Manuela González González

Curso 2008-2009

Hago constar que el presente trabajo fue realizado en la Universidad Central Marta Abreu de Las Villas como parte de la culminación de los estudios de la especialidad de Ciencias de la Computación, autorizando a que el mismo sea utilizado por la institución, para los fines que estime conveniente, tanto de forma parcial como total y que además no podrá ser presentado en eventos ni publicado sin la autorización de la Universidad.

_____	_____
Firma del autor	Firma del autor
Yaise Georgina Hidalgo Alvarez	Leandro Miguel López Vilacha

Los abajo firmantes, certificamos que el presente trabajo ha sido realizado según acuerdos de la dirección de nuestro centro y el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

_____	_____
Firma del tutor	Firma del jefe del Seminario

AGRADECIMIENTOS

A mis padres, por todo el sacrificio que realizaron, y todo el apoyo que me dieron.

A mi familia, por apoyarme y ayudarme en todo momento.

A Nene, por enseñarme a vivir la vida alegremente.

A Mirlandys, Titi, Mima, y Eduardo por ser mi segunda familia.

A mis amigos, que siempre me dieron su mano cada vez que los necesité.

A los que me hicieron reír las veces que estaba triste.

A mi tutor Carlos García por todo el tiempo que me dedicó, y las cosas que me enseñó.

A Luisa, y Manuel Castro, por crearme la base para la investigación.

A todos mis profesores, por contribuir en mi formación profesional.

A todos aquellos que no he mencionado y de una forma u otra me dieron su ayuda.

Gracias a todos.

De corazón

Yaise.

A mis padres, por aconsejarme y darme la oportunidad de llegar a donde estoy.

A mi abuela que siempre rezó por mí y confió en mí.

A Yisi que es mi guía, mi novia y mi mejor amiga.

A mi familia que sembró en mí el interés por el estudio.

A la familia de mi novia y en especial a mi suegra Blanqui.

A mi tutor por su ayuda y por ser un ejemplo para mí.

A mis compañeros de estudio por hacer que mi tiempo con ellos fuera divertido.

A todos gracias, muchas gracias.

Leandro Miguel.

DEDICATORIA

A la mejor mamita del mundo, por todo su amor y comprensión

Su eterna niñita.

A mi mamá que es una sola pero para mí vale por un millón

Su niño grande.

PENSAMIENTO

“La verdadera sabiduría está en reconocer
la propia ignorancia”
Sócrates.

RESUMEN

En la actualidad se disponen de numerosas herramientas para asistir a los diseñadores en proceso de diseño de una base de datos, sin embargo, por lo general, estas no contienen información acerca del mundo real y la forma en que el mismo opera, y delegan en los usuarios para que proporcionen la información acerca del dominio de la aplicación. El reconocimiento dentro de un esquema conceptual de la semántica de las interrelaciones y su clasificación es un aspecto importante en el diseño conceptual. En este trabajo se propone una ontología que permite la validación semántica de las interrelaciones de un esquema conceptual. Se presenta además un prototipo de aplicación implementado en Java para lograr el intercambio con la ontología.

ABSTRACT

Nowadays there are several tools to assist designers in the process of designing a database, however, they generally do not contain information about the real world and how it operates, and delegates it to users so they can provide information about the application domain. Recognition within a conceptual framework for the semantics of interactions and their classification is an important aspect in the conceptual design. This paper proposes an ontology that enables semantic validation of the interrelationships of a conceptual scheme. It also presents a prototype application implemented in Java to achieve the exchange with the ontology.

CONTENIDO

INTRODUCCIÓN

El diseño de bases de datos es un proceso nada trivial porque se basa en la comprensión de los requisitos por parte de los diseñadores y de la precisión con que los mismos puedan reflejar el universo del discurso que se esté modelando. Sin embargo, un diseñador no tiene que conocer el dominio de la aplicación y debe apoyarse en los usuarios para el análisis de requisitos. La fase de modelación conceptual de una base de datos se centra en la construcción de una representación del problema que se enmarca en algún dominio. Los diseñadores de bases de datos generan los modelos conceptuales o *scripts* utilizando: a) gramáticas de modelación conceptual, por ejemplo, la gramática del modelo Entidad-Relación y b) métodos de modelación conceptual y trabajos dentro de un contexto organizacional (Weber, 2002). La mayoría de los avances se han logrado en el desarrollo de metodologías y directrices para ayudar a los diseñadores en el diseño conceptual de una base de datos, sobre todo en entender cómo aplicar los constructores del modelo (Siau et al., 1997; Bodart et al., 2002; Shanks et al., 2002). Sin embargo, también sería útil si el diseñador pudiera tener a su disposición el conocimiento acerca de un dominio de aplicación en forma de un repositorio (Lloyd-Williams, 1997).

Las ontologías han sido propuestas como una vía para representar el conocimiento de la realidad y, en algún nivel, permitir la interoperatividad (Swartout, 1999; Gualtleri et al., 2005; Knight et al., 2006). En esencia, una ontología de dominio proporciona conocimiento acerca de los términos más relevantes del dominio. Si una base de conocimientos de estos términos estuviera disponible y pudiera ser procesada automáticamente, esto podría agilizar el proceso de diseño y hacer los resultados más relevantes. Las ontologías ayudan a capturar la semántica de un dominio actuando como mediador para el significado de los términos. La semántica, para el propósito de esta investigación, se define como el significado de los términos utilizados en un modelo conceptual, es decir, las palabras y frases que representan a las entidades y las interrelaciones. Primero, el conocimiento de un dominio, almacenado en una ontología, puede ser útil durante el diseño de la base de datos ya que la ontología puede sugerir los términos (por ejemplo entidades e interrelaciones) que debieran aparecer en un dominio específico y como ellos están relacionados con otros términos. En una ontología las interrelaciones se corresponden con reglas de negocios y tienen implicaciones para las restricciones de integridad. Segundo, en un diseño parcial la comparación de constructores con aquellos almacenados en la ontología pudiera poner de manifiesto construcciones omitidas.

Se han logrado avances en el desarrollo de metodologías para el diseño de bases de datos, se han propuesto metodologías que apoyan diversos diseños conceptuales y lógicos y se han llevado a cabo investigaciones para automatizar varios aspectos de las fases del diseño de una base de datos (Alter, 1999; Dennis et al., 2000). Esto ha conducido al desarrollo de herramientas CASE (Computer Aided Software Engineering) con capacidades sofisticadas de dibujo. Estas herramientas han probado su utilidad para el chequeo de los aspectos sintácticos y estructurales del esquema conceptual (Stewart et al., 2003), sin embargo pudieran ser más útiles si las mismas tuvieran acceso al conocimiento acerca de la semántica de un dominio de aplicación.

Por todo lo anterior se formula el siguiente **problema de investigación**: actualmente las herramientas CASE orientadas al diseño de bases de datos no realizan una validación semántica del esquema conceptual, por lo que su inclusión en una herramienta CASE permitirá aumentar la calidad de los esquemas obtenidos.

Preguntas de investigación:

1. ¿Qué aspectos deben tenerse en cuenta para diseñar una ontología que permita la validación de frases verbales?
2. ¿Qué criterios serán utilizados por el usuario para clasificar las frases verbales?

El **objetivo general** de esta tesis es diseñar una ontología para la validación semántica de las frases verbales de las interrelaciones en un esquema conceptual.

Para lograr este objetivo se plantean los siguientes **objetivos específicos** de la investigación:

1. Caracterizar el estado del arte del uso de las ontologías como herramienta en el diseño conceptual de bases de datos.
2. Diseñar una ontología para la validación semántica de interrelaciones.
3. Desarrollar una aplicación interactiva que inserte en la ontología los tipos de entidades y tipos de interrelaciones de un esquema conceptual para su validación semántica por las reglas implementadas en la ontología.

Los resultados de esta investigación serán incluidos en una herramienta de diseño de bases datos, desarrollada por el Laboratorio de Bases de Datos, y que permitirá dotar a dicha herramienta de la capacidad de validar semánticamente un esquema conceptual utilizando como base de conocimientos una ontología de dominio.

Como resultado del estudio realizado, y del desarrollo del trabajo sobre el cumplimiento de los objetivos se define la siguiente **hipótesis de investigación**: una ontología de dominio puede ser utilizada para validar semánticamente frases verbales en el proceso de diseño conceptual de bases de datos.

La hipótesis de investigación se comprobará utilizando una aplicación interactiva que verifique el proceso de validación semántica llevado a cabo por las reglas implementadas en la ontología.

Para exponer los resultados de este trabajo se ha decidido incluir 3 capítulos, organizados de la siguiente manera:

El Capítulo 1 está dedicado al marco-teórico referencial sobre el estado del arte del uso de las ontologías de dominio como herramienta en el diseño conceptual de bases de datos. El Capítulo 2 se muestra el proceso de diseño de una ontología para la validación de interrelaciones binarias, y el desarrollo de la aplicación interactiva. El Capitulo 3 a través de un caso de estudio, muestra el proceso de validación semántica de las interrelaciones de un esquema conceptual.

1. Las ontologías en el diseño conceptual de bases de datos

En este Capítulo se hace una exposición del estado del arte del uso de las ontologías de dominio como herramienta en el diseño conceptual de bases de datos. En el estudio se abordan los trabajos referentes a las ontologías y sus características más importantes.

1.1 Ontologías: Definición, principales aplicaciones, lenguajes formales para expresarlas y herramientas para su creación

En los últimos años el desarrollo de ontologías ha estado moviéndose del dominio de los laboratorios de Inteligencia Artificial a los escritorios de los expertos de un dominio dado.

Una ontología define un vocabulario común para investigadores que necesitan compartir información en un dominio. Ella contiene definiciones de conceptos básicos y sus relaciones que pueden ser interpretadas por una máquina.

Compartir el entendimiento común de la estructura de información entre personas y agentes de software es una de las más importantes metas al desarrollar ontologías (Musen, 1992; Gruber, 1993).

Permitir la reutilización de conocimiento de un dominio fue una de las fuerzas conductoras detrás recientes trabajos en la investigación sobre ontologías. Por ejemplo, modelos para diferentes dominios necesitan representar la noción de tiempo. Esta representación incluye las nociones de intervalo de tiempos, puntos en el tiempo, medidas relativas de tiempo, y cosas por el estilo. Si un grupo de investigadores desarrollan tal ontología en detalle, otros podrían simplemente reusarla en sus dominios. Además, si necesitamos construir una ontología grande, podemos integrar varias ontologías existentes que describen porciones del dominio más grande.

También se puede reusar una ontología general, tal como la ontología UNSPSC, y extenderla para describir un dominio de interés.

La explicitación de suposiciones de un dominio, que subyacen bajo una implementación, permite cambiar esas suposiciones fácilmente si el conocimiento del dominio cambia. Suposiciones modificadas explícitamente acerca del mundo en algún lenguaje de programación hacen que las suposiciones no solo sean difíciles de hallar sino también difíciles de cambiar, en particular para

alguien sin competencias en programación. Además, las especificaciones explícitas del dominio de conocimiento son útiles para nuevos usuarios que deben aprender el significado de los términos del dominio.

La separación del conocimiento del dominio del conocimiento operacional es otro uso común de las ontologías. Se puede describir la tarea de configuración de un producto a partir de sus componentes de acuerdo a especificaciones requeridas e implementar un programa que hace independiente esta configuración de los productos y componentes en sí (McGuinnes et al., 1998).

Analizar el conocimiento de un dominio es posible una vez que una especificación declarativa de los términos está disponible. El análisis formal de los términos es extremadamente valioso al intentar reusar ontologías existentes y al extenderlas.

Desarrollar una ontología es comparable a definir un conjunto de datos y sus estructuras para que otros programas los usen. Métodos de resuelven problemas, aplicaciones independientes del dominio, y agentes de software usan ontologías y bases de conocimiento construidos a partir de ontologías como datos.

Mientras la ontología sigue siendo un área fecunda de investigación en el campo de la filosofía, es actualmente materia de investigación, desarrollo, y aplicación en disciplinas relacionadas con la computación, la información y el conocimiento.

1.1.1 ¿Qué es una ontología?

La palabra Ontología proviene del griego y es la conjunción de los términos ontos y logos que denotan existencia y mundo respectivamente.

El desarrollo de estudios e investigaciones sobre el tema de las Ontologías, propician el surgimiento de determinados conceptos que se convierten en la base para el entendimiento del tema.

Existen diversas definiciones del concepto de ontología, una de las más ampliamente aceptadas es la siguiente de Thomas Gruber (Gruber, 1993):

"*Una Ontología es una especificación formal y explícita de una conceptualización compartida*"

De esta definición se desprende que una ontología es un modelo abstracto de algún fenómeno del mundo sobre el que se identifican los conceptos relevantes, que son representados por un lenguaje de representación formal y que es aceptado como mínimo por el grupo de personas que pretendan usarlas.

El término "conceptualización" se refiere a que toda ontología desarrolla un modelo abstracto del dominio o fenómeno del mundo que representa. Este modelo abstracto se basa esencialmente en el empleo de conceptos, atributos, valores y relaciones.

Con "especificación explícita" indica que una ontología supone la descripción y representación de un dominio concreto mediante conceptos, atributos, valores, relaciones, funciones, etc., definidas explícitamente. Las máquinas no pueden dar nada por supuesto o por obvio, y todo conocimiento (por básico que parezca, mientras sea necesario) debe ser explícitamente representado.

El término "formal" refleja que cualquier representación (concepto, atributo, valor, etc.) ha de ser expresada en una ontología mediante un formalismo siempre idéntico, de manera que pueda ser reutilizada y leída por cualquier máquina independientemente del lugar o de la plataforma o idioma del sistema que lo emplee.

Y por último y con gran importancia, el término "compartida" hace referencia a que una ontología debe, en el mejor de los casos, dar cuenta de conocimiento aceptado, como mínimo por el grupo de personas que deben usarla. En efecto, una ontología cumple con este término cuando dicha conceptualización y su representación formal y explícita ha sido favorablemente acogida por todos los usuarios de la misma. Ello permite por una parte distinguir claramente las ontologías de las bases de datos (en las que también puede hablarse de conceptualización y especificación formal y explícita mediante conceptos, atributos y valores, pero en la que el creador no tiene que lograr el consenso de nadie). Sin embargo, al mismo tiempo plantea una gran dificultad, pues en la práctica es casi imposible conseguir el consenso de todos los involucrados en un dominio específico. De ahí que, en la práctica, se considere irrealizable una ontología de carácter genérico o global, y sin embargo, se desarrollen ontologías en ámbitos mucho más restringidos, porque alcanzar aquí el consenso es factible (un banco para desarrollar servicios online para sus clientes, una empresa que desea una ontología sobre el conocimiento generado internamente por ella, etc.). Cuanto más genérico es el ámbito, en mayor medida el

proceso hasta alcanzar el consenso se produce mediante una guerra de estándares inicial (varios organismos lanzan sus ontologías, esperando que cada una de ellas alcance el consenso de los demás), de las que surgen con el tiempo 2 ó 3 estándares de factores por área o sector, normalizándose finalmente hasta alcanzar una variante con el consenso de todos.

Una definición de ontología más concreta la ofrece (Weigand, 1997):

"*Una ontología es una base de datos donde se describen conceptos del mundo o algún dominio en especifico, sus propiedades y como se relacionan los conceptos entre sí*"

Por tanto, aunque en filosofía una ontología es una explicación sistemática de la existencia, en los sistemas basados en el conocimiento, lo que existe es exactamente lo que se puede representar, y lo que se representa, mediante un formalismo declarativo, se conoce con el nombre de universo de discurso. El universo de discurso de una ontología es el conjunto de objetos que están representados en ella y sobre los cuales se puede hablar y razonar.

Cuando se habla de ontologías como "sistemas de representación de conocimiento" se debe especificar a que tipo de sistemas se refiere. En realidad, las ontologías se están empleando en todo tipo de aplicaciones informáticas en las que sea necesario definir concretamente el conjunto de entidades relevantes en el campo de aplicación determinado, así como las interacciones entre las mismas. Algunas ontologías se crean con el mero objetivo de alcanzar una comprensión del universo de discurso pertinente, ya que su creación impone una especificación detallada. Otras ontologías han sido creadas con un propósito general.

Las ontologías, consideradas como repositorios formales de conocimiento, siguen un ciclo de vida que modela desde su construcción, refinamiento, modificaciones, uso o explotación hasta su retiro. Alrededor de este ciclo se sitúa el reto de la disponibilidad de las ontologías, que incluye la necesidad de metodologías de construcción, herramientas que las soporten, métodos de evaluación, comprobación y metodologías de evolución como gestión de cambios y de versiones, entre otros. Las ontologías no son formalismos cerrados y están sujetos a procesos evolutivos. Es por eso que metodologías y herramientas que soporten estos procesos se hacen esenciales. El proceso de desarrollo de ontologías debe tener el apoyo necesario tanto desde el punto de vista metodológico como de herramientas que lo faciliten.

1.1.2 Componentes de una ontología

Independientemente del ámbito en que se desarrollen las ontologías, la base para su desarrollo es la conceptualización junto con un vocabulario para referirse a las entidades de un dominio particular. Las ontologías para representar el conocimiento precisan los siguientes componentes:

- **Clases** (o conceptos): Son las ideas básicas que se intentan formalizar de algún fenómeno o dominio del mundo. Las clases se suelen organizar en taxonomías a las que se les pueden aplicar los mecanismos de herencia. Pueden ser clases de objetos, métodos, planes, estrategias, procesos de razonamiento, etc.
- **Propiedades** (o *slots* o roles): Describen los rasgos y atributos de los conceptos.
- **Restricciones** (o facetas): Restricciones sobre los *slots*; la cardinalidad, tipo y rango, son facetas de los *slots*. La cardinalidad se refiere al número de valores que puede tener un *slot*. Cadena de caracteres (*string*), Números, verdadero/falso (*boolean*) e instancias son tipos de valores comunes. El rango de un *slot* es usado para los tipos de valores Instancia y específica cuales objetos en la ontología pueden asignarse a dicho *slot*.
- **Instancias**: Se utilizan para representar objetos determinados de un concepto.
- **Relaciones**: Representan la interacción y enlace entre los conceptos del dominio. Suelen formar la taxonomía del dominio. Por ejemplo: subclase-de, parte-de, parte-exhaustiva-de, conectado-a, etc. Es habitual que el formalismo sólo permita representar relaciones binarias. Cada una de las ontologías suele permitir el uso de un número acotado y cerrado de relaciones. Cada una de las relaciones puede estar dotada de determinadas propiedades que determinan su comportamiento.
- **Funciones**: Son un tipo concreto de relación donde se identifica un elemento mediante el cálculo de una función que considera varios elementos de la ontología. Por ejemplo, pueden aparecer funciones como categorizar-clase, asignar-fecha, etc.
- **Axiomas**: Son teoremas que se declaran sobre relaciones que deben cumplir los elementos de la ontología. “Para todo A que cumpla la condición C, entonces A es B”. Permiten dejar constancia de que ciertos valores de propiedades introducidos son coherentes con las restricciones de la ontología, o bien inferir posteriormente valores de atributos que no se han introducido explícitamente. De esta forma, a través de los axiomas es posible inferir conocimiento no codificado explícitamente en la ontología.

