UNIVERSIDAD CENTRAL
"MARTA ABREU" DE LAS VILLAS.

FACULTAD MATEMÁTICA – FÍSICA – COMPUTACIÓN

"Creación de una Ontología para representar palabras de origen africano de uso en nuestro país."

Tesis presentada en opción al Título Académico de Licenciado en Ciencia de la Computación.

AUTOR: *JUAN PABLO RODRÍGUEZ VEITÍA*
TUTORES: *MSC. MANUEL CASTRO*
DRA. LUISA GONZÁLEZ
DRA. GEMA VALDÉS

VILLA CLARA, CUBA.
2006

Dedicatoria

Antes de nada le dedico este trabajo en especial a mi madre Isora, quien es mi ángel del alma, mostrándome siempre su confianza y amor convirtiéndose en la responsable principal de haber logrado terminar mi licenciatura, a mis padres Juan Manuel y Aleandro, mi madrastra Noris, quienes me han brindado su apoyo tanto en los tiempos buenos, como en los tiempos malos, y de llevar a cabo mis estudios cubriendo todos los aspectos que he necesitado.

A mis hermanas Ismary, Yaima y Yodelaine, así como mi hermano Yodelkis, quienes de una forma muy especial se han preocupado por mi en todo momento.

A mi esposa Yanet y mi hija Betsy que al pensar en momentos que todo estaba perdido estubieron presente para brindarme apoyo, ánimo y fe durante gran parte de mi carrera, así como también, a mis abuelos Felio y América, que sin ellos no hubiese tenido razón para escribir estas líneas, considerándolos como el motor impulsor de mi carrera y sus consejos mis propias armas para continuar.

A mis tíos, primos y demás familiares, quienes que de una forma u otra me apoyaron y ayudaron en momentos difíciles.

A mis vecinos y amistades que me regalaron parte de su apoyo y lealtad durante la carrera.

Agradecimientos

A mis tutores Manuel Castro, Luisa González y Gema Valdés quienes me aguantaron mis rezagos y atrasos, los cuales no lo desanimaron, poniendo empeño en que este trabajo estuviera terminado de la mejor forma posible.

A los profesores de la ENU “Primero de Mayo” de la ciudad de Santa Clara, especialmente a la directora Maira y a los profesores de computación Irinka y Yosnelvis.

A todos los que han tenido que ver con esta tesis y no haya mencionado, también mis más sinceros agradecimientos.

Resumen

El trabajo realizado consiste en un estudio sobre las ontologías y la implementación de una ontología. Desde hace algunos años se vienen trabajando cada vez con más fuerza en el desarrollo de herramientas dirigidas a la construcción de ontologías y al desarrollo de ontologías en muchos ámbitos de la actividad humana.

El trabajo plantea diferentes definiciones y clasificaciones de ontología según se referencia en la bibliografía consultada. Se caracterizan las herramientas más importantes en el área de la ingeniería de las ontologías así como los lenguajes utilizados para crearlas. A partir del análisis de las características de las herramientas y la tendencia de desarrollo mundial en esta área se selecciona una de estas herramientas para aplicarla a la solución de un problema y se muestra paso a paso como se utiliza la misma de forma que pueda servir de guía para la aplicación de esa herramienta a otros problemas.

La aplicación desarrollada tiene como objetivo representar palabras de origen africano de uso en Cuba mostrando sus propiedades fundamentales (nombre, tipo y uso) y significado que dependen del campo semántico y del reservorio en que se use dicha palabra. También interesa tener acceso a las fuentes donde aparece referenciada la palabra sin tener en cuenta el significado que tiene es ese contexto, en el caso en que aparecen en obras literarias se requiere guardar el nombre de la obra.

Índice

Introducción

Las ontologías han ido cobrando cada día más importancia en el ámbito de la computación al irse generalizando su aplicación a diferentes áreas como el procesamiento de lenguaje natural, la interoperabilidad entre sistemas heterogéneos, la indexación de sitios web, el modelado de empresas, la enseñaza, etc.

En la literatura se encuentran diferentes definiciones sobre ontologías que han ido cambiando en la medida en que se va ganando en conocimiento y se van formalizando hasta el punto de aparecer la Ingeniería de Ontologías como una rama más de las ciencias. De igual manera se han ido estableciendo diversas clasificaciones atendiendo a diferentes criterios y se han desarrollados herramientas de diversa índole para la construcción y utilización de las ontologías, se tiene que se han desarrollado herramientas para el desarrollo, la fusión e integración de ontologías, la evaluación, herramientas para el almacenamiento y preguntas, etcétera, y se desarrollaron diferentes lenguajes que permiten la creación de ontologías como XML, OWL, etc.

El objetivo general de este trabajo es realizar un estudio sobre ontologías.

Los objetivos específicos del trabajo son: realizar un resumen de las principales definiciones y clasificaciones que se encuentran en la literatura sobre las ontologías así como hacer una caracterización de las herramientas que con mayor frecuencia han sido o están siendo utilizadas en este momento en el área de las ontologías. A partir de los criterios de expertos reconocidos en el área se seleccionará una herramienta y se utilizará en la creación de una ontología para representar palabras de origen africano de uso en nuestro país con la que se explicará cómo es que se utiliza la herramienta.

Entre las características más mencionadas del español de Cuba y de su identidad cultural en general, una es la relacionada con el contacto de nuestra variante lingüística con las lenguas africanas. Sin embargo, los estudios científicos sobre el tema requieren de un proceso metodológico muy complejo que ha impedido un desarrollo acelerado de esta esfera de las investigaciones hispanísticas; solamente en las últimas décadas se han podido continuar las

hipótesis de trabajo planteadas por figuras como Fernándo Ortiz en el siglo pasado.

Dentro de las figuras lingüísticas africanas llegadas a Cuba con la trata de esclavos, una de las más importantes junto con la yoruba es la familia bantú, que se habla en una gran extensión desde Gabón hasta Sudáfrica. De esta familia las lenguas que más han aportado al español de Cuba son las localizadas en la cuenca del río Congo como es el *kikongo,* de cuyas matrices parte más del 90% de los datos que se ejemplifican en este trabajo.

La Universidad Central de Las Villas tiene en nuestro país una tradición de más de 30m años en el análisis de lenguas africanas, y gracias al intercambio con las Universidades de Alcalá (España) y Simón Kimbangu (República Democrática del Congo) se realizó un trabajo de campo en el año 2000 en la zona donde se habla el kikongo, por lo que se constató la procedencia de los datos recogidos en Cuba. El resultado de estos análisis obtuvo en el año 2003 el Premio de la Academia de Ciencias de Cuba.

Sin embargo se hace imprescindible en la nueva etapa de trabajo la organización de los datos recopilados para la confección de un futuro glosario de bantuismos que funcionan en Cuba. Estos vocablos están presentes en tres reservorios fundamentales: el Religioso (Religión de Palo Monte), el Español Coloquial y el General (representado por la Real Academia de la Lengua).

Capítulo 1: "Caracterización de las Ontologías."

Las ontologías han estado llamando la atención en los últimos años en el mundo de la informatización. La investigación sobre las ontologías comenzó a inicio de los años 90 y se a acelerado extendiéndose a la comunidad de tecnología *web* en los últimos años.

Muchas personas hablan hoy en día sobre las ontologías. Sin embargo, pueden haber algunos conceptos erróneos sobre qué es una ontología, qué es la ingeniería ontológica, y cuan útil puede ser, etc

En este capítulo se aborda como objetivo principal los aspectos fundamentales referentes a las ontologías y sus características más importantes

1.1 Concepto y definición de ontología.

1.1.1 Concepto de ontología.

Las ontologías, hacen posible una semántica para construir los metadatos que sirven para la estructuración del contenido. Un metadato no es más que un dato estructurado sobre la información, o sea, información sobre información, o de forma más simple, datos sobre datos. Los metadatos en el contexto de la *web*, son datos que se pueden guardar, intercambiar y procesar por medio del ordenador y que están estructurados de tal forma que permiten ayudar a la identificación, descripción clasificación y localización del contenido de un documento o recurso *web* y que, por tanto, también sirven para su recuperación. Una ontología es una especificación de una conceptualización, esto es, un marco común o una estructura conceptual sistematizada y de consenso no sólo para almacenar la información, sino también para poder buscarla y recuperarla. Una ontología define los términos y las relaciones básicas para la compresión de un área del conocimiento, así como las reglas para poder combinar los términos para definir las extensiones de este tipo de vocabulario controlado.

Se trata de convertir la información en conocimiento mediante unas estructuras de conocimiento formalizadas (las ontologías) que referencien los datos, por medio de los metadatos, bajo un esquema común normalizado sobre

algún dominio del conocimiento. Los metadatos no sólo especificarán el esquema de datos que debe aparecer en cada instancia, sino que también podrán contener información adicional de cómo hacer deducciones sobre ellos, es decir, cómo establecer axiomas que podrán, a su vez, aplicarse en los diferentes dominios del conocimiento almacenado.

Las ontologías proceden del campo de la Inteligencia Artificial; son vocabularios comunes para las personas y aplicaciones que trabajan en un dominio. Según el Grupo de Trabajo en Ontologías del consorcio W3C, una ontología define los términos que se usan para describir y representar un cierto dominio. Se utiliza la palabra "dominio" para denotar un área específica de interés o un área de conocimiento (física, aeronáutica, medicina, contabilidad, fabricación de productos, etc). Toda ontología representa cierta visión del mundo con respecto a un dominio. Por ejemplo, una ontología que defina "ser humano" como "espécimen vivo o muerto correspondiente a la especie *Homo sapiens*; primate bípedo que pertenece a la familia de los homínidos, como los chimpancés, gorilas y orangutanes" expresa una visión del mundo totalmente distinta a la de una ontología que lo defina como "sujeto consciente y libre, centro y vértice de todo lo que existe; todos tienen la misma dignidad, pues han sido creados a imagen y semejanza de Dios".

En las aplicaciones reales, sin embargo, una ontología es una identidad computacional, y no se debe considerar como una entidad natural que se descubre, sino como recurso artificial que se crea. Una ontología se comprende que es un **conocimiento común y compartido** de un dominio, que se puede comunicar entre científicos y sistemas computacionales. La última característica (se pueden compartir y reutilizar en aplicaciones diferentes), explica en parte el gran interés suscitado en los últimos años en la creación e integración de ontologías.

1.1.2 Definiciones de ontología.

En la filosofía, significa teoría de existencia. Esto trata de explicar qué existe en el mundo y cómo el mundo es configurado introduciendo un sistema de categorías críticas para considerar las cosas y sus relaciones intrínsecas.

Según Gruber [1], en 1993, la ontología desde el punto de vista de la Inteligencia Artificial, se define como “la especificación explícita de conceptualización” la cual es aceptada ampliamente en la comunidad de la Inteligencia Artificial.

Desde el punto de vista de los sistemas basado en conocimiento, es definido como “una teoría de conceptos, vocabulario usado como bloques de información de sistemas de procesamiento”, definición dada por R. Mizoguchi [2]. Las bases de conocimiento son necesarias cuando se construye un modelo de resolución de problema en el mundo. En este caso las ontologías son dividido en dos tipos: *la Ontología de la Tarea* para problema que resuelve el proceso y *la Ontología del Dominio* para el dominio donde la tarea es realizada.

En 1993, Gruber define otro concepto sobre ontologías, emitiendo que estas eran acuerdos sobre conceptualizaciones compartidas. Las conceptualizaciones compartidas incluyen los frameworks conceptuales para modelar el conocimiento del dominio; los protocolos específicos (content-specific) para la comunicación entre los agentes; y los acuerdos sobre la representación de las teorías del dominio.

1.2 Tipos de ontologías.

Cuando hablamos de ontologías como "sistemas de representación de conocimiento" debemos especificar a qué tipo de sistemas nos referimos. 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.

Según Uschold [3], una ontología necesariamente incluirá un vocabulario de términos y una especificación de su significado (definiciones e interrelaciones entre conceptos) que impone estructura al dominio y restringe las posibles interpretaciones.

En 1998, se distinguen tres tipos fundamentales de ontologías:

- **Ontologías de un dominio**, en las que se representa el conocimiento especializado pertinente de un dominio o subdominio, como la medicina, las aplicaciones militares, la cardiología, etc.

- **Ontologías genéricas**, en las que se representan conceptos generales y fundacionales del conocimiento como las estructuras parte / todo, la cuantificación, los procesos o los tipos de objetos.

- **Ontologías representacionales**, en las que 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 ontologies*).

A estos tres tipos, Guarino [4] en1998 añade las ontologías que se crean para una actividad o tarea específica, como por ejemplo la venta de productos o el diagnóstico de una enfermedad y las ontologías creadas para una aplicación específica.

En KACTUS identificaron 4 tipos de ontologías de acuerdo a su alcance de aplicabilidad:

- **Ontología de la aplicación**: usadas por la aplicación. Ontología de procesos de producción, de diagnóstico de fallas, de diseño intermedio de barcos, etc.

- **Ontología del dominio**: específicas para un tipo de artefacto, generalizaciones sobre tareas específicas en algún dominio. Por ejemplo, ontología del proceso de producción de hidrocarburos, de la red eléctrica, de barcos, etc.

- **Ontologías técnicas básicas**: describe características generales de artefactos. Por ejemplo: componentes, procesos, funciones.

- **Ontologías genéricas**: describe la categoría de más alto nivel.

Otra forma de identificar ontologías es desde su punto de vista. Por ejemplo según el tipo de agente al que vayan destinadas:

- **Ontologías lingüísticas**: se vinculan a aspectos lingüísticos, esto es, a aspectos gramáticos, semánticos y sintácticos destinados a su utilización por los seres humanos.

- **Ontologías no lingüísticas**: destinadas a ser utilizadas por robots y agentes inteligentes.

- **Ontologías mixtas**: combinan las características de las anteriores.

Según 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, etc.

1.3 Clasificación de las ontologías.

Las personas mediante las ontologías se representan en su cabeza el mundo que los rodea. Estas ontologías no son explícitas, ya que no se detallan en un escrito ni se establecen de forma jerárquica o matemática. Todos usamos 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 compartimos 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 más de un padre y una madre biológicos, que los padres tienen o han tenido padres... No necesitamos ser explícitos con este conocimiento, pues todo el mundo lo sabe. Sin embargo, cuando se tratan términos poco 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. En cuanto dos sistemas de información (sistemas ERP, bases de datos, bases de conocimiento) intentan comunicarse, aparecen problemas semánticos que dificultan o imposibilitan la comunicación entre ellos. Los problemas semánticos son de dos tipos: **de dominio** y **de nombre**. Los conflictos de dominio aparecen cuando conceptos similares en cuanto a significado, pero no idénticos, se representan en distintos dominios. Un ejemplo de esto, es el concepto representado por *Trabajador* en una base de datos (BD) que puede representar a un trabajador cualificado, mientras que en otra BD se puede utilizar *Trabajador* para cualquier trabajador, sea o no cualificado. Ambos conceptos están muy relacionados, pero no son equivalentes ni se deben mezclar. Usando ontologías, se podría especificar que el primer concepto sería una especialización del segundo; y un sistema de razonamiento automático basado en ontologías impediría, por ejemplo, que se contratara para tareas cualificadas a trabajadores no cualificados.

Los conflictos de nombre son de dos tipos: sinónimos y homónimos. Los sinónimos ocurren cuando los sistemas usan distintos nombres para referirse al mismo concepto. Por ejemplo, una BD puede usar *Trabajador* para el mismo concepto que otra usa *Empleado*. En ese caso, se podría usar una ontología que definiera como idénticos los dos términos. Así, las aplicaciones que manejaran esas bases de datos sabrían como llevar datos de una a otra.

Los homónimos surgen cuando los sistemas usan el mismo nombre para representar cosas distintas. Por ejemplo, en una aplicación de una compañía de seguros, *Conductor* representa a una persona que tiene contratada una póliza particular con la compañía; mientras que, en una aplicación de una compañía de taxis, *Conductor* representa a un trabajador que conduce un taxi de la compañía. Como es de suponer, si se intentara integrar automáticamente ambas aplicaciones basándose en que ambas usan el mismo término

(*Conductor*) para significar lo mismo, se produciría el desastre más absoluto: al dar de baja a un conductor de taxi se le quitaría su póliza de seguros, con lo que no podría conducir ni su propio coche (al menos, no legalmente); y, al dar de alta a un asegurado, se le daría de alta como taxista, aunque no tuviera la licencia de taxista. Sólo una ontología explícita le puede comunicar a una aplicación que su *Conductor* no guarda ninguna relación con el de otra.

Las ontologías explícitas se pueden expresar de muchas maneras. Como mínimo, deben incluir un vocabulario de términos, con la definición de cada uno. Por ejemplo, la ontología empresarial *Enterprise Ontology* (EO) define así Venta: “Una Venta es un acuerdo entre dos Entidades Legales para el intercambio de un Producto por un Precio de Venta. Normalmente, el Producto es un bien o servicio y el Precio de Venta es monetario, aunque se incluyen otras posibilidades”. Las ontologías sencillas suelen representarse como una jerarquía de conceptos relacionados y ordenados.

Dependiendo del grado de formalidad, las ontologías explícitas se clasifican en informales, semi-informales, semi-formales y formales. Las primeras se expresan directamente en cualquier 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.

1.4 Algunos aspectos importantes de las ontologías.

1.4.1 Méritos concretos que una ontología puede proporcionar.

1- Vocabulario común.

La descripción de las necesidades mundiales ha designado un vocabulario entre las personas involucradas. El papel fundamental de una ontología contribuye a él.

2- Explicación de lo que queda implícito.

En todas las actividades humanas, nosotros encontramos presuposiciones/asunciones que quedan implícitas. Los ejemplos típicos incluyen las definiciones de condiciones comunes y básicas, las relaciones y restricciones entre ellos, y los puntos de vista para la interpretación de los fenómenos y estructura común a las tareas a que están normalmente comprometidos. Cualquier base de conocimiento construida esta basada en una conceptualización. Una ontología es una explicación del conocimiento implícito. Una representación explícita de las asunciones y la conceptualización es más que una explicación simple. Aunque podría ser duro, propiamente apreciado por las personas que no tienen experiencias en tal representación, su contribución al conocimiento rehusado y compartido es más que la expectativa considerando que la simplicidad ha sido una de las causas cruciales de prevenir conocimiento que se comparte y reusa.

3- Estructura de Datos

Una ontología en una base de datos. En este sentido, una ontología nos proporciona una estructura de datos apropiada para la descripción de información e intercambio.

4- Sistematización del conocimiento

La sistematización de conocimiento requiere establecer correctamente la relación vocabulario/conceptos en cuanto a la referencia de qué personas describen los fenómenos, teorías y designaciones bajo consideración. Una ontología así contribuye a proporcionar el espinazo de la sistematización del conocimiento.

5- Estandarización

El éxito de las industrias modernas ha sido logrado gracias a la estandarización de varios componentes. Nosotros no podemos evitar tal estandarización en el conocimiento exitoso que procesa la investigación y las actividades en el mundo real.

6- La razón del diseño

Los ejemplos típicos para ser explicados incluyen la intención de los diseñadores de artefactos, eso es, parte de *la razón del diseño*. Una ontología contribuye a la explicación de las asunciones, condiciones previas implícitas requeridas por los problemas para resolver así la conceptualización del objeto designado que refleja aquellas asunciones.

7- La función meta-modelo

Un modelo normalmente se construye en la computadora como una abstracción del blanco real y una ontología nos proporciona los conceptos y relaciones entre ellos que son usadas como bloques del modelo. Así, una ontología especifica los modelos a construir dando pautas y restricciones que debe satisfacerse. Esta función es vista como *meta nivel*.

1.4.2 Composición de una ontología.

Según Gruber, las ontologías se componen de:

- **conceptos**: son las ideas básicas que se intentan formalizar. Los conceptos pueden ser clases de objetos, métodos, planes, estrategias, procesos de razonamiento, etc.

- **relaciones**: representan la interacción y enlace entre los conceptos de un dominio. Suelen formar la taxonomía del dominio. Por ejemplo: subclase-de y parte-de.

- **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: asignar-fecha, categorizar-clase, etc.

- **instancias**: se utilizan para representar objetos determinados de un concepto.

- **reglas de restricción o axiomas**: son teoremas que se declaran sobre relaciones que deben cumplir los elementos de la ontología. Por ejemplo: "Si A y B son de la clase C, entonces A no es subclase de B", "Para todo A que cumpla la condición B1, A es C", etc. Los axiomas, junto con la herencia de conceptos, permiten inferir conocimiento que no esté indicado explícitamente en la taxonomía de conceptos.

1.4.3 Cuestiones a tener en cuenta para diseñar ontologías.

A la hora de diseñar una ontología debemos tener en cuenta 5 cuestiones claves:

- **claridad**: una ontología debe poder comunicar de manera efectiva el significado de sus términos. Las definiciones serán lo más objetivas posibles y deben explicarse también en lenguaje natural.

- **coherencia**: una ontología debe permitir hacer inferencias que sean consistentes con las definiciones.

- **extensibilidad**: deben anticiparse nuevos usos para así poder permitir extensiones y especializaciones.

- **especificidad**: se debe especificar a nivel de conocimiento, sin que dependa de una codificación particular a nivel de símbolo.

- **precisión**: debe hacerse la menor cantidad de "suposiciones" acerca del mundo modelado.

1.4.4 Conceptos claves en relación a las ontologías.

Conceptos clave en relación a las ontologías son:

- **Clase:** Es un objeto que define una categoría. Describe conceptos en el dominio del discurso.

- **Subclase:** Es en sí misma una clase, pero que es hija de alguna otra clase.

- **Clase jerárquica:** La compuesta por una colección de clases conectadas por relaciones "es un tipo de" (*class hierarchy*).

- **Casos (instancias):** Ejemplos específicos pertenecientes a alguna clase, esto es, objetos de una clase.

- **Roles o Propiedades (slots):** Propiedades de cada concepto que describen varias características y atributos del concepto. Ayudan a definir las características de las clases.

- **Restricciones:** Se utilizan para definir qué tipo de valor puede contener un slot particular, valores permitidos, número de valores, etc. También se denominan restricciones de roles.

- **Valor:** Describe una propiedad que se aplica a alguna clase o *instancia*.

