INGENIERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN

PROYECTO FIN DE CARRERA

APO:

APLICACIÓN PARA LA

POBLACION DE ONTOLOGÍAS

Autor: Sebastián Lendinez Teixeira

Tutora: Elena Castro Galán

Director: Francisco Javier Calle Gómez

AGRADECIMIENTOS

“Dedico este proyecto a toda mi familia y amigos, y en especial tengo que dedicárselo a mi esposa e hijo, por su comprensión en mis momentos de estrés y por el tiempo sacrificado en este largo viaje”

“Aprovecho estas líneas también para agradecer a mis tutores la confianza depositada en mí, y dedicarles también este trabajo, ya que sin su ayuda y tiempo dedicados no habría llegado a buen puerto.”

ÍNDICE

Índice de Ilustraciones

Índice de Tablas

Introducción

1. Introducción

"…la ontología es el estudio de los conceptos a priori que residen

en el entendimiento y tienen su uso en la experiencia,

llevando la noción hacia un sentido más inmanentista."

KANT

La ontología es una antigua disciplina que en sentido filosófico, se define como un esquema específico de categorías que refleja una visión específica del mundo. Apartándonos del aspecto filosófico, podemos definir ontología como una teoría que explica cómo un individuo, grupo, lenguaje o ciencia entiende un determinado dominio. Pero desde el punto de vista informático las ontologías son como teorías que especifican un vocabulario relativo a un cierto dominio. Este vocabulario define entidades, clases, propiedades, predicados y funciones y, las relaciones entre estos componentes. Las ontologías toman un papel clave en la resolución de interoperabilidad semántica entre sistemas de información y su uso.

La definición más aceptada de ontologías es la dada por Gruber (1993) y extendida por Studer (98), que la describe como *"una especificación explícita y formal sobre una conceptualización compartida"*. La interpretación de esta definición es que las ontologías definen conceptos y relaciones de algún dominio, de forma compartida y consensuada; y que esta conceptualización debe ser representada de una manera formal, legible y utilizable por los ordenadores.

Las ontologías fueron desarrolladas en el ámbito de la inteligencia artificial para facilitar el intercambio de conocimiento, el conocimiento de las ontologías se establece para que se use de manera "consensuada" y compartida por distintos sistemas, que deberán comprometerse con el vocabulario que se maneja en ontologías, también han de implementarse las ontologías en lenguaje entendible o computable por máquina, esto es el aspecto formal.

La disponibilidad de información semántica en la base de conocimiento permite formular al usuario consultas más expresivas y precisas, e implementar un sistema capaz de utilizar elementos conceptuales para determinar correspondencias entre consultas y contenidos. La presencia de una ontología del dominio que estructura y relaciona la información de acuerdo a su significado permite

construir un buscador donde los usuarios especifican sus criterios de búsquedas en función de los conceptos y atributos modelizados.

Tras la aparición de los distintos lenguajes y estándares se hizo necesario el idear cauces que permitieran un acceso uniforme y flexible, para un usuario final, de cara a la utilización y el mantenimiento del conocimiento compartido en forma de ontología Es en ese preciso momento cuando surge la idea de crear APO, un editor de ontologías, que permitirá la manipulación y población de ontologías. Se busca crear una herramienta que no requiera de un conocimiento altamente técnico, y que pueda se manipulada por cualquier especialista o conocedor de un dominio específico.

Cada día más especialistas trabajan en entornos especializados, entornos en ocasiones muy técnicos que dificultan la traducción o búsqueda de conceptos en diversas lenguas. Estas necesidades demandan la necesidad de herramientas que permitan guardar y buscar esta información, así como poder compartirla con otros usuarios, conocedores del mismo entorno, e incluso de entornos afines o relacionados. Compartir y reutilizar conocimientos son una baza más a favor de la creación de herramientas de este estilo.

Estado del arte

- **Ontología**
- **Herramientas para la edición de Ontologías**
- **Planes de Futuro**
- **Conclusiones finales**

2. Estado del Arte

El Estado del Arte es una de las primeras etapas que debe desarrollarse dentro de una investigación, puesto que su elaboración, que consiste en "ir tras las huellas" del tema que se pretende investigar, permite determinar cómo ha sido tratado el objeto a estudiar, cómo se encuentra en el momento de realizar la propuesta de investigación y cuáles son las tendencias existentes.

Es una etapa larga que podemos dividir en dos subfases:

- **Fase heurística**: se procede a la búsqueda y recopilación de las fuentes de información, que pueden ser de muchas características y diferente naturaleza.
- **Fase Hermenéutica**: Durante esta fase cada una de las fuentes investigadas se leerá, se analizará, se interpretará y se clasificará de acuerdo con su importancia dentro del trabajo de investigación. A partir de allí, se seleccionarán los puntos fundamentales.

2.1. Ontología

Una ontología es una descripción formal y detallada de términos, abstracciones, instancias, propiedades, restricciones, clases, jerarquías, relaciones, características, etc., de un dominio en particular, que puede ser compartido y/o reutilizado; por lo que debe ser legible y libre de ambigüedades **[40]**. En términos prácticos desarrollar una ontología incluye: definir las clases, organizar las clases en una jerarquía taxonómica (superclase/sub-clase), definir slots y describir valores permitidos para esos slots. Para más información ver **[Anexo C].**

Para guiar y evaluar diseños, se necesitan diseños objetivos que estén fundados en el propósito de un resultado, en vez de que estén basados en nociones a priori naturales o verdaderas. Existe un conjunto preliminar de criterios de diseño para ontologías cuyo propósito es compartir conocimiento y la interoperabilidad entre programas basados en conceptualización compartida, entre los que se citan: claridad, coherencia, extensibilidad, sesgo de codificación mínimo y un mínimo compromiso ontológico para soportar las actividades de conocimiento compartido previsto.

Los artículos sobre ontologías llevan casi veinte años de gran actualidad en la literatura profesional, como se puede comprobar haciendo búsquedas en bases de datos e Internet. A pesar de esto, sigue sin haber consenso sobre el significado de este concepto. Probablemente la representación gráfica conocida como el espectro de las ontologías ha generado cierta confusión en algunos lectores. El origen de esta confusión puede deberse a la mezcla de varios tipos de sistemas de organización del conocimiento, cuya distinta finalidad llevó a definirlos de diferente forma. De hecho, construir una

ontología formal para un sistema no siempre supone una mejora y en muchas ocasiones es suficiente con una representación menos compleja.

Una ontología "define los términos usados para describir y representar un área de conocimiento" **[1]** (como medicina, arte, etc.). Las ontologías se utilizan por la gente y por sistemas computacionales para compartir información de un dominio (entendiendo como dominio una porción determinada de un área de conocimiento).

Existe un proceso de abstracción, y podemos representar el mundo completo en el que las distintas ontologías pueden existir (ontología genérica). Este mundo tendrá muchas ontologías actuando dentro de él y con la posibilidad de interactuar entre ellas. A su vez podemos considerar el mundo dividido en porciones más pequeñas llamadas Ambientes.

Estos Ambientes son porciones individuales del mundo. Cada Ambiente tiene sus propias características que lo diferencian de los otros Ambientes. En estos Ambientes en donde se sitúan las ontologías específicas (conocidas por profesionales que se desenvuelven en dicho Ambiente). El cuerpo de conocimiento (KnowledgeBody) que es quien formará parte de la ontología, y el que se irá enriqueciendo, a medida que actúa en el ambiente, y con otras ontologías.