1.1.3 Otros conceptos relacionados con ontologías

Otros conceptos destacables a la hora de hablar de ontologías son:

Anotación: Es el proceso de relleno de instancias a partir de texto libre. Existen dos maneras de anotar texto: la más usual es la inclusión de etiquetas semánticas dentro del texto que se está procesando. Esto implica que el formato del texto debe ser editable y procesable. En el caso de no disponer de este formato, la segunda manera consiste en rellenar las instancias directamente en el modelo, dejando el texto original sin modificar.

Herencia: Propiedad de la relación 'es_un' que permite que las clases relacionadas (heredadas) cuenten con los atributos de la clase con la cual se relacionan (clase padre).

Herencia múltiple: Se da cuando una clase dada hereda o cuenta con las propiedades de dos clases padre con las que establece dos relaciones del tipo 'es_un'.

Derivación: Organización de las clases de la ontología en un árbol de jerarquía mediante sucesivas relaciones 'es_un' (también llamadas kind_of, is_a, o herencias) con la propiedad de herencia. Esta organización permite el encadenamiento sucesivo de herencias desde las clases de nivel superior a las clases situadas en niveles inferiores, llamadas clases derivadas.

Primitiva: categoría de una ontología que no puede ser definida en términos de otras categorías en la misma ontología. Un ejemplo de una primitiva es el concepto del tipo Punto en la geometría de Euclides. El significado de una primitiva no está determinado por una definición con una forma cerrada (closed-form), sino por axiomas que especifican cómo se relaciona a otras primitivas. Una categoría que es una primitiva en una ontología debe no ser primitiva en un refinamiento (refinement) de aquella ontología.

Base de conocimiento: término informal para referirse a una colección de información que incluye una ontología como un componente. Además de una ontología, una base de conocimiento debe contener información especificada en un lenguaje declarativo tal como reglas lógicas o sistemas expertos, aunque también incluye información no estructurada o formalizada expresada en lenguaje natural o en lenguaje de procesado.

Refinamiento: Un refinamiento (matización) de cada categoría de una ontología A, a alguna categoría de otra ontología B, se denomina un refinamiento de A. Cada categoría en A debe corresponder a una categoría equivalente en B, pero algunas de A deben ser equivalentes a no

primitivas en B. El refinamiento define un orden parcial de ontologías: si B es un refinamiento de A, y C es un refinamiento de B, entonces C es un refinamiento de A; si dos ontologías son refinamientos una de la otra, entonces deben ser isomórficas.

1.1.4 Tipos de ontologías

La amplia utilización de las ontologías en aplicaciones informáticas ha propiciado que se distingan los aspectos más representativos para determinados campos de aplicación. En tal sentido se distinguen tres tipos fundamentales de ontologías (Steve et al., 1998):

- Ontologías de dominio: Se representa el conocimiento especializado pertinente de un dominio o subdominio, como la medicina, las aplicaciones militares, la cardiología, etc.
- Ontologías genéricas: Se representan conceptos generales y fundacionales del conocimiento como las estructuras parte/todo, la cuantificación, los procesos o los tipos de objetos.
- Ontologías representacionales: Se especifican las conceptualizaciones que subyacen a los formalismos de representación del conocimiento, por lo que también se denominan meta-ontologías (*meta-level* o *top-level*).

A estos tres tipos, (Guarino, 1998) añade las ontologías que se crean para una actividad o tarea especifica (denominada *task ontologies*), como por ejemplo la venta de productos o el diagnostico de una enfermedad y las ontologías creadas para una aplicación especifica. Se podría pensar que al implementar ontologías de este tipo, se debería tratar de representar todo el conocimiento disponible de cada clase en cuestión, pero realmente esto no es recomendable según (Noy et al., 2001) sólo el conocimiento mas relevante y que es realmente necesario para la aplicación, pues de otra manera seria mas ineficiente su utilización, se comparte conocimiento inútil, y los procesos de inferencia son más lentos.

Hay otras posibles clasificaciones de ontologías atendiendo a diversos criterios, una de ellas podría hacerse atendiendo a su destino. Según este criterio, se pueden distinguir los siguientes tipos de ontologías (Guarino, 1996):

- Ontologías lingüísticas: Están relacionadas con aspectos lingüísticos, gramáticos, semánticos y sintácticos destinados a su utilización por los seres humanos.
- Ontologías no lingüísticas: Para ser utilizadas por robots y agentes inteligentes.

- Ontologías mixtas: Combinan las características de las anteriores.

Otra forma de identificarlas es por el grado o nivel de abstracción y razonamiento lógico que permitan:

- Ontologías descriptivas: incluyen descripciones, taxonomías de conceptos, relaciones entre los conceptos y propiedades, pero no permiten inferencias lógicas.
- Ontologías lógicas: permiten inferencias lógicas mediante la utilización de una serie de componentes como la inclusión de axiomas, reglas, etc.

Las personas mediante las ontologías se representan en su cabeza el mundo que los rodea. Estas ontologías no son explicitas, ya que no se detallan en su escrito ni se establecen de forma jerárquica o matemática. Se usan ontologías en las que *Automóvil* representa un medio de transporte y tiene cuatro ruedas. ¿Se formaliza este tipo de ontologías? No, sería innecesario: los automóviles son tan usuales que todos comparten la información de lo que son. Lo mismo ocurre al pensar en el dominio familiar: se sabe que una familia dispone de varios miembros, que un hijo no puede tener mas de un padre y una madre biológicos, que los padres tienen o han tenido padres, etc. No se necesita ser explícitos con este conocimiento, pues todo el mundo lo sabe. Sin embargo, cuando se tratan términos pocos comunes o cuando se quiere que estos términos sean procesados por máquinas, se precisa explicitar las ontologías; esto es, desarrollarlas de forma que las máquinas comprendan.

Las máquinas carecen de las ontologías con las que nosotros contamos para entender el mundo y comunicarse entre ellas; por eso necesitan ontologías explícitas. Las ontologías explícitas se pueden expresar de muchas maneras. Como mínimo, deben incluir un vocabulario de términos, con la definición de cada uno.

Dependiendo del grado de formalidad, las ontologías explícitas se clasifican en informales, semi-informales y formales. Las primeras se expresan directamente en un lenguaje natural. Las segundas se expresan en una forma estructurada y restringida de algún lenguaje natural. Las terceras se expresan en lenguajes estructurados, como RDF. Por último, las ontologías formales definen los términos mediante lenguajes lógico-matemáticos cuyos símbolos se definen exactamente y sin ambigüedades; en consecuencia, estas ontologías permiten emplear teoremas y demostraciones. Los dos últimos tipos de ontologías permiten que las aplicaciones puedan usar las definiciones de los conceptos del dominio y sus relaciones. Así como los tres primeros tipos

de ontologías pueden contener términos ambiguos o inconsistentes, las ontologías formales no los permiten.

1.1.5 Aspectos a tener en cuenta para el diseño de una ontología

A la hora de diseñar una ontología no se puede olvidar que la misma debe comunicar de manera efectiva el significado de sus términos. Las definiciones serán lo más objetivas posibles y deben explicarse también en lenguaje natural, siendo así la claridad un aspecto esencial.

Otro aspecto a tener en cuenta será la coherencia, de manera que la ontología debe permitir hacer inferencias que sean consistentes con las definiciones.

La extensibilidad es también muy importante, deben anticiparse nuevos usos para así poder permitir extensiones y especializaciones.

En las ontologías se deben especificar a nivel de conocimiento, sin que dependa de una codificación particular a nivel de símbolo, por esta razón la especificidad no pude faltar en las mismas.

Por último la precisión será un aspecto primordial, en el diseño se debe hacer la menor cantidad de suposiciones acerca del mundo modelado.

1.1.6 Aplicaciones de las ontologías

Las ontologías proporcionan una comprensión compartida del conocimiento de un dominio, que puede ser comunicada entre personas y sistemas heterogéneos. Para poder reutilizar conocimientos de otros sistemas es necesario conocer y estar conforme con la terminología y su significado. Por ello, se llaman acuerdos ontológicos a los acuerdos terminológicos necesarios para reutilizar conocimiento. Se trata de convertir la información en conocimiento.

El amplio desarrollo de la investigación en el campo de las ontologías, permite un aumento notable de los usos de éstas, basados en los estudios de Gruber (Gruber, 1995) y de Gruninger y Lee (Gruninger et al., 2002), donde se refieren como los más frecuentes en el campo de la denominada Web semántica, la ingeniería del conocimiento o los sistemas de información.

Algunas de las aplicaciones de las ontologías son:

- Web semántica: Una de las aplicaciones en la Web semántica es la indexación de un sitio Web con apoyo de una ontología terminológica y comienza con la extracción de los

términos más relevantes de cada página, después de asociar a estos términos conceptos candidatos, se evalúa la capacidad de representación de la página de cada uno de estos conceptos, que determina su nivel de representatividad, y finalmente se construye el índice.

- Ingeniería del conocimiento: En el procesamiento del lenguaje natural una ontología puede mantener la definición de elementos gramaticales del lenguaje y sus relaciones, permitiendo, por ejemplo, el análisis sintáctico de un texto.
- Sistemas de información: Una de las aplicaciones que con frecuencia se referencia en la literatura es la interoperatividad entre sistemas heterogéneos, donde las ontologías se presentan como una solución para lograr una integración inteligente. Con una ontología terminológica, se pueden organizar los términos que son usados en interacciones entre sistemas heterogéneos, de manera que reconozca cuándo una aplicación está usando un término que es más general o más específico que otro que está en uso por otra aplicación.

En la actualidad las ontologías son ampliamente usadas en portales web, las colecciones multimedia, diseño de documentos web, agentes inteligentes, comercio electrónico, gestión de la imagen audiovisual, etc. En efecto, las ontologías, en principio, han de potenciar el intercambio de datos en contextos informáticos y digitales gracias a los fundamentos semánticos que se encuentran en ellas.

Una aplicación de las ontologías relacionado con el tema de esta tesis es la integración en un sistema de varias bases de datos. Al integrar información proveniente de distintas fuentes surge un gran problema a resolver: la heterogeneidad de la misma. Esta heterogeneidad es tanto a nivel sintáctico como semántico. A nivel sintáctico la interoperabilidad significa integrar datos que están presentes en los sistemas de información en diferentes lenguajes y representaciones de datos. Para alcanzar este nivel de integración generalmente se emplea XML (eXtensible Markup Language). Por otro lado, al hecho que un término pueda representar distintos conceptos o un concepto pueda ser representado por distintos términos se lo denomina heterogeneidad semántica. Para resolver este problema se utilizan las ontologías.

En esta dirección se expondrán más adelante algunos trabajos relacionados con la integración de bases de bases de datos y el diseño conceptual de bases de datos mediante el uso de ontologías.

1.1.7 Principales ventajas y desventajas de las ontologías

En la actualidad, la principal ventaja que aportan las ontologías, desde el punto de vista de las bases de datos y de la recuperación de información, tiene que ver con el empleo simultáneo de bases de datos de muy distinta naturaleza, características y formato a las que poder interrogar simultáneamente para recuperar la información buscada. Estas bases de datos pueden ser de contenido tan dispar como documentación textual poco estructurada (la clásica que forma parte de los buscadores en Internet), documentación fotográfica, documentación geográfica, museográfica, urbanística, espacios naturales, etc.

Otra de las ventajas de las ontologías es la posibilidad de utilizar la información preexistente en dichas bases de datos a través de la propia ontología sin tener que renunciar a la base de datos original de partida, de manera que pueden convivir y seguir empleándose simultáneamente bases de datos iniciales y ontología. Tan sólo se debe tener presente la necesidad de actualizar la ontología con los nuevos datos que vayan añadiéndose a las bases de datos. Tampoco es necesario introducir de nuevo manualmente -aunque podría hacerse- las bases de datos en la nueva estructura que hemos desarrollado con la ontología. Se puede automatizar el proceso de volcado de la información mediante el desarrollo informático de programas que transformen el formato de la base de datos inicial en el formato y lugar adecuado en la nueva estructura ontológica. Lógicamente, cuanto más estructurada se halle la información inicial, más sencilla es la transformación correspondiente.

Una de las principales desventajas, en cambio, radica en la imposibilidad de utilización directa de las ontologías previamente desarrolladas. Es preciso crear programas de interrogación de la ontología, pues las herramientas existentes apenas permiten la visualización de los datos previamente introducidos. Eso exige la participación de especialistas en programación dentro de los equipos de desarrollo de ontologías si deseamos utilizarla posteriormente.

Otra desventaja surge al tratar de sacar el máximo partido de las amplísimas posibilidades de recuperación que proporcionan las ontologías, lo que se consigue mediante el desarrollo de buscadores denominados semánticos. Estos buscadores semánticos involucran la programación de complejos algoritmos que echan mano de herramientas de procesamiento de lenguaje natural poco desarrollados aún. En consecuencia, las ontologías posibilitarían una mejora sustancial en la precisión de la recuperación gracias a la posibilidad real de “entender” los términos y conceptos

presentes en las preguntas, pero para ello dependen de complejos algoritmos de comprensión del lenguaje natural que no han alcanzado todavía un desarrollo suficiente. En resumen, suponen una herramienta con grandes posibilidades que, sin embargo, dependen para desarrollar todo su potencial de avances en otras áreas ajenas como la Inteligencia Artificial o el Procesamiento del Lenguaje Natural.

1.1.8 Lenguajes formales para expresar ontologías

Son varias las herramientas y los lenguajes existentes para el desarrollo de ontologías. Las herramientas de construcción de ontologías son similares unas a otras, aún cuando muestren problemas de convergencia y de adaptación a los cambiantes lenguajes. Los lenguajes de marcado de las ontologías están todavía en fase de desarrollo, aunque con una progresión muy clara y evidente y con claras implicaciones en la propia elaboración y puesta en marcha de estos dispositivos de representación del conocimiento.

Existen múltiples lenguajes asociados con ontologías, sin embargo, no todos ellos van a permitir el mismo nivel de expresividad. Al seleccionar un lenguaje para la definición de ontologías, deben de tenerse en cuenta los siguientes requisitos:

- Debe poseer una sintaxis bien definida para ser capaz de poder leer las ontologías, que se definan mediante este lenguaje. Este es un requisito bastante obvio.
- Debe tener semánticas bien definidas, para poder ser capaz de comprender las ontologías.
- Debe tener suficiente expresividad para poder capturar varias ontologías.
- Debe ser fácilmente maleable desde/hacia otros lenguajes ontológicos.
- Debe ser eficiente a la hora de realizar razonamiento.

RDF y RDFS

RDF (*Resource Description Framework)* es un marco de trabajo desarrollado por el W3C. Es un lenguaje de etiquetado, creado mediante sintaxis XML, que define un modelo de datos para describir recursos, mediante enunciados en forma de tripletas sujeto-predicado-objeto. Como RDF es un vocabulario XML, puede describir recursos de forma externa a éstos o de forma embebida, siempre y cuando estos recursos tengan sintaxis XML.

Por otro lado, RDFS *(RDF Schema)* es un vocabulario RDF que permite describir recursos mediante una orientación a objetos similar a la de muchos lenguajes de programación como Java.

Para ello, proporciona un mecanismo para definir clases, objetos y propiedades; relaciones entre clases y propiedades; y, restricciones de dominio y rango sobre las propiedades. Además, RDF cuenta con semánticas y un modelo gráfico intuitivo proporcionados por un modelo teórico inspirado por la lógica de primer orden.

OIL

Desarrollado en el marco del proyecto On-To-Knowledge y el proyecto IBROW4. Es el primer lenguaje de representación de ontologías basado en estándares W3C (World Wide Web Consortium). Basado en la Web para la representación e inferencias, estructurado por varias capas de sublenguajes, entre ellas tenemos como capa base RDF, a la que cada una de las capas subsiguientes añade alguna funcionalidad y mayor complejidad. Esto posibilita, teóricamente, la reutilización de agentes diseñados para RDFS, que podrían procesar ontologías OIL. Utiliza la sintaxis del lenguaje XML y está definido como una extensión de RDFS. Se basa tanto en la lógica descriptiva (declaración de axiomas) y en los sistemas basados en frames (taxonomías de clases y atributos). Pretende conseguir la interoperabilidad semántica entre recursos Web. La principal carencia de este lenguaje es la falta de expresividad para declarar axiomas. OIL se encuentra estructurado en varias capas de sublenguajes. La capa base o núcleo de OIL coincide plenamente con RDFS y cada una de las capas superiores añade funcionalidad y complejidad a su capa subyacente. Esto posibilita, teóricamente, la reutilización de agentes diseñados para RDFS, que podrían procesar ontologías OIL. Entre las limitaciones de OIL se pueden destacar las siguientes: no ofrece la posibilidad de sobreescritura de valores heredados de una superclase; presenta falta de expresividad en la declaración de axiomas (reglas); y no soporta dominios concretos.