- **Tipo:** Define el tipo de valor (como cadena de caracteres, número, booleano, etc).

- **Cardinalidad:** Define cuántos valores puede tener un *slot* individual (máximo y mínimo).

- **Herencia:** Es el proceso por el cual las subclases e *instancias* de alguna clase heredan propiedades y valores definidos más arriba en la jerarquía.

- **Variable:** Espacio vacío que puede llenarse preguntando a clases e instancias. Cada variable comienza con un signo de interrogación.

- **Relación:** Nuevo conocimiento que se obtiene por deducción, partiendo del conocimiento que se encuentra en la ontología. Las relaciones utilizan variables.

1.4.5 Otros conceptos de las ontologías.

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

- **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 (refinement)**: Un refinamiento de cada categoría de una ontología A, a alguna categoría de otra ontología B, lo cual 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.

1.5 Aplicaciones, beneficios y características de las ontologías.

1.5.1 Aplicaciones de las ontologías.

Las ontologías se usan para favorecer la comunicación entre personas**,** organizaciones y aplicaciones, lograr la interoperabilidad entre sistemas informáticos**,** razonar automáticamente y para la ingeniería de software.

Las ontologías favorecen la comunicación entre personas, organizaciones y aplicaciones porque proporcionan una comprensión común de un dominio, de modo que se eliminan confusiones conceptuales y terminológicas. Los problemas derivados de la falta de comprensión común entre personas revisten una gran importancia en la ciencia y en la tecnología.

En los campos de la Inteligencia Artificial, la Teoría de Decisiones y la Teoría de Sistemas Distribuidos (campos muy relacionados con la Web semántica), sucede algo parecido: los investigadores de un campo no pueden leer fácilmente los resultados de los investigadores de los otros, pues se usan diferentes perspectivas y términos para las mismas ideas y conceptos. Construyendo una ontología común para los tres campos, las investigaciones de un campo serían inmediatamente aplicables a los otros.

1.5.2 Beneficios de las ontologías.

Los beneficios de utilizar ontologías se pueden resumir de la siguiente forma:

- Proporcionan una forma de representar y compartir el conocimiento utilizando un vocabulario común.
- Permiten usar un formato de intercambio de conocimiento.

- Proporcionan un protocolo específico de comunicación.
- Permiten una reutilización del conocimiento.
- Sirven de repositorios para la organización del conocimiento.
- Sirven de herramienta para la adquisición de información.
- Sirven de herramientas de referencia en la construcción de sistemas de bases de conocimiento que aporten consistencia, fiabilidad y falta de ambigüedad a la hora de recuperar información.
- Normalizan los atributos de los metadatos aplicables a los documentos.
- Crean una red de relaciones que aporta especificación y fiabilidad.
- Permiten la integración de diferentes perspectivas de usuarios.
-Permiten el tratamiento ponderado del conocimiento para recuperar información de forma automatizada.
- Permiten la construcción automatizada de mapas conceptuales y mapas temáticos.
- Permiten la reutilización del conocimiento existente en nuevos sistemas.
- Permiten la interoperatividad entre diferentes sistemas.
- Establecen modelos normativos que permiten la creación de la semántica de un sistema y un modelo para extenderlo y transformarlo en diferentes contextos.
- Sirven de base para la construcción de lenguajes de representación del conocimiento.

1.5.3 Características de las ontologías.

Algunas de las características de las ontologías son:

- **Pueden existir ontologías múltiples:** si el propósito de una ontología es hacer explícito algún punto de vista, en algunos casos, necesitamos combinar dos o más ontologías. Cada ontología introduce conceptualizaciones específicas.

- **Se pueden identificar distintos niveles de abstracción estableciendo una topología de ontologías:** se puede caracterizar una red de ontologías usando multiplicidad y abstracción. Al no poder realizar una descripción completa del mundo, se puede pensar una estrategia de construcción gradual que vaya de abajo hacia arriba.

- **Multiplicidad de la representación:** un concepto puede ser representado de muchas formas, por lo que pueden coexistir múltiples representaciones del mismo concepto.

- **Mapeo de ontologías:** se pueden establecer las relaciones entre los elementos de una o más ontologías para establecer generalizaciones, especializaciones, conexiones, etc.

1.6 Lenguajes asociados a las ontologías.

Entre los principales lenguajes podemos destacar los siguientes:

1.6.1 Ontolingua:

Es una forma de lenguaje para la representación y el desarrollo compartido de una ontología, creado por KSL (Knowledge Systems Lab) en la Universidad de Stanford. Se diseña agregando representación mediante frames y traduciendo funcionalidades al KIF (Knowledge Interchange Format), que no es más que un lenguaje basado en la lógica para la representación del conocimiento. Este puede traducir desde y hasta algunos lenguajes lógicos de descripción como: *Loom*, *Epikit*, etc.

Ontolingua usa una notación tipo LISP para expresar la ontología, pero no soporta máquina de inferencia. No es adecuado para el Diseño de Bases de Datos debido a su poco poder de expresividad. Proporciona un ambiente de desarrollo con un conjunto de funciones para la construcción de la ontología como: *browse*, *create*, *edit*, *modif.,etc,* y una biblioteca de ontologías modulares y reusables. Aunque había sido un lenguaje importante para la representación de la ontología desde que comenzó su desarrollo, no es recientemente activo debido al advenimiento de los lenguajes como XML. A continuación se muestra un ejemplo de código de Ontolingua:

```
(define-class Tutoring-objective (?t-obj)
"Attributes are also represented as slots."
:def (and (individual ?t-obj)
(value-type ?t-obj Tutoring.policy Policy))
```

:axiom-def
(subclass-partition
Tutoring-objective
(setof Transfer-of-knowledge Remedy)))

Ésta es una aplicación de una clase “Tutoring objective” en una ontología de tarea para el sistema ITS (Intelligent Tutoring System) donde la instancia es llamada *?t-obj*. Los paréntesis en la segunda línea significan comentario. *:def* y *:axiom-def* les permite a los usuarios escribir una condición y definición necesaria (condición necesaria y suficiente), respectivamente. Los predicados-funciones, *value-type*, *subclass-partition* y *setof* significan que una instancia es más “individual” que un conjunto, el tipo de valor de un slot desde el punto de vista semántico, particiona una clase dentro de subclases.

1.6.2 RDF(S): *Resource Description Framework.*

Es un framework para descripción del desarrollo de los metadatos, desarrollada por el W3C (consorcio WWW). Este emplea un modelo “triple” <objeto, atributo, valor>, muy conocido en la comunidad de la Inteligencia Artificial, donde el objeto es llamado recurso, que representa una página *web*. Un mismo “triple” puede ser un objeto y un valor. El valor puede ser una cadena de caracteres o un recurso. El objeto y el valor son considerados como un nodo y el atributo como un enlace entre los nodos. Así, un modelo RDF forma una red semántica. RDF tiene una sintaxis basada en XML (llamada serialización), qué hace que se parezca al lenguaje XML común.

Los metadatos son datos sobre datos. En el contexto de las informaciones procesadas en Internet cualquier recurso contiene información que es considerada como una instancia de una cierta clase.

El siguiente esquema RDF (fig.1.1) contiene clases y metaclases, lo cual los usuarios pueden definir cualquier clase y relación. *Rdfs:Resource* y sus dos subclasses: *rdf:sClass* y *rdfs:Property* son las metaclases importantes.

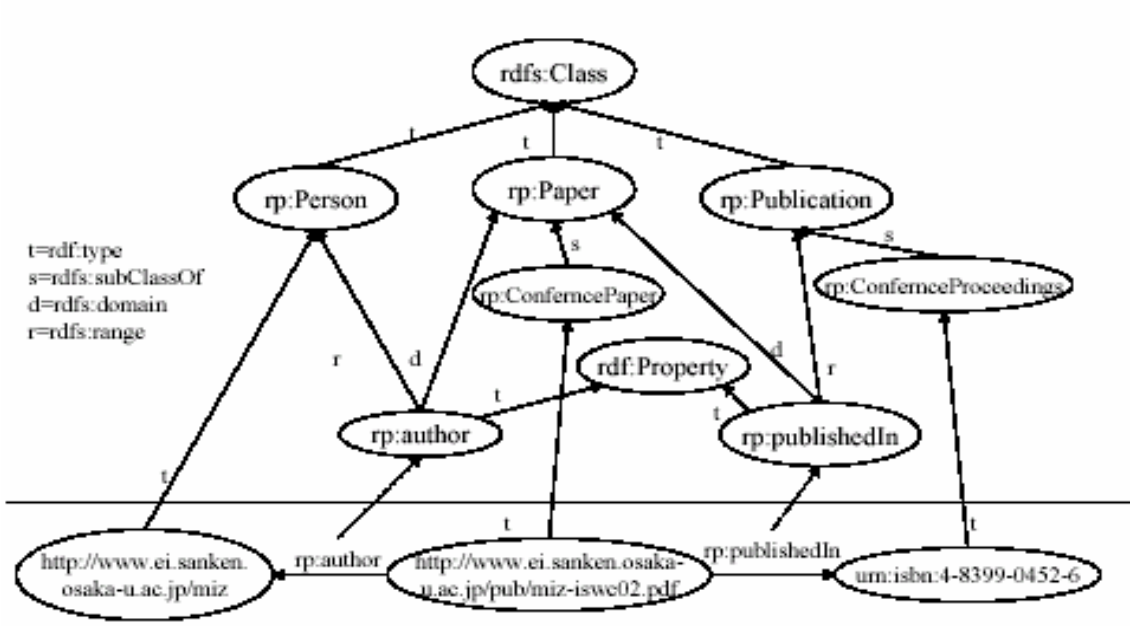

Fig.1.1: Esquema RDF.

Cada clase ordinaria definida en el esquema de RDF es una instancia de *rdfs:Class*. De la misma manera, cada propiedad y relación definida en el esquema de RDF es una instancia de *rdfs:Property*. Por ejemplo, *rdf:subclass-* es una instancia de *rdfs:Property* y es una relación empotrada. Esto muestra que los atributos y las relaciones no son distinguibles en el esquema RDF. Las relaciones y los atributos se definen globalmente, esto es, independientemente de cualquier clase de lenguajes basados en frames, en la cual un atributo es definido como un slot de cada clase. Esto viene de las convenciones de DL.

La siguiente figura muestra una jerarquía de la clase simple del esquema RDF que incluye algunas aplicaciones orientados a clases. El código de la jerarquía del esquema RDF sería el siguiente:

```
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
<rdfs:Class rdf:ID="Paper">
<rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-
schema#Resource"/>
</rdfs:Class>
<rdfs:Class rdf:ID="ConferencePaper">
<rdfs:subClassOf rdf:resource="#Paper"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Person"/>
```

<rdfs:Class rdf:ID="Publication"/>
<rdfs:Class rdf:ID="ConferenceProceedings">
<rdfs:subClassOf rdf:resource="#Publication"/>
</rdfs:Class>
<rdf:Property rdf:ID="title">
<rdfs:domain rdf:resource="#Paper"/>
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</rdf:Property>
<rdf:Property rdf:ID="name">
<rdfs:domain rdf:resource="#Person"/>
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</rdf:Property>
<rdf:Property rdf:ID="author">
<rdfs:domain rdf:resource="#Paper"/>
<rdfs:range rdf:resource="#Person"/>
</rdf:Property>
<rdf:Property rdf:ID="publishedIn">
<rdfs:domain rdf:resource="#Paper"/>
<rdfs:range rdf:resource="#Publication"/>
</rdf:Property>
</rdf:RDF>

1.6.3 SHOE: *Simple HTML Ontology Extensions*.

Fue el primer lenguaje de etiquetado para diseñar ontologías en la Web. Este lenguaje nació antes de que se ideara la Web Semántica. Las ontologías y las etiquetas se incrustaban en archivos HTML. Este lenguaje permite definir clases, interrelaciones, reglas de inferencia, pero no negaciones o disyunciones. Se utiliza para mejorar búsquedas en la Web, no tiene construcciones para representar restricciones en un dominio que son necesarios para el Diseño de Base de Datos. Con él se desarrollaron muchos editores, buscadores, APIS, etc.; pero este proyecto fue abandonado a medida que se desarrollaron OIL y DAM; aunque también existe una serialización de este lenguaje en XML.

1.6.4 OIL: *Ontology Inference Layer*.

Este lenguaje, derivado en parte de SHOE, fue impulsado también por el proyecto de la Unión Europea On-To-Knowledge. Utiliza ya 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). OIL posee varias capas de sub-lenguajes, entre ellas destaca la capa base que es RDFS, a la que cada una de

las capas subsiguientes añade alguna funcionalidad y mayor complejidad. La principal carencia de este lenguaje es la falta de expresividad para declarar axiomas.

1.6.5 DAML + OIL:

Este lenguaje nació fruto de la cooperación entre OIL y DARPA y unifica los lenguajes DAML (*DARPA's Agent Markup Language*) y OIL (*Ontology Inference Layer*). Se basa ya en estándares del W3C. El lenguaje DAML se desarrolló como una extensión del lenguaje XML y de Resource Description Framework (RDF) y 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 máquina. 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

1.6.6 OWL: *Ontology Web Language* o Lenguaje de Ontologías para la Web.

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 RDF, 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.).

1.6.7 **KIF**: *Knowledge Interchange Format*

Es un lenguaje para representar ontologías basadas en la lógica de primer orden. KIF está basado en la lógica de predicados con extensiones para definir términos, metaconocimiento, conjuntos, razonamientos no monotónicos, etc.; y pretende ser un lenguaje capaz de representar la mayoría de los conceptos y distinciones actuales de los lenguajes más recientes de representación del conocimiento. Se trata de un lenguaje diseñado para intercambiar conocimiento entre sistemas de computación distintos, diferentes lenguas, etc.; y no para la interacción entre seres humanos.

1.7 Clasificación de las ontologías.

En el año 2000, Sowa [6] clasifica las ontologías según por el grado de axiomatización como:

1- Terminológicas: Definen términos y sus relaciones, pero no axiomas y definiciones expresadas en lógica o algún tipo de lenguaje interpretable por computadoras que pueda ser traducido automáticamente a la lógica.

Ejemplos

EDR Electronic Dictionary Research	440 000 conceptos
Word Net	166 000 conceptos

2- Formales: Explicada anteriormente.

Y por dependencia del contexto:

1- Ontologías de dominio: Específicas a un dominio con restricciones en contenido y relaciones.

EngMath Ontology

2- Ontologías TOVE (Toronto Virtual Enterprise): Modelado de procesos empresariales.

Enterprise Design Ontology.

Project Ontology.

Scheduling Ontology.
Service Ontology.

3- Generales: Vocabulario relacionado a cosas, tiempo, etc.
CYC provee gran cantidad de conocimiento humano general.

4- Meta Ontologías: Similares a las de dominio con conceptos genéricos a través de diferentes áreas de conocimientos. Describen conceptos como estado, evento, acción, etc.
The Meteorology Ontology.

5- Ontologías de tareas: Proveen un vocabulario sistemático de los términos usados para resolver problemas asociados con tareas particulares. Tiene conceptos como “observación, hipótesis, objetivo, etc.”

Capítulo 2: "Herramientas."

En los últimos años, el número de herramientas para la construcción de ontologías desarrolladas por las comunidades americanas y europeas son altas. Cuando una nueva ontología va a ser construida, varias preguntas básicas forman incógnitas relacionado a las herramientas a usar como: ¿Qué herramientas apoyan el proceso de desarrollo de la ontología? ¿Cómo se almacenan las ontologías (en bases de datos o ficheros ASCII)? ¿La herramienta tiene máquina de inferencia? ¿Se puede traducir (o exportar) la herramienta en otro lenguaje de ontología diferente? ¿Cuál es la calidad de las traducciones?

En este segundo capítulo se tiene como objetivo principal describir las características principales de las herramientas. Dentro de estas características mencionaremos:

- 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.

- Uso, donde se analizará la existencia de editores gráficos.

2.1 Apolo.

Apolo es una herramienta ontológica para el desarrollo de aplicaciones. El diseño fue motivado por experiencias de trabajo de compañeros de la rama industrial quienes desearon el uso de técnicas del modelo de conocimiento, pero requerían un ambiente y una sintaxis fácil para su uso y entendimiento (fig 2.1).

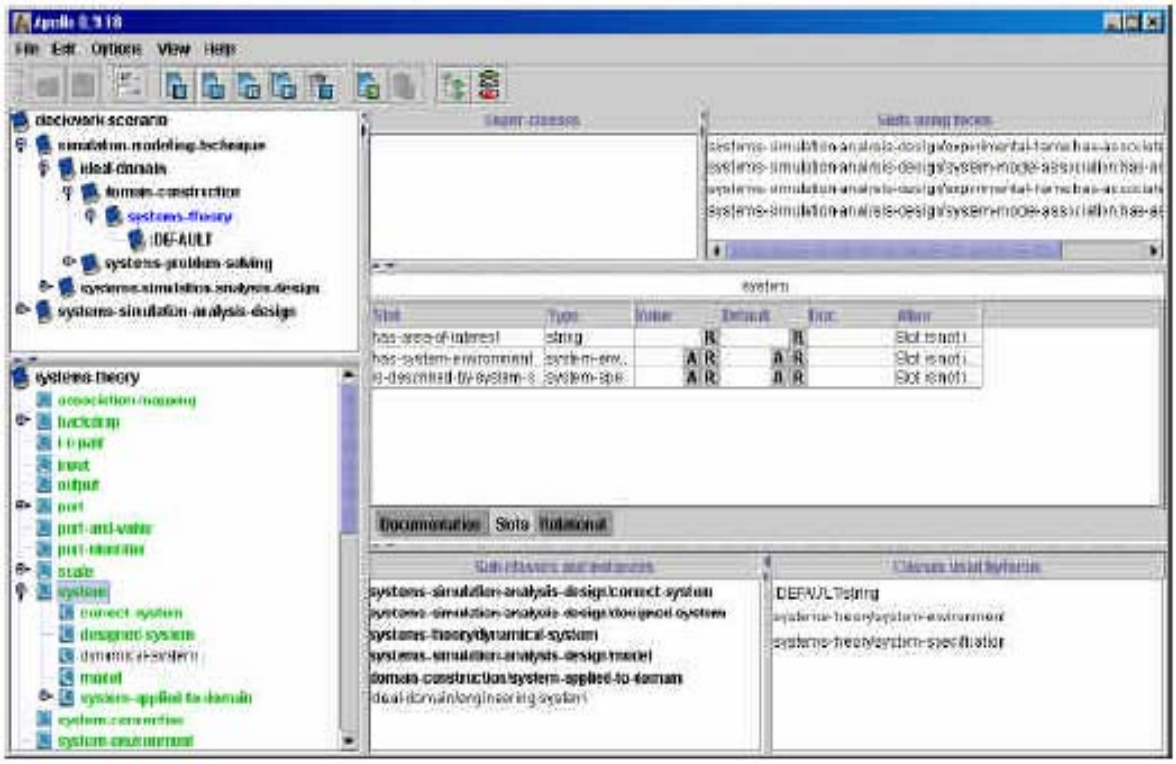

Fig.2.1: Vista de la herramienta Apolo.

Una representación jerárquica de ontologías se muestra en el lado izquierdo superior. La jerarquía de clases e instancias se muestra en el lado izquierdo inferior. Una vez seleccionado una clase o instancia, es mostrada en la parte derecha de la pantalla. Los slots y valores de una clase o instancia pueden ser adicionado usando una interfaz de estilo de hoja de cálculo.

Apolo soporta todas las primitivas básicas del modelo de conocimiento: las ontologías, las clases, las instancias, las funciones y relaciones. Mientras se edita, Apolo realiza todo el chequeo de consistencia, por ejemplo, detectar el

uso de clase no definida. Apolo tiene su propio lenguaje interno para almacenar las ontologías, y además puede exportar la ontología dentro de diferentes representaciones de lenguajes, según desee el usuario. Apolo es implementado en Java.

2.2 LinkFactory®.

LinKFactory® es un sistema de dirección de ontología formal desarrollado por Language & Computing, diseñado para construir y manejar las ontologías formales idioma-independientes muy grandes y complejas. El sistema LinKFactory® consiste en 2 componentes mayores: LinKFactory® Server, y LinKFactory® Workbench, ambas desarrolladas en Java.

LinKFactory® salva los datos en una base de datos relacional. El acceso a las bases de datos está resumido por un conjunto de funciones, donde estas funciones son accesibles a los clientes del software a través de una API estandarizada que permite construir aplicaciones al inicio de la base de datos semántica sin requerir conocimientos privados de la estructura interna de la base de datos. Este componente es capaz de ¨comunicarse¨ con múltiples usuarios existentes con plataformas independiente (Windows, Solaris, UNIX y Linux).