Las ontologías se componen de:

1. **Objetos e instancias**. Podrían ser una representación de un objeto en la realidad (por ejemplo la silla de la habitación). También se conocen como instancias, o individuals.
2. **Clases**, que representan a un conjunto de objetos. Aunque la palabra "concepto" se utiliza a veces como sinónimo **[2]**, otros autores **[3]** relacionan estas dos definiciones diciendo que las "clases" son concretas representaciones de los "conceptos" (ideas que tenemos en la mente).
3. Las **propiedades** que estos objetos pueden tener, siempre dentro de nuestro dominio de interés. También se conocen como roles, slots o atributos.
4. **Relaciones** que pueden haber entre dichos objetos, que la mayoría de lenguajes ontológicos agrupan dentro de las propiedades de dichos objetos.

Una ontología se expresará en un lenguaje basado en sistemas lógicos (lenguaje ontológico) que servirá para almacenar nuestro conocimiento sobre el dominio de interés. Algunos autores utilizan el término teoría para referirse a estas ontologías, ya que a veces son relativamente pequeñas y se pueden completar con otras teorías **[4]**. Las ontologías tienen características diferenciadoras respecto a una base de datos que almacene la información de nuestro sistema, donde las tablas sean clases, las tuplas de las tablas instancias y las relaciones entre instancias relaciones simples o múltiples entre los identificadores de las tablas. Los elementos diferenciadores entre una ontología y la información almacenada en una base de datos son:

- Permiten que los objetos tengan propiedades (serían los equivalentes a campos de una tabla de base de datos) que no estén en la definición de la clase a que pertenecen. Cuando definimos una tabla que contiene la información de un objeto, hemos de indicar a priori el número de campos y el tipo de datos a almacenar en cada campo.
- Permiten múltiples definiciones de clases y propiedades. En una base de datos normalmente existe una tabla fija para el almacenamiento de información de un objeto.
- La información sobre un objeto no tiene por qué estar solamente en un documento (aunque es cierto que existen bases de datos distribuidas, en estos lenguajes se permite mayor libertad). Además, puede ir apareciendo información en documentos distintos.
- No se supone nunca que se tiene un conocimiento total de nuestro dominio de interés, por eso existen frases como "las pizzas tienen al menos un ingrediente", junto con información de algunas pizzas en las que no sepamos ningún ingrediente.
- Al estar basados en lenguajes lógicos, podrán procesarse inferencias (deducciones), que podrán realizar sistemas no humanos. No se pueden realizar muchas inferencias automáticas basándonos en una base de datos.

2.1.1. Tipología de lenguajes ontológicos

Existe una gran cantidad de lenguajes ontológicos, lo que nos obliga a realizar una taxonomía de dichos lenguajes. Una primera clasificación sería **[5]**:

1. **Lenguajes para representaciones gráficas** (denominados *Graphical notations*). Estos lenguajes no se basan en lenguajes lógicos, sólo en la sintaxis de las ontologías (clases, instancias, propiedades que las relacionan entre sí). No nos sirven de mucha utilidad por su deficiencia en lógica para poder realizar inferencias, sí son la base para la gran mayoría de lenguajes ontológicos y para representar ontologías gráficamente. Dentro de este grupo encontramos:
    - **Las redes semánticas**, entendidas como un conjunto de nodos relacionados entre sí por flechas. Hay sistemas ontológicos fuertemente relacionado con ellos, como WorldNet **[6]**, o sistemas de enseñanza basados en conceptos, como AHA! (Adaptative Hypermedia Architecture) o AIMS (Agentbased Information Management System) **[7].**
    - **Los mapas de conceptos** (denominados en inglés *topics maps* **[8]**), entendidos como una ampliación de las redes semánticas incluyendo el concepto de ocurrencias, que son referencias a fuentes externas al mapa de conceptos (por ejemplo una referencia bibliográfica). Sistemas como Citeseer **[9]** utilizan estas uniones externas (almacena

uniones con los documentos en otros lugares fuera de Citeseer). Ejemplos de lenguajes basados en este sistema son el XML Topic Map (XTM **[10]**) y el ISO/IEC 13250 **[11].**

- **UML** (Unified Modeling Language **[12]**), que como lenguaje que permite representar objetos y sus relaciones (aparte de otras muchas posibilidades) se puede entender como uno de estos lenguajes.

- **RDF** (Resource Description Framework **[13]**, que desarrollaremos más en profundidad en apartados siguientes.

2. **Lenguajes basados en lógica**. Los lenguajes lógicos se pueden definir como lenguajes formales para representar la información, de forma que se puedan inferir conclusiones. Conllevan una sintaxis, que define el tipo de sentencias que pueden darse en el lenguaje, y una semántica, que define el significado de dichas sentencias. Los lenguajes utilizados para describir ontologías, y que se basan en lenguajes lógicos son los lenguajes basados en lógica. Estos lenguajes se considerarán como leguajes ontológicos, ya que incluyen en su sintaxis definiciones lógicas que permiten utilizar razonadores lógicos para inferir resultados. Según la lógica utilizada tenemos:

   - **Basados en Lógica Descriptiva** (DL *Description Logics*). En este caso tenemos OIL, DAML+OIL, OWL. Estos lenguajes son a los que mayor énfasis dedicaremos en este proyecto, por lo que los describiremos con más detalle en otro apartado.

   - **Basados en lógica de primer orden** (FOL *First Order Logic*), como es el caso de KIF (Knowledge Interchange Format) y sus sub-lenguajes. Se puede decir que la lógica de primer orden comprende a su vez la lógica descriptiva. También trataremos sobre KIF en los siguientes apartados.

   - **Basados en Reglas** (denominadas en inglés como *Rules*), como RuleML **[14]**, LP/Prolog **[15]**. Una regla en este caso se entiende como "unidades de conocimiento autocontenida que implican algún tipo de razonamiento" **[16]**. En cierto modo se puede decir que la lógica clásica es un subconjunto de las posibles reglas. Otros ejemplos de reglas son reglas de reacción (cuando pase X, hacer Y), y otros que veremos más adelante cuando hablemos de RuleML y SWRL.

   - **Gráficos conceptuales** (CG *conceptual graphs*), sistemas basados en los gráficos de Charles Sanders Peirce y utilizados para sistemas de inteligencia artificial.

   - **Basados en lógicas de mayor orden**, como LBase. El mayor orden en este caso se refiere sintácticamente, es decir poder relacionar conjuntos de elementos de discusión (y no sólo parejas) directamente, sin hacerlo a pares.

- **Lógicas no clásicas** (F-logic, Non-Monotonic). Estas lógicas no se basan en el concepto de verdadero o falso, sino en grado de verdad (probabilidad) de una aserción. Estos lenguajes no serán tratados en este estudio.

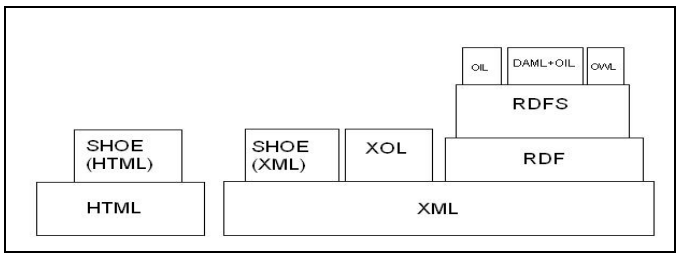
- **Lógicas probabilísticas o difusas** (fuzzy logics) Estos lenguajes suelen ser utilizados por sistemas de redes neuronales, que no se estudiarán en este proyecto.

Como en muchos otros sistemas, cuando más restrictiva es la definición (en este caso la lógica utilizada) más sencillo es su manejo (razonadores más rápidos, descripciones más sencillas, pero más difícil es que nuestros problemas reales se adapten a dichos sistemas). Los lenguajes basados en lógica que se acaban de enumerar se han ordenado de más restrictivo a menos, siendo el más restrictivo los basados en Lógica Descriptiva.