DAML + OIL

Resultado de la cooperación entre OIL y DARPA unificando los lenguajes DAML (DARPA's Agent Markup Language) y OIL (Ontology Inference Layer). Se basa en estándares del W3C. El lenguaje DAML se desarrolló como una extensión del lenguaje XML y RDF, donde para extender el nivel de expresividad de RDFS, DAML+ OIL hereda muchas de las características de OIL, pero se aleja del modelo basado en clases (frames) y potencia la lógica descriptiva. Es más potente que RDFS para expresar ontologías. En la última revisión del lenguaje (DAML+OIL) ofrece ya un rico conjunto de elementos con los cuales se pueden crear ontologías y marcar la

información para que sea legible y comprensible por los sistema de computo. También funciona como formato de intercambio. Sin embargo, este lenguaje presenta algunas carencias debido a su complejidad conceptual y de uso, complejidad que se intentó solventar con el desarrollo de OWL. No obstante, se desarrollaron muchas aplicaciones que utilizan DAML-OIL y también existen herramientas para convertir DAML a OWL.

OWL

OWL (Web Ontology Language) es un lenguaje de etiquetado semántico para publicar y compartir ontologías en la Web. Se trata de una recomendación del W3C, y puede usarse para representar ontologías de forma explícita, es decir, permite definir el significado de términos en vocabularios y las relaciones entre aquellos términos (ontologías). En realidad, OWL es una extensión del lenguaje RDF y emplea las tripletas de RDFS, aunque es un lenguaje con más poder expresivo que éste. Se trata de un lenguaje diseñado para usarse cuando la información contenida en los documentos necesita ser procesada por programas o aplicaciones, en oposición a situaciones donde el contenido solamente necesita ser presentado a los seres humanos. OWL surge como una revisión al lenguaje DAML-OIL y es mucho más potente que éste. Al igual que OIL, OWL se estructura en capas que difieren en la complejidad y puede ser adaptado a las necesidades de cada usuario, al nivel de expresividad que se precise y a los distintos tipos de aplicaciones existentes (motores de búsqueda, agentes, etc.).

OWL utiliza la conexión proporcionada por RDF para añadir las siguientes capacidades a las ontologías:

- Capacidad de ser distribuidas a través de varios sistemas
- Escalable a las necesidades de la Web
- Compatible con los estándares Web de accesibilidad e internacionalización
- Abierto y extensible

OWL fue diseñado para cumplir los siguientes objetivos:

- Las ontologías OWL deben ser públicas y apropiadas para el compartimiento, con el objetivo de que sistemas diferentes puedan referenciar una misma ontología.
- OWL permite que las ontologías ínter operen entre sí cuando los mismos conceptos son representados en ontologías diferentes, propiciando una red de ontologías.

- Debe ser posible detectar inconsistencias entre diferentes ontologías que son contradictorias.
- OWL debe propiciar un balance entre expresividad y tratamiento computacional, que permita el razonamiento.
- OWL debe ser fácil de usar e intuitivo.
- OWL debe ser compatible con otros estándares como XML o UML.
- OWL debe ser compatible con la internacionalización (uso en diferentes lenguajes).

OWL se divide en tres sublenguajes OWL-Lite, OWL-DL y OWL-Full, cada uno de los cuales proporciona un conjunto definido sobre el que trabajar. OWL-DL basado en lógica descriptiva, tiene semánticas bien definidas y es soportado por algoritmos completos de razonamiento. OWL-DL está compuesto por capas de subconjuntos de RDFS, permitiendo una estricta separación de clases, propiedades y sujetos. OWL-Full es más completo e incluye RDFS sin restricciones. OWL-Lite es más sencillo OWL-Lite ya que es un subconjunto de OWL-DL

1.1.9 Herramientas para el diseño de ontologías

Existen herramientas para diseñar gráficamente ontologías, que permiten que el diseñador vea la información que está codificada en la ontología de forma gráfica, reduciendo así la sobrecarga de información. Cada una de estas herramientas tienen características que permiten valorar cuan efectivas son al utilizarlas para definir una ontología, por lo que en la presente sección, de las herramientas más populares y representativas, se realiza una caracterización basada fundamentalmente en:

- Descripciones generales de las herramientas, la cual incluye información sobre el desarrollo de dichas herramientas.
- La arquitectura del software y evolución de la herramienta, que incluye la arquitectura de la herramienta, cómo la herramienta puede ser extendida con otras funcionalidades y módulos, cómo las ontologías son almacenadas (bases de datos, fichero texto, etcétera).
- Interoperabilidad con el desarrollo de otros lenguajes y herramientas de ontologías, que incluye las capacidades de interactuar entre ellas.
- Representación del conocimiento, donde se presenta el paradigma de representación del conocimiento que está debajo del modelo de conocimiento de la herramienta. Esta

representación del conocimiento es muy útil para saber qué y cómo puede ser modelado el conocimiento con la herramienta. Se analizará si la herramienta mantiene cualquier lenguaje para la construcción de axiomas.

- Servicios de inferencias, se analizará si la herramienta tiene alguna máquina de inferencia, y si dicha herramienta presenta algún chequeo de restricción y/o consistencia.

Sobre la base de estos aspectos a continuación se realiza una caracterización de algunas de las herramientas disponibles para el desarrollo de ontologías.

OntoEdit

OntoEdit es un ambiente de diseño que soporta el desarrollo y mantenimiento de una ontología usando los medios gráficos. El proceso de desarrollo de la ontología en OntoEdit es basado sobre su propia metodología, y es originalmente basado en Common KADS. Dos herramientas, OntoKick y Mind2Onto, son preparadas para soportar la fase de captura de la ontología. OntoKick es diseñada para especialistas que están familiarizados con el proceso de desarrollo del software, e intentan construir estructuras relevantes para edificar la descripción de la ontología informal. Mind2Onto es una herramienta gráfica para capturar relaciones informales entre los conceptos. Es fácil de usar porque tiene una buena interfaz visual y permite una identificación individual de relaciones entre conceptos. Sin embargo, esto es necesario para convertir el mapa dentro de una organización más formal para generar una ontología. Su paradigma trabaja sobre el modelo de representación de lenguajes neutral tanto para los conceptos, relaciones y axiomas. Esta herramienta le permite al usuario editar una jerarquía de conceptos o clases, donde estas clases pueden ser abstractas o concretas, lo cual indica si se permite o no realizar instancias directamente de las clases. Una clase puede tener varios nombres, lo que esencialmente es una manera de definir sinónimos para esa clase. Como la mayoría de otras herramientas, OntoEdit emplea la arquitectura cliente/servidor, donde las ontologías son manejadas en un servidor y acceden múltiples clientes, y se modifica solo uno. Además esta herramienta está basada en plugin. Todas las versiones de OntoEdit están disponibles en una versión libre y otra profesional, donde las versiones profesionales incluyen además un conjunto de plugins.

OILEd

OILEd es un editor para la ontología gráfica desarrollado por la Universidad de Manchester, que permite al usuario construir ontologías usando DAML+OIL. El modelo de conocimiento de

OILEd está basado sobre el modelo de conocimiento DAML+OIL. Las clases son definidas en términos de sus superclases y restricciones de propiedad con axiomas adicionales que capturan las relaciones extensas como el disjoint. Un aspecto importante de OILEd es el uso del razonador FaCT, para la clasificación de ontologías y el chequeo de consistencia, lo cual permite al usuario describir sus clases de ontologías. DAML+OIL Esquema RDF se usa para cargar y guardar las ontologías. Además, la herramienta leerá y escribirá las jerarquías de concepto en puro RDF y dará las definiciones de la ontología como HTML para la navegación y como SHIQ para la clasificación más tarde por el razonador FaCT.

Protégé

Desarrollada por el grupo Stanford Medical Informatics en la Stanford University School of Medicine con la ayuda de National Library of Medicine, National Science Foundation, y Defense Advanced Research Projects Agency. De todas las herramientas existentes, Protégé, es la de más amplio uso actualmente. Es un software libre y de código abierto, usado por numerosos expertos en diferentes dominios. Permite el desarrollo de ontologías al hacer más fácil el trabajo de forma simultánea con clases e instancias. Así, una instancia singular puede ser usada en el mismo nivel de la definición de una clase, y una clase puede ser almacenada como una instancia. Por su parte, los atributos (slots), que inicialmente fueron empleados solo dentro de las clases, ahora son elevados al mismo nivel que las clases, y pueden tener significado sin pertenecer a alguna de ellas. A su vez, a efectos de facilitar el ingreso de datos, cada clase se asocia a un Form, y cada slot se asocia a un SlotWidget (objeto de edición del slot según el tipo de dato que sea), de forma que la interfase al usuario pueda ser diseñada y modificada para facilitar al usuario el ingreso de instancias.

Protégé es un ambiente visual de diseño y registro de ontologías, en las primeras versiones orientado a frames y slots desarrollado en Java, puede correr en forma independiente en un PC y funciona perfectamente bajo Windows. Dispone de un conjunto importante de plugins con orígenes diversos, con la posibilidad de que sean conectados otros de estos módulos externos y proporcionar una funcionalidad adicional, haciendo así a Protégé extensible, aumentando su utilidad y adaptación según las necesidades. En el 2003 fue extendido para soportar OWL a través de la creación del OWL plugin, el cual también proporciona interfaces para razonadores de lógica descriptiva.

Las ontologías creadas en Protégé pueden ser exportadas a varios formatos como RDF(S), OWL y XML(S). En las versiones más recientes se tiene la posibilidad de editar clases y sus características, acceder a motores de razonamiento, editar y ejecutar consultas y reglas, comparar ontologías, visualizar relaciones entre conceptos y obtener instancias usando procesamientos configurables por el usuario.

Protégé tiene su propio lenguaje interno para definir ontologías, PAL (Protégé Axiom Language), pero permite también trabajar con RDF y OWL de modo transparente. El PAL plugin, agrega funcionalidad a las reglas y restricciones básicas de Protégé mediante la incorporación de un editor de reglas, verificador sintáctico, la ejecución y verificación de las reglas y consultas.

Cada regla o consulta se compone de un nombre (: PAL-NAME), una descripción (: PAL-DESCRIPTION), un rango que es la declaración de las clases y eventualmente los slots que estarán involucrados (: PAL-RANGE), y una declaración de la regla o consulta en KIF (: PAL-STATEMENT). Las reglas se pueden asociar a clases o slots, como restricciones, lo que hace opcional la declaración de esas clases en el slot de rango de la regla. De todas maneras para completitud siempre se declaró el rango de las clases para las instancias involucradas en Consultas y Restricciones.

El razonador a utilizar puede verificar y evaluar las reglas, consultas o restricciones, indicando si son correctas, dando advertencias y/o presentando las instancias que verifican las consultas o que violan las reglas.

Constituye un entorno abierto y fácil de extender, que ha generado en torno suyo toda una comunidad que contribuye activamente a ampliar el entorno con todo tipo de contribuciones en forma de plugins, haciendo de esta herramienta un entorno sumamente potente.

Como contrapartida, no tiene muy desarrolladas las verificaciones sintácticas, ni tampoco indicadores útiles para que el desarrollador pueda encontrar el origen de los problemas. Además tiene problemas para manejar reglas y grandes consultas. Protégé, brinda varias posibilidades que permiten hacer extensivas otras funcionalidades que se necesiten incorporar a las ontologías; así como el uso de determinadas facilidades que se integran al mismo. Particularmente, en Protégé esto se logra a través de los plugins.

1.2 Las ontologías como herramientas en el diseño conceptual de bases de datos

Las ontologías han sido propuestas como un medio importante de representación del conocimiento para el diseño de bases de datos.

Las bases de datos mantienen su utilidad de forma tradicional y dirigida al comercio y constituyen el núcleo de todo sitio web con fines comerciales; para llevar a cabo esta función es importante realizar un buen diseño y en particular cuando se necesitan mezclar bases de datos tradicionales y basadas en web. Sin embargo, el diseño de bases de datos es difícil y se ha considerado un arte y se reconoce que escasean los buenos diseñadores.

Se dispone de numerosas herramientas para asistir a los diseñadores, sin embargo estas no contienen información acerca del mundo real y la forma en que el mismo opera, y delegan en los usuarios para que proporcionen toda la información acerca del dominio de la aplicación.

Un próximo paso en lograr herramientas inteligentes, sería la inclusión del conocimiento del mundo real, organizado por dominios de aplicación.

Un método para capturar el conocimiento del mundo real es utilizar una ontología que contenga los hechos y las relaciones del dominio de aplicación. Las ontologías han recibido un notable reconocimiento como un mecanismo útil para hacer esto.

Una ontología define los términos básicos y las relaciones comprendidas en el vocabulario de un área, así como reglas para combinar términos y relaciones entre términos.

Una ontología puede tener términos de muy alto nivel o específicos a un dominio.

En el área del diseño de bases de datos se ha reconocido la utilidad de las ontologías para clasificar entidades e interrelaciones y contribuir a su automatización.

El reconocimiento dentro de un esquema conceptual de la semántica de las interrelaciones y su clasificación es un aspecto importante en el diseño conceptual. En este contexto (Storey, 1993) hace un análisis de la semántica de algunas interrelaciones poco conocidas y cómo estas interrelaciones pueden ser capturadas ya bien sea manual o utilizando una herramienta y se evalúa el impacto que tienen estas interrelaciones dentro de una base de datos. Se presenta una taxonomía de siete tipos de interrelaciones semánticas (*inclusion, possession, attachment,*

attribution, antonym, synonym, case) y un prototipo de sistema para examinar la semántica de las interrelaciones. El analizador obtiene la cardinalidad de la interrelación y trata de encontrar la mejor interpretación de la misma, luego hace inferencias acerca de los atributos, llaves, restricciones de integridad e interrelaciones omitidas por último identifica las interrelaciones por su sinónimo y se eliminan las interrelaciones redundantes.

En (Dey et al., 1999) se propone un marco de trabajo para el análisis de las interrelaciones, el cual ayuda a determinar interrelaciones ternarias y de grado mayor que en ocasiones son incorrectamente representadas y se definen reglas para la correcta representación de las interrelaciones n-arias. Se analiza la dependencia de existencia de una interrelación. En el análisis de las interrelaciones se tiene en cuenta: la cardinalidad, el grado, la recursividad y las restricciones entre las interrelaciones.

A pesar de que la interrelación es una parte importante del modelo Entidad-Relación el significado de la frase verbal que las identifica es un tema poco tratado y es la clave para cualquier análisis semántico de un esquema conceptual.

En esta dirección (Storey, 2001; Storey et al., 2004) proponen una ontología para la clasificación de las interrelaciones basado en las frases verbales, la cual permite entender, capturar y refinar la semántica de una aplicación. Esta ontología clasifica una frase verbal en una o más categorías y en ocasiones depende de la interacción con el usuario. Una de sus aplicaciones es la comparación de interrelaciones entre varias bases de datos.

En (Sugumaran et al., 2006) se propone una metodología de apoyo a la creación y evaluación de bases de datos que hace uso del conocimiento sobre un dominio de aplicación específico almacenado en una ontología de dominios. La metodología propuesta es implementada en un prototipo llamado OMDDE (*Ontology Management and Database Design Environment*), que es utilizado para la creación de esquemas conceptuales Entidad-Relación. La arquitectura del sistema está compuesta por un cliente HTML y un servidor Web el cual consiste de: 1) un componente administrador de ontologías, 2) un componente para el diseño de base de datos, y 3) repositorios de ontologías.

En (Storey, 2005; Purao et al., 2005) se presenta una ontología basada en el dominio de una aplicación para la clasificación de las frases verbales de las interrelaciones. Se define una estructura de tres capas, donde el núcleo está conformado por la clasificación de las frases

verbales según las categorías fundamentales propuestas, las otras dos capas están dadas por dos factores adicionales que influyen en la clasificación: el primero es el contexto local, facilitado por las entidades que forman parte de la interrelación, es decir los nombres que rodean la frase verbal, y el segundo es el contexto externo, que representa el dominio en el que la interrelación se está usando.

Como se observa en la Figura1, la primera capa refleja las categorías fundamentales para la categorización de las frases verbales. La segunda y tercera capa representan el contexto local, y global respectivamente, proporcionando los dos factores adicionales.

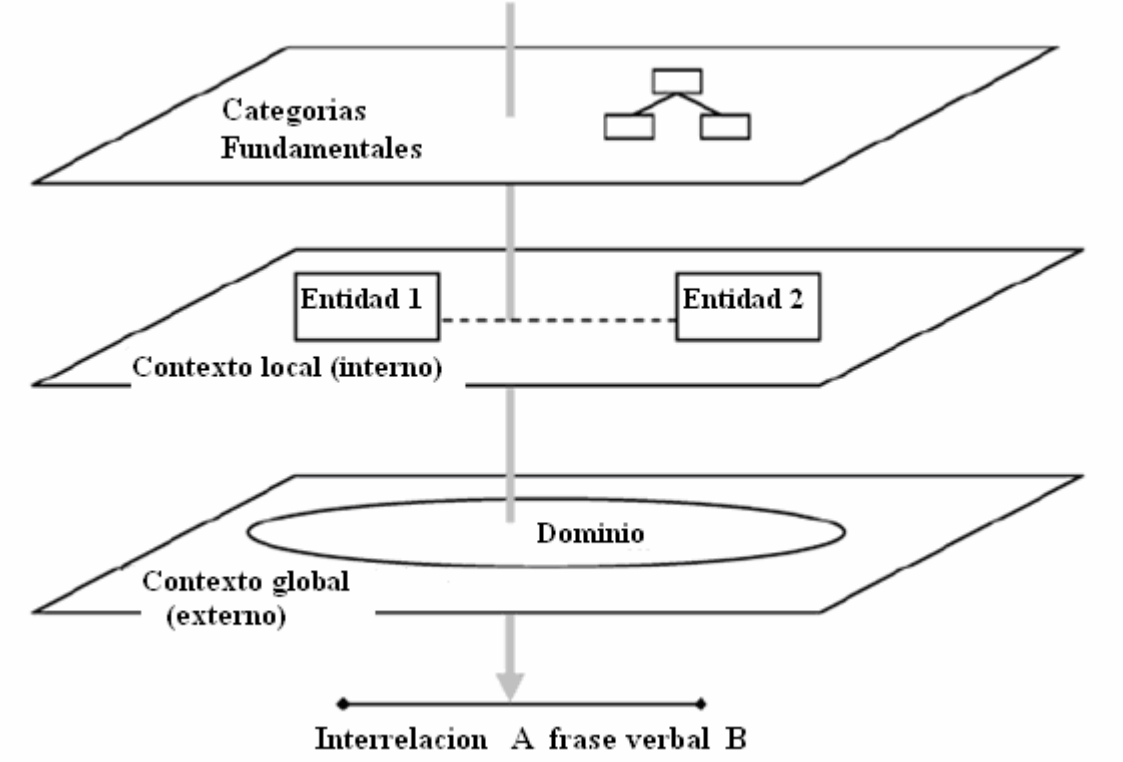


Figura 1: Capas para la clasificación de entidades e interrelaciones.