LinkFactory® les permite a los usuarios navegar y modelar algunas ontologías como se muestra figura 2.2:

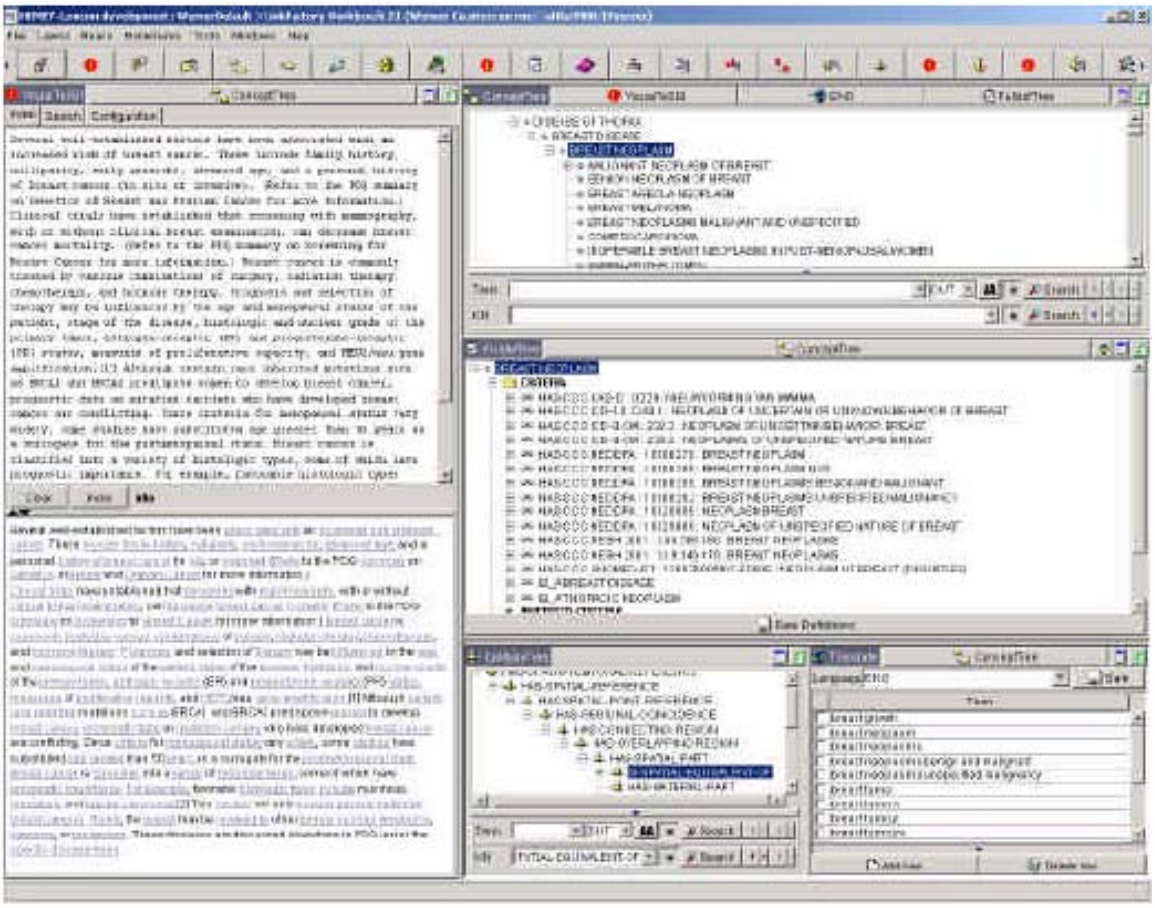

Fig.2.2: Vista de la herramienta LinkFactory®.

2.3 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.

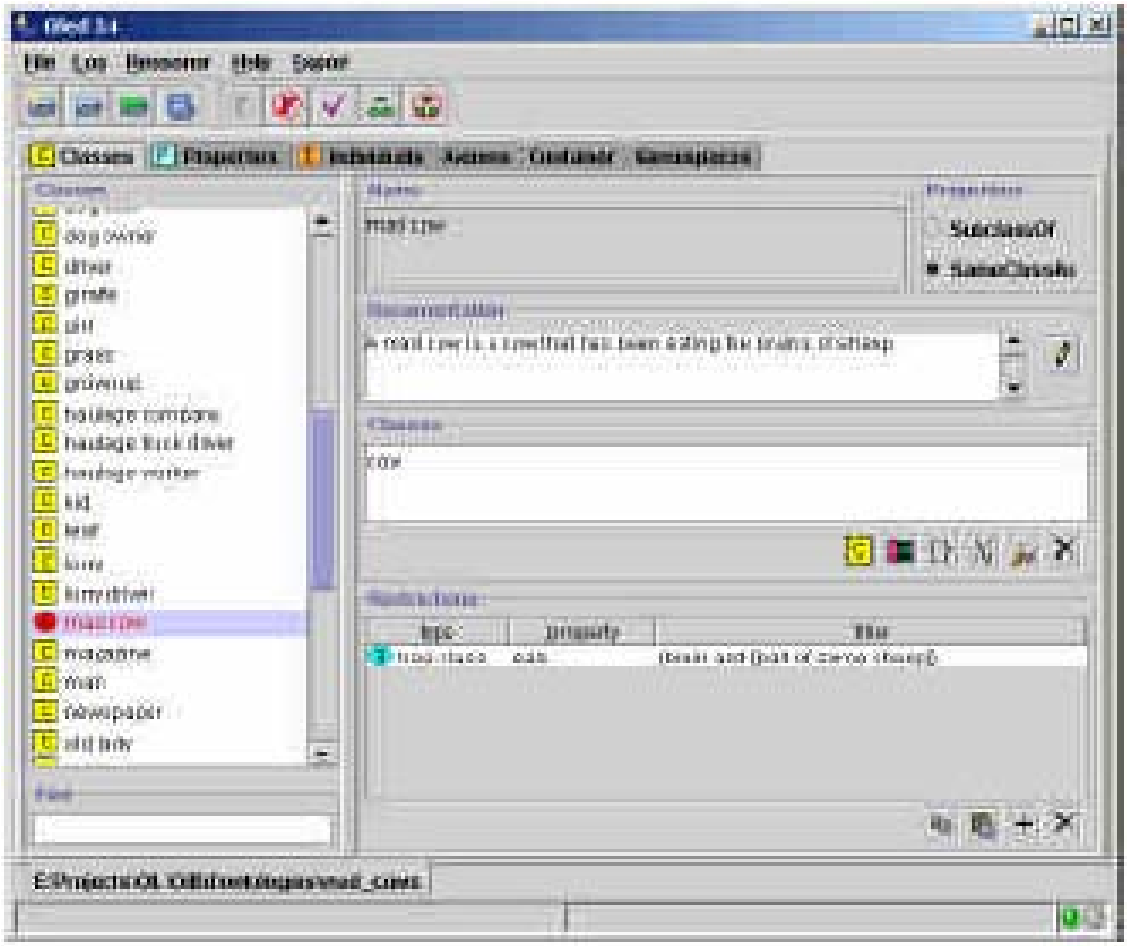

Fig.2.3: Vista de la herramienta OILEd.

Un aspecto importante de OILEd es el uso del razonador FaCT (según Horrocks [5]) para la clasificación de ontologías y el chequeo de consistencia, lo cual permite al usuario describir sus clases de ontologías. DAML+OIL Esquema RDF (marzo del 2001) 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.

OILEd versión 3.4 está implementado en Java y está libremente disponible en el sitio web de OILEd, aunque se requiere una dirección de correo electrónico para su transmisión(download).

2.4 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 preparados para soportar la fase de captura de la ontología. OntoKick es diseñada para informáticos quienes están familiarizados con el proceso de desarrollo de los softwares, 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. La figura 2.4 muestra una vista de esta herramienta.

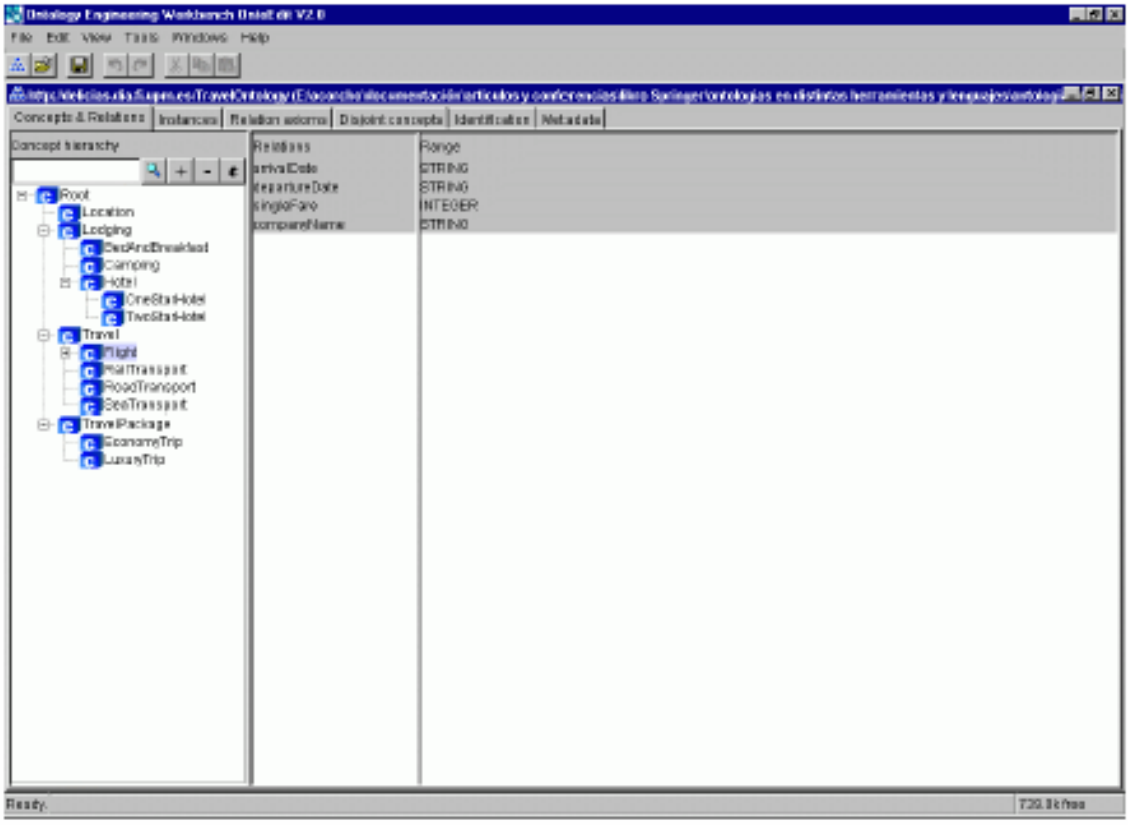

Fig.2.4: Vista de la herramienta OntoEdit.

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.

Su funcionalidad está extendida por:

1- Un plugin de inferencia para el chequeo de consistencia, clasificación y ejecución de reglas.

2- Ingeniería colaborativa de ontologías.

3- Un servidor de ontologías para la administración de bibliotecas de ontología.

OntoEdit Versión Profesional 2.0 y 2.5 incluye la funcionalidad 1, mientras que la versión 3.0 incluyen las funcionalidades 2 y 3.

2.5 Ontolingua Server.

Ontolingua Server es un conjunto de herramientas y servicios que admite la construcción de ontologías compartidas entre grupos distribuidos, y ha sido desarrollado por el Laboratorio de Sistemas de Conocimiento (KSL, sus siglas en inglés) en la Universidad de Stanford. La arquitectura del servidor de ontología en esta herramienta proporciona el acceso a una biblioteca de ontologías, traductores a lenguajes (Prolog, CORBA IDL, CLIPS, Loom, etcétera), y un editor para crear un navegador de ontología como se muestra en la figura 2.5:

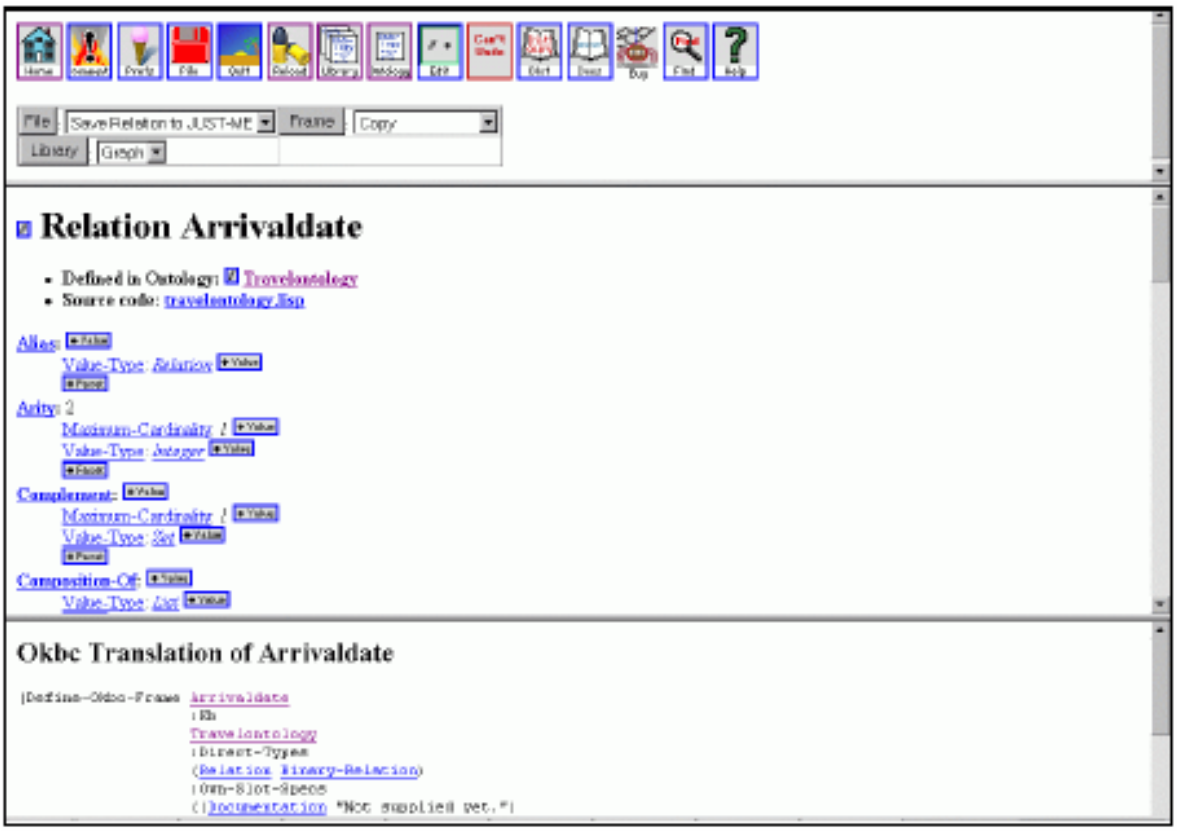

Fig.2.5: Editor de Ontolingua Server.

Los navegadores remotos pueden navegar y editar ontologías, y las aplicaciones tanto remotas como locales pueden acceder cualquiera de las

ontologías en la librería de ontologías usando el protocolo OKBC (Open Knowledge Based Connectivity).

2.6 OntoSaurus.

OntoSaurus fue desarrollado por el Instituto de Ciencias de Información (ISI, sus siglas en inglés) en la Universidad de California del Sur. Esta herramienta consiste en dos módulos: un servidor de ontología que usa LOOM como Sistema de Representación del Conocimiento, y un servidor navegador de ontología que dinámicamente son vistas como páginas HTML (incluyendo imágenes y documentaciones textuales), que muestra la jerarquía de la ontología, como se muestra a continuación (figura 2.6):

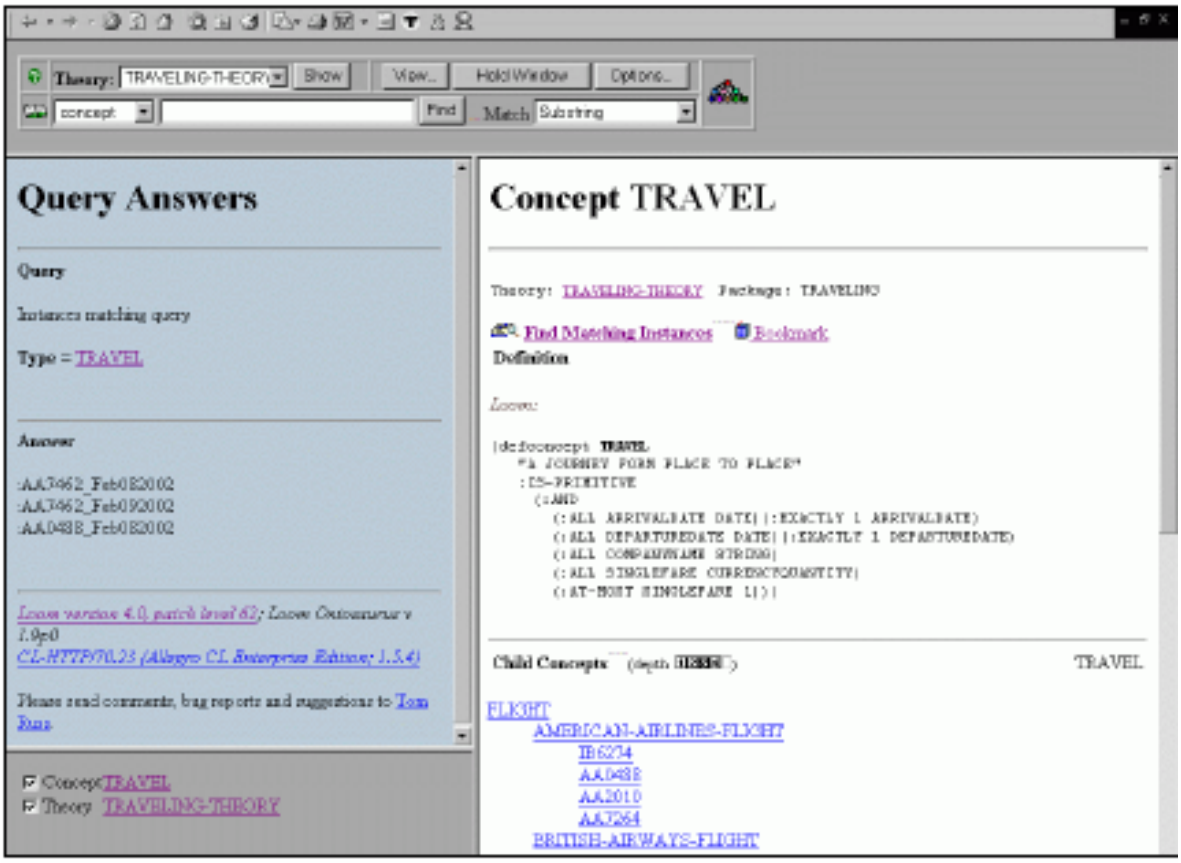

Fig.2.6: Jerarquía de la Ontología en la herramienta OntoSaurus.

La ontología puede ser editada por formas HTML, y existen traductores de LOOM para Ontolingua, KIF, KRSS and C++.

2.7 OpenKnoME.

KnoME contiene una gran colección de herramientas para el desarrollo de ontologías. Tigger es una parte importante de esta colección, desarrollada para la adquisición rápida del conocimiento de los expertos usando ingeniería ontológica. Juntos, estas herramientas forman el OpenKnoME. Ellos han sido desarrollados por la Universidad de Manchester sobre varios programas de ontologías médicas y farmacéuticas.

Las APIs en KnoME proporcionan una distinción entre la ontología y los clientes que lo usan. La ontología es presentada como un servicio en lugar de estructuras de datos. Además, KnoME permite navegar, explorar, visualizar, e interrelacionar clases.

El KnoME se ha usado para coordinar el desarrollo de la ontología para proyectos de terminología médica en Estados Unidos; en Inglaterra para un proyecto ante las drogas (NHS); así como en otras esferas internacionales e industriales.

OpenKnoME versión 5.4 está implementada en Smalltalk, y está libremente disponible en el sitio web: *www.topthing.com*, al igual que los manuales y tutoriales acerca de esta herramienta.

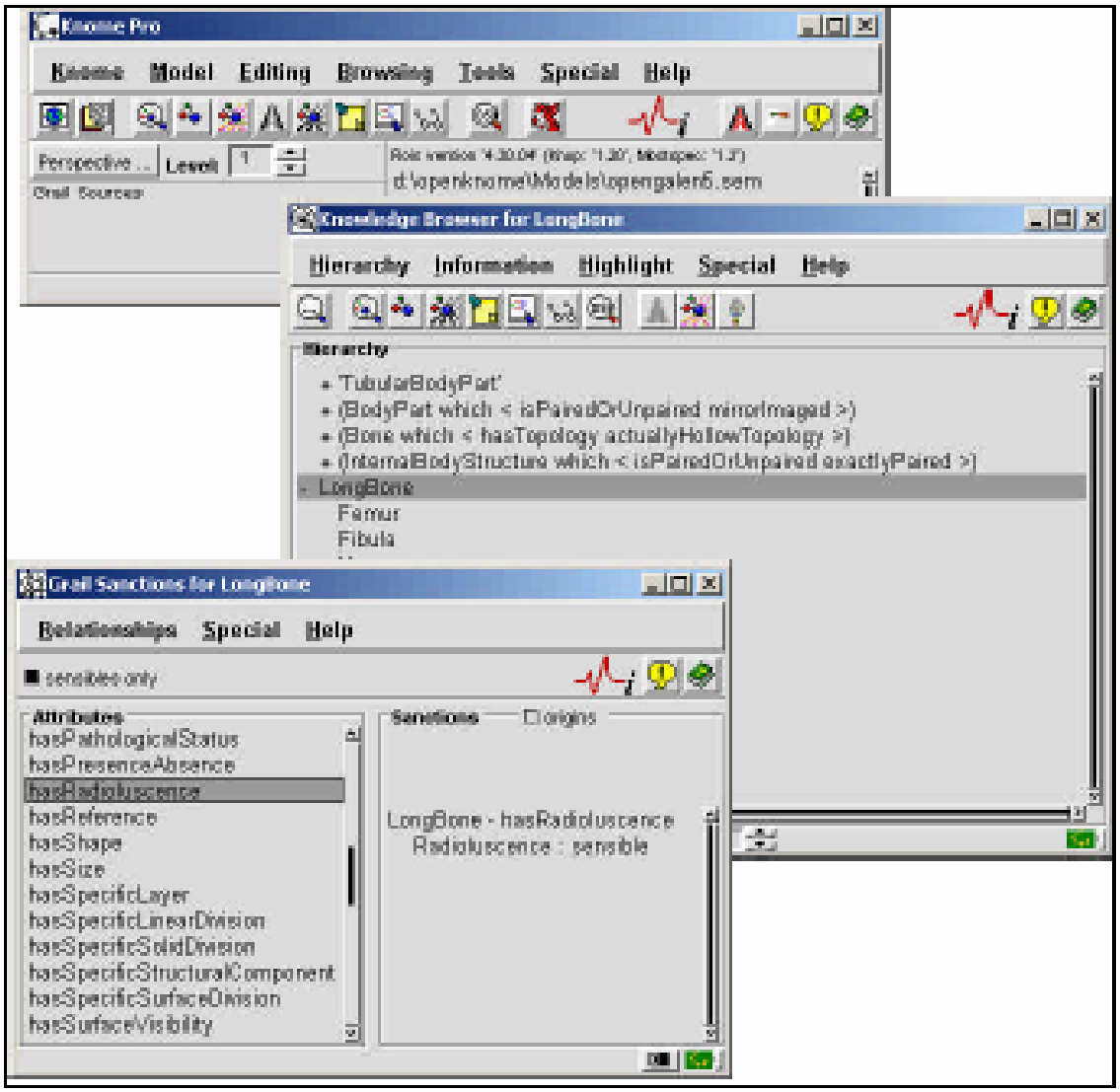

Fig.2.7: Vista de la herramienta OpenKnoME.