2.1.2. Lenguajes de especificación de ontologías

Se trata de un conjunto de lenguajes utilizados para la representación de ontologías, parsers, lenguajes de consulta, entornos de desarrollo, módulos de gestión (almacenamiento, acceso y actualización) de ontologías, módulos de visualización, conversión de ontologías, y otras herramientas y librerías. Forman parte de las tecnologías creadas para desarrollar también la Web Semántica.

Podemos ver una relación que se establece entre diferentes lenguajes en la siguiente figura **[017]** (figura 2.1).



2.1 Relación entre Lenguajes

- **XML**: Aunque no está diseñado explícitamente para definir ontologías, representa una tecnología apropiada para ser aplicada a la Web semántica, es el estándar más extendido hoy en día para estructurar datos y documentos, y nos servirá de introducción para el resto de lenguajes.

  Extensible Markup Language (XML), es un formato de texto muy simple y flexible derivado de SGML, posee una sintaxis bien definida capaz de ser entendible por los humanos y muy fácil de analizar gramaticalmente.

La desventaja de este lenguaje es que no aporta ningún mecanismo para tratar la información a nivel semántico, que es lo que más nos interesa (entre otras cosas) a la hora de establecer una ontología.

Otra desventaja importante es que el XML no posee características destinadas a la especificación de ontologías ya que no es un lenguaje especialmente diseñado para el desarrollo de ontologías.

- **RDF y RDF Schema:** Resource Description Framework (RDF) [Infraestructura para la Descripción de Recursos]. **[18]**. Provee una base para procesar metadatos. El fundamento o base de RDF es un modelo para representar propiedades designadas y valores de propiedades.

  La sintaxis RDF utiliza el XML. El principal objetivo de RDF es definir un mecanismo para describir recursos que sean neutrales al dominio de aplicación.

  RDF dispone de un sistema de clasificación muy parecido a los sistemas de modelado y programación orientada a objetos.

  Denominamos Schema al conjunto de categorías. Las categorías se organizan en una jerarquía y proporcionan extensibilidad a través de un refinamiento de subcategorías.

  Enriquece el modelo básico proporcionando un vocabulario básico para RDF que se supone que tiene cierto significado semántico.

  RDF, al tratarse de una infraestructura para la descripción de recursos, y que sirve como base para procesar metadatos, está influido por varias comunidades de representación de datos (HTML, SGML, XML). Su modelo básico de datos se basa en tres elementos, *Sujeto*, *predicado* y *objeto*.

  En general una sentencia siempre podemos leerla de ésta forma:

  <Sujeto> TIENE <Predicado> <Objeto>

- D**AML+OIL:** Es un lenguaje de ontologías, aunque su mayor utilización es el entorno Web. Está basado en estándares como RDF y XML, pero añaden el rigor que ofrece una descripción lógica. Las ontologías pueden escribirse tanto en XML utilizando Namespaces **[46]** como en RDF.

  Como hemos dicho, DAML+OIL está basado en RDF, pero se trata de un lenguaje mucho más expresivo y completo, de hecho desde un punto de vista formal DAML+OIL puede verse como una descripción lógica muy expresiva.

  En parte la expresividad de éste lenguaje viene determinada por los constructores de clases que son soportados por este lenguaje. En la siguiente figura (figura 2.2) podemos ver una tabla resumen con todos los constructores soportados **[19]**.

Constructor	DL Syntax	Example
`intersectionOf`	$C_1 \sqcap \ldots \sqcap C_n$	Human $\sqcap$ Male
`unionOf`	$C_1 \sqcup \ldots \sqcup C_n$	Doctor $\sqcup$ Lawyer
`complementOf`	$\neg C$	$\neg$Male
`oneOf`	$\{x_1 \ldots x_n\}$	{john, mary}
`toClass`	$\forall P.C$	$\forall$hasChild.Doctor
`hasClass`	$\exists P.C$	$\exists$hasChild.Lawyer
`hasValue`	$\exists P.\{x\}$	$\exists$citizenOf.{USA}
`minCardinalityQ`	$\geq nP.C$	$\geq$2hasChild.Lawyer
`maxCardinalityQ`	$\leq nP.C$	$\leq$1hasChild.Male
`cardinalityQ`	$=n\,P.C$	$=$1 hasParent.Female

2.2 Constructores soportados DAML+OIL

- **OWL:** (Ontology Web Language) Es un lenguaje de marcado para la publicación de ontologías en la World Wide Web, sus objetivos son facilitar un modelo de marcado construido sobre RDF, de hecho muchas de las primitivas RDF son reutilizadas por OWL, y está codificado en XML, que permite representar ontologías a partir de un vocabulario más amplio y una sintaxis más fuerte que la que permite RDF. OWL apoya el intercambio y la reutilización de ontologías por lo que se permite a una ontología importar de otra ontología. Cuando una ontología importa de otra, todas las clases, propiedades y definiciones individuales que tiene la ontología importada, están disponibles para su uso en la ontología que importa.

  Puede ser utilizado para representar de forma explicita el significado de términos pertenecientes a un vocabulario y definir las relaciones que existen entre ellos. OWL es una revisión del lenguaje de especificación de ontologías DAML+OIL, que aprovechando éste lenguaje ha aprendido de sus fallos y deficiencias, y ha conseguido ser un lenguaje mucho más expresivo y completo. De hecho se trata de una recomendación de la W3C junto a RDF.

  Las principales diferencias entre OWL y DAML+OIL son las siguientes:

  - OWL permite el uso de propiedades simétricas, las cuales no están permitidas en DAML+OIL. Para definir éstas propiedades se utiliza la primitiva: *owl: SymetricProperty*.
  - OWL, al igual que DAML+OIL soporta relaciones disjuntas, pero éstas no pueden expresarse con la primitiva *daml: disjointUnionOf*. Éste efecto se consigue con la combinación de dos primitivas, *owl :unionOf* y *owl:disjointWith.*
  - OWL utiliza las primitivas de RDF sin renombrar, al contrario de lo que hacía DAML+OIL.

- Por tanto, en OWL muchas primitivas propias de DAML+OIL han sido renombradas, significando lo mismo, pero con diferente nombre.

OWL se divide en tres sublenguajes *OWL-Lite*, *OWL-DL*, *OWL-Full*,

- OWL Lite →Como hemos mencionado anteriormente, es el sublenguaje más sencillo que posee OWL. Se trata de un lenguaje muy simple, y posee un razonamiento muy eficaz, a cambio obtenemos un nivel de expresión muy limitado. Posee las funcionalidades básicas para la definición de clases y algunos recursos muy limitados para especificar restricciones. En cuanto a la cardinalidad, solo permite restringir la cardinalidad a 0 ó 1.

- OWL DL → Adopta el nombre (DL) de Description Logic, que se trata de la precursora de la ontología en la búsqueda de modelos formales de representación de conocimiento. Si queremos que un documento RDF sea válido en éste sublenguaje deberemos adaptarlo extendiéndolo en varios aspectos y restringiéndolo en otros. Éste sublenguaje ofrece el máximo poder expresivo sin perder computabilidad ni la capacidad de ser decidible, ya que éste es el precio que hay que pagar cuando se tiene un lenguaje muy potente y con mucho poder expresivo, esto quiere decir que la aplicación de las reglas de inferencia no se transforman necesariamente en un algoritmo (computable), y que no puede ser determinada la veracidad o falsedad de todo enunciado decidible.