En la primera capa están las categorías fundamentales que consisten en clases de interrelaciones que ofrecen una categorización de las interrelaciones encontradas en el diseño conceptual de bases de datos. Se definen tres categorías generales de las interrelaciones que reflejan cómo ellas aparecen en diseños conceptuales de bases de datos: S*tatus*, *Change of Statu*s e *Interaction*. Estas categorías son apropiadas ya que permiten capturar tres orientaciones importantes de una entidad hacia otra en la modelación conceptual de bases de datos.

La categoría *Status* representa la orientación duradera que captura la consecuencia de eventos. La noción de durabilidad puede ser diferente en distintos ámbitos, lo que pone de relieve la necesidad de tener en cuenta el contexto externo o de dominio, la tercera capa en la ontología. Para muchas bases de datos, también es importante la consecuencia del evento. Esta categoría capta esta consecuencia. Por ejemplo, el caso <adquirir una propiedad> puede dar lugar a una orientación duradera de la propiedad entre un propietario y un valor. La categoría de status capta este hecho como una cosa duradera o permanente con respeto a la otra. Estas interrelaciones duraderas se expresan: una entidad **(**A**)** <es algo> con respecto a la otra entidad **(**B**)**, por ejemplo, A <es dueño de> B.

La Figura 2 muestra la jerarquía de las clasificaciones primitivas de la categoría *status*. Las cinco principales subcategorías: *structural, influential, temporal, spatial, attitudinal*, son etiquetas que comunican el significado inherente en esta subcategoría. De éstas, *spatial* y *attitudinal* son primitivas. La subcategoría *spatial* se refiere a las interrelaciones que describen la localización de entidades en el espacio como Ciudad <está sobre> Longitud. La subcategoría *attitudinal* capta la disposición emocional de una entidad a otra, como gustos y disgustos. Un ejemplo de esta primitiva es Cliente <le gusta> Producto.

Las otras subcategorías, structural, influential y temporal están conformadas por primitivas agrupadas según su significado. La Tabla 1 muestra ejemplos de frases verbales clasificados en las diferentes primitivas.

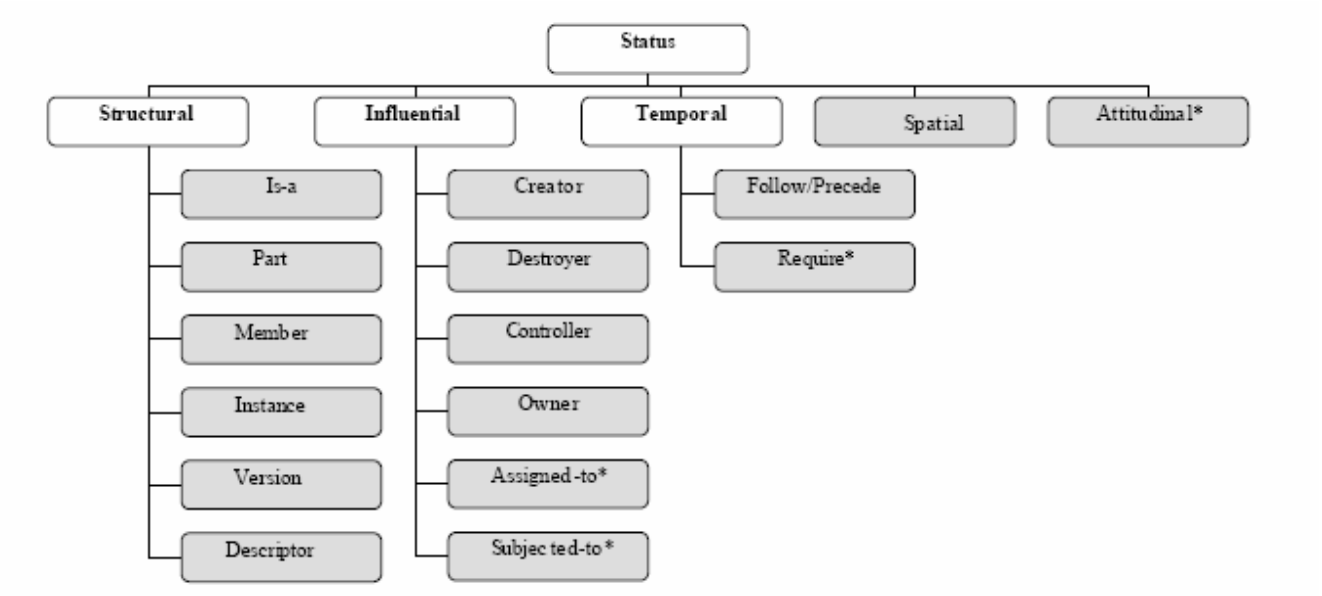


Figura 2: Primitivas de la categoría Status.

	Primitive	Example
1	A <is a> B	Pilot <is an> Employee
2	A <is member of> B	Professor <is member of> Department
3	A <is part of> B	Car <has> Engine
4	A <is instance of> B	Video Tape <is copy of> Movie
5	A <is version of> B	Draft <is version of> Manuscript
6	A <is descriptor of> B	Document <defines> Task
7	A <is creator of> B	Author <writes> Book
8	A <is destroyer of> B	Tennant <terminates> Lease
9	A <is owner of> B	Company <owns> Building
10	A <is in control of> B	Manager <leads> Team
11	A <is assigned to> B	Employee <assigned to> Project
12	A <is subjected to> B	Industry <regulated by> Law
13	A <follows or precedes> B	Rental <follows> Reservation
14	A <requires> B	Construction <requires> Approval
15	A <is next to> B	Office <is-next-to> Elevator
16	A <has attitude towards> B	Customer <likes> Product

Tabla 1: Ejemplos de frases verbales pertenecientes a las primitivas de Status.

La categoría *Change of Status* representa la manifestación de un evento como la transición de un estado a otro. La conceptualización de esta categoría se basa en una extensión, logrando la comprensión de cómo cada primitiva en la categoría status pueden evolucionar durante el ciclo de vida empresarial. Cada etapa en la relación ciclo de vida, se asigna a diferentes primitivas de status. Las primitivas pueden ser utilizadas para ilustrar el ciclo de vida que recorren los procesos de creación, obtener propiedad y destrucción. El ciclo de vida puede ser dividido de forma lógica en: la intención, la tentativa de adquirir, la transición a la adquisición, la intención de renunciar, la tentativa de renunciar, y la transición a la renunciación. La Tabla 2 muestra esta información superpuesta en los diferentes estados dentro del ciclo de vida. La subcolumna bajo las primitivas de change of status muestra el significado capturado en cada una de ellas: intent, attempt y transition.

	Primitive		Example
1	A <wants-to-be owner of> B	*intent*	Customer <wants to own> Product
2	A <attempts-to-become owner of> B	*attempt*	Customer <orders> Product
3	A <becomes owner of> B	*transition*	Customer <receives> Product
	Status Primitive: Customer <owns> Product		
4	A <dislikes-being owner of> B	*intent*	Company <wants to sell> Product
5	A <attempts-to-give-up ownership of> B	*attempt*	Company <offers> Product
6	A <gives-up ownership-of> B	*transition*	Company <sells> Product

Tabla 2: Ejemplos de frases verbales pertenecientes a las primitivas de Change of Status.

La categoría *interaction* describe la comunicación de corta duración entre dos entidades o el funcionamiento de una entidad en otra. La interacción puede causar un cambio en una de las entidades. Por ejemplo, una entidad puede "manipular" otra (manipulate), o causar circulación de los demás a través del tiempo o el espacio (transmit, receive). Dos entidades pueden interactuar sin causar cambios (comunicate, observe). Una entidad puede interaccionar con otra también por medio del rendimiento (operate, serve). La Figura 3 muestra las primitivas de la categoría *Interaction* con ejemplos en la Tabla 3.

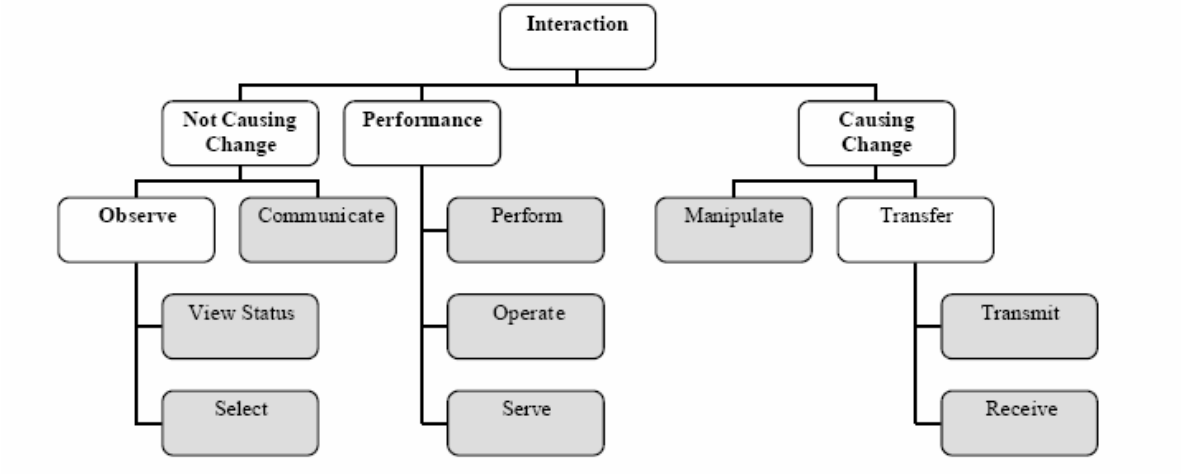


Figura 3: Primitivas de la categoría Interaction.

	Primitive	Example
1	View Status	Analyst <analyses> Requirements
2	Select	Customer <selects> Product
3	Communicate	Modem <negotiates with> Phone Line
4	Perform	Developer <tests> Software
5	Operate	Pilot <flies> Plane
6	Serve	Employee <serves> Customer
7	Manipulate	Instructor <grades> Exam
8	Transmit	Bank <remits> Payment
9	Receive	Warehouse <receives> Shipment

Tabla 3: Ejemplos de frases verbales pertenecientes a las primitivas de Interaction.

Estas categorías proporcionan una clasificación completa de todas las interrelaciones en el diseño conceptual de bases de datos (Purao et al., 2005).

La segunda capa captura el contexto interno, se toma en cuenta la naturaleza de las entidades en torno a la frase verbal, necesaria para la comprensión de la interrelación. En este epígrafe se muestra una clasificación de entidad que comprende las categorías: actor, acción y artefactos.

- Los actores son entidades que son capaces de ejecutar una acción independiente
- Las acciones son la realización de un acto.
- Los artefactos son objetos inanimados que no sean capaces de una acción independiente.

Después de clasificadas las entidades, las primitivas pueden ser especificadas válidas para cada par de tipos de entidad, por ejemplo no tiene sentido interrelaciones donde la frase verbal sea de tipo Perform y las entidades sean actor, esta primitiva es válida cuando una entidad es actor y la otra acción. En las Tablas 4 y 5 se definen las interrelaciones válidas para cada categoría.

Entity 1	Entity 2	Valid Status Primitives															
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Actor	Actor	√	√		√			√	√	√	√	√				√	√
Actor	Action									√	√	√	√				√
Actor	Artifact						√	√	√	√	√	√					√
Action	Action	√		√	√	√							√	√	√		
Action	Artifact						√						√				
Artifact	Artifact	√		√	√	√	√									√	

Tabla 4: Validación de las primitivas de Status basadas en el contexto Entidad.

Reglas para la validación de las interrelaciones pertenecientes a la categoría Status según el contexto entidad:

1. Ambas entidades deben pertenecer a la misma categoría.
2. Ambas entidades deben pertenecer a la categoría Actor.
3. Ambas entidades deben pertenecer a la misma categoría, y deben ser de tipo Action o Artifact.
4. Ambas entidades deben pertenecer a la misma categoría.
5. Ambas entidades deben pertenecer a la misma categoría, y deben ser de tipo Action o Artifact.
6. Una de las entidades debe pertenecer a la categoría Artifact.
7. Una entidad debe pertenecer a la categoría Actor, y la otra a Actor o Artifact.
8. Una entidad debe pertenecer a la categoría Actor, y la otra a Actor o Artifact.
9. Una de las entidades debe pertenecer a la categoría Actor.
10. Una de las entidades debe pertenecer a la categoría Actor.

11. Una de las entidades debe pertenecer a la categoría Actor.
12. Una de las entidades debe pertenecer a la categoría Action.
13. Ambas entidades deben pertenecer a la categoría Action.
14. Ambas entidades deben pertenecer a la categoría Action.
15. Ambas entidades deben pertenecer a la misma categoría, y deben ser de tipo Action o Artifact.
16. Al menos una de las entidades pertenece a la categoría Actor.

Entity 1	Entity 2	Valid Interaction Primitives								
		1	2	3	4	5	6	7	8	9
Actor	Actor			√			√			
Actor	Action	√	√		√					
Actor	Artifact	√	√	√		√	√	√	√	√
Action	Action									
Action	Artifact	√	√		√					
Artifact	Artifact	√	√	√		√	√	√	√	√

Tabla 5: Validación de las primitivas de Interaction basadas en el contexto Entidad.

Reglas para la validación de las interrelaciones pertenecientes a la categoría Interaction según el contexto entidad:

1. Una de las entidades debe ser Actor o Artifact y la otra Action o Artifact.
2. Una de las entidades debe ser Actor o Artifact y la otra Action o Artifact.
3. Las entidades pueden ser cualquier combinación de Actor y Artifact.
4. Una de las entidades debe ser Actor o Artifact, y la otra debe ser Action.
5. Una de las entidades debe ser Actor o Artifact y la otra Artifact.
6. Las entidades pueden ser cualquier combinación entre Actor y Artifact.
7. Una de las entidades debe ser Actor o Artifat, y la otra Artifact.
8. Una de las entidades debe ser Actor o Artifat, y la otra Artifact.
9. Una de las entidades debe ser Actor o Artifat, y la otra Artifact.

Las limitaciones señaladas para las primitivas de Status se aplican a las de Change of Status debido a que estas últimas captan el ciclo de vida de las primitivas de status.

La tercera capa captura el contexto externo, es decir, el dominio en el que la interrelación se utiliza, reflejando la naturaleza dependiente del dominio de las frases verbales. Considere la posibilidad del uso de la frase verbal <Abrir> en el contexto de un teatro y de un banco. En un teatro la interrelación Actor <Abre> Puerta, su frase verbal se clasifica en la primitiva <manipulates> de la categoría Interaction, mientras que en el banco la interrelacion, Cajero <Abre> Cuenta, la misma frase verbal pertenece a la primitiva <is-creator-of>.

Esta ontología fue desarrollada en un prototipo que realiza las clasificaciones de las frases verbales de forma interactiva y sólo acepta interrelaciones binarias y no hace la comparación de las cardinalidades de las interrelaciones. La arquitectura de este prototipo consiste en un módulo de clasificación (lo que facilita la clasificación de una determinada interrelación), y un módulo de comparación (facilitando la comparación de las interrelaciones a fin de evaluar su equivalencia). La base de conocimientos y las interfaces de usuarios creadas, tienen funcionalidad en dirección al primer módulo, es decir, la clasificación de nuevas interrelaciones.

1.3 Conclusiones parciales

Del análisis crítico de la revisión bibliográfica referente al uso de las ontologías como herramienta de ayuda a la modelación conceptual de bases de datos se concluye que:

1. La validación semántica depende del dominio de la aplicación por lo que resulta difícil definir un criterio de validez generalizado.
2. Las ontologías pueden utilizarse como base de conocimiento de términos para asistir en el diseño conceptual de bases de datos.
3. Los actuales desarrollos en el uso de las ontologías de dominio en el diseño conceptual de bases de datos muestran resultados en forma de prototipos de aplicaciones.
4. La clasificación reportada en la literatura de las entidades e interrelaciones en categorías permite disponer de criterios en el proceso de validación semántica de esquemas conceptuales.
5. El diseño de una ontología de varios niveles es una alternativa adecuada como soporte de información para el proceso de validación de esquemas conceptuales.

2. Diseño de una ontología para la validación semántica de interrelaciones de esquemas conceptuales

A partir del marco teórico-referencial de esta tesis, este Capítulo estará dedicado a fundamentar la elaboración de una ontología que permita la validación semántica de esquemas conceptuales Entidad Relación.

2.1 Diseño de la ontología para la validación semántica de interrelaciones

El diseño de la ontología que se describe a continuación se basa en los resultados publicados en (Purao et al., 2005) los cuales fueron descritos en el Capítulo 1.

Para la representación de la ontología se seleccionó como lenguaje a OWL porque es unos de los más completos ya que añade más vocabulario; es decir es mucho más expresivo que otros para describir relaciones, como por ejemplo que *RDF-Schema*. Además OWL, permite razonar y definir operaciones sobre la información y describir relaciones de más alto nivel entre clases, sin entrar en detalles de cómo se estructura dicha información. A la hora de diseñar un sistema, el desarrollador no tiene que pensar tanto en qué campos o atributos componen cada clase, sino modelar cómo se relacionan unas con otras. Otra ventaja de OWL es que está basado en XML, lo cual facilita la compartición de información entre diferentes sistemas.