2.8 SymOntoX.

SymOntoX (Symbolic Ontology Manager XML), es un prototipo de software para las ontologías del dominio. Ha sido desarrollado por LEKS (Laboratory for Enterprise Knowledge and Systems).

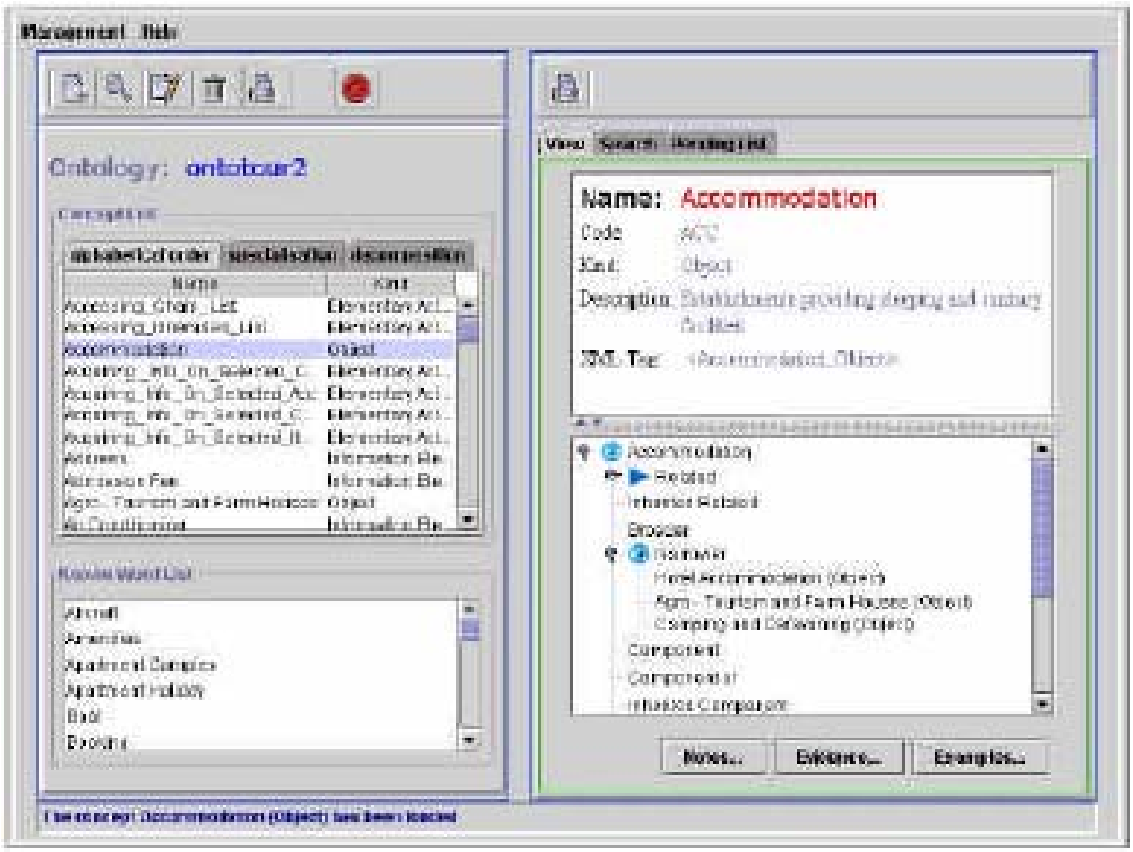

Fig.2.8: Vista de la herramienta SymOntoX.

SymOntoX, que es el sucesor de SymOntos, está desarrollándose y experimentándose actualmente dentro de un Proyecto Europeo relacionado con el dominio del turismo.

En SymOntoX, los conceptos y relaciones del dominio son modelados acorde con OPAL (Object, Process, Actor modelling Language, o lo que es lo mismo en español, Lenguaje de modelación de Objeto, Proceso, Autor), una metodología para la representación de la ontología desarrollado por LEKS.

Con OPAL los conceptos son organizados por medio de tres ideas del modelado primario: actor, proceso y objeto:

- Actor: cualquiera entidad del dominio que puede activar o realizar un proceso (por ejemplo, Turista, Agencia de Viaje).

- Proceso: una actividad específica de acuerdo con la satisfacción de un objetivo (por ejemplo, Hacer_una_Reservación).

- Objeto: una entidad pasiva donde opera un proceso (por ejemplo, Hotel, Vuelo).

Además de las ideas del modelado primario anterior, OPAL propone las siguientes ideas del modelado complementario:

- Componente de información: determinada información que pertenece a la estructura de información de un Actor o un Objeto (por ejemplo, Información_de_Vuelo, Dirección_de_Hotel).

- Elemento de información: elemento de información atómico que es parte de un Componente de Información (por ejemplo, Precio_de_Vuelo, Numero_de_Habitaciones).

- Acción: actividad que representa un componente del proceso (por ejemplo, Pedir_Habitación).

- Objetivo: estado deseado que un autor busca alcanzar (por ejemplo, Ir_de_Vacaciones).

- Estado: característica de valores que las variables instanciadas de una clase pueden tener (por ejemplo, Vuelo_Lleno).

- Regla: es una expresión que restringe los posibles valores de una instancia de una clase (regla de restricción) o que permita derivar nueva información (regla de producción) (por ejemplo, Boleto_30_días_Antelación).

Las ideas del modelado anterior son necesarias para definir los conceptos. Según OPAL, los conceptos son unidos juntos por medio de varias relaciones ontológicas, Especialización, Descomposición, Predicción, Similitud y Relación.

SymOntoX se ha concebido para ser un servicio disponible en Internet usando un navegador web común.

Es principalmente basado en XML (todos los datos se guardan en una base de datos de XML) y tecnologías de Java, para garantizar flexibilidad máxima, interoperatividad y plataforma independiente.

SymOntoX puede manejar diferentes ontologías, diferentes tipologías de usuarios y diferentes modos de uso. Un usuario puede registrarse como un *User* (con sólo leer los derechos), como un *SuperUser* (el o ella pueden insertar nuevos conceptos, pero sólo como propuestas), o como el *OntologyMaster* (quién es el responsable de la ontología), donde a su vez este tiene la tarea de aceptar o negar las propuestas hechas por los *SuperUsers*.

SymOntoX admite una forma basada en la interfaz gráfica de usuario. Todos los datos (ontología, conceptos, instancias) son almacenados en una base de datos XML.

2.9 WebODE.

WebODE es un integrado ambiente de trabajo desde el punto de vista del diseño de la ontología sobre el desarrollo de la metodología.. Este solo incluye las ontologías del nivel de conocimiento, y lo traduce a lenguajes diferentes. WebODE es diseñado sobre la base de una arquitectura general como se muestra en la figura 2.9:

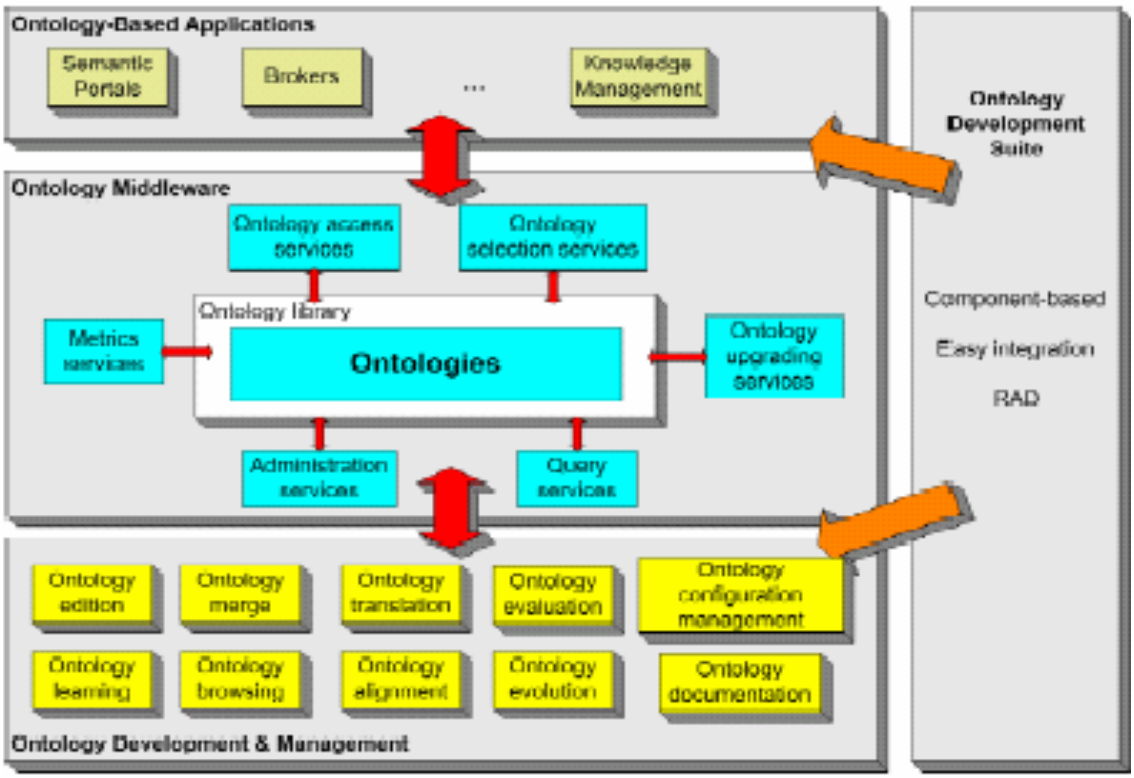

Fig.2.9: Arquitectura general de WebODE.

Mientras Protégé-2000 y OntoEdit son basados en la arquitectura plug-in, WebODE es basado en una arquitectura cliente-servidor que proporciona gran extensibilidad y utilidad, permitiendo el crecimiento de nuevos servicios y el uso de servicios existentes. Las ontologías son almacenadas en una base de datos SQL con el objetivo de lograr una gran eficacia en el caso de una ontología grande. Tiene como gran importancia también la posibilidad de exportar e importar servicios desde y hacia estructuras *XML*, y la traducción de servicios dentro y desde varios lenguajes de ontologías como: *RDF(S), OIL, DAML+OIL, CARIN, F-logic*.

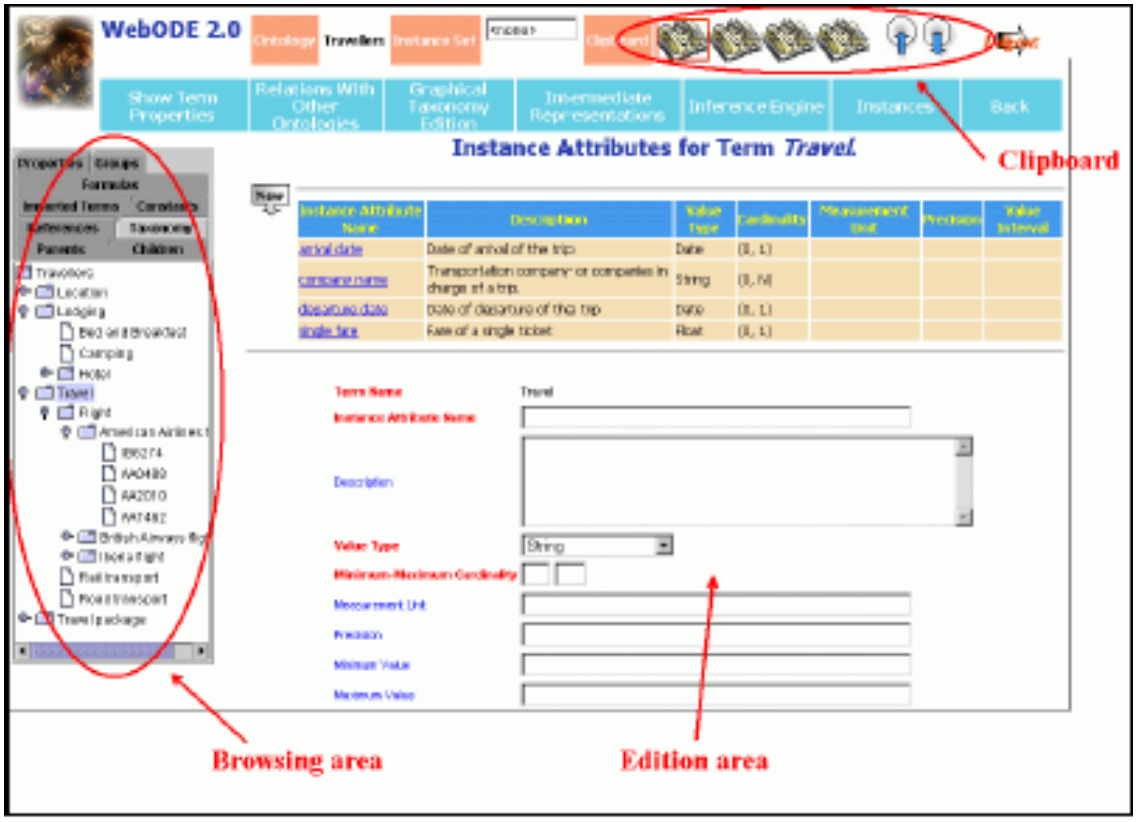

Fig.2.10: Vista de la herramienta WebODE.

WebODE es basada en una interfaz gráfica de usuario, lo cual permite navegar por todas las relaciones definidas en la ontología, además contiene una máquina de inferencia, un constructor de axiomas, y un servicio de documentación. Tiene un buen chequeo de restricciones como por ejemplo, las restricciones de tipos, de valores numéricos, de cardinalidades.

WebODE ha sido desarrollado en Ciao Prolog. Un subconjunto del protocolo OKBC ha sido definido en Prolog para su uso en esta máquina de inferencia.

2.10 WebOnto.

WebOnto es una herramienta desarrollada por el Instituto Medio de Conocimientos (KMI, sus siglas en inglés) de la Universidad Open en Inglaterra.

WebOnto admite un navegador colaborativo para la creación y edición de ontologías, la cual es representada en el Lenguaje de Modelación del Conocimiento (OCML).

Como rasgos principales tiene la administración de ontologías que usan una interfaz gráfica (como se muestra en la figura 2.11) que permite la generación automática de instancias desde definiciones de clases, la inspección de elementos teniendo en cuenta la herencia de propiedades y el chequeo de consistencia, entre otras facilidades.

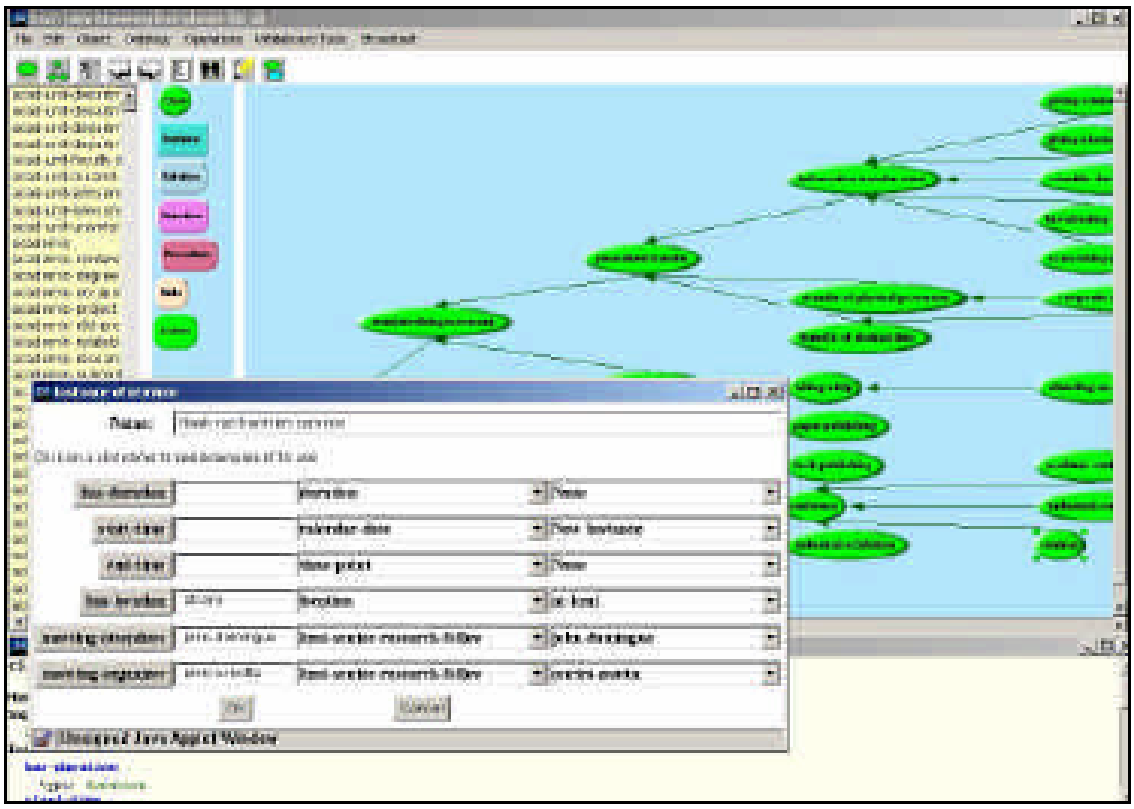

Fig.2.11: Vista de la herramienta WebOnto.

El servidor de WebOnto es un servicio libre y cuenta con una biblioteca de más de cien ontologías y pueden ser utilizadas sin restricciones en el acceso.

2.11 Protégé.

Esta herramienta se utiliza para el desarrollo de ontologías y sistemas basados en el conocimiento, creada en la Universidad de Stanford. Protégé está desarrollada en JAVA y puede funcionar perfectamente bajo WINDOWS.

Protégé es poderoso en la fase de diseño y utilización de la ontología: Ofrece facilidades para la adquisición del conocimiento y la mezcla de

ontologías existentes. Es extensible mediante módulos o plugin para aumentar su utilidad. Se ha usado durante muchos años para la adquisición del conocimiento, fundamentalmente *conocimiento del dominio* para crear *ontologías de dominio*.

El modelo de conocimiento de Protégé es OKBC. Esto hace que esta herramienta incluya además de las clases, herarquías de clases con múltiples herencias, que emplee una interfaz de usuario que facilita la creación de una estructura de *frames* con clases, formularios, slots e instancias de una forma integrada. Protégé contiene un diseño ontológico gráfico e interactivo.

Las aplicaciones desarrolladas con Protégé son empleadas en la resolución de problemas y toma de decisiones en dominios particulares.

A continuación se muestra un ejemplo de la herramienta Protégé, donde se observa en la parte izquierda las jerarquías de clases con herencias múltiples. En el centro muestra informaciones detalladas de la clase seleccionada, incluyendo las propiedades (slots) que describen las instancias de la clase. En la parte inferior derecho se muestra un formulario para una instancia de una determinada clase.

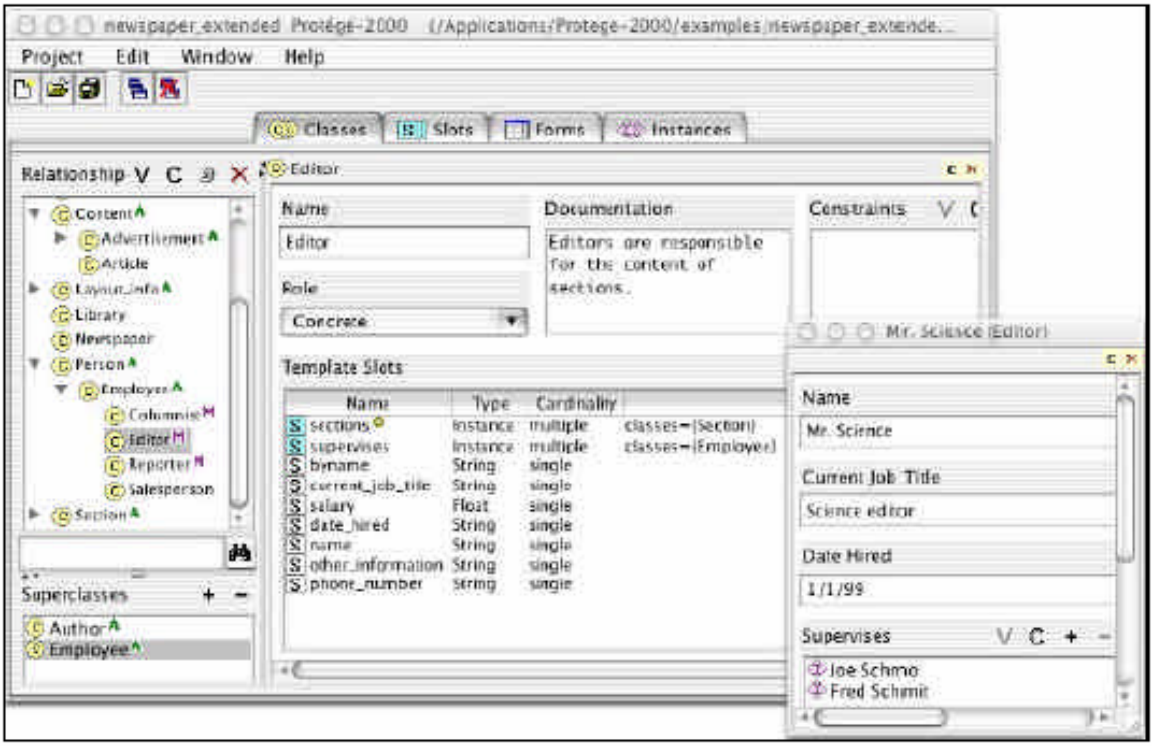

Fig.2.12: Vista de la herramienta Protégé-2000.

Protégé tiene como ventajas importantes:

- Utiliza un conjunto de APIs para diferentes superlenguajes que facilitan la creación de aplicaciones sobre la ontología.

- Tiene su propio lenguaje llamado Protégé Axiom Language (PAL).