- OWL Full → Éste sublenguaje como hemos mencionado arriba, se trata del sublenguaje perteneciente a OWL más completo. Ofrece el máximo poder expresivo, pero debe pagar a cambio el alto precio de perder la capacidad de ser decidible, es decir no siempre puede dar una respuesta ante cualquier consulta que se realice al modelo.

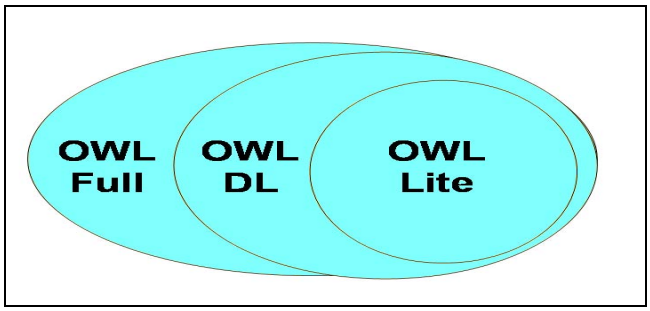
  El vocabulario de OWL Full es el mismo que el de OWL DL, pero con la ventaja de que se puede combinar libremente con sentencias propias de RDF-Schema, pudiéndose mezclar sin ningún tipo de restricción clases, propiedades y valores. Esto nos da la posibilidad que un recurso pueda ser al mismo tiempo instancia, clase e individuo, podemos deducir de esto que se trata de un lenguaje de 2º orden y que por tanto no es decidible. Cabe hacer notar que no existe ninguna demostración formal que afirme que se trata de un lenguaje de lógica de 2º orden.

Cada uno de estos lenguajes es una extensión de su predecesor. Podemos hacer las siguientes afirmaciones, pero no a la inversa:

- Cada ontología OWL Lite es una ontología OWL DL.

- Cada ontología OWL DL es una ontología OWL Full.
- Cualquier modelo definido en OWL Lite u OWL DL forma un esquema RDF válido.

Para ver éstas relaciones más claramente, se muestra la siguiente figura (figura 2.3).



2.3 Relación de sublenguajes OWL

Una de las novedades más importantes que incorpora éste lenguaje son las **c**ondiciones necesarias y suficientes**.** A las clases que incorporan éste tipo de condiciones se les denominan *clases definidas* y a las que incorporan condiciones necesarias se les llaman *clases primitivas*.

Las clases definidas se utilizan para clasificar todas las clases que cumplan una determinada condición. Veamos un ejemplo.

Un *Estudiante_de_tercero* , será cualquier estudiante que está cursando alguna asignatura del tercer curso. En éste caso, *Estudiante_de_tercero* sería una clase definida. Con ésta condición, es suficiente para reconocer cualquier estudiante que esté matriculado en alguna asignatura de tercero.

Todos los proyectantes, estudiantes que realizan el proyecto fin de carrera, tienen al menos una asignatura de tercero que es el proyecto en si, por tanto *Proyectantes* se trata de una clase primitiva.

Debe quedar claro que el calificador solo debe clasificar cosas debajo de las clases definidas, no de las primitivas. En éste caso el razonador clasificará automáticamente la clase primitiva *Proyectante*, debajo de la clase definida *Estudiante_de_tercero*.

2.1.3. ¿Para qué se necesita una ontología?

Una ontología es uno de los principales componentes para crear una nueva *base terminológica*, porque se parte del principio de que ningún término puede existir antes de haber sido relacionado con un concepto**. [30].**

El objetivo es que esta red conceptual se convierta en el eje central en la búsqueda, elección, estructuración y continua evaluación de los términos. Los conceptos resultantes no son más que una formalización posible de una parte del conocimiento y serán punto de referencia para la unidad terminológica

Hay que tener en cuenta que, aunque a nivel conceptual tratemos con abstracciones, éstas han de formalizarse por medio de unidades léxicas, expresiones pertenecientes al lexicón primario, imágenes **[32]** o incluso responder a un amplio tono de grises que cubra el espacio existente entre estos tres extremos.

Para estructurar e introducir los conceptos en la BD en primer lugar se ha de *ubicar conceptualmente* el término extraído o adquirido. Se trata de determinar a qué concepto se refiere el término o viceversa teniendo en cuenta que puede tratarse de una unidad léxica que no sea un concepto relevante, una conclusión a la que no siempre se llega desde el principio. Si creemos que se trata de un concepto necesario, y no ha sido recogido con anterioridad, habrá que crearlo. Si el concepto responde a una especificación de otro ya existente, se ubicará como hijo de éste; es decir, un concepto siempre habrá que clasificarlo (decir qué tipo de concepto es) y ubicarlo en la ontología por medio de inferencias a partir del texto y de nuestro conocimiento del dominio oncológico. Una vez situado en el editor ontológico se le añaden unas características mínimas y su concepto inmediatamente superior. En el caso de tratarse de una RELATION además habrá que añadir su relación inversa, y si se trata de un ATTRIBUTE habrá que añadir un número, color, etc. Lo ideal es que cada concepto cuente con un abanico de vínculos con otros conceptos que fluctúe entre 10 y 15 **[33]**.

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

- **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.

2.1.4. Líneas futuras

La lingüística de corpus ha puesto de manifiesto la necesidad de derivar la descripción lingüística de un análisis detallado de la lengua usada de forma natural, ya que este estudio puede ayudar a revelar muchas regularidades (e irregularidades) en nuestro uso de la lengua, que antes no se habían observado, o pueden ayudarnos a verlas de forma más uniforme, con una perspectiva más amplia y con índices de frecuencia relativa más fiables. Por tanto, la lingüística de corpus se ha convertido durante los últimos veinte años es una herramienta primordial de la lexicografía **[34]**. En el trabajo termino gráfico también es necesaria una *evidencia mensurable*, aunque se utilice abiertamente la competencia lingüística del experto, que permita formular conclusiones más objetivas que hasta ahora se hacían desde la introspección o la consulta a nativos **[35]**.

La adquisición de información léxica puede hacerse de varias formas: manualmente, a partir de diccionarios en formato magnético, o a partir de textos en formato electrónico **[31]**. Es obvio que la primera está cayendo en desuso por la cantidad de tiempo que implica; en cuanto a la segunda, el diccionario presenta limitaciones a la hora de adquirir información porque está destinado a un usuario que se supone tiene conocimiento acerca de la estructura del lexicón, por lo que presenta una información estática que no se corresponde con la dinamicidad de la lengua real. Por tanto, los corpus textuales informatizados se han convertido en la principal fuente de adquisición de conocimiento. Los córpora pueden ofrecer información léxica muy relevante, sobre todo en aspectos relativos a los hábitos colocacionales de las unidades léxicas o sus propiedades combinatorias, y son una herramienta de gran utilidad para la extracción de ejemplos reales de uso, así como en el enriquecimiento y refinamiento de la información ya contenida en un lexicón computacional **[47].**

Desde que el proyecto COBUILD utilizó esta iniciativa pionera en la compilación de diccionarios, el concepto de *corpus* suele entenderse necesariamente como conjunto de textos en formato electrónico aunque en lo concerniente a otro tipo de características como tamaño o representatividad, hasta la actualidad no existe consenso **[48].** El único punto de acuerdo es que el tamaño del corpus está supeditado al fin para el que se ha recopilado. En este sentido, los córpora utilizados para trabajos termino gráficos son más reducidos que los del discurso general porque se

ciñen a un tema de trabajo que suele ser muy restringido **[36]** y el incremento de su tamaño no implica necesariamente una mayor representatividad.