Entre las herramientas de creación de ontologías se seleccionó a Protégé, por ser la herramienta más robusta comparada con las otras que se analizaron. La plataforma Protégé, soporta dos caminos principales para la construcción de ontologías, que son Protégé Frames y Protégé OWL. Para el desarrollo de las ontologías se selecciona la segunda metodología; una ontología OWL puede incluir las descripciones de clases, propiedades y sus instancias.

La ontología que se propone en este trabajo está formada por una capa, que incluye las categorías fundamentales y las restricciones impuestas sobre las categorías de interrelaciones. Las categorías fundamentales consisten en clases de interrelaciones que proporcionan una clasificación de las interrelaciones encontradas en la modelación conceptual de bases de datos. Las restricciones son reglas que se aplican en dependencia de la clasificación dada a un tipo de interrelación. Esta capa es independiente del contexto del dominio del discurso que se esté

modelando. La elaboración de la capa del dominio de la aplicación propuesta en (Purao et al., 2005) queda fuera del alcance de este trabajo y será acometida como continuación de esta investigación.

El diseño de una ontología que contiene la clasificación de interrelaciones permite utilizarla como base de conocimiento en la validación semántica de esquemas conceptuales. De esta manera una herramienta de diseño puede suministrarle como instancias a la ontología los tipos de entidades y tipos de interrelaciones previamente clasificados, y a continuación se evalúan las reglas correspondiente que permiten identificar si una interrelación es válida o no. Un prototipo de esta aplicación se presenta en el Capítulo 3.

2.1.1 Definición de clases, atributos e instancias

Las clases base de la ontología para la representación de interrelaciones son: (ver Figura 5)

Entity: Esta clase representa las entidades. Contiene como subclases los tipos de entidades. Cada entidad será una instancia de la subclase a la que pertenece.

VerbPhrase: Esta clase representa las frases verbales. Contiene como subclases la clasificación de las frases verbales. Cada frase verbal será una instancia de la subclase que representa la primitiva a la que pertenece (ver Figura 4).

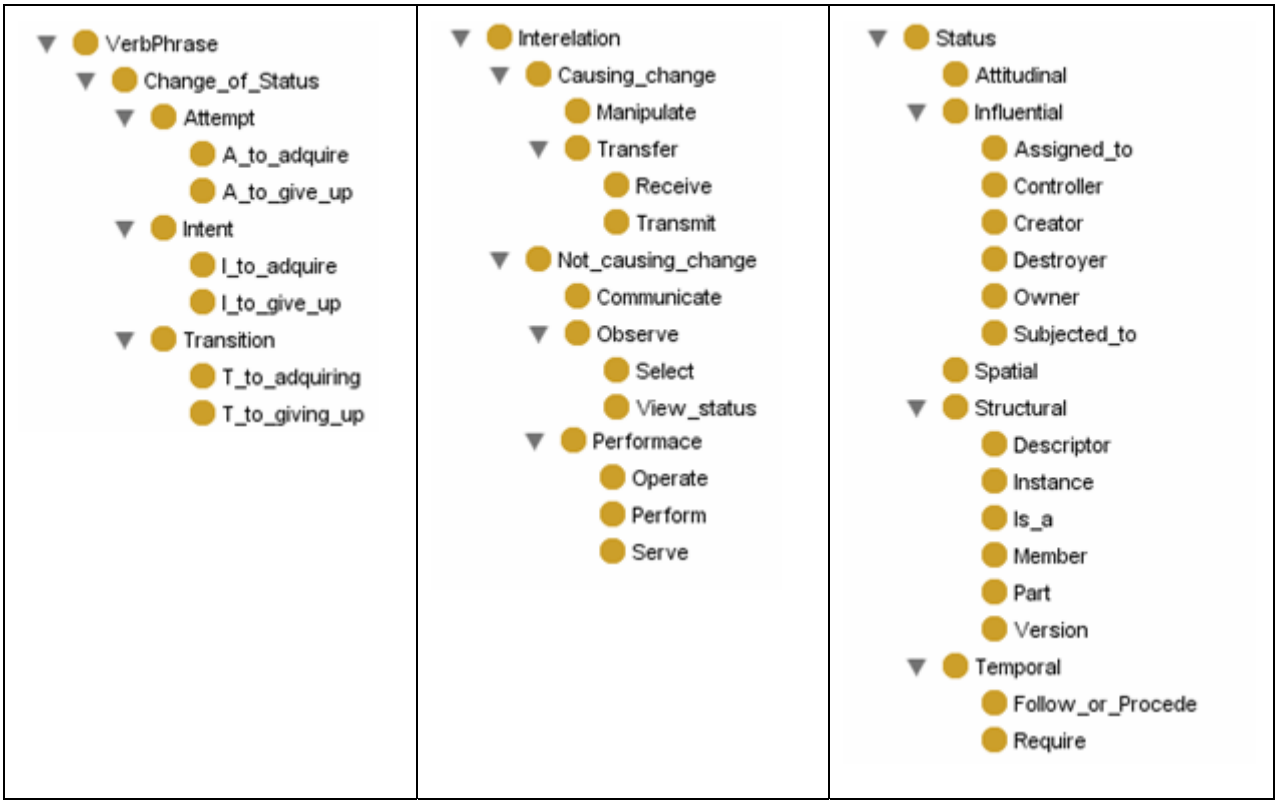


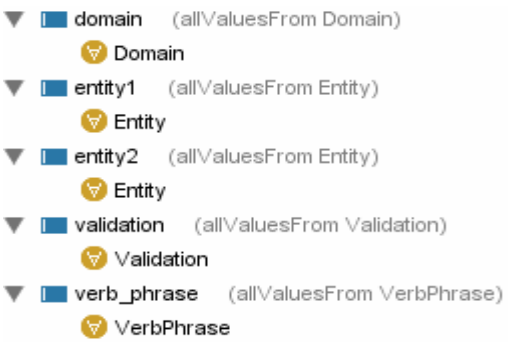
Figura 4: Subclases de la clase VerbPhrase.

Domain: Esta clase representa los dominios. Sus instancias son los diferentes dominios de las interrelaciones.

Validation: Esta es una clase auxiliar, necesaria para darle valor Válido a la propiedad *validate* de la clase patrón (si es válido) después de analizadas las reglas.

Patrón: Esta clase representa todas las interrelaciones. Cada interrelación será una instancia de esta clase. Contiene atributos que son instancia de las clases anteriores, conformando un patrón de la interrelación.

Propiedades y sus restricciones:



Las cinco propiedades de objeto relacionan el individuo de patrón con los individuos de las otras clases.

Todas las propiedades están restringidas por el cuantificador universal (todos) indicando que todos los valores de esa propiedad pertenecerán a la clase indicada (allValuesFrom Clase)

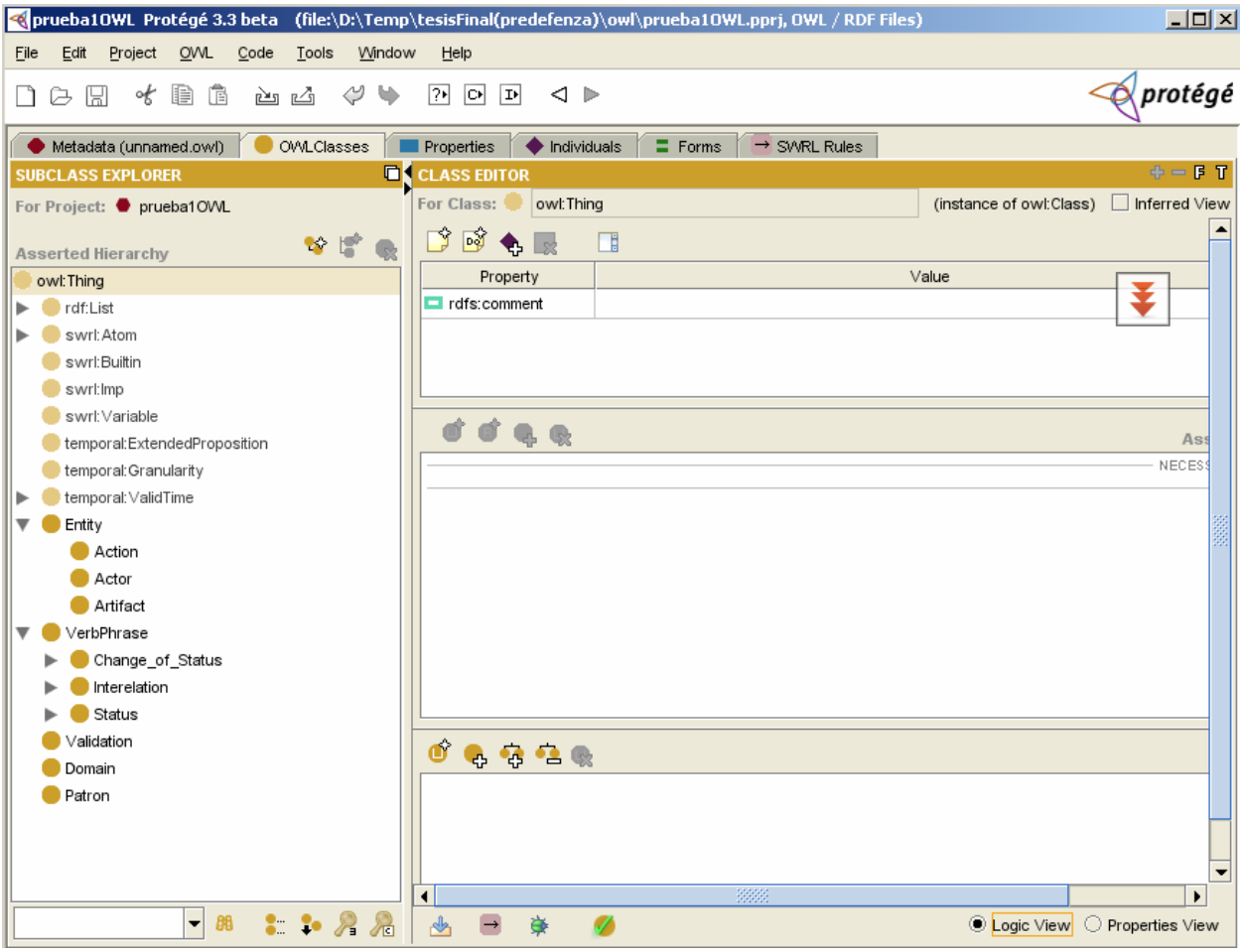


Figura 5: Definiciones de las clases en el Protégé.

2.1.2 Definición de las reglas en la ontología

Para la definición de las reglas se utilizó el Lenguaje de Reglas de la Web Semántica (SWRL, *Semantic Web Rule Language*).

Una ontología en OWL contiene una secuencia de hechos y axiomas (de subclases, de equivalencia de clases, restricciones de propiedades). SWRL propone extenderlos con axiomas de regla.

Un axioma de regla consiste en un antecedente (cuerpo) y un consecuente (cabeza), cada uno de los cuales está compuesto por un conjunto (puede ser vacío) de átomos.

En la sintaxis inteligible por seres humanos de SWRL, una regla tiene la siguiente forma:

Antecedente => consecuente

De manera informal, una regla puede interpretarse como una indicación de que si el antecedente es cierto, entonces el consecuente también es cierto. Por tanto una regla en SWRL tiene la forma de una relación de implicación entre la cabeza y el cuerpo.

Utilizando la sintaxis (notación lógica clásica), si un patrón con entidades de tipo Actor y frase verbal de tipo Is_a es válido, se escribiría de la siguiente forma:

Patrón(?x) ∧ Actor(?y) ∧ entity1(?x, ?y) ∧ Actor(?z) ∧ entity2(?x, ?z) ∧ Is_a(?a) ∧ verb_phrase(?x, ?a) → validation(?x, Válido)

Donde Patrón, Actor e Is_a representan clases, entity1, entity2, verb_phrase y validation son propiedades y ?x,?y ,?z y ?a son variables.

Interpretación: Si ?x en una instancia de la clase Patrón y ?y una instancia de la clase Actor y la propiedad entity1 del individuo ?x tiene como valor ?y y ?z es una instancia de la clase Actor y la propiedad entity2 del individuo ?x tiene como valor ?z y ?a es una instancia de la clase Is_a y la propiedad verb_phrase del individuo ?x tiene como valor ?a implica que la propiedad validation del individuo ?x tomará valor Válido.

Para la definición de las reglas en la herramienta Protégé se activó el tab SWRL en Menú Project → Configure… (ver Figura 6).

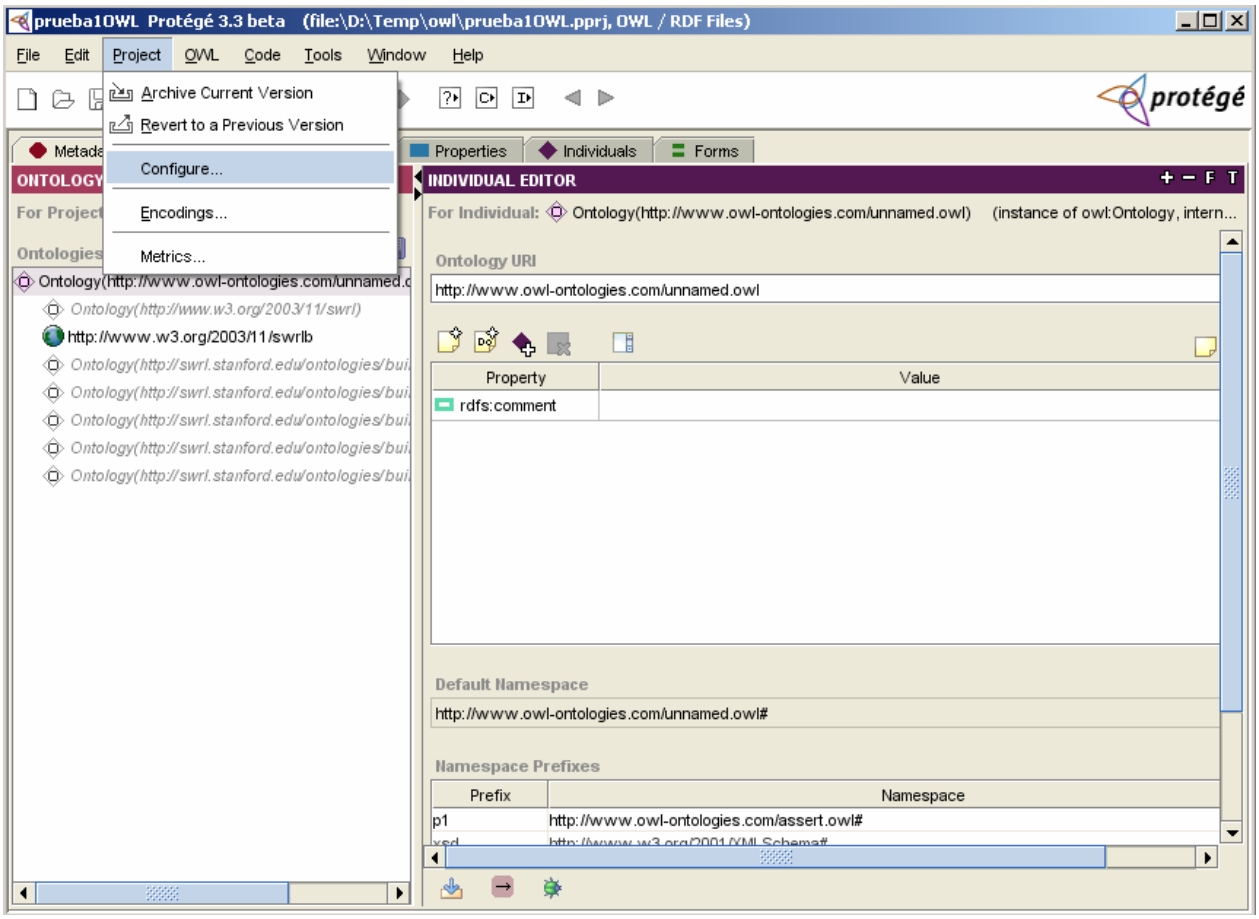


Figura 6: Activación del SWRL Tab.

Luego se seleccionó la pestaña SWRL Rules (ver Figura 7) y se activó el botón Activate SWRL para realizar la edición de las reglas.

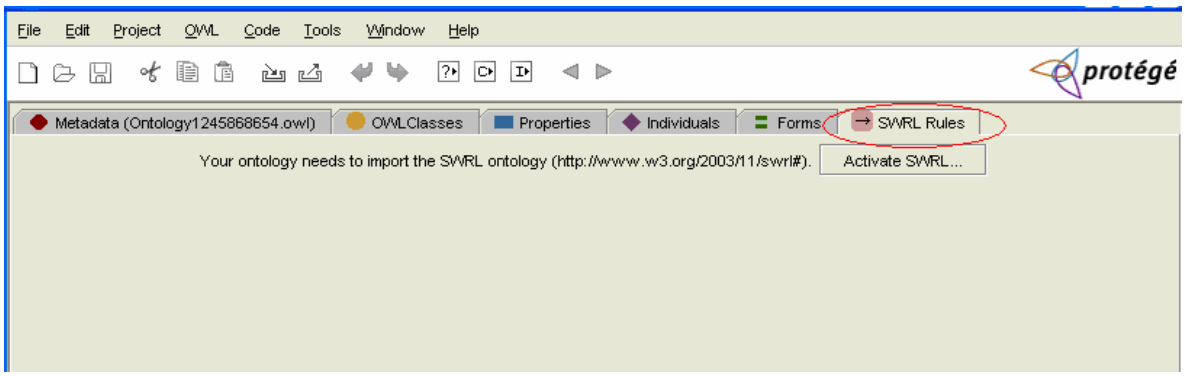


Figura 7: SWRL tab activado.

Finalmente en la Figura 8 se muestra el conjunto de reglas definidas.