- Tiene su propia máquina de inferencia.

- Puede utilizar recursos de multimedia, etc.

Todas estas características la convierten en la más utilizada actualmente en el desarrollo de aplicaciones, por lo que la utilizaremos para construir una ontología sobre palabras.

Capítulo 3: "Implementación de una Ontología para representar palabras de origen africano que se utilizan en Cuba."

3.1 Planteamiento del problema.

La aplicación debe de presentar palabras que tienen como propiedades:

- *nombre*
- *tipo (sustantivo, adjetivo, verbo).*
- *uso (general, restringido).*

Según su *tipo*, las palabras tienen propiedades adicionales las cuales se exponen a continuación:

Para los *sustantivos* interesa:

- *Número (singular, plural).*
- *Género (femenino, masculino).*

Para los *adjetivos* interesa:

- *Número (singular, plural).*
- *Género (femenino, masculino).*

Para los *verbos* interesa:

- *Conjugación (primera, segunda, tercera).*

Debe tenerse en cuenta que cada palabra tiene un *significado* que depende del *campo semántico* y del *reservorio* en que se use dicha palabra.

Los *campos semánticos* se listan a continuación:

- *Comidas y bebidas.*
- *Plantas.*
- *Animales.*
- *Religión.*
- *Hombre social.*
- *Hombre físico.*
- *Objeto.*
- *Fenómenos de la naturaleza.*

- *Nombres propios.*

Los *reservorios*, constituyen una medida de la cantidad de personas que usan la palabra, a continuación se indican los tipos de reservorios en orden creciente de uso.

- *Religiones afrocubanas.*
- *Español coloquial.*
- *Español general.*

También interesa tener acceso a las *fuentes* donde aparece referenciada la palabra, sin interesar precisar *reservorio* ni *campo semántico*, solo saber donde aparece recogida dicha palabra y con qué *significado*; a este tipo de información se le denomina *Información lexicográfica*. Para la *Información lexicográfica*, interesa la *fuente* y una lista de sus posibles *significados*.

Como que estas palabras pueden aparecer en *obras literarias*, también es de interés saber el *título* de las obras donde aparece cada palabra, a este tipo de información se le denomina *Testimonio literario*. Existen palabras que han sufrido cambios en su significado, otras las han mantenido, e incluso, otras palabras que han mantenido su significado y lo han ampliado, a este tipo de información se le denomina *Fenómeno observado*.

Como información a recuperar, interesa:

- Todas las palabras por reservorios.
- Palabras para cada campo semántico.
- Las palabras que han sufrido cambios.
- Cuáles palabras son sustantivos.
- Cuáles palabras son verbos.
- Cuáles palabras son adjetivos.
- Palabra y significado.
- Palabra y autor que referenció a la misma.

La construcción de la ontología se hará con Protégé y a continuación se muestran los pasos a seguir para su implementación.

Teniendo en cuenta el problema planteado se definieron las siguentes clases:

- *Palabra.*
- *Obra_Literaria.*
- *Campo_Semántico.*
- *Reservorio.*
- *Fuente.*
- *Palabra_Cambiada.*
- *Significado.*
- *Información_Lexicográfica.*
- *Test_Literario.*

Para la clase *Palabra* se defieron las subclases:

- *Sustantivo*.
- *Verbo*.
- *Adjetivo*.

3.2 Creación de proyectos.

Un proyecto en Protégé consiste en el desarrollo de una ontología o estructura de conocimiento. Los elementos que se pueden ir creando son fundamentalmente clases, slots, formularios, instancias y consultas, aunque la herramienta es modular y permite adicionar más componentes de una forma sencilla. Cada uno de estos elementos dispone de una etiqueta en la ventana principal de la herramienta, seleccionando cada una de ellas podemos elegir el tipo de elemento concreto sobre el que se va a trabajar.

3.3 Creación del proyecto.

3.3.1 Selección del proyecto.

Al iniciar Protégé se abre una ventana de aplicación donde se selecciona el proyecto de ontología a construir, bien sea uno nuevo o uno ya guardado anteriormente.

Existen distintos tipos de proyectos al crear uno nuevo, por defecto se opta por la opción de “Default Files (.pont and .pins)”, la cual se almacena en tres ficheros:

- Un fichero de proyecto, extensión .pprj, donde se guarda información variada sobre configuración.
- Un fichero de ontología, extensión .pont donde se guarda la definición de los términos de la ontología (clases, propiedades, relaciones, etc.).
- Un fichero de instancias, extensión .pins, donde se guarda la base de conocimiento.

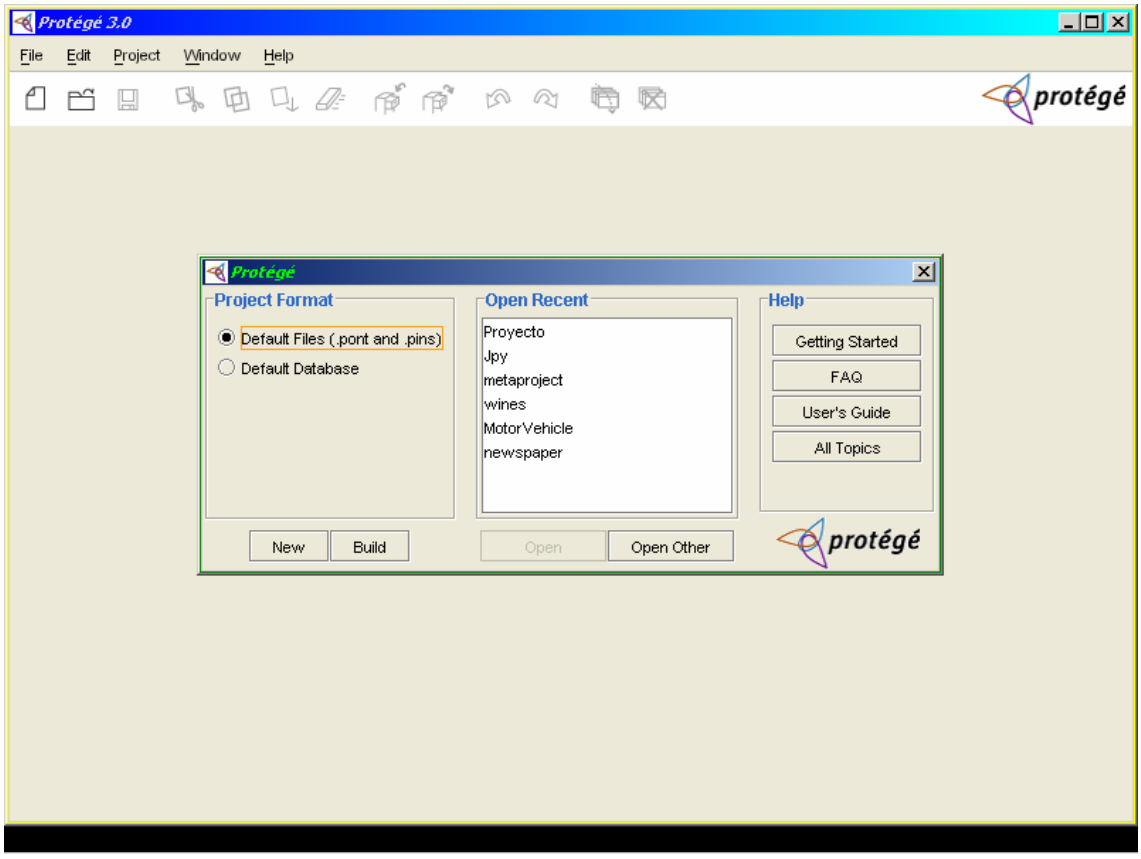

Fig.3.1: Ventanas de selección de proyectos.

Una vez abierto un nuevo proyecto aparece la ventana principal de Protégé donde se puede empezar ya a desarrollar la nueva ontología (figura 3.2).

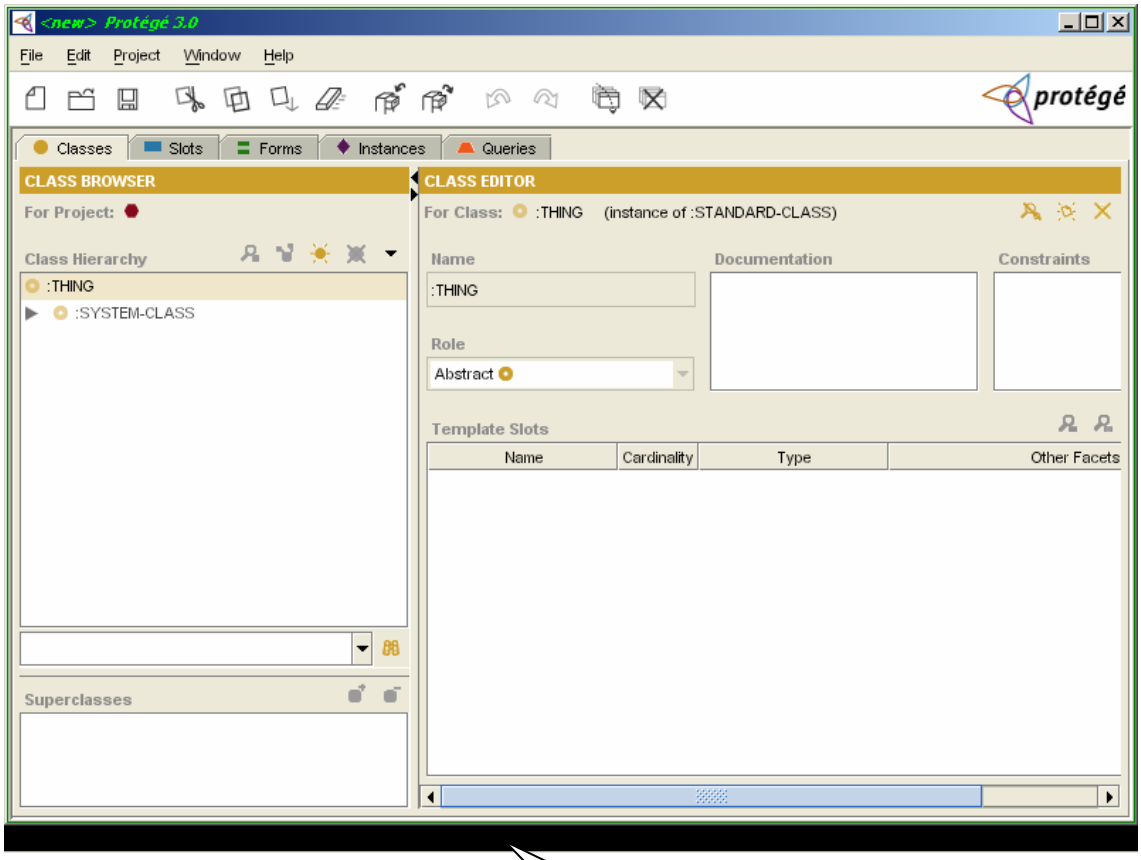

Fig.3.2: Ventana principal de Protégé.

Solo son visibles las clases internas :THING y :SYSTEMCLASS.

3.3.2 Guardar el proyecto.

Para guardar el proyecto (figura 3.3), pinchar en el botón ”Save Project” (o elegir en el menú “Project/Save CTL+S ”).

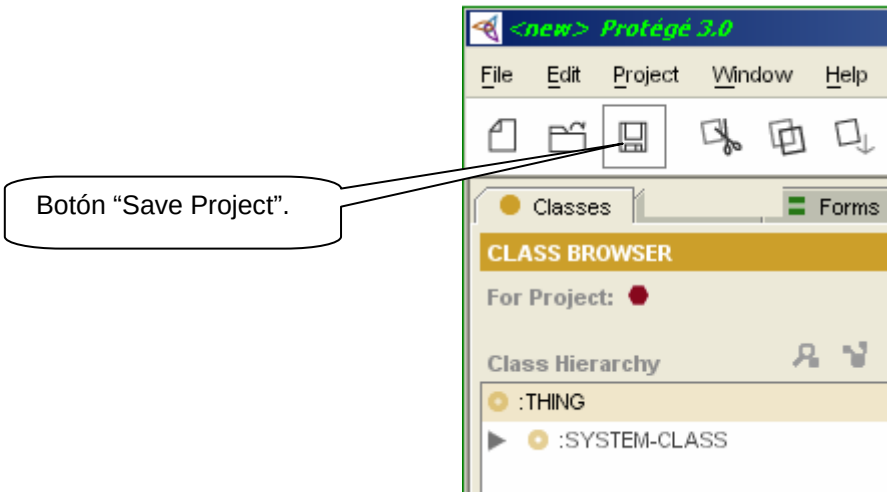

Fig.3.3: Vista para salvar el proyecto.

Al hacer clic en las opciones antes mencionadas, aparece (figura 3.4):

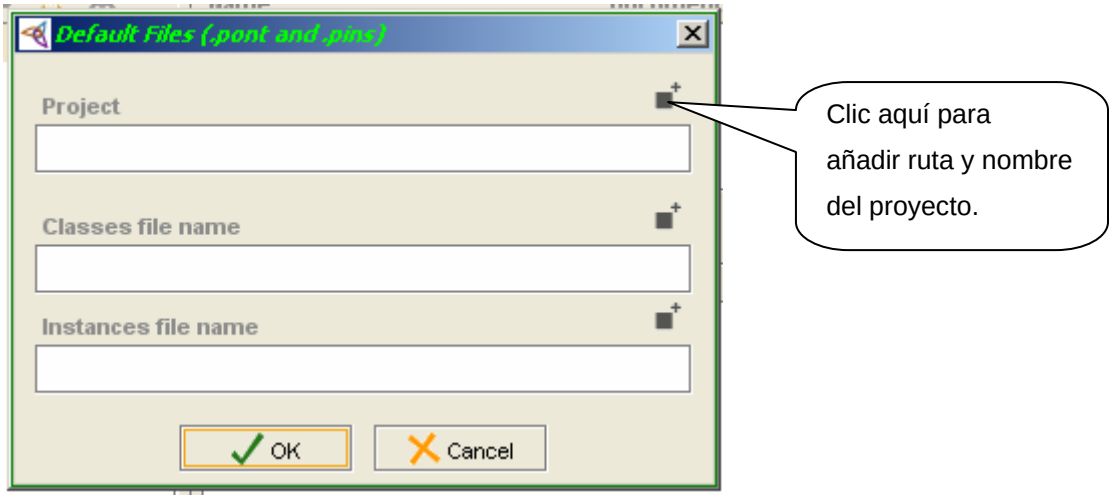

Fig.3.4: Ventana donde se define el camino para salvar el proyecto.

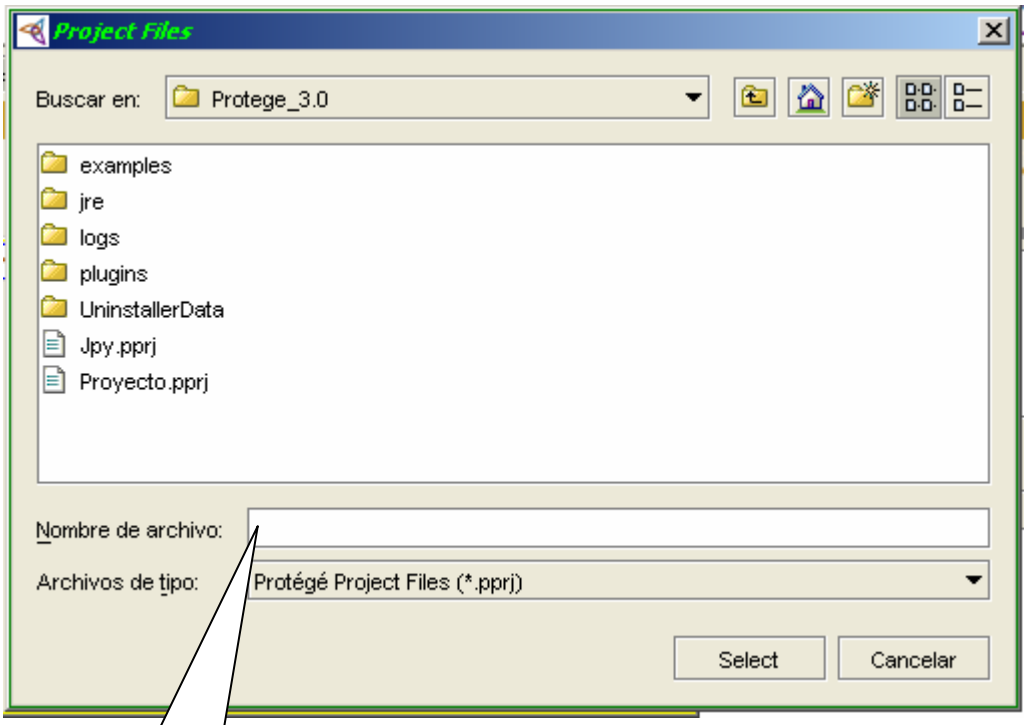

Fig.3.5: Vista para nombrar el proyecto a salvar.

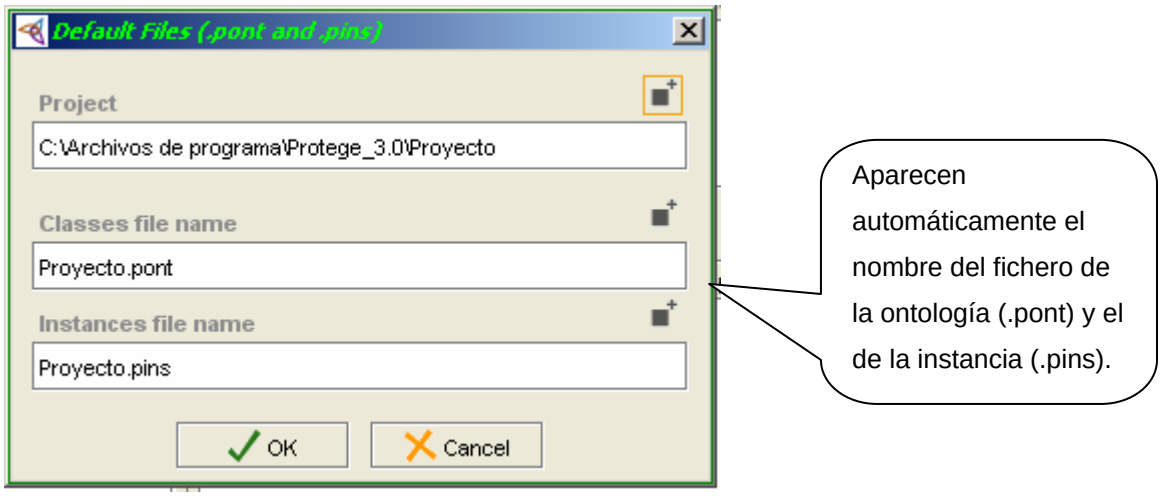

Fig.3.6: : Ventana donde se define el camino para salvar el proyecto (con el camino y nombre incluido).

3.4 Crear clase.

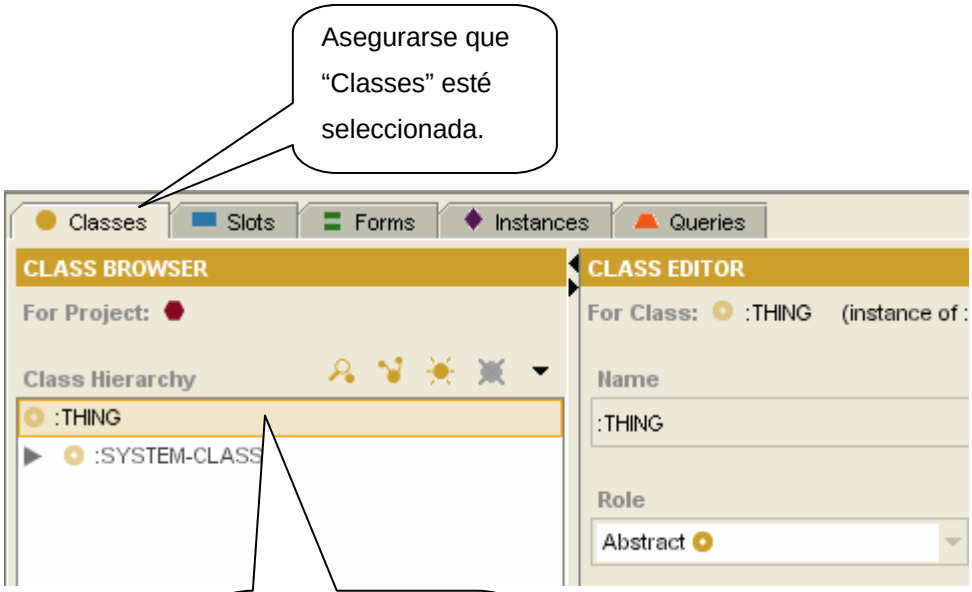

Fig.3.7: Vista para crear clases.

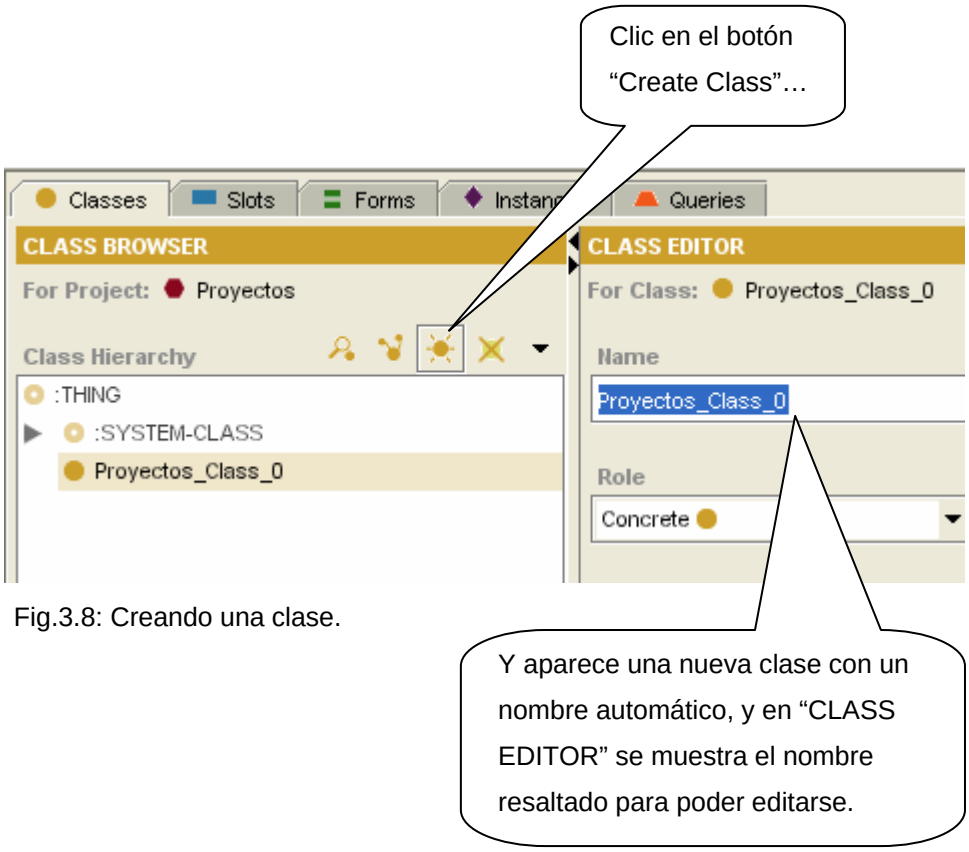

Fig.3.8: Creando una clase.