Para crear este Corpus hay que tener unos objetivos precisos, el estudio del subdominio a tratar. La adquisición de conocimiento experto a partir de córpora relevante puede llevar a cabo además en varias lenguas de trabajo como pueden ser el inglés y el español, se requiere la ayuda profesional de expertos para asegurar la representatividad de la documentación. Se podría por lo tanto crear un corpus multilingüe que a su vez contiene textos comparables y paralelos, es decir, tanto textos similares en todas sus lenguas como originales y sus traducciones. Aunque no existe acuerdo respecto a la conveniencia de usar traducciones en terminografía, hay que tener en cuenta que en algunos ámbitos como el biomédico "gran número de publicaciones son traducciones que, en muchos casos, aportan soluciones neológicas a lagunas terminológicas en la lengua término. Mientras que en español se debate la validez del trabajo con textos traducidos, en los textos en inglés no se plantea si se trata de una traducción o no, cuando en muchos casos, los artículos científicos son de hecho traducciones de lenguas minorizadas en el ámbito científico al inglés" **[37]**. Esta postura también la comparte **[38]** Teubert, **[35]** Langlois y Heid **[39]**, ya que ven el potencial para extraer candidatos a equivalentes de traducción.

Para poder recopilar material para un corpus, lo ideal sería obtener la información de: i) textos extraídos de Internet; ii) enciclopedias, manuales y publicaciones en CD-ROM; iii) textos escaneados por su riqueza de vocabulario definicional (manuales del subdominio a tratar). La selección del material se haría teniendo en cuenta criterios como cantidad, calidad, simplicidad, documentación, pertenencia al dominio de especialidad, fecha de producción y condición lingüística del texto, factualidad, tipo textual, nivel de tecnicidad y receptores del texto **[28]**. Todo ello para poder satisfacer las necesidades de la amplia gama de usuarios potenciales que servirá de plataforma para representar y consultar el conocimiento adquirido.

Todo esto nos hace pensar en las posibles cualidades que se le pueden dar a las ontologías resumidas 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.

2.1.5. Conclusiones sobre las ontologías

Se necesita convertir la información en conocimiento mediante unas estructuras de conocimiento formalizadas (las ontologías) que referencien los datos, por medio de 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 que trate el conocimiento almacenado. Este conocimiento se podrá integrar en bases de datos, donde usuarios comunes no sólo encontrarán la información precisa, sino que podrán realizar inferencias de forma automática buscando información relacionada con la que se encuentra situada en otros medios y con los requerimientos de las consultas realizadas por los usuarios. Además, podrán intercambiar sus datos siguiendo estos esquemas comunes consensuados, e incluso podrán reutilizarlos. Además es importante un lenguaje estándar que especifique dichas ontologías con mayor precisión. Con ese fin se han creado lenguajes como OWL, que es una especificación del W3C para especificar ontologías.

Además las ontologías requieren de un lenguaje lógico y formal para ser expresadas. En la inteligencia artificial se han desarrollo numerosos lenguajes para este fin, algunos basados en la lógica de predicados, como KIF y Cycl que ofrecen poderosas primitivas de modelado, y otros basados en *frames* (taxonomías de clases y atributos), que tienen un mayor poder expresivo, pero menor poder de inferencia; e incluso existen lenguajes orientados al razonamiento como Description Logic y Classic.

Todas estas necesidades surgen por el hecho de poder unificar criterios y esquemas reutilizables, así como mejorables. *En un lenguaje de ontologías se pretenderá un alto grado de expresividad y uso.*

Pero surge un problema, ¿cómo poblar estas ontologías?, se podría hacer de forma manual, pero puede suponer un trabajo tedioso, además de conocer el medio y la ontología (así como sus reglas de diseño), por lo que se echan en falta una serie de herramientas. Por un lado un editor de ontologías que permita el enriquecimiento de las mismas, así como la edición de estas. Estos editores unidos a herramientas de perfeccionamiento de ontologías o de corrección de estas, hacen de las ontologías un instrumento útil en multitud de áreas **[Anexo D]**.

Tras la exhaustiva búsqueda de información puedo resaltar que las posibles aplicaciones y usos de las ontologías son:

- Repositorios para la organización del conocimiento.

- Servir de herramienta para la adquisición de información.
- Servir 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.
- Normalizar los atributos de los metadatos aplicables a los documentos.
- Crear una red de relaciones que aporte especificación y fiabilidad.
- Permitir compartir conocimiento.
- Posibilitar el trabajo cooperativo al funcionar como soporte común de conocimiento entre organizaciones, comunidades científicas, etc.
- Permitir la integración de diferentes perspectivas de usuarios.
- Permitir el tratamiento ponderado del conocimiento para recuperar información de forma automatizada.
- Permitir la construcción automatizada de mapas conceptuales y mapas temáticos.
- Permitir la reutilización del conocimiento existente en nuevos sistemas.
- Permitir la interoperatividad entre sistemas distintos.
- Establecer modelos normativos que permitan la creación de la semántica de un sistema y un modelo para poder extenderlo y transformarlo entre diferentes contextos.
- Servir de base para la construcción de lenguajes de representación del conocimiento.

2.2. Herramientas para la edición de Ontologías

Existen numerosos editores para la creación de ontologías, ya que los lenguajes más expresivos como DAML+OIL y OWL, son lenguajes muy complicados, y es muy fácil cometer errores trabajando directamente con ellos. Las herramientas para la creación de ontologías incorporan soporte para incluir documentación en la ontología, importar y exportar ontologías a diferentes formatos y lenguajes, manejo de librerías de ontologías, etc.

La mayoría de los kits de herramientas de desarrollo de ontología proporcionan un entorno integrado para crear y editar ontologías, la verificación de errores e incoherencias, navegar por múltiples ontologías, y compartir y reutilizar los datos existentes mediante el establecimiento de asignaciones entre las diferentes entidades ontológicas. Sin embargo, su diseño de la interfaz de usuario (look & feel) y el uso están inspirados en el estilo tradicional basado en los paradigmas KR,

cuya limitado y metódico marco de empinadas curvas de aprendizaje, supone que sea difícil de utilizar para el usuario promedio.

El éxito de estos proyectos se medirá por el aumento en el uso de estas herramientas por parte de todos los usuarios, desde principiantes a expertos de dominio y los conocimientos técnicos. El fomento de su uso se logra a través de Plug-ins adicionales, tutoriales, talleres y el contacto con la comunidad.

En éste apartado vamos a hacer un resumen de los editores de ontologías más potentes, destacando sus características y funcionalidades, haciendo más hincapié en aquellas más utilizadas.

2.2.1. LinkFactory

Es un editor de ontologías desarrollado por Language & Computing **[20]**. Se trata de un editor que se puede utilizar no sólo para construir ontologías sencillas, sino también ontologías muy grandes y complejas independientes del lenguaje formal.

El sistema LinkFactory posee dos componentes muy importantes, el LinkFactory Server (actúa de servidor) y el LinkFactory Workbench (está en el lado del cliente). Los dos componentes están desarrollados en Java.

El LinkFactory Server se encarga de almacenar los datos en una base de datos relacional. El acceso a la base de datos desde el lado del cliente se realiza a través de unas determinadas funciones, las cuales son accedidas desde el software del cliente a través de unas APIs estandarizadas que permiten construir aplicaciones en la parte superior de la base de datos semántica sin tener total conocimiento de la estructura interna de la base de datos.

El Workbench es un marco de trabajo dinámico implementado en JAVA beans. Si utilizamos los beans independientemente obtendremos vistas limitadas de la ontología y de la representación del conocimiento, mientras que si combinamos los diferentes beans podemos obtener unas vistas y un manejo muy potente de la ontología.

Vamos a definir ahora las características principales de la representación del conocimiento:

- Existen mecanismos para combinar varias ontologías.
- Posibilidad de especificación de condiciones necesarias y suficientes para la definición de conceptos individuales.
- Al definir nuevos conceptos en los términos del lenguaje natural, éstos se auto clasifican como definiciones formales.