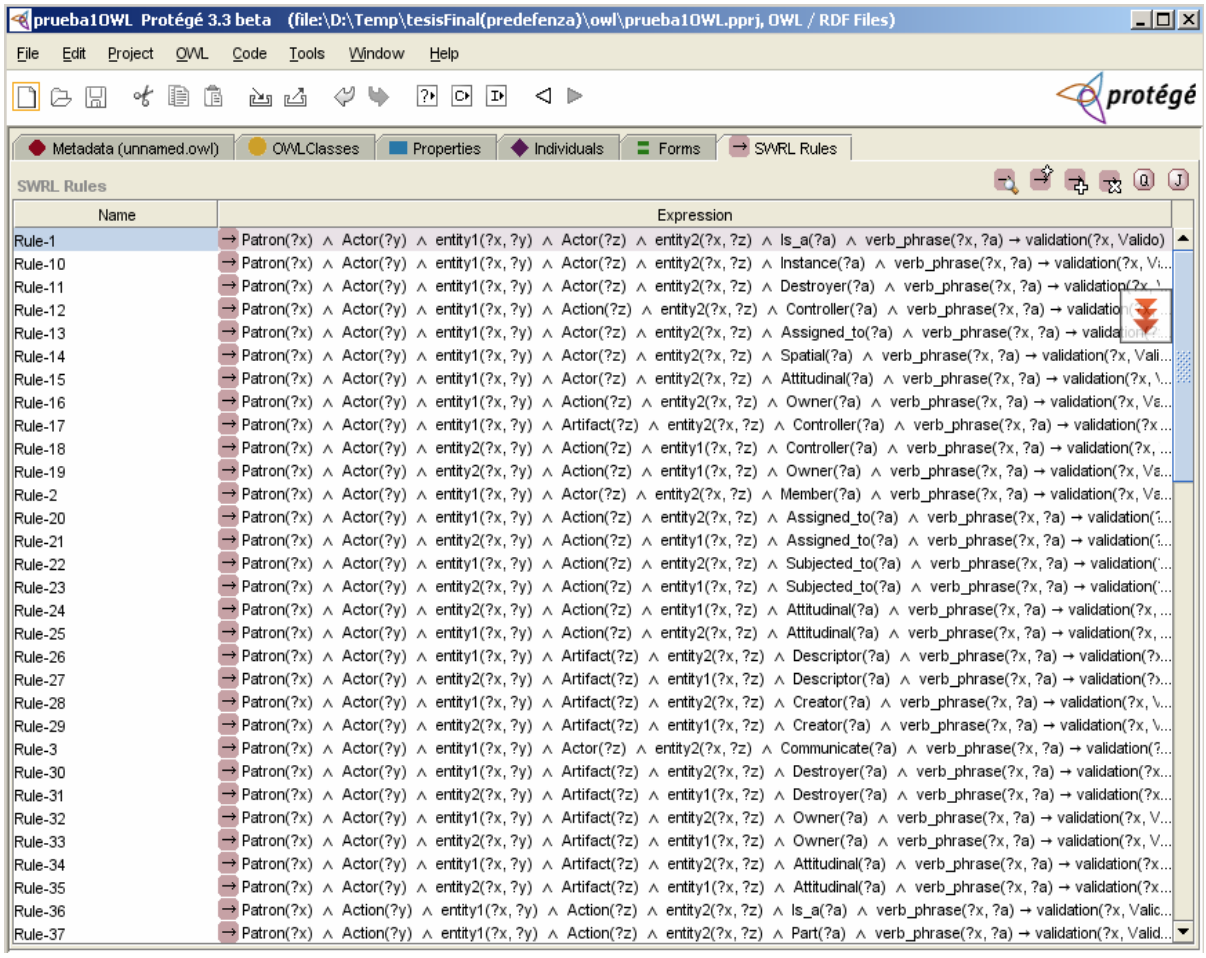


Figura 8. Conjunto de reglas.

La ontología quedará guardada en un fichero owl, al cual, mediante un prototipo de aplicación, se le agregará las instancias

2.2 Diseño de la aplicación interactiva

Con vista a lograr un buen funcionamiento de la aplicación interactiva se diseñó un conjunto de clases que realizan las operaciones necesarias para satisfacer los objetivos de la misma.

2.2.1 Descripción de las principales clases y sus métodos

Primeramente se diseñó la clase más importante Operaciones_Xml que contiene los métodos encargados de la lectura y escritura del fichero owl, el cual tiene una sintaxis de xml:

<?xml version="1.0"?>

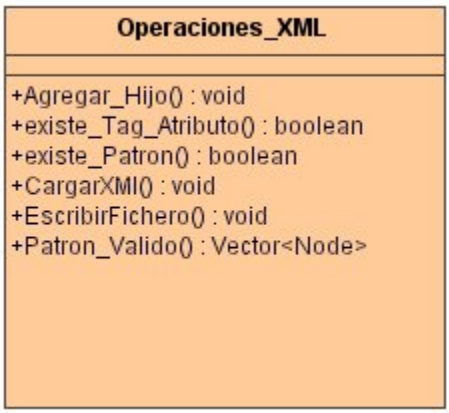


Figura 9: Clase principal.

package ValSemRel;

public class Operaciones_XML {

private static void Agregar_Hijo (Node nodoprincipal, Document doc, NodeList list,

String clase,String atributo,String valor)

Este método busca en el fichero owl si el atributo que se va a entrar está, si es así lo inserta creando un nuevo atributo del tipo "rdf:resource" y le agrega delante al valor el símbolo de numero (#), en caso contrario al nuevo atributo a crear se le especifica la clase a la que pertenece y sería de tipo "rdf:ID", el valor se escribe normalmente.

Ejemplo:

Si existe el atributo:

<entity1 rdf:resource="#profesor"/>

En caso contrario:

<entity1>

<Actor rdf:ID="profesor"/>

</entity1>

private static boolean existe_Tag_Atributo(Document doc, NodeList list,String nombreNodo,String NombreAtributo) {

list = doc.getElementsByTagName(nombreNodo); //function que devuelve la lista de nodos.

int i = 0;

while (i < list.getLength()&& (list.item(i).getAttributes().getLength()== 0

```
            || !list.item(i).getAttributes().item(0).getNodeValue().equals(NombreAtributo))) {
            i++;
        }
        return i < list.getLength();
    }
```

Este es el método que busca en la lista de nodos si el atributo existe, si lo encuentra devuelve verdadero, en caso contrario devuelve falso.

private static boolean existe_Patron(Document doc,Vector<Par_Comparar> vectorComparar)
Busca en la lista de nodos Patrón si existe la interrelación que se quiere insertar comparando cada atributo que lo conforma. Devuelve verdadero si el Patrón existe, falso en otro caso.

public static void CargarXMl(String NombreFicheroEntrada,String NombreFicheroSalida,String ent1,String Clasif_ent1, String dom,String ent2,String Clasif_ent2, String verbp, String Clasif_verbp)
Este método carga el fichero owl, en un document y escribe en él los datos de la nueva interrelación, si esta no se encuentra en el fichero. Para realizar estas operaciones se auxilia de los métodos privados que son la clave para intercambiar con el fichero.
Obtención del fichero:

```
DOMParser parse = new DOMParser();
parse.parse(NombreFicheroEntrada);
Document doc=parse.getDocument();
```

```
static void EscribirFichero(Document doc, String XMLFileName) {
    FileWriter file = null;
    OutputFormat of = new OutputFormat("XML", "UTF-8", true);
    XMLSerializer serializer = new XMLSerializer();
    serializer.setOutputFormat(of);
    file = new FileWriter(XMLFileName);
    serializer.setOutputCharStream(file);
    serializer.serialize(doc);
```

```
serializer.endDocument();
file.close();
  }
```

Este método es el encargado de escribir finalmente los datos en el fichero owl.

```
public static Vector<Node> Patron_Valido (String NombreFicheroEntrada) throws
SAXException, IOException{
        DOMParser parse = new DOMParser();
        parse.parse(NombreFicheroEntrada);
        Document doc = parse.getDocument();
        NodeList list = doc.getElementsByTagName("Patron");
        int i = 0;
        Vector<Node> patrones = new Vector<Node>();
        while (i < list.getLength()) {
           String valorInvestigar;
           String NombreCampo;
           int j = 0;
           Boolean band = true;
           while (j < list.item(i).getChildNodes().getLength()) {
              Node aux = list.item(i).getChildNodes().item(j);
               if (aux.getAttributes().getLength() == 0) {
           valorInvestigar = aux.getChildNodes().item(0).getAttributes().item(0).getNodeValue();
                 NombreCampo = aux.getNodeName();
                 } else {
                  valorInvestigar = aux.getAttributes().item(0).getNodeValue();
                 NombreCampo = aux.getNodeName();
              }
               Integer index = 0;
              boolean comparacion = false;
```

```
            if (((NombreCampo.equals("validation"))||(NombreCampo.equals("Validation")))
&& ((valorInvestigar.equals("#Valido"))||(valorInvestigar.equals("Valido")))) {
                patrones.add(list.item(i));
              }
             j++;
           }
           j = 0;
           i++;
        }  return  patrones;

    }
```

Este método busca en la lista de nodos Patrón los patrones que tienen en su atributo *Validation* valor válido y devuelve un vector con todos estos nodos.

Las otras clases representan la parte visual de la aplicación permitiéndole al usuario la entrada de los datos y la obtención de los resultados..

public class NewJFrame extends javax.swing.JFrame

public class status extends javax.swing.JDialog

public class changeOfStatus extends javax.swing.JDialog

public class interaction extends javax.swing.JDialog

public class PatronValido extends javax.swing.JDialog

2.2.2 Diagrama de actividades

Un diagrama de actividades describe la secuencia de actividades que se pueden realizar, definiendo un orden en las distintas tareas, permite establecer procesos en paralelo o estar sujeto a condiciones lógicas. Su objetivo no es relacionar actividad con objetos, sólo comprender qué actividades son necesarias y cuáles son sus relaciones de dependencia.

En el diagrama de actividades de la Figura 10 se muestra las diferentes actividades que se realizan para la validación de las interrelaciones. En este diagrama se puede apreciar, ejemplos de procesos paralelos, y actividades sujetas a condiciones.

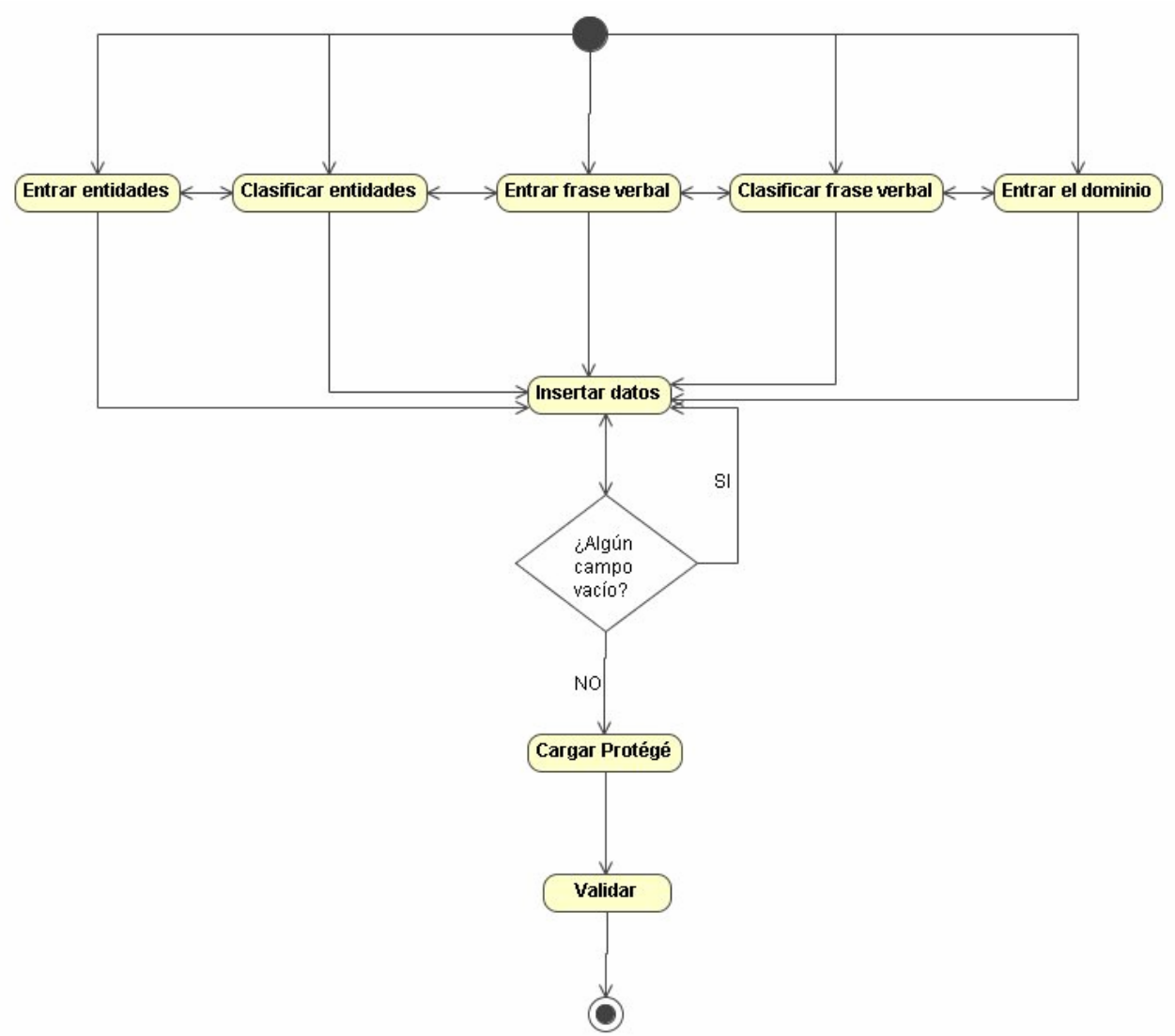


Figura 10: Diagrama de actividades.

2.2.3 Diagrama de implementación

Los diagramas de implementación se usan para modelar la configuración de los elementos de procesamiento en tiempo de ejecución y de los componentes, procesos y objetos de software que viven en ellos.

En la Figura 11 se presenta el diagrama de implementación donde se modelan dentro de un nodo físico las componentes que existen como entidades en tiempo de ejecución.

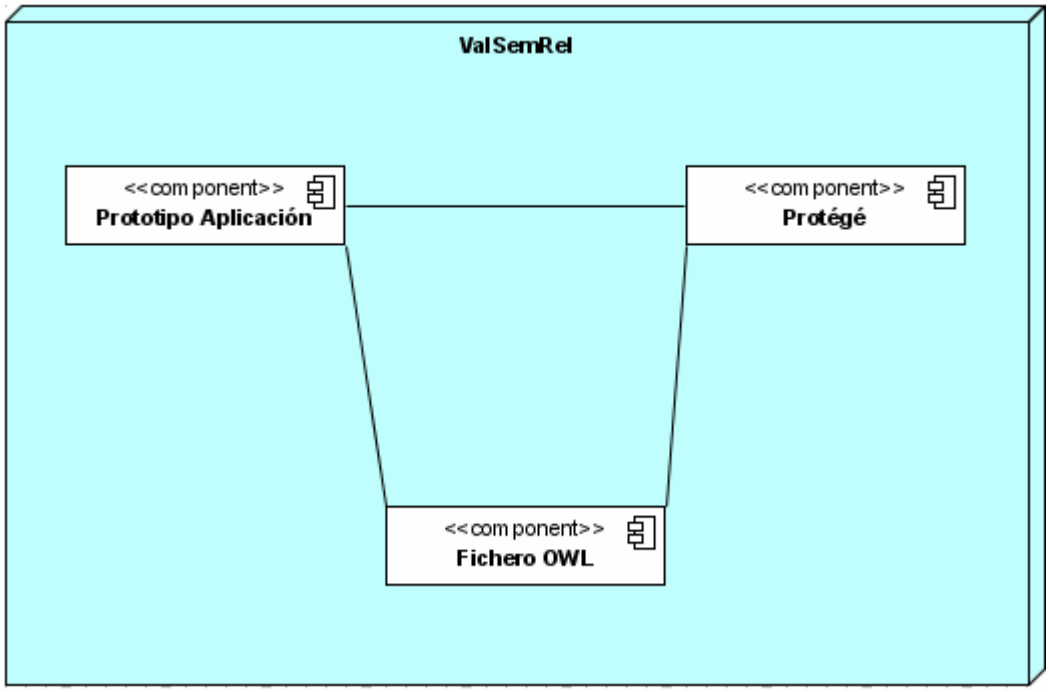


Figura 11: Diagrama de Implementación.

La componente prototipo de aplicación inserta nuevos datos en el fichero owl (ontología), luego ejecuta el Protégé con el fichero cargado y se realiza el proceso de validación, modificando el fichero, una vez terminado este proceso, desde el prototipo de aplicación se obtiene del fichero los resultados finales.

La aplicación se implementó en NetBeans IDE, entorno de desarrollo visual de código abierto para aplicaciones programadas mediante Java, uno de los lenguajes de programación más difundidos en la actualidad.

2.3 Conclusiones parciales

1. Se elaboró una ontología que contiene la clasificación de las interrelaciones, la cual puede ser utilizada para la validación semántica de esquemas conceptuales Entidad Relación.
2. Se diseñó e implementó un prototipo de aplicación que permite agregar como instancias a la ontología un esquema conceptual Entidad Relación para su posterior validación semántica por parte de las reglas implementadas en la ontología.

3. PRESENTACIÓN DE UN CASO DE ESTUDIO

En este capítulo se tomó un ejemplo sencillo de un esquema conceptual de base de datos para mostrar el proceso de validación semántica a través de un prototipo de aplicación implementado como parte de este trabajo.