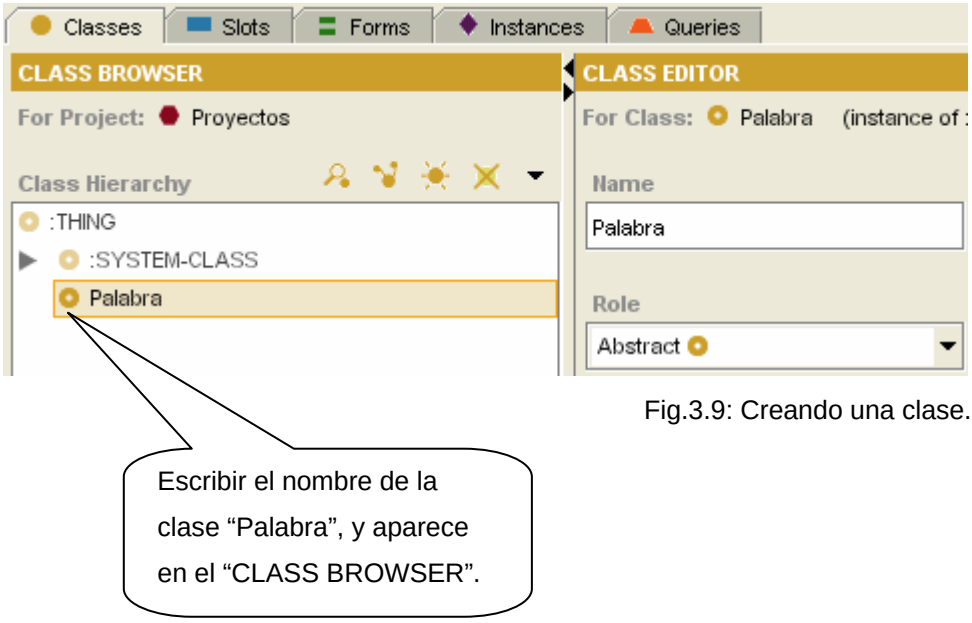

Fig.3.9: Creando una clase.

3.5 Crear subclases y clases.

3.5.1 Crear una subclase.

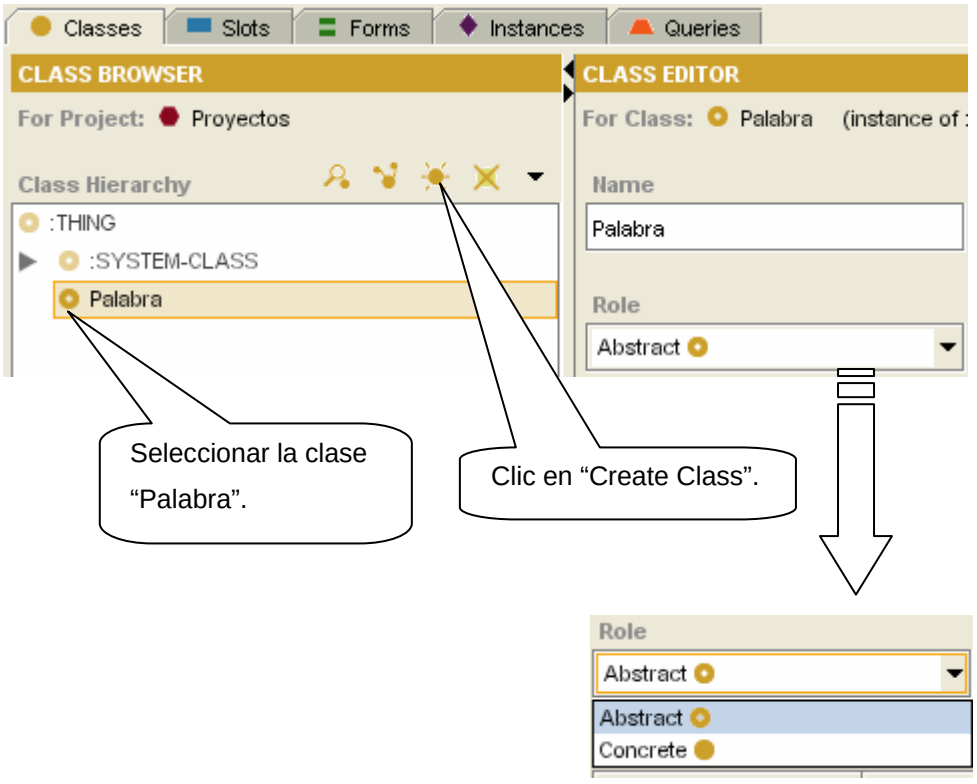

Fig.3.10: Tipos (role) de clases.

Las clases pueden ser “Abstractas” (Abstract) o “Concretas” (Concrete) (figura 3.10). La diferencia está en que a la hora de instanciar las clases, las abstractas simplemente no se instancian, mientras que las clases concretas sí.

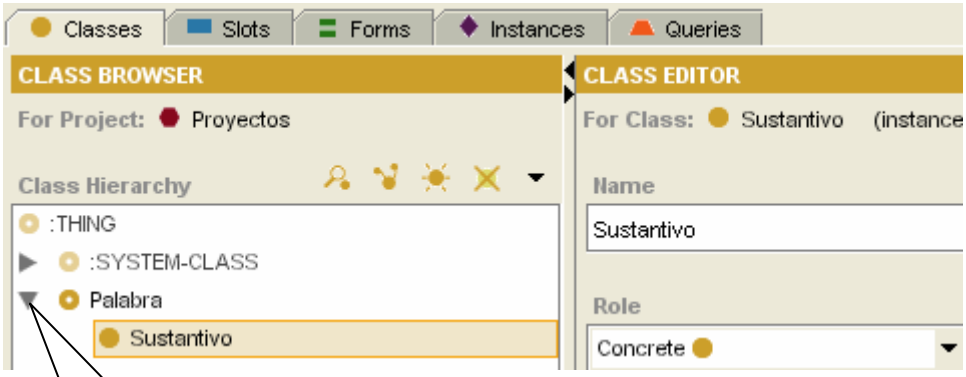

Fig.3.11: Creando subclases.

Luego de renombrar la clase…

Observa que cuando se crea la primera subclase de una clase, aparece un triángulo a la izquierda. Este triángulo puede pincharse para expandir o contraer el árbol de clases.

3.5.2 Crear otra subclase.

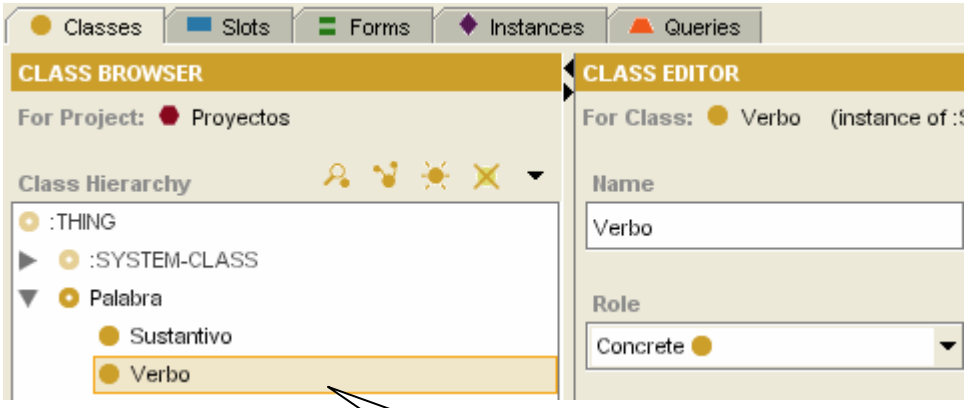

Fig.3.12: Creando otra subclase.

Para crear otra subclase de “Palabra”:

1- Seleccionar de nuevo la clase “Palabra”.

2- Clic en “Create Class”.

3- Renombrar la clase.

3.5.3 Crear otra clase.

Una vez creada la clase “Adjetivo”, subclase de “Palabra”, se crea la clase “Obra_Literaria” (figura 3.13)…

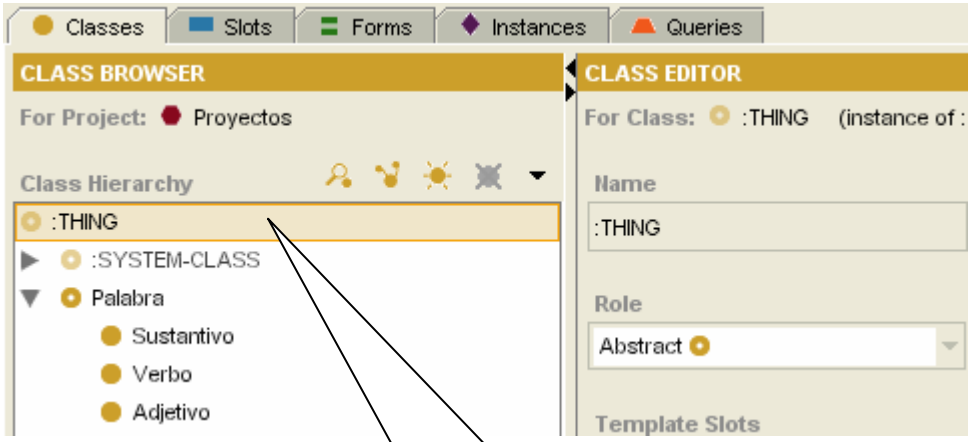

Fig.3.13: Crear otra clase.

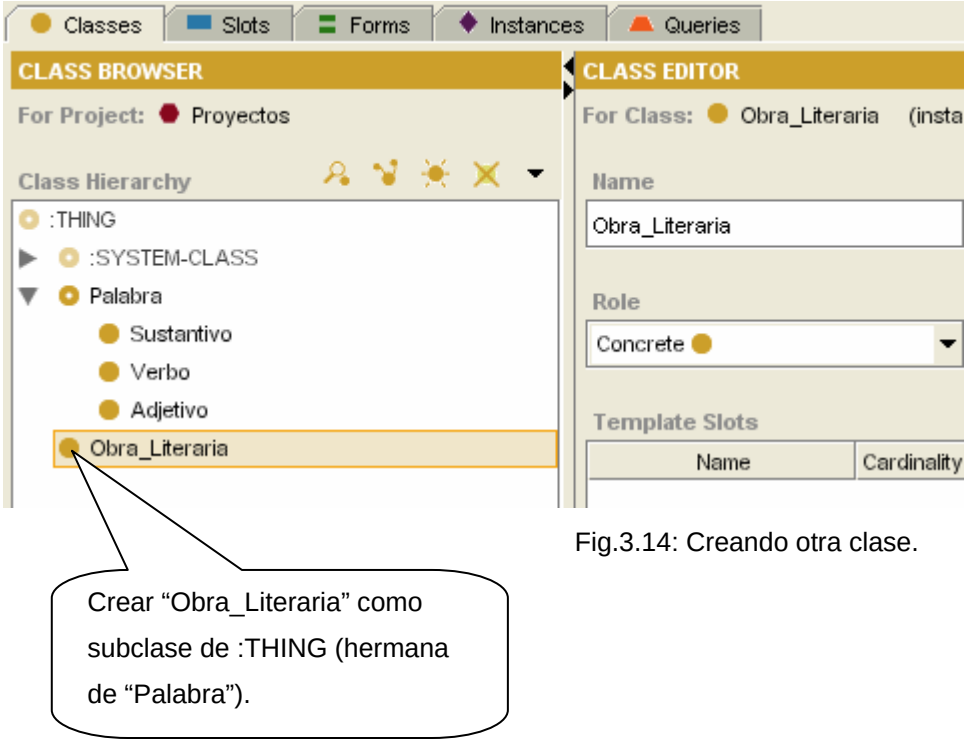

Fig.3.14: Creando otra clase.

Así sucesivamente hasta quedar como se muestra a continuación (figura 3.15):

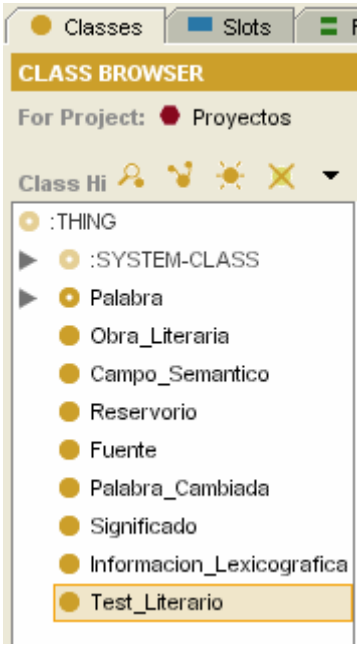

Fig.3.15: Vista donde se representa todas las clases del proyecto.

3.6 Crear slots, tipos de valores y cardinalidad.

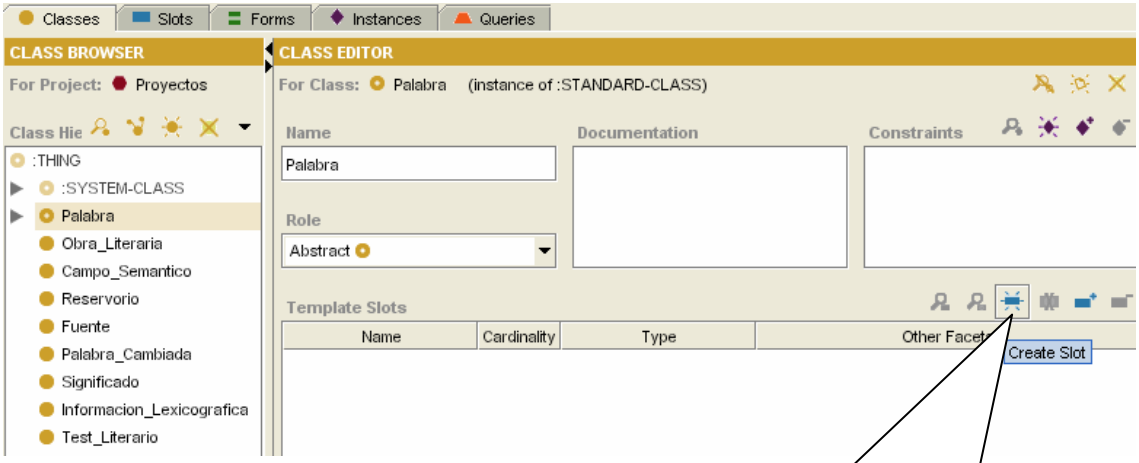

Fig.3.16: Vista de creación de slots.

Para crear un slot a la clase “Palabra”:

1- seleccionar la clase en el “CLASS BROWSER”.

2- Clic en “Create Slot”.

Los slots tienen varias características (figura 3.17), aunque las fundamentales son:

- **Cardinality:** que puede ser *single* o *multiple*, para indicar que el slot admite un único valor o una lista de valores
- **Type:** que se refiere al tipo de valor que almacena un slot. Puede ser *boolean, integer, float, string, instance, class* o *any*.
- **Domain:** el dominio de un slot es la clase (o clases) para la que se ha definido.

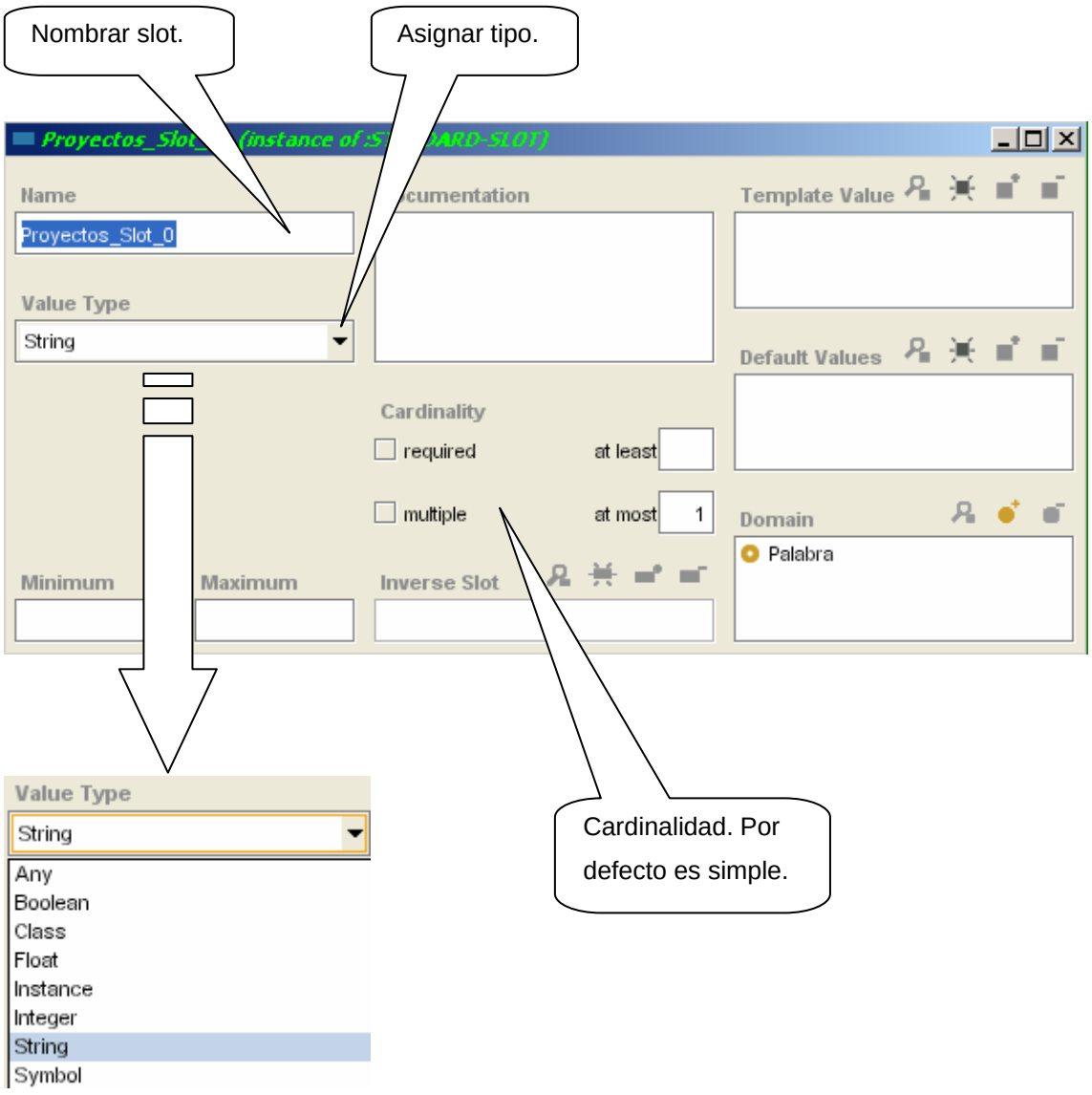

Fig.3.17: Creando slots con sus tipos.

Tipos de valores:

Any, cualquier tipo, o lo que es lo mismo, el tipo vacío.

Boolean, tipo de dato booleano (verdadero o falso).

Class, si los valores permitidos pueden ser directamente clases.

Float, dato real.

Instance, si los valores permitidos para el slot son instancias de alguna clase.

Integer, dato entero.

String, cadena de caracteres.

Symbol, tipo de dato enumerativo.

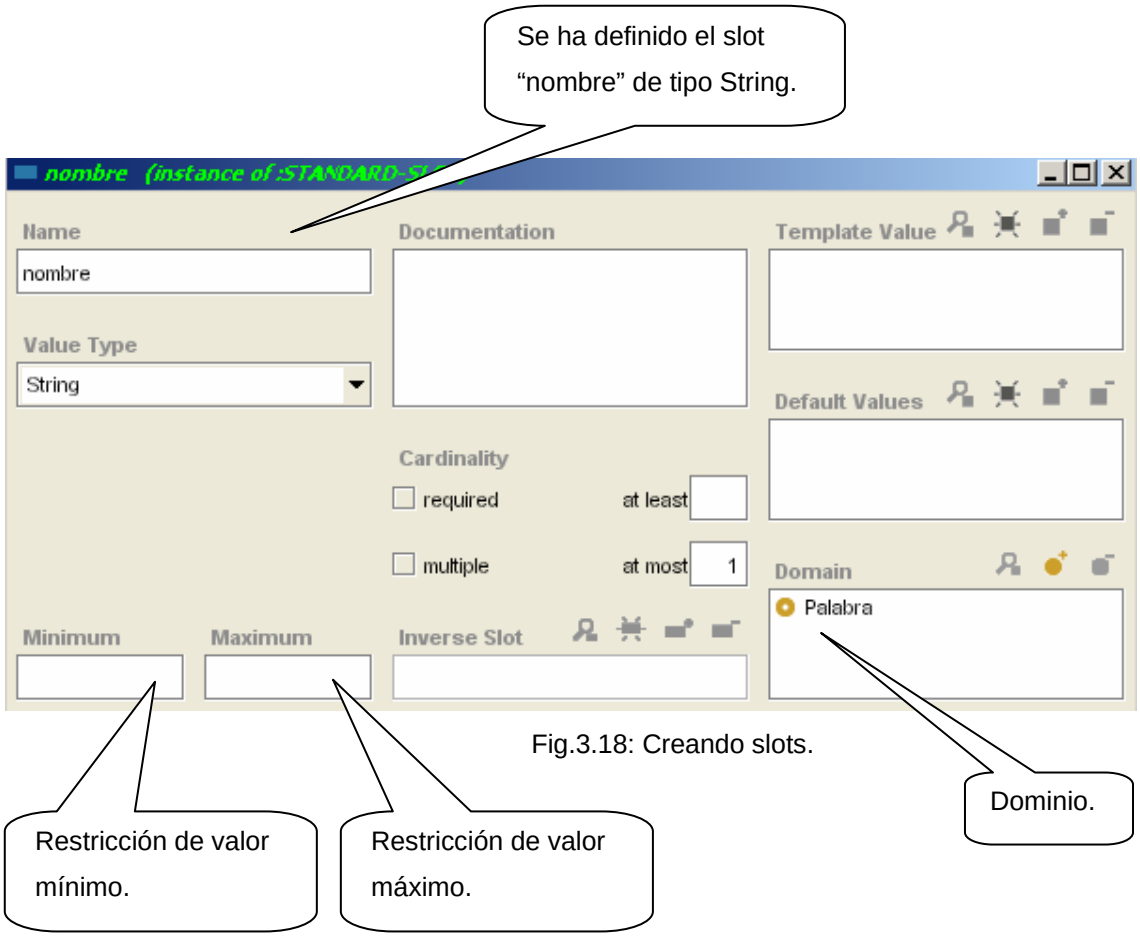

Fig.3.18: Creando slots.