2.2.2. OILEd

Se trata de un editor gráfico de ontologías. Permite desarrollar ontologías en diversos lenguajes de especificación. Inicialmente se orientó al diseño de ontologías en DAML+OIL, pero actualmente es compatible con RDF, RDFs y OWL**. [21].**

Otra característica importante de Oiled es que proporciona un razonador conocido como "FaCT" que se utiliza principalmente para comprobar si una ontología es consistente o no.

2.2.3. OntoEdit

Editor gráfico de ontologías **[22]**. Al igual que los anteriores permite representar o definir una ontología gráficamente.

La gran ventaja de ésta herramienta es que además de permitir editar ontologías permite almacenarlas en una base de datos relacional. Es más extensible y escalable que Oiled, ya que permite ampliar su funcionalidad mediante plug-ins. Al igual que Oiled soporta numerosos lenguajes de especificación como DAML+OIL, RDFs, OXML y Frame-Logic.

2.2.4. WebODE

Herramienta sencilla pero potente para la creación de ontologías desarrollada por Ontology & Knowledge Reuse Group **[23]**. Se trata de un editor de ontologías de libre acceso que está disponible a través de la Web. Basado en la metodología Methontology. Actualmente permite exportar sus ontologías a diversos lenguajes como: RDF(S), OIL, DAML+OIL, OWL, WebODE's XML, UML, etc.

2.2.5. Ontolingua Server

Herramienta de libre acceso creada por el KSL (Knowledge Software Laboratory) de la Universidad de Stanford **[24]**. Éste software es accesible desde cualquier navegador Web, y permite crear, editar, modificar y usar ontologías. Permite exportar sus ontologías a diversos lenguajes como KIF, Ontolingua, OKBC, LOOM, etc.

2.2.6. SWoop editor

Es una herramienta para la creación, edición y depuración de ontologías OWL **[42]**. Fue producido por el laboratorio MIND en la Universidad de Maryland, College Park, pero ahora es un proyecto de código abierto para todos.

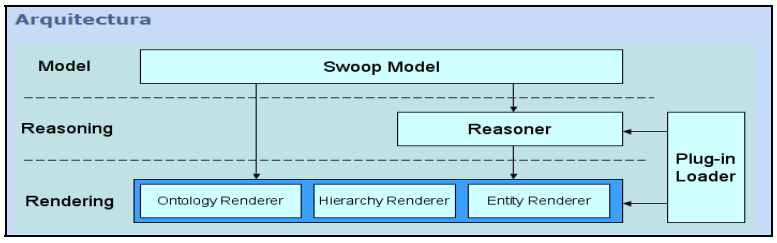
La mayoría de los kits de herramientas de desarrollo ontológico proporcionan un entorno integrado para crear y editar ontologías, la verificación de errores e incoherencias (con un razonador de ontologías), navegar por múltiples ontologías, y compartir y reutilizar los datos existentes mediante el establecimiento de asignaciones entre las diferentes entidades ontológicas. Sin embargo, su diseño de la interfaz de usuario (look & feel) y su uso están inspirados en el estilo tradicional basado en los paradigmas KR, pero su empinada curva de aprendizaje lo limita, por lo que es difícil de utilizar para el usuario medio.

Por otra parte, se puede considerar la posibilidad de una hipermedia inspirada en el editor ontológico que utiliza un *navegador Web metafórico* para su diseño y uso. Esta herramienta es más eficaz (en términos de aceptación y el uso) para el usuario medio de Internet por presentarse más simple, coherente y familiar para hacer frente a las entidades de la Web Semántica. Basándose en esta hipótesis, se presentó el editor de ontologías SWOOP que significaba una herramienta rápida y fácil para la navegación y el desarrollo de ontologías OWL.

- SWOOP emplea un navegador similar a los utilizados para navegar en la red : Una barra de direcciones donde una URI de una ontología (o clase / propiedad / persona) se puede introducir directamente para su carga; hipervínculo basado en la navegación a través de las entidades ontológicas (la barra de direcciones URL cambiará en consecuencia cuando proceda); botones clásicos (Atrás, Siguiente, etc.) para la barra transversal, y marcadores que permiten ser guardados para poder ser referenciados posteriormente.
- Edición en línea: Toda la edición en SWOOP se realiza en línea con HTML renderizado, usando diferentes códigos de color y estilos de fuente para destacar los cambios en la ontología. Por ejemplo. representaciones diferentes de axiomas añadidos frente a axiomas borrados y axiomas referenciados. Deshacer / rehacer son opciones que dispone de un Log de cambios de la ontología y opción de retroceso.
- Diseñado para OWL Rec. Soporte para múltiples ontologías (Navegación, Mapeo, Comparación).
- Funciones avanzadas:

Ejecución "buena y completa" de consultas con conjunctive ABox (escritas en RDQL) en una ontología, utilizando Pellet.

- Partición de las ontologías automáticamente por su transformación en un E-connection.
- Un modo de depuración: Al exponer el flujo interior de trabajo en una "Descripción de *Logic Tableaux reasoner*" (pellets), de una forma comprensible y legible, donde las explicaciones dadas se ofrecen para ayudar a los usuarios a comprender la causa de las incoherencias detectadas en las ontologías.
- Soporte de anotación y colaboración**:** Un Plug-in en SWOOP que permite a los usuarios escribir y compartir anotaciones sobre cualquier entidad ontológica (clase / propiedad / persona). Además, establece que se puede unir a la anotación para el intercambio de mensajes.
- Una arquitectura basada en el paradigma de Modelo-Vista-Controlador (MVC) (figura 2.4).



2.4 Arquitectura Swoop

- Carga de ontologías de forma similar a un razonador de ontologías.
- Vistas múltiples
- Prestación de apoyo a la edición.
- Plugin basado en el sistema. Carga de nuevos racionamientos y los racionamientos de forma automática.

- Como plan de futuro resalta la importancia del concepto de reutilización (búsqueda y fusión): El algoritmo de búsqueda se ha diseñado y combina las palabras clave con DL, basado en encontrar conceptos relacionados en las ontologías existentes. Después de haber encontrado estos conceptos / propiedades en ontologías externas, SWOOP puede ayudar al usuario en la vinculación de los datos mediante E-Connections.
- SMORE (que se publicará por separado).
- DIG Interface para razonadores ontológicos.

- XMLS (esquema XML -> convertidor de ontología OWL).

El conocimiento de lenguajes ontológicos y su correcta utilización, lo que conlleva a un conocimiento y manejo adecuados, fue lo que motivó su creación. Pero dada la expresividad de la lógica, a los recién llegados se les plantean dificultades para comprender las inferencias de OWL, así como los errores cometidos en la fijación de una ontología OWL, ya que la mayoría de los razonadores de ontologías solo producen un informe de inferencias (o errores) de la ontología, sin explicar cómo o por qué se han dado. Aún siendo expertos de DL resulta difícil depurar errores en las grandes y complejas ontologías. Los dos tipos principales de errores de semántica o lógica encontrados en una ontología OWL son:

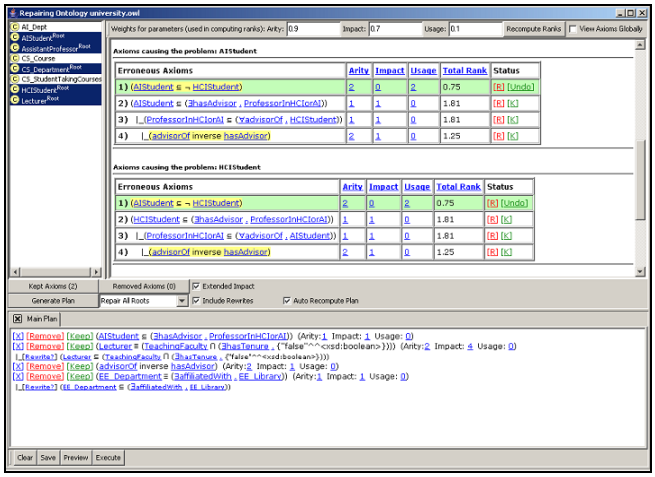
- Insastifiabiliadad de conceptos (Unsatisfiable Concept). Esto se produce cuando la definición de un concepto contiene una contradicción que impide que el concepto tenga un modelo, es decir, el concepto se ve obligado a *no* tener ningún individuo.
- Ontología inconsistente (Inconsistent Ontology). Esto ocurre cuando los axiomas de una ontología contienen una contradicción que impiden a la ontología tener un modelo, por ejemplo, cuando afirmamos que la ontología de un individuo pertenece a un concepto insastifiable.