3.1 Caso de estudio

El esquema conceptual presentado está conformado por cinco tipos de interrelaciones y seis tipos de entidades:

Entidad 1	**Frase verbal**	**Entidad 2**
Alumno	Es_miembro	Grupo
Profesor	Imparte	Asignatura
Profesor	Enseña	Grupo
Profesor	Pertenece	Departamento
Aula	Asignada_a	Grupo

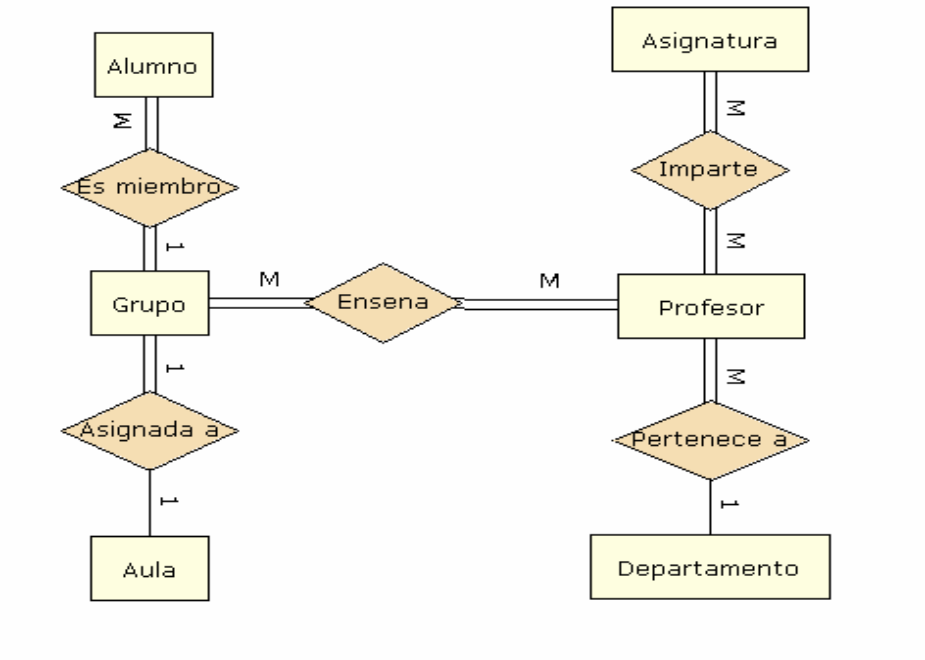


Figura 12. Diagrama ER a validar semánticamente por la ontología.

3.2 Inserción de la interrelaciones en la ontología

Primeramente se comienza la entrada de datos en la aplicación, la misma le permite al usuario seleccionar de las categorías propuestas la clasificación de las interrelaciones.

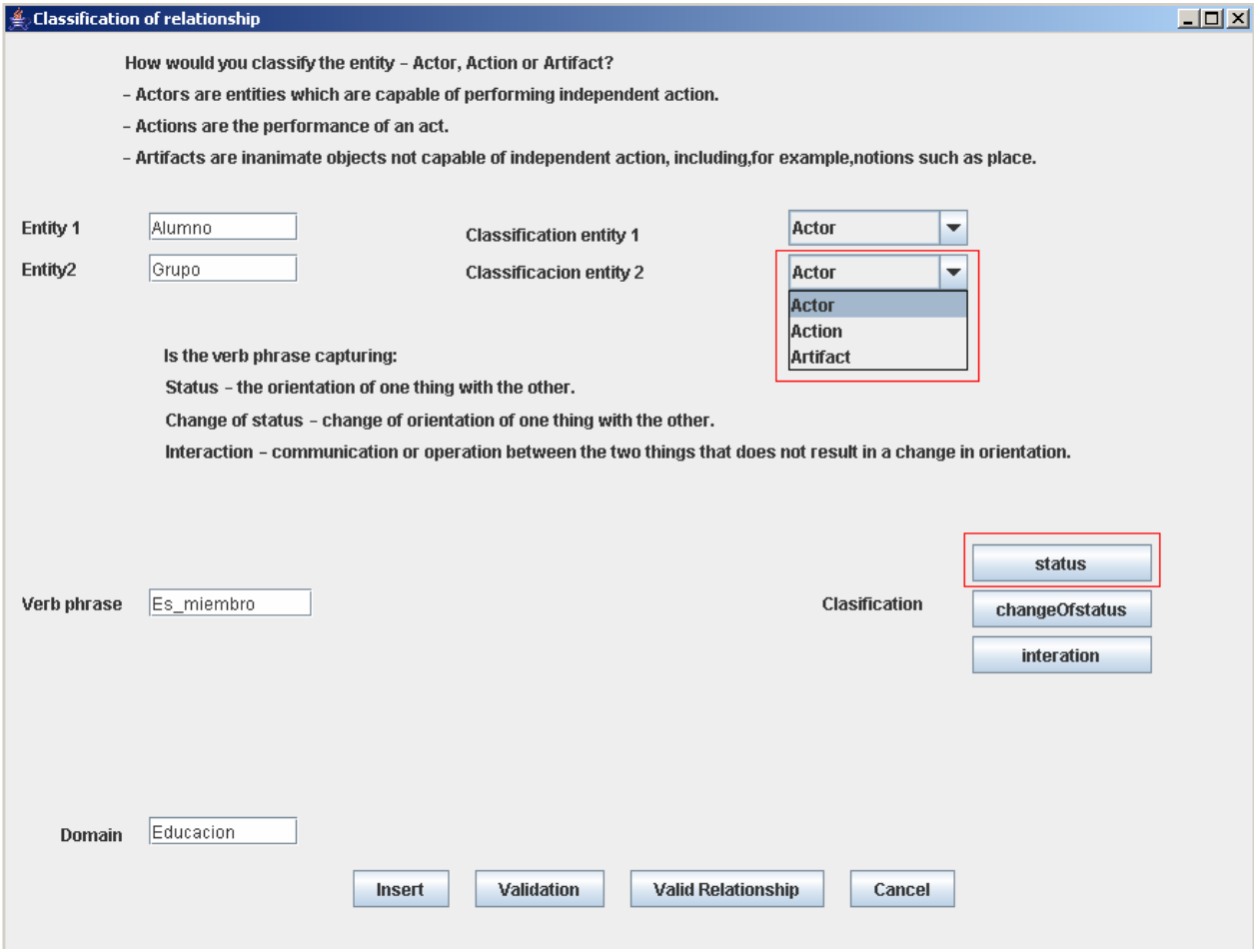


Figura 13: Aplicación que inserta las interrelaciones en la ontología.

Las entidades, la frase verbal y el dominio de la interrelación a insertar, son los valores de los campos Entity1, Entity2, Verb phrase y Domain respectivamente.

Luego se selecciona la clasificación de cada entidad. En la parte superior de la aplicación se muestra la especificación de cada tipo de entidad (ver Figura 13).

En la primera interrelación del esquema conceptual ambas entidades son actores.

Se propone además las categorías para la clasificación de la frase verbal, al seleccionar una de estas se muestra una nueva ventana para la selección de la primitiva a la que pertenece la frase verbal.

La frase verbal Es_miembro pertenece a la categoría Status.

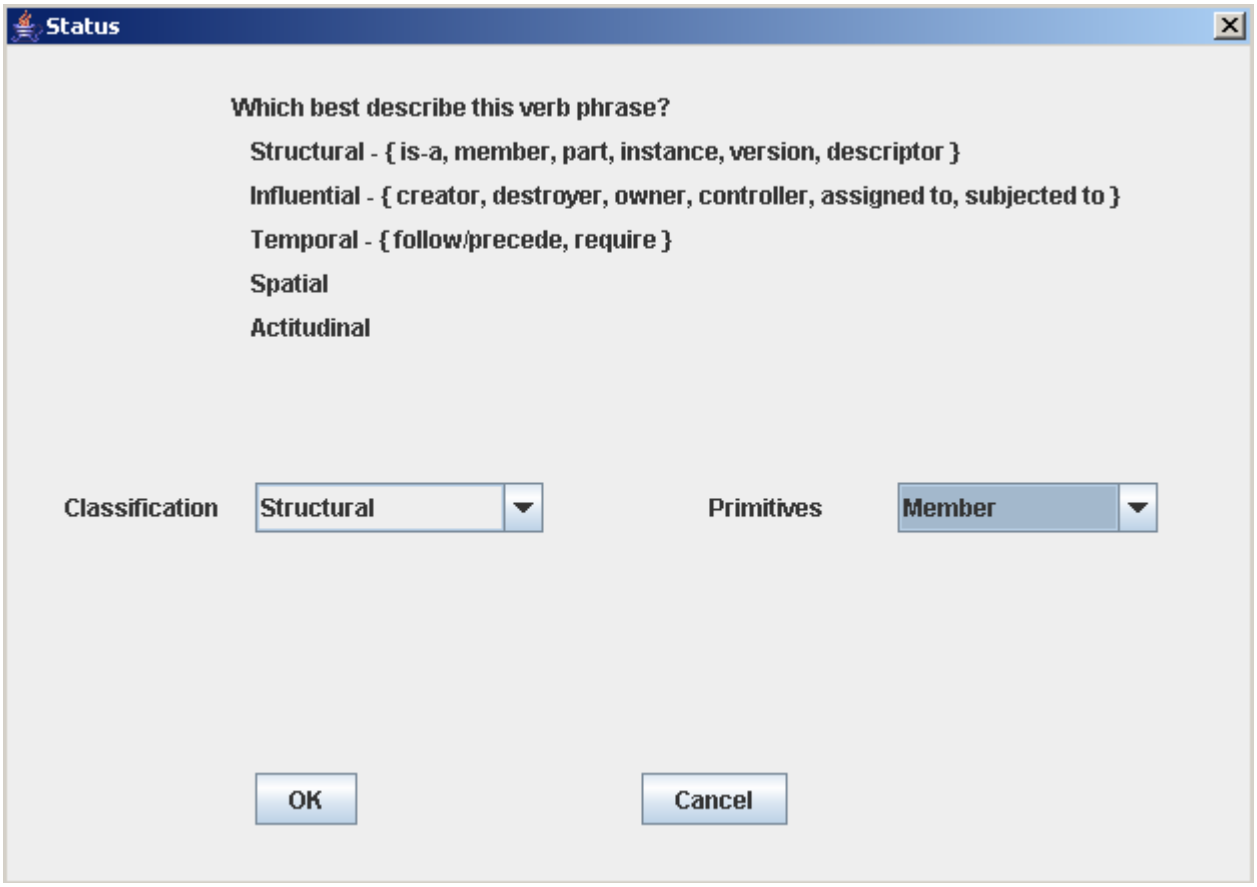


Figura 14: Ventana que permite seleccionar la clasificación de la frase verbal para la categoría Status.

Luego de seleccionar la primitiva, presionar el botón OK para que la primitiva quede insertada en la ventana principal.

El proceso de selección de primitivas es similar para las otras dos categorías. En las Figuras 15 y 16 se muestran las ventanas para la selección de las primitivas de las categorías *Change of Status* e *Interaction* respectivamente.

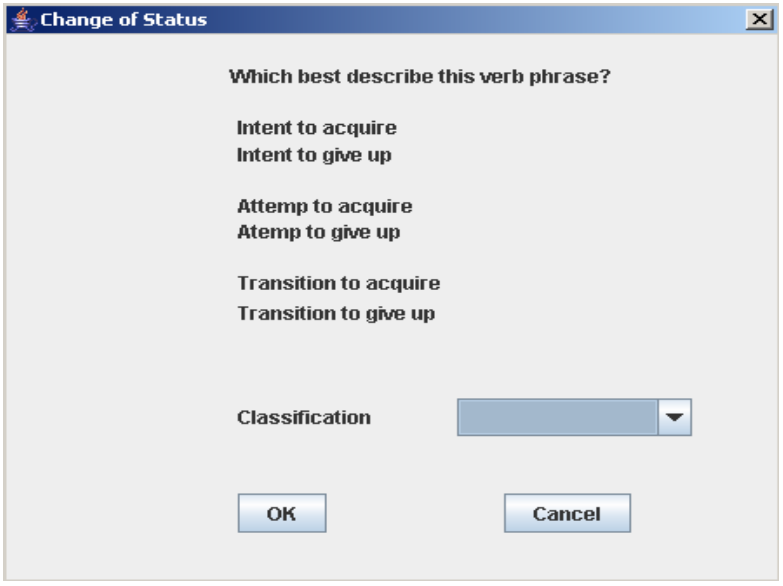


Figura 15: Ventana que permite seleccionar la clasificación de la frase verbal para la categoría *Change of Status*.

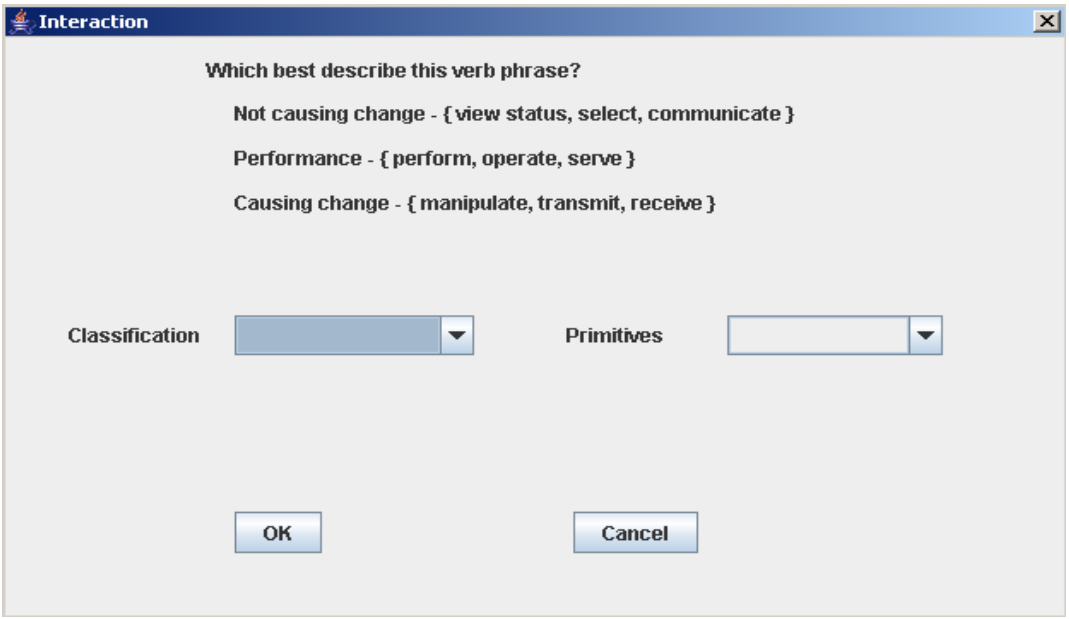


Figura 16: Ventana que permite seleccionar la clasificación de la frase verbal para la categoría *Interaction*.

Después de concluir la entrada de datos se continúa con la inserción en el fichero owl presionando el botón *Insert* como se muestra en la Figura 17.

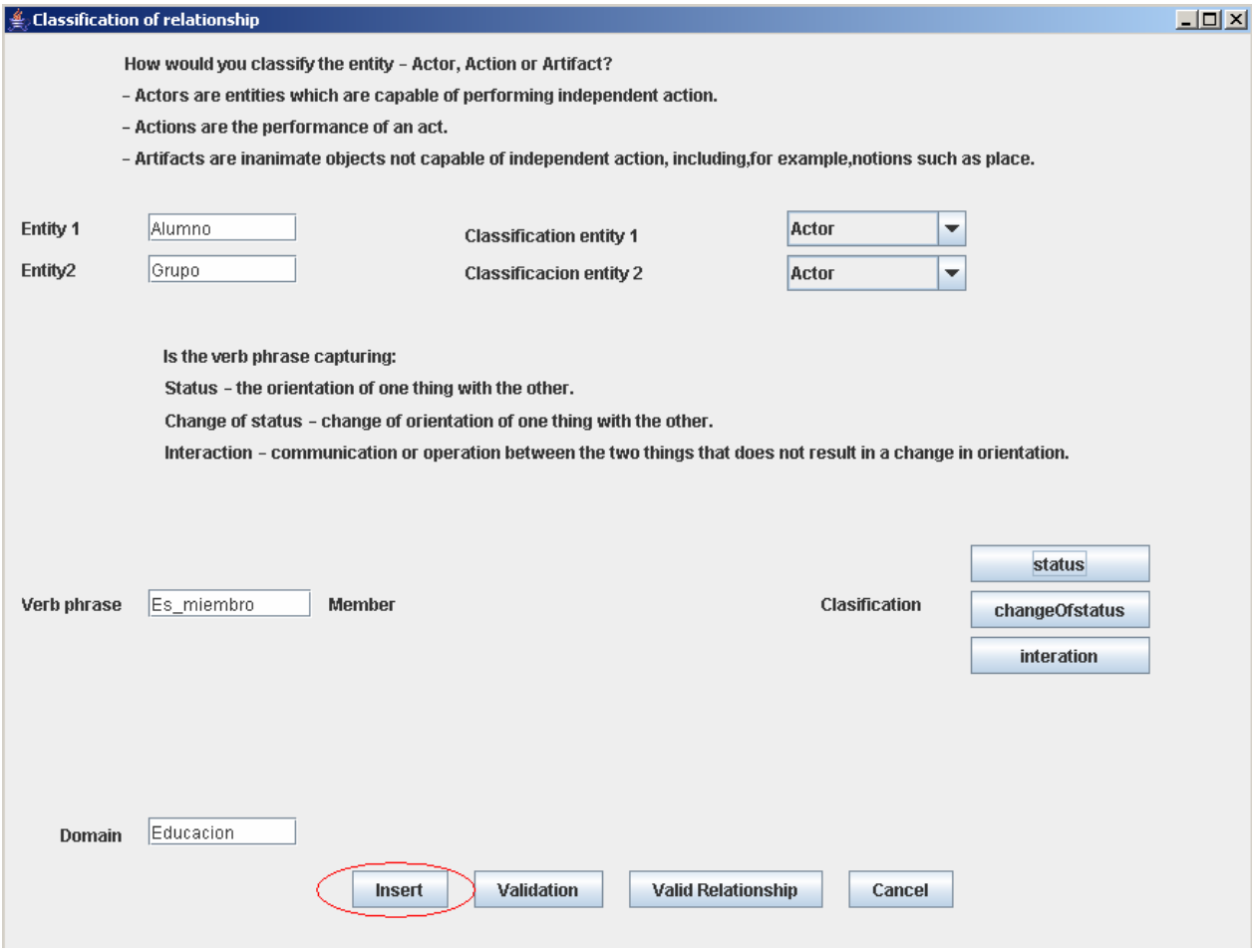


Figura 17: Ventana que muestra los datos que se van a insertar en la ontología.

Si los datos fueron insertados correctamente se muestra un mensaje informándolo.