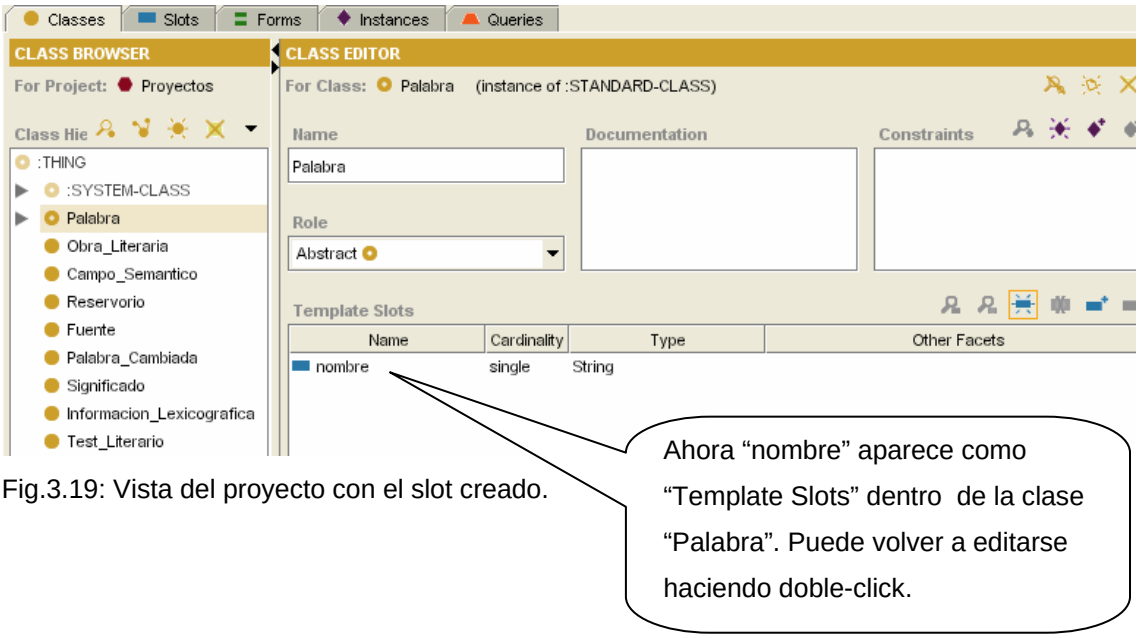

Fig.3.19: Vista del proyecto con el slot creado.

Análogamente se definen otros dos slots en esta clase, así como en las demás clases. Para esta clase (figura 3.20):

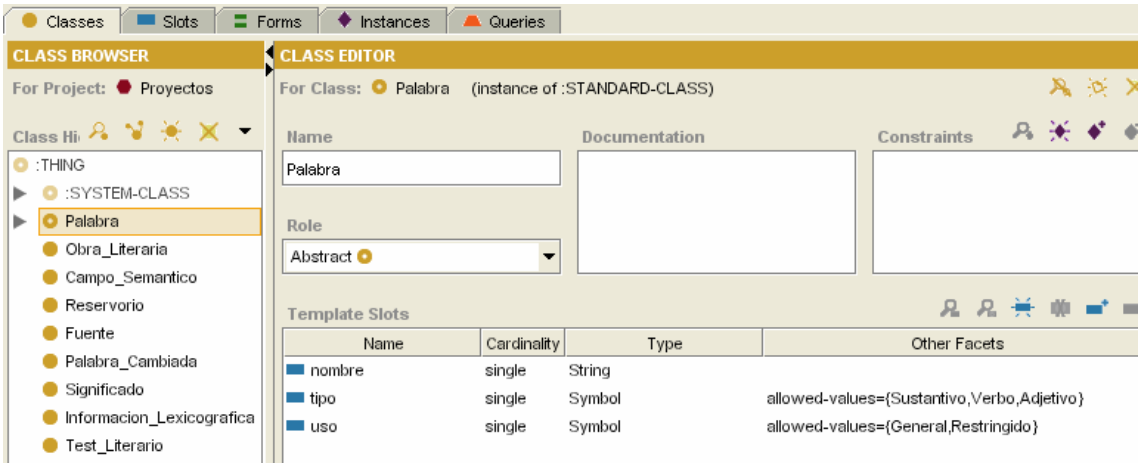

Fig.3.20: Clases con sus slots definidos.

Visualización de los slots:
- nombre
- cardinalidad
- tipo de valor
- otros valores.

3.7 Relaciones.

Para hacer las relaciones necesarias en este diseño, el modo de realización es como sigue (figura 3.21 y 3.22):

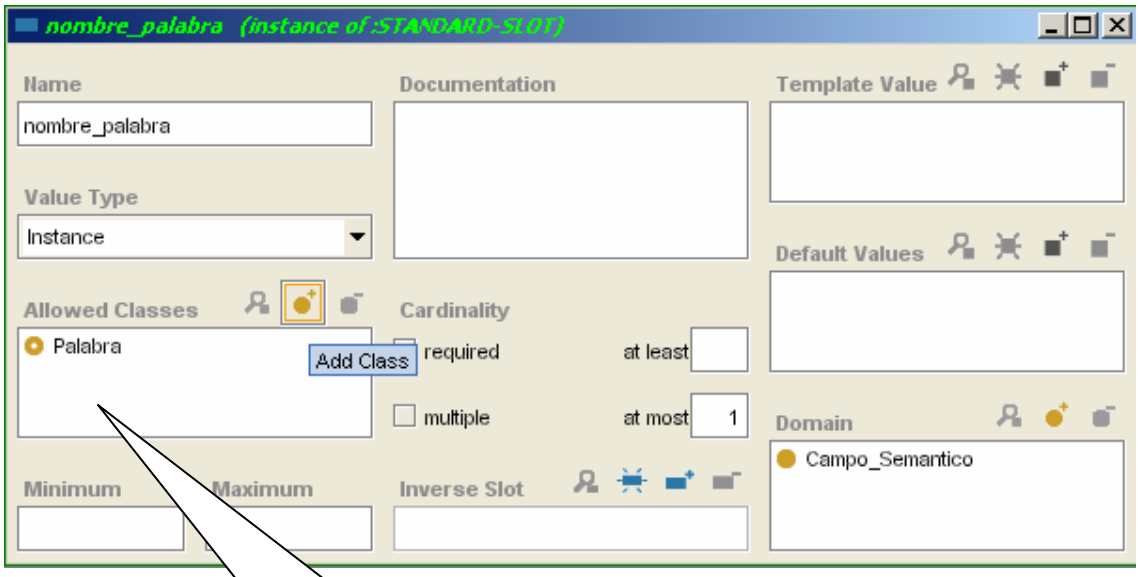

Fig.3.21: Relaciones.

"nombre_palabra" es una instancia a la clase "Palabra" (tipo de valor "Instance").

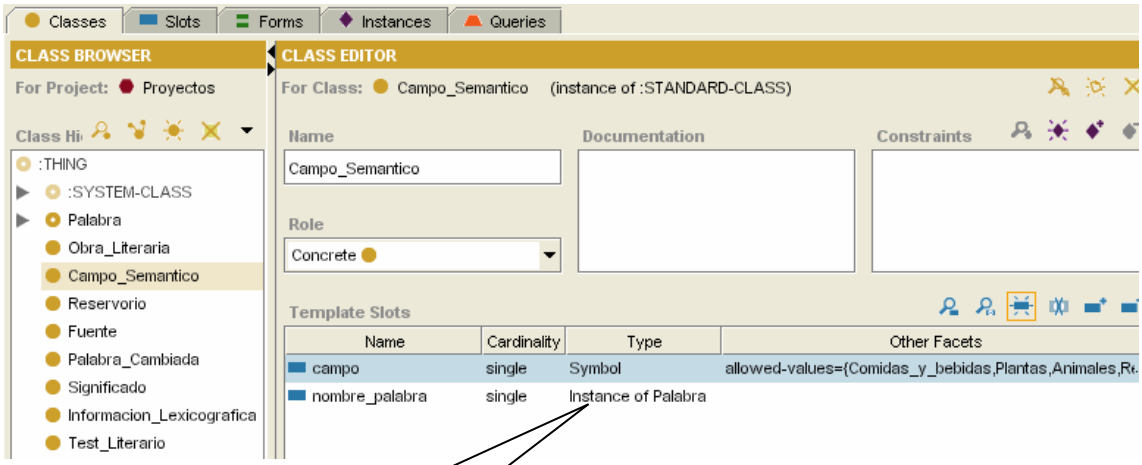

Fig.3.22: Relaciones.

Se observa la instancia mencionada anteriormente.

3.8 Utilización de formularios.

Los formularios son la utilidad que ofrece Protégé para la introducción de información en las diferentes instancias de nuestra estructura de conocimiento. Cada clase creada tendrá su propio formulario, el cual es creado inicialmente de forma automática por Protégé pero que puede ser fácilmente modificado.

Para ver los formularios se debe seleccionar la etiqueta “Forms”, tal y como se muestra en la figura 3.23:

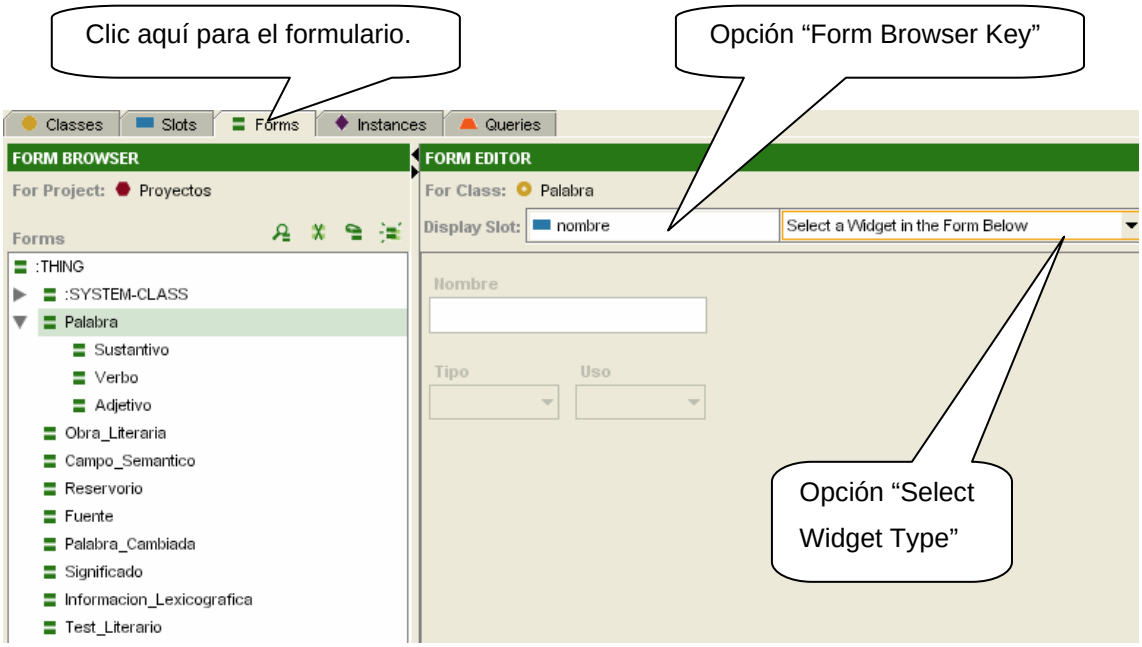

Fig.3.23: Vista de formularios.

En la parte izquierda de la ventana se puede seleccionar la clase de la que se quiere observar su formulario, mientras que en la parte derecha aparece la disposición de los slots que componen la clase. Los objetos de cada slot pueden ser desplazados y modificados sobre dicha ventana.

La opción “Form Browser Key” permite indicar como quieren mostrarse las instancias de una clase concreta, bien por medio de su identificador interno, o bien por medio del valor asignado a algún slot concreto. Si se modifica esta opción, el resultado puede verse al volver a la sección de “Instances” donde ahora las instancias de una clase son nombradas según el valor del campo seleccionado.

La opción “Selected Widget Type” permite cambiar el tipo de objeto que está asociado a un slot para introducir su valor en el formulario. Por defecto, Protégé asigna un determinado objeto, por ejemplo un cuadro simple de texto, pero pueden ser posibles otros más en función del tipo de slot. Para modificarlo no hay más que seleccionar el slot correspondiente y después seleccionar una de las opciones que nos ofrece Protégé en dicha opción de “Selected Widget Type”.

3.9 Crear instancias.

Los objetos del dominio se representan en Protégé como instancias de las clases definidas en la ontología. En realidad, las instancias son la verdadera base de conocimiento, y la estructura de clases, propiedades y relaciones forman el esquema (o la estructura) del conocimiento.

Las instancias se editan en tres secciones (figura 3.25):

- “CLASS BROWSER“, donde se muestra la jerarquía de clases pero ahora indicando entre paréntesis el número de instancias definidas para cada clase concreta.
- “INSTANCE BROWSER”, donde se muestran las instancias de una clase seleccionada en navegador de clases. Para identificar cada instancia se visualiza el valor de uno de sus slots. El slot utilizado para visualizar la instancia no es fijo y es posible decir a Protégé qué slot se utilizará en el navegador de instancias (es denominado *Display slot,* figura 3.24*)*.

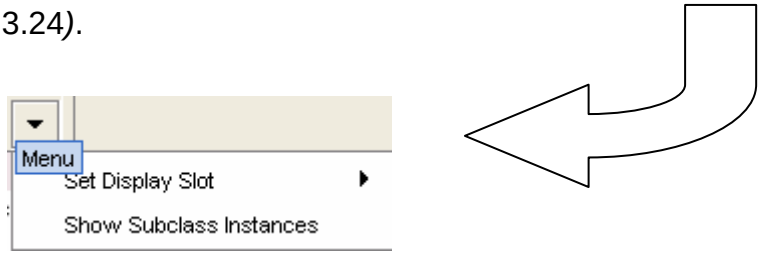

Fig.3.24: Menú.

- “INSTANCE EDITOR”, donde se muestran en un formulario los campos editables de una instancia (cada campo está asociado a un slot).

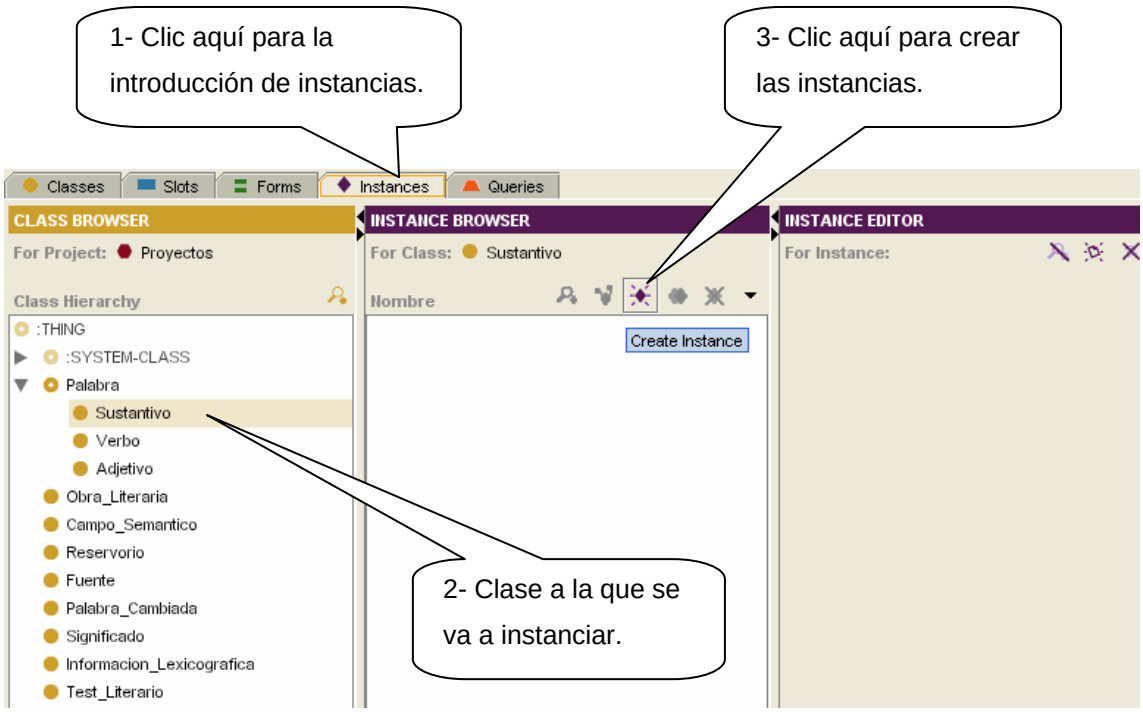

Fig.3.25: Instancias.

Luego:

• Aparece una nueva instancia en el “INSTANCE BROWSER”.

• El nombre de la instancia es generado automáticamente.

• Junto al nombre de la clase aparece un número entre paréntesis indicando el número de instancias (este valor sólo aparece en la pestaña “Instances”).

• En el “INSTANCE EDITOR” aparecen los slots definidos para la clase Editor como campos de texto editables para esa instancia.

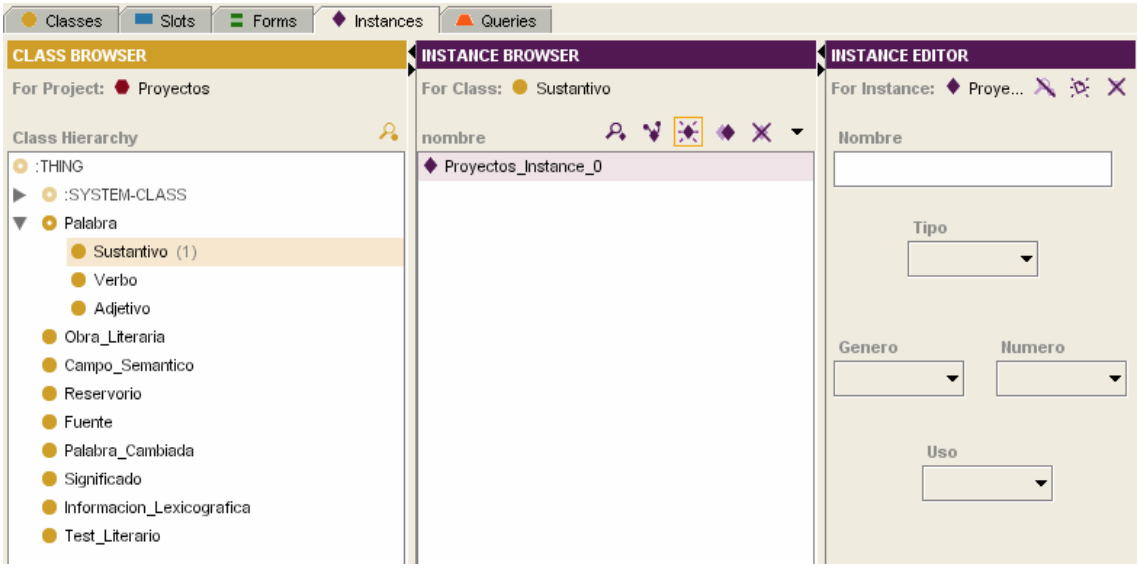

Fig.3.26: Vista de la introducción de instancias.

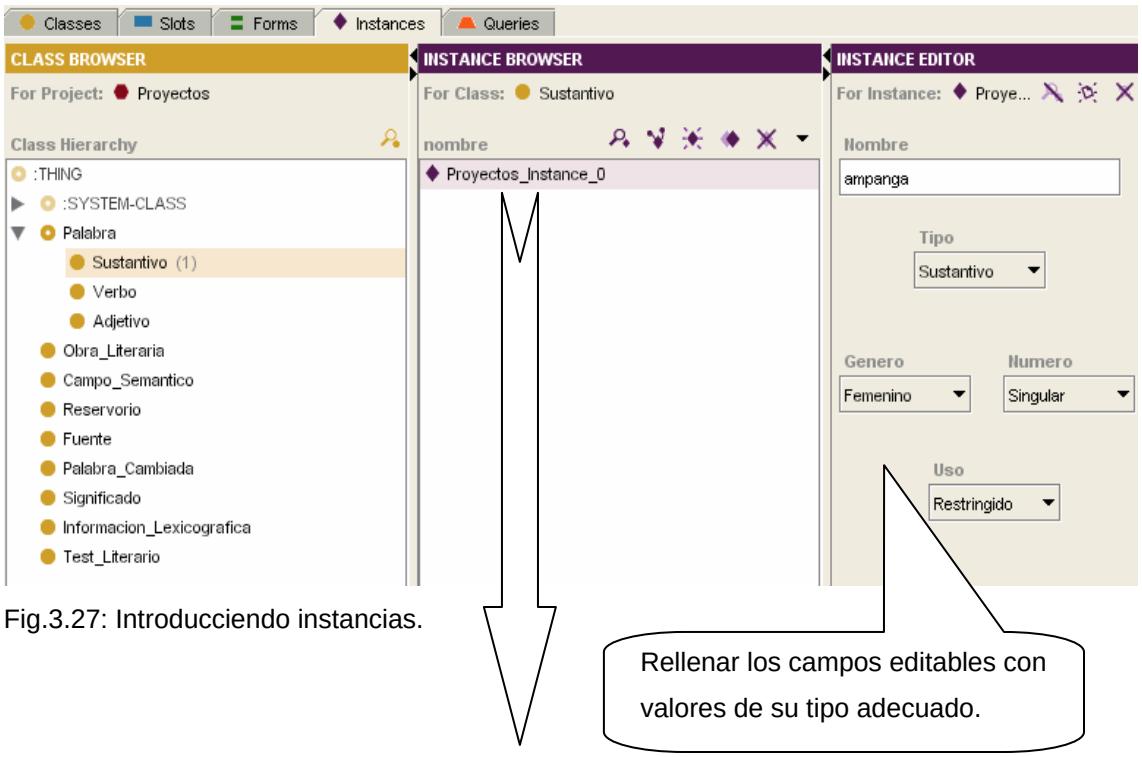

Fig.3.27: Introducciendo instancias.

• Protégé nombra automáticamente las instancias asignando un valor a este slot (figura 3.27).

• Por defecto, este es el nombre que aparece en el “INSTANCE BROWER” (figura 3.27).

• El slot utilizado para visualizar una instancia (denominado DISPLAY SLOT) se puede configurar, para ello clic en el icono para configurar la visualización de las instancias (figura 3.28).

• Si se selecciona el slot “nombre” para las instancias de “Sustantivo”, el nombre de las instancias cambian tal y como a continuación (figura 3.28).

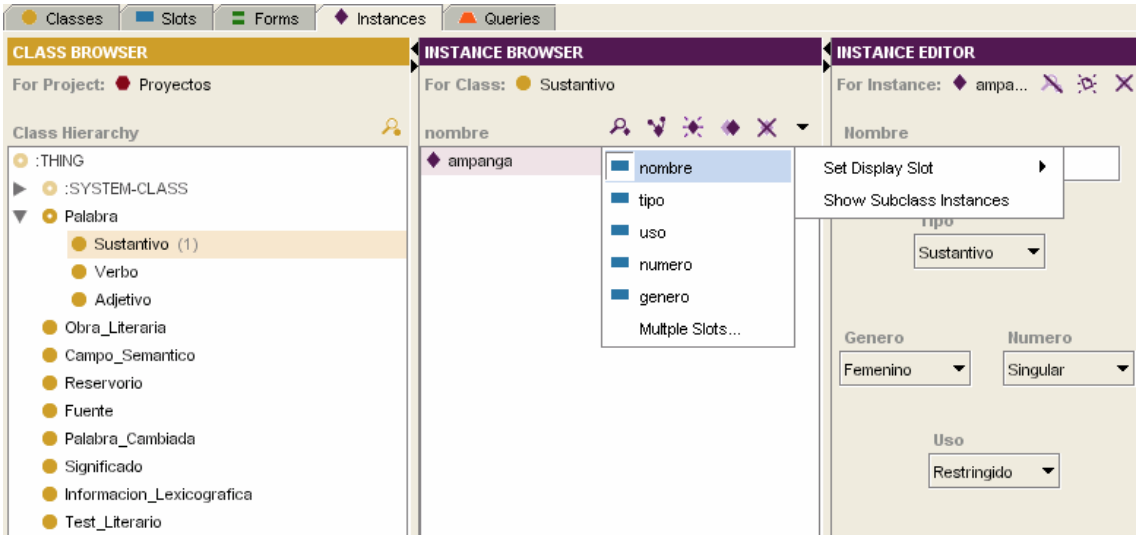

Fig.3.28: Introducciendo instancias.

De forma análoga se continúa el diseño quedando como sigue:

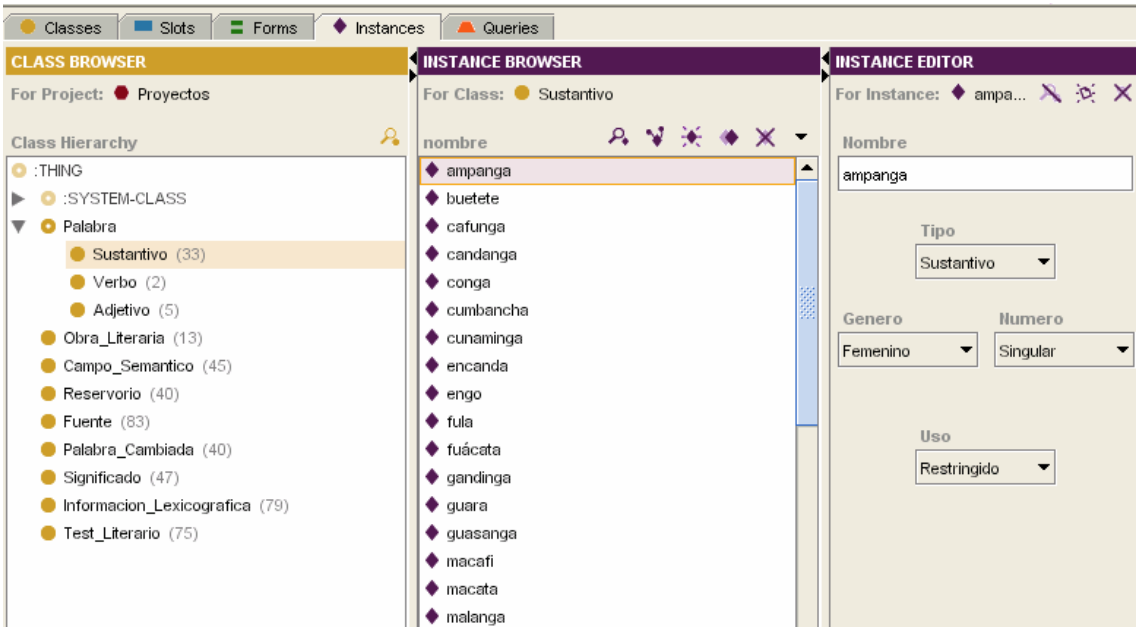

Fig.3.29: Vista del proyecto con sus instancias.

3.10 Crear consultas.

A la hora de responder las necesidades o resultados del trabajo, se necesita hacer consultas (figura 3.30):

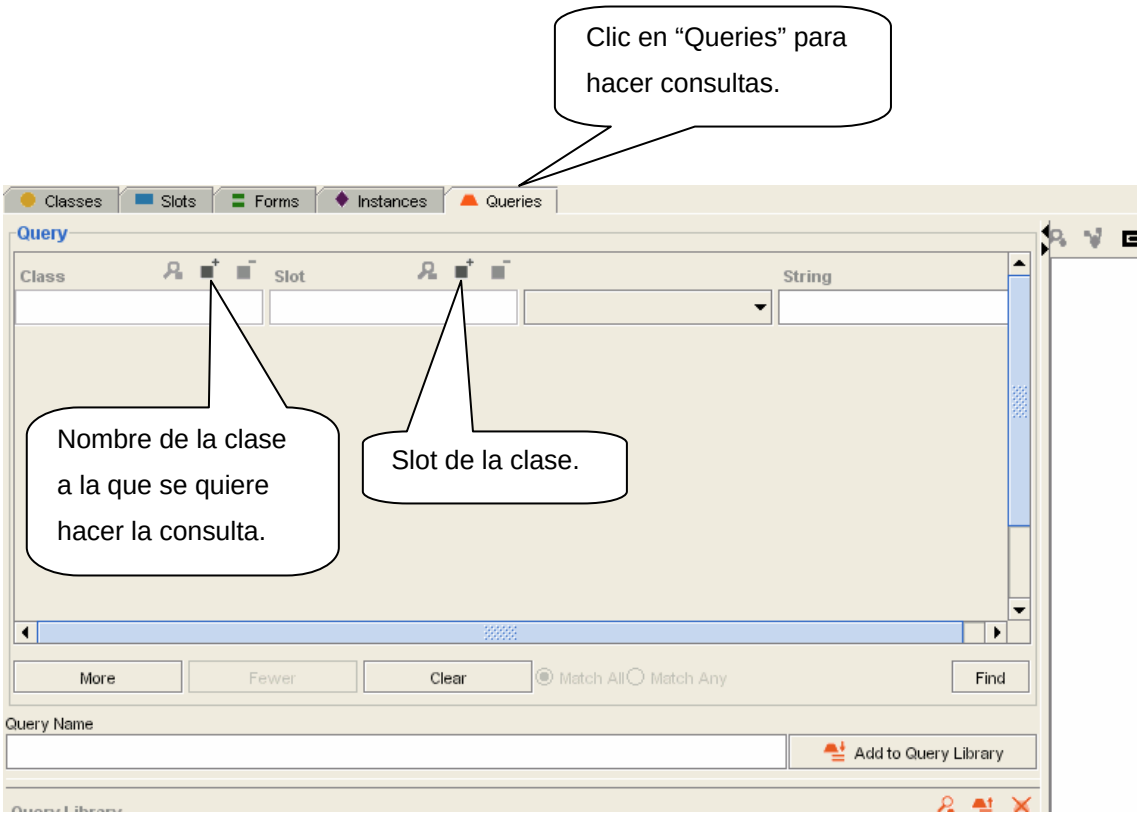

Fig.3.30: Vista para crear consultas.

Se muestra un ejemplo con la consultad que muestra todas las palabras que son sustantivos (figura 3.31):

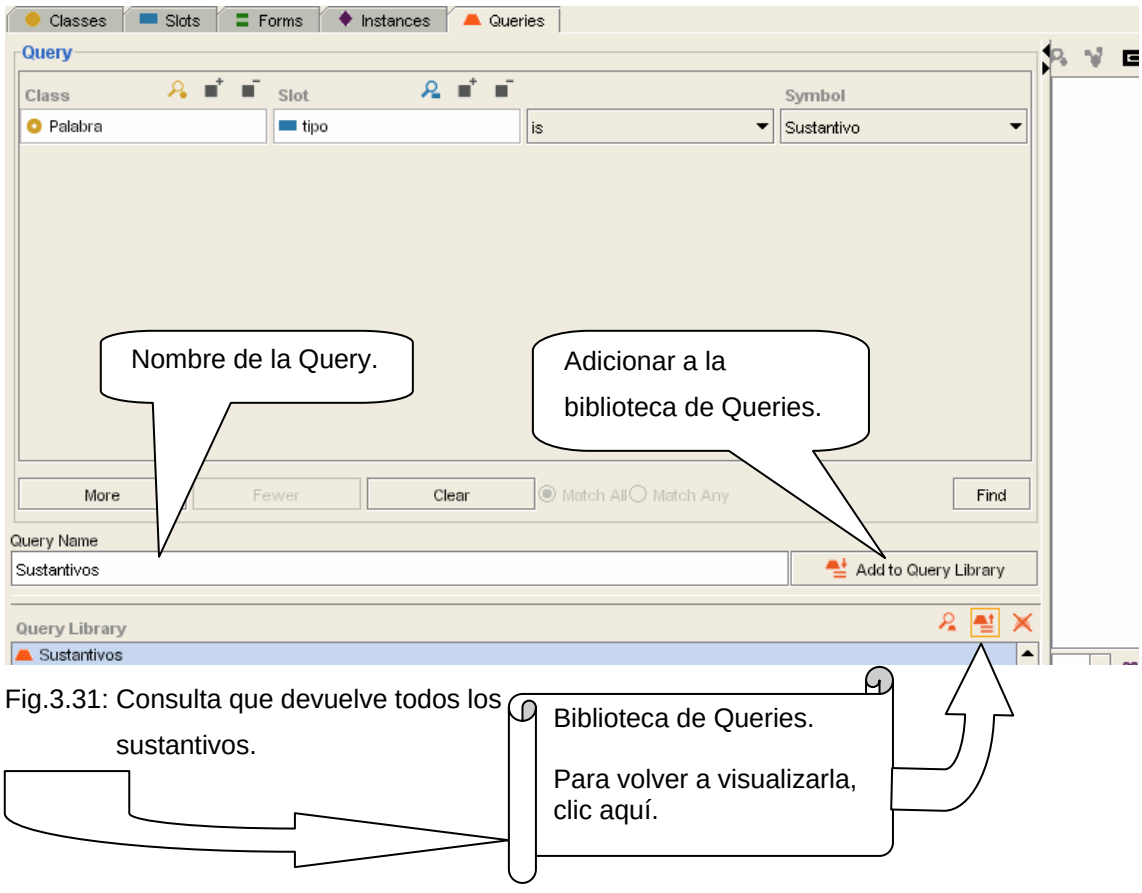

Fig.3.31: Consulta que devuelve todos los sustantivos.

El botón “Find” realiza entonces la búsqueda de la Query, como muestra la figura 3.32:

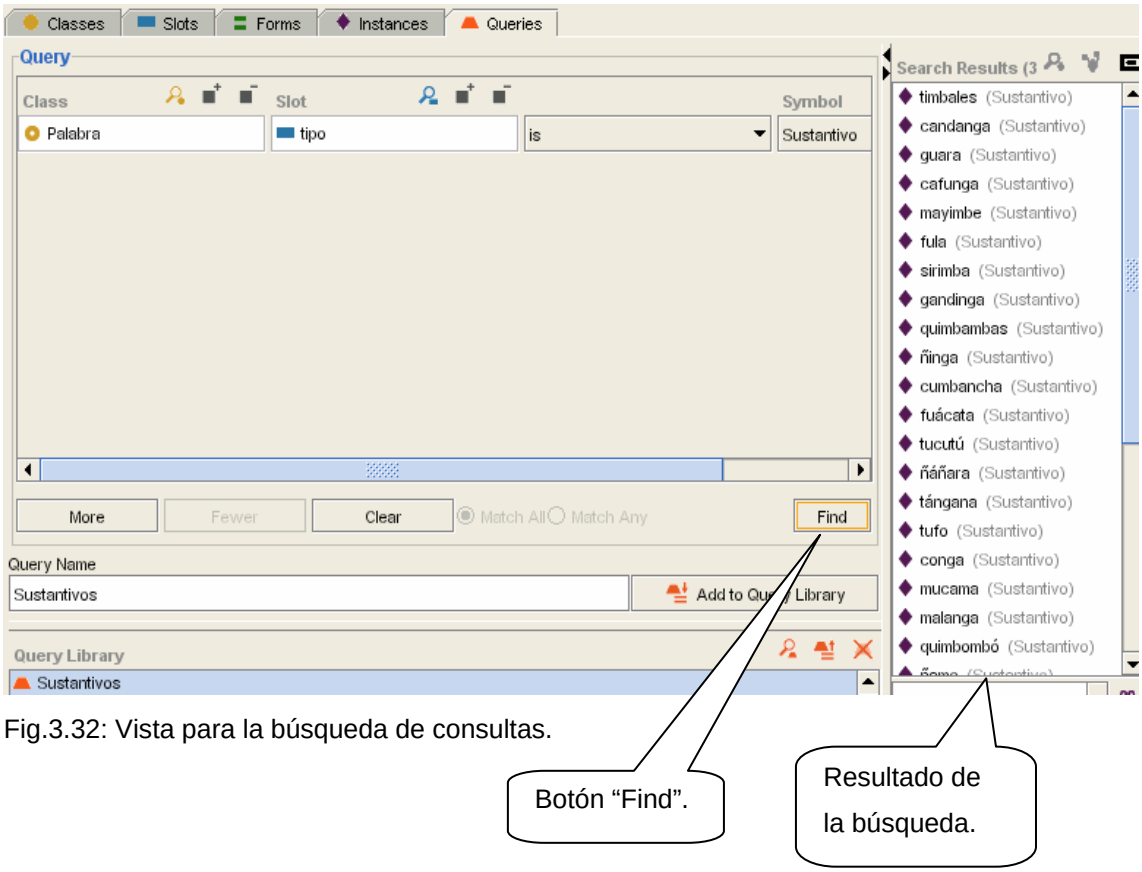

Fig.3.32: Vista para la búsqueda de consultas.

Protégé tiene la posibilidad de exportar los resultados obtenidos para un fichero texto en la misma ubicación donde está guardado el proyecto (figura 3.33), que por defecto se nombra ***protege_query_results.txt.***

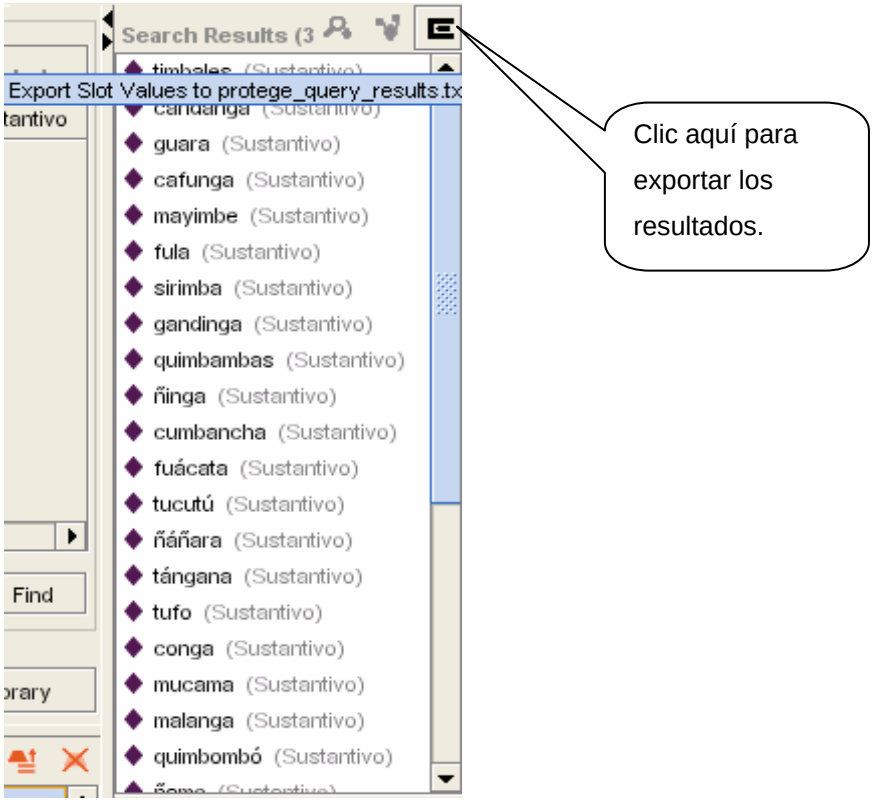

Fig.3.33: Vista para exportar los resultados.

Conclusiones

Se realizó un estudio de las ontologías resumiendo las definiciones y clasificaciones actualmente aceptadas a nivel mundial, así como cuestiones de interés sobre las mismas.

Se realizó una búsqueda bibliográfica exhaustiva que permitió reconocer las herramientas más utilizadas actualmente y se determinó la herramienta que más posibilidades está ofreciendo en este momento para el desarrollo de aplicaciones con ontologías.

Se estudió la herramienta Protégé y las facilidades que la misma ofrece para el desarrollo de ontologías aplicando la misma a la solución de un problema de utilidad en el área de Letras.

Se muestra de manera detallada como se aplica la herramienta a la solución del problema.

Recomendaciones

Completar la ontología con la incorporación de todas las palabras de origen africano que se utilizan en Cuba.

Desarrollar una interfaz gráfica que permita la incorporación de estas palabras a personas que no son del área de Computación.

Mejorar el diseño de la aplicación realizada que permite recuperar las palabras atendiendo a los diferentes criterios planteados por los especialistas de Letras.

Ampliar el número de consultas que realiza el sistema.

Referencias bibliográficas

[1] Gruber, T.R. (1993). A Translation Approach to Portable Ontology Specification. *Knowledge Acquisition* 5, pp.199-220.

[2] Mizoguchi R. (1997). Roles of shared ontology in AI-ED research, pp.537-544.

[3] Uschold, M. (1996). Ontologies: Principles, Methods and Applications. *Knowledge Engineering Review* 11.

[4] Guarino N. (1998). Formal Ontologies and Information Systems, pp. 3-15.

[5] Horrocks (1999). Sexta Conferencia Internacional de Programación Lógica y Razonamiento de Autómatas.

[6] Sowa (2000). Sowa Knowledge Representation : Logical, Philosophical, and Computational Foundation. Brooke Cole Publishing. Ca 2000.

Bibliografía

1. Booch, G.; Rumbaugh, J.; Jacobson, I. "*The Unified Modeling Language userguide*: Addison-Wesley.". Addison-Wesley. 1997.

2. Chandra. Ontology of tasks and methods. 1998.

3. [DAML] ***http://www.daml.org/***

4. Feigenbaum, E. A. "The art of artificial intelligence: Themes and case studies of knowledge engineering", pp. 1014-1029. 1977.

5. Gómez-Pérez, A. "Criteria to Verify Knowledge Sharing Technology", Technical Report KSL 95-10, Knowledge Systems Laboratory, Universidad de Stanford. 1995.

6. [KIF] ***http://logic.stanford.edu/kif/kif.html***

7. Jasper, R. and M. Uschold. A framework for Understanding and Classifying Ontology Applications, Canadá. Octubre, 1999.

8. McGuinness, D.L.; Abrahams, M.K.; Resnick, L.A.; Patel-Schneider, P.F.;Thomason, R.H.; Cavalli-Sforza, V. and Conati, C. "Classic Knowledge Representation System Tutorial". 1994.

9. Ontolingua (1997). Ontolingua System Reference Manual.

10. [PROTÉGÉ] ***http://protege.stanford.edu/***

11. [ONTOLINGUA] ***http://www.ksl.stanford.edu/software/ontolingua/***

12. [ONTOWEB] *http://ontoweb.aifb.uni-karlsruhe.de/About/Deliverables/D1.4-v1.0.pdf*