El objetivo de SWoop fue Diseñar *un estándar* de servicios DL que atendiese específicamente a la reparación y la depuración de inconsistencias lógicas en ontologías OWL. Otra motivación, para su creación, fue demostrar que la buena depuración no sólo ofrece apoyo a los usuarios en el control de su modelado, sino también anima a experimentar con mayor libertad con las expresiones, y ayudaría a comprender y conocer las ontologías a través del proceso de depuración (PDep).

De sus módulos podemos resumir:

- Justificación de premisas (identificar el axioma) para cualquier vinculación arbitraria obtenida.
- Detección de errores y explicación de la raíz del problema.
- Detección de errores en la obtención de la raíz. Un eficiente procedimiento iterativo.
- Reparación de ontologías (figura 2.5). Este servicio actúa como una guía para la reparación de la ontología, para ayudar al usuario a comprender y evaluar las distintas opciones de reparación disponibles. Propone soluciones de corrección.
- Gestión de Cambios y versiones de la ontología. SWoop tiene una versión ontológica característica que apoya a la hora de hacer y deshacer los cambios (con logging), junto a la

capacidad de control y archivo de diferentes versiones de la ontología. Alternativamente, si un usuario tiene dos versiones de ontología diferentes, una consistente y otra inconsistente, una diferencia entre las versiones puede ser localizada usando en orden el formato conciso de renderizado de SWoop, utilizado para determinar los posibles cambios entre las versiones. Al examinar estos cambios producidos, y observando los “bugs” (fallos) producidos por los cambios, los usuarios pueden encontrar y eliminar las definiciones de entidad erróneas y los axiomas en la ontología necesarios.



2.5 Reparación de Ontologías,Swoop

- Protocolo Annotea. el Protocolo Annotea para cliente de Swoop (que se encuentra integrado) se utiliza para publicar el conjunto de cambios existentes, además de un comentario, de esta forma otros usuarios de “Annotea store” pueden ver estos cambios en su contexto y los comentarios que se hicieron, y aplicar los cambios para ver sus efectos y publicar las respuestas. Estos intercambios persistentes proporcionan un repositorio de casos reales para su posterior estudio por los modeladores.

Merecen mención algunas de sus particularidades como las siguientes:

- Ontologías, clases, propiedades, y los individuos se encuentran en un nivel alto, de manera accesible.
- Se puede "ver la fuente" de las ontología y de sus entidades en un número común de sintaxis (por ejemplo, RDF / XML, OWL Abstract Sintaxis, Turtle).
- La ontología tiene una gestión de cambios con amplios mecanismos de rollback y undo.
- También atribuye y distribuye los cambios en la ontología.
- Busca a través de múltiples ontologías y "encuentra todas las referencias".

- Compara entidades usando el ResourceHolder.
- Exportación de Ontologías directamente a un almacén remoto WebDav.
- Depura ontologías utilizando Pellet (explicaciones de clases no fiables y ontologías incompatibles).
- Navegación HiperTextual.
- Tiene un registro de cambio de versiones.
- Permite utilizar imágenes.

Es de resaltar que dispone de varios Plug-ins, caracterizados por estar en formato .jar.

Como Plug-ins accesibles dispone de "Plugin Pellet" que es de código abierto, basado en Java OWL-DL y que es un razonador de ontología desarrollado en MINDSWAP. Se utiliza para realizar la coherencia de la ontología en tiempo de ejecución.

También tiene otros en preparación como "Natural Language Entity Renderer": su finalidad es la de divulgar Información. Se caracteriza por un algoritmo que proporciona un lenguaje natural (NL) paráfrasis de la ontología OWL, cuya meta es garantizar la fluidez (lectura) de la salida y la precisión en términos de significado. Por su formalismo lógico de descripción, el algoritmo usa el Visitor Design Pattern para construir un árbol patrón de la ontología con intervalos textuales y con relaciones de términos antológicos.

2.2.7. Protégé

Protégé **[41]** es un editor de ontologías y bases de conocimiento gratis y abierto (*actualmente se encuentra disponible la versión 4.0 en formato libre*). Se encuentra basado en Java y soporta Frames, XML Schema, RDF y OWL y además cuenta con un ambiente "Plug-ad-play".

Sirve de apoyo de una sólida comunidad de desarrolladores y académicos, así como el gobierno y usuarios corporativos, los cuales están utilizando las soluciones de Protégé en áreas tan diversas como la biomedicina, la recogida de datos y modelado corporativo.

Protégé al igual que otros sistemas basados en marcos describen ontologías declarativas, estableciendo explícitamente cual es la jerarquía de clases y a qué clases individuales pertenece.

Protégé proporciona soporte para la construcción del marco basado en ontologías y es también una sólida plataforma para ontologías OWL.

- Protégé-OWL es un editor incorporado en la parte superior de la fuente abierta OWL API Support para OWL 2.
- Entradas directas a memoria, conexión con Pellet y FaCT++.

- Soporte para editar múltiples ontologías.
- Permite la creación, importación y exportación de fichas configurables por el usuario.
- Su marco GUI es Configurable. Además de permitir la creación, importación y exportación de la configuración del usuario.
- Presenta múltiples vistas alternativas de la misma ontología.
- Tear-off y clonación de componentes.
- Atajos de teclado.
- Apoyo del arrastre y suelte.
- Carga lenta de componentes y plug-ins para mejorar la velocidad y el uso de la memoria.
- OWL API para OWL 2.0 proveé un modelo eficiente en memoria.
- El marco plug-in es compatible con OSGi Equino (fácilmente extensible).
- La aplicación genérica del marco está separada del Kit editor OWL.
- En la modularización hace un uso inteligente de los repositorios local / global para manejar las dependencias de importación. Además permite la carga de múltiples ontologías en un solo trabajo.
- Permite el cambio entre las ontologías de forma dinámica.
- Sugerencias UI para mostrar en las declaraciones hechas en la ontología.
- Refactorización: fusión de ontologías y la eliminación de las importaciones redundantes.
- Refactorización: moviéndo axiomas entre ontologías. Permite Renombrar (incluyendo varias entidades), manipulación y separaciones de los diferentes axiomas; una rápida creación de clases definidas; transforma en diversas restricciones (incluyendo recubrimiento(covering)); conversión de IDs a etiquetas de conversión y traslado de axiomas entre ontologías.
- Infiere axiomas que muestra en la mayoría de las vistas estandar.
- Interfaz directa con FaCT++.
- Interfaz directa a Pellet reasoner .
- Plug-ins de Reasoners.

Siguiendo con las características de OWL, las siguientes peculiaridades son importantes a la hora de resaltar en el aspecto de la edición de OWL.

- Prestación de conformidad de las entidades de la ontología, usando fragmentos URI o valores de anotación.
- Análizar descripción OWL (también admite nombres en las anotaciones).
- Incorporado el cambio de apoyo que permite deshacer los cambios y compuestos.
- Histórico de autocompletado y expresiones.
- Resaltado de sintaxis.
- Creación automática de documentos y etiquetas IDs para nuevas entidades.
- Normas de edición SWRL.

Con referencia a los plug-ins que existen hoy día resaltamos:

- Arquitectura altamente conectable con el apoyo de muchos de los diferentes tipos de Plug-in incluyendo vistas, menú de acciones, reasoners, preferencias, gestores circulares y más.
- Actualización automática de nuevos Plug-ins y versiones.
- Muchos Plug-ins disponibles incluyendo reasoners, matrices, secuencias de comandos, árboles existenciales, extracción de textos, explicaciones, procesamiento de la ontología, pruebas de marco, generación de lenguaje natural y más.
- Permite reconfigurar la interfaz de usuario añadiendo y eliminando tablas y vistas a través de sus pestañas, así como ver los menús.

Se espera actualizar, con el fin de mantener , Protégé 4.0 para que sea lo más estable posible de cara a los usuarios y contribuyentes, ya que en los últimos años se ha producido una gran cantidad de cambios en el fondo de la API de OWL. Algunas de las modificaciones de la lengua han sido inevitablemente incompatibles - la mayoría de vocabulario, y también algunos cambios estructurales. Es una herramienta que pretende que tanto plug-ins, como interfaz de usuario y ontologías sean estables y compatibles.

2.2.8. WSMO Studio

WSMO Studio **[29]** es un software libre, de servicio Web Semántica y de modelización de procesos de negocio basado en el entorno de servicios Web de modelos ontología (WSMO). WSMO Studio está disponible como un conjunto de plug-ins de Eclipse que se puede ampliar y personalizar por 3 partes.

WSMO Studio entre sus características incluye:

- Es un editor integrado con la ontología WSML Reasoner (MINS, KAON2, Pellet, IRIS) de las comprobaciones de coherencia y consulta de ontologías
- Editor de elementos WSMO (servicios Web, los objetivos, mediadores)
- SAWSDL editor para añadir anotaciones semánticas a documentos WSDL
- Procesos de Negocios de Modelado de Semántica de acuerdo con el Business Process Modeling Ontology.
- Diseñador de coreografías, para WSMO coreografías céntricas.
- Importación y exportación de:
    - WSML.
    - Un subconjunto de OWL-DL.
    - RDF.
    - Representación XML de WSML.
- Front-end para la ontología / servicio / objetivos de repositorios
    - Integrado ORDI repositorio.
    - IRS-III adaptador (3ª Parte) .
    - WSMX adaptador.
    - Front-end para el servicio de descubrimiento componentes: EPFL QoS basado en el descubrimiento.
    - Integrado WSML Validador.
    - WSML editor de texto con sintaxis de colores.
    - Eclipse FMAM basado Axiom editor (3ª Parte).

WSMO Studio es en parte financiado por la Unión Europea IST proyectos DIP (6 º PM-507483), InfraWebs (6 º PM-511723), SemanticGov (6 º PM-027517) y Super (6 º PM-026850).*WSMO Studio* está disponible bajo una licencia LGPL. Añade una serie de Plug-ins interesantes como:

- IRS-III, que es un front-end que integra en WSWO Studio. Este Plug-in ofrece soporte para la importación y exportación de entidades WSMO, tales como objetivos, mediadores, ontologías y Web Services para el razonamiento y para el servidor. Además permite a los usuarios alcanzar objetivos mediante el IRSIII directamente desde el editor, mediante un simple apuntar y hacer click.

- El Editor de Axiom es un GUI para la construcción de expresiones lógicas complejas en WSML. Axiom Editor es un conductor de la ontología, dirigido para facilitar al usuario la construcción gráfica de complejas expresiones lógicas (llamados axiomas) sobre la base del conjunto de ontologías disponibles. Los axiomas son escritos en un subconjunto del lenguaje de modelado Web Service Web (WSML). No es necesario saber WSML con el fin de usar el Editor de INFRAWEBS Axiom. Los principales usuarios de esta herramienta se prevé que sean los proveedores de semántica WSMO basada en servicios Web. El Editor de axiomas de INFRAWEBS se desarrolla en el marco del FP6 IST Project 511723IST INFRAWEBS en el Instituto de Tecnologías de la Información - Academia de las Ciencias de Bulgaria por el equipo encabezado por el Dr.Gennady.

2.2.9. Resumen

En la (tabla 1) se presenta el resumen de todas las herramientas para edición de ontologías vistas en éste documento, destacando los aspectos más relevantes de cada una de ellas.

2.3. Planes de futuro

En Europa las tendencias actuales giran en torno a una de las áreas más importantes de la tecnología informática: permitir una Web Semántica y el comercio electrónico. Concentran sus esfuerzos en torno a la divulgación de esta tecnología a la industria. Naturalmente, esto incluye la educación y los esfuerzos de investigación para garantizar la durabilidad de los efectos y el apoyo de la industria. El propósito general de muchos de los proyectos vigentes es la aplicación “Web Semántica” que permitiría a *los usuarios* (expertos de dominio) crear "grupos de discusión" donde los usuarios anotan cualquier tema de interés a bajo nivel, y estas anotaciones se expresan utilizando los idiomas definidos en la iniciativa Web Semántica y los intercambios P2P ocurren en un modelo. Pero el *usuario final* no tiene que ser consciente de ello de inmediato. Para un *usuario final,* es simplemente una forma de expresar y recuperar el conocimiento, con otros compañeros de los usuarios de una forma mucho más específica y éxito que lo que normalmente permite la Web hoy día, y las limitaciones ontológicas que se dan.

Por lo tanto, el conocimiento Web dedica sus esfuerzos a las siguientes áreas principales:

- **De extensión a la industria.** El objetivo principal de la Web como divulgación del conocimiento a la industria es de promover un mayor conocimiento y más rápida asimilación de las tecnologías de "Web Semántica" en Europa en plena sinergia con la actividad de investigación. Esta divulgación contribuirá a reducir el tiempo necesario para transferir la tecnología a la industria y al mercado.

Herramienta	Extensibilidad	Lenguajes compatibles	Almacenamiento de la Ontología	Acceso	Consistencia
LinkFactory	Flexible y Escalable	DAML+OIL, RDF Schema, XML	En BBDD.	Vía Web a través de navegador	Reglas de restricción y parámetros formales para comprobarla.
OILEd	No	OIL, RDFs, Daml+Oil, Owl, SHIQ, HTML	En ficheros *.daml*.	XML Schema muy limitado	A través de su razonador FaCT.
OntoEdit	Mediante plug-ins	XML, RDFs, Flogic, Daml+Oil	OXML	No proporciona acceso Web.	A través del plug-in OntoBroker.
Protegé	Mediante plug-ins	XML, RDFs, XML, Daml+Oil, Owl, Flogic, HTML.	BBDD compatibles con JDBC, ficheros de texto, etc.	A través de Servlets o Applets.	Mediante Plug-ins
WebODE	Mediante plug-ins	XML, RDFs, Oil, Daml+Oil, Flogic, Prolog, HTML	En BBDD compatible con Java	Vía Web a través de navegador.	Utiliza restricciones, verificadores de consistencia y otros parámetros para chequearla.
Ontolingua Server	Mediante plug-ins	KIF, Prolog, LOOM, OKBC, IDL, CML, Clips	En el servidor.	Vía Web a través de navegador.	La comprobación de la consistencia se hace en tiempo real, mientras