Después de insertada la interrelación se continua insertando las siguientes de igual forma.

3.3 Validación de las interrelaciones

Cuando todas las interrelaciones fueron insertadas en el fichero owl se presiona el botón *Validation*, este carga el fichero en el Protégé para analizar las reglas y validar las interrelaciones.

En el Protégé seleccionar el Menú Project-Configure…

En la nueva ventana marcar SWRL Tab:

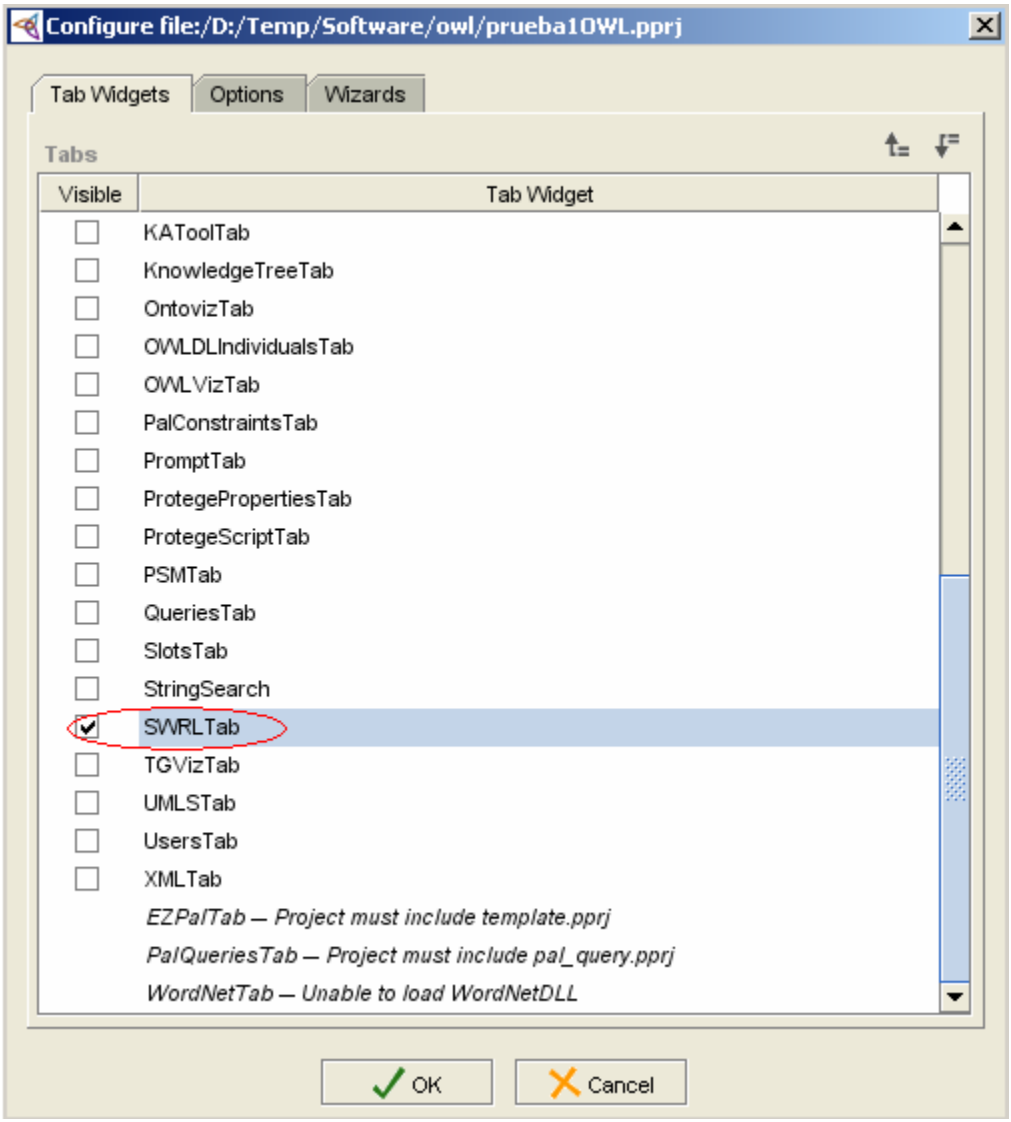


Figura 18: Ventana para activar Tab.

Seleccionar la pestaña SWRL Rules y correr el motor de inferencia Jess, que SWRL utiliza, como se explica en los siguientes pasos:

Paso1: Presionar el botón J para activar el Jess (ver Figura 19).

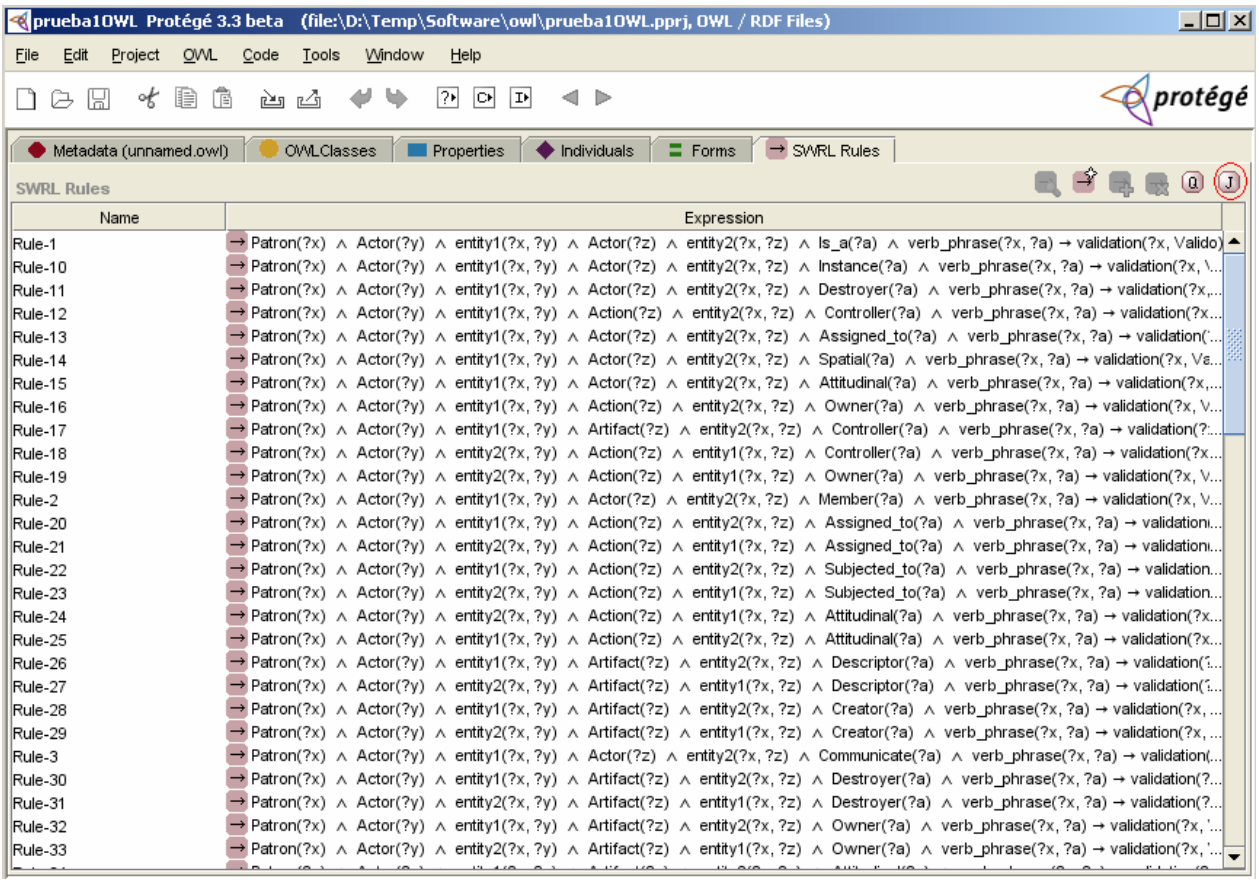
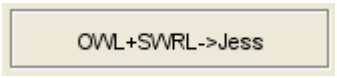


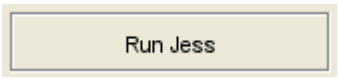
Figura 19: Ventana para activar el motor de inferencia.

Una vez activado el motor de inferencia se muestra en la parte inferior de la ventana los botones para realizar la inferencia (ver Figura 20).

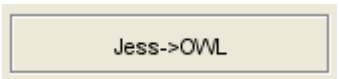
Paso2: Presionar el botón OWl+SWRL->Jess.



Paso3: Presionar el botón Run Jess.



Paso4: Presionar el botón Jess->OWL.



Luego se salva el fichero para que guarde la inferencia de las reglas y se termina la ejecución de Protégé.

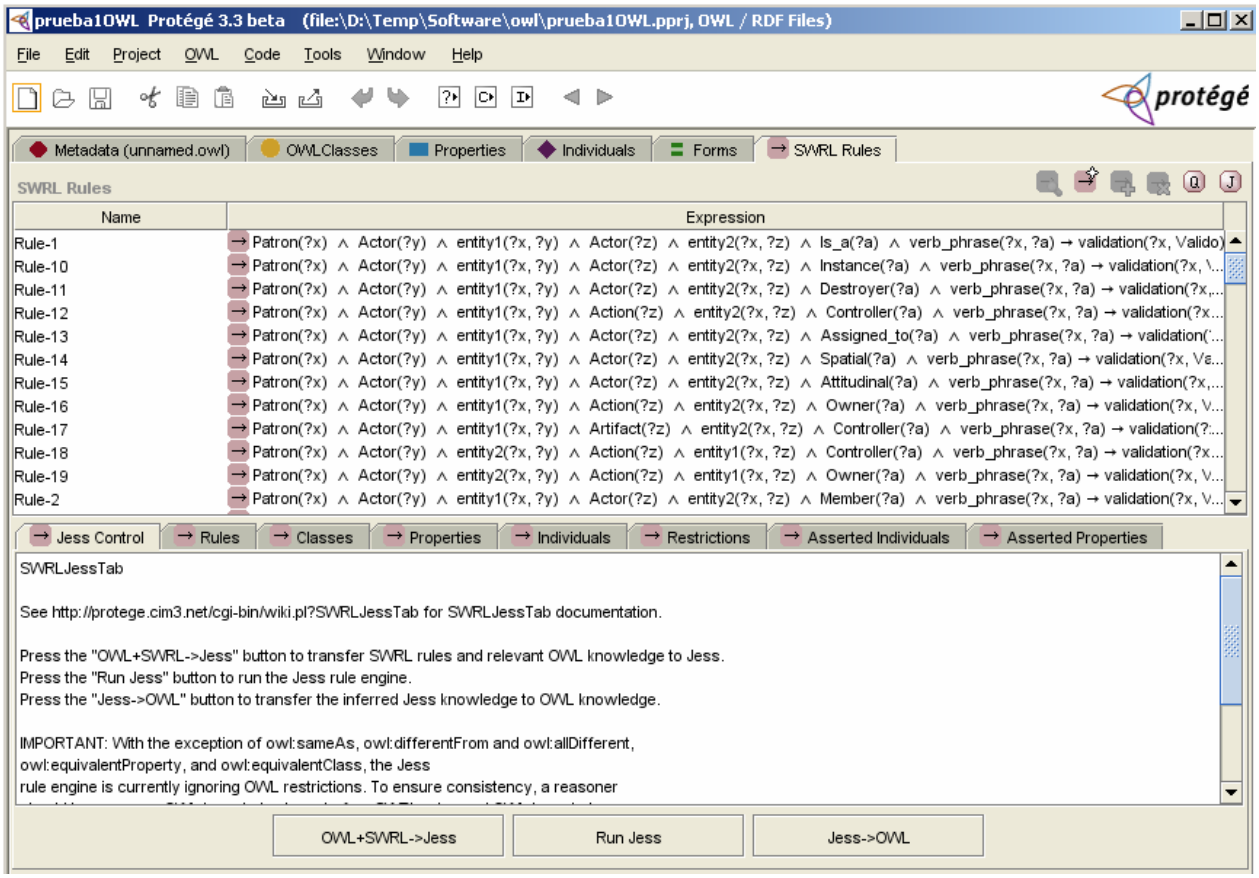


Figura 20: Ventana para la inferencia de las reglas.

3.4 Resultados de la validación semántica

Para la visualización de las interrelaciones válidas, se debe presionar el botón *Valid Relationship*, y a continuación se muestran en una nueva ventana las interrelaciones clasificadas como válidas, tal y como aparece en la Figura 21.

Figura 21: Resultados de la validación semántica del esquema conceptual.

3.5 Conclusiones parciales

1. La utilización de esta ontología como base de conocimiento y la inferencia de reglas dio resultados satisfactorios en la validación de interrelaciones de un esquema conceptual.
2. La interacción entre el prototipo de aplicación y la ontología muestra la factibilidad de su integración en una herramienta CASE.

CONCLUSIONES

De acuerdo a los resultados de este trabajo se puede concluir que:

1. Se elaboró una ontología que contiene la clasificación de los tipos de entidades y tipos de interrelaciones utilizadas en la modelación conceptual de bases de datos.
2. Se implementaron un conjunto de reglas en la ontología que permiten la validación semántica de interrelaciones a partir de una clasificación previa dada por el diseñador de la base de datos.
3. Se diseñó e implementó un prototipo de aplicación que permite insertar como instancias en la ontología las interrelaciones de un esquema conceptual.

RECOMENDACIONES

1. Extender la ontología de manera que incluya una capa que recopile la semántica del dominio de la aplicación.
2. Integrar la aplicación ERECASE con la ontología.

REFERENCIAS

Alter, S. (1999): Information Systems: A Management Perspective. Addison-Wesley. *p*

Bodart, F., Pate, A., Sim, M. & Weber, R. (2002) Should optional properties be used in conceptual modeling? A theory and three empirical tests. Informations Systems Research, 12(4), *pp* 384–405.

Dennis, A. & Wixom, B. H. (2000) Systems Analysis and Design. . John Wiley, *pp*

Dey, D., Storey, V. C. & Barron, T. M. (1999) Improving Database Design through the Analysis of Relationships. ACM Transactions on Database Systems, 24(4), *pp* 453-483.

Gruber, T. R. (1993) A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition, 5, *pp* 199-200.

Gruber, T. R. (1995) Towards Principles for the Design of Ontologies used for Knowledge Sharing. International Journal of Human-Computer Studies, 43(5-6), *pp* 7-928.

Gruninger, M. & Lee, J. (2002) Ontology Applications and Design. Communications of the ACM, 45(2), *pp* 39-41.

Gualtleri, A. & Ruffolo, M. (2005): An ontology-based framework for representing organizational knowledge. Proc. of the International Conference on Knowledge Management (I-KNOW '05), (June/July), *pp* 71–78.

Guarino, N. (1996) Understanding, Building, and Using Ontologies. *pp*

Guarino, N. (1998): Formal Ontology and Information Systems. Proc. of the FOIS'98., *pp* 3-15.

Knight, C., Gašević, D. & Richards, G. (2006) An ontology-based framework for bridging learning design and learning content. Educat. Technol. Society, 9(1), *pp* 23–37.

Lloyd-Williams, M. (1997): Exploiting domain knowledge during the automated design of object oriented databases. In Proc. of the ER 1997, LNCS 1331 *pp*

McGuinnes, D. L. & Wright, J. R. (1998) Conceptual modelling for configuration: A description logic-based approach. Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 12, *pp* 333-344.

Musen, M. A. (1992) Dimensions of Knowledge Sharing and Reuse. COMPUTERS AND BIOMEDICAL RESEARCH, 25, *pp* 435-467.

Noy, N. F. & D.L. McGuinness (2001): Ontology Development 101: A Guide to Creating Your First Ontology. Available at

http://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html. *Retrieved:* 15 March 2004, 2004

Purao, S. & Storey, V. C. (2005) A multi-layered ontology for comparing relationship semantics in conceptuals models of databases. Applied Ontology, (1), *pp* 117-139.

Shanks, G., Tansley, E., Nurelini, J., Toblin, D. & Weber, R. (2002): Representing Part-Whole Relationships in Conceptual Modeling: An Empirical Evaluation. Proc. of the ICIS 2002, *pp*

Siau, K., Wand, Y. & Benbasat, I. (1997) The Relative Importance of Structural Constraints and Surface Semantics in Information Modeling. Information Systems, 22(23), *pp* 155-170.

Steve, G., Gangemi, A. & Pisanelli, D. (1998): Integrating Medical Terminologies with ONIONS Methodology. http://saussure.irmkant.rm.cnr.it. *Retrieved:* May, 2008

Stewart, D. B. & Arora, G. (2003) A tool for analyzing and fine tuning the real-time properties of an embedded system. IEEE Transactions on Software Engineer, 29(4), *pp* 311-326.

Storey, V. C. (1993) Understanding Semantic Relationships. VLDB Journal, 2(4), *pp* 455-488.

Storey, V. C. (2001): Understanding and Representing Relationship Semantics in Database Design. Proc. of the NLDB 2000, LNCS 1959, *pp* 79-90.

Storey, V. C. (2005) Comparing Relationships in Conceptual Modeling: Mapping to Semantic Classifications. IEEE Transactions on Knowledge and Data Engineering, 17(11), *pp* 1478-1489.

Storey, V. C. & Purao, S. (2004): Understanding Relationships: Classifying Verb Phrase Semantics. Proc. of the ER 2004, LNCS 3288, *pp* 336-347.

Sugumaran, V. & Storey, V. C. (2006) The Role of Domain Ontologies in Database Design: An Ontology Management and Conceptual Modeling Environment. ACM Transactions on Database Systems, 31(3), *pp* 1064-1094.

Swartout, W. (1999) Ontologies. IEEE Intelligent Systems, (Jan./Feb.), *pp* 18–19.

Weber, R. (2002): Conceptual modeling and ontology: Possibilities and pitfalls. Proc. of the ER 2002, LNCS 2503, *pp* 1-2.

Weigand, H. (1997): A Multilingual Ontology based Lexicon for News Filtering -The TREVI project-. http://www.itaca.it/itaca/it/Trevihom.htm. *Retrieved:* December, 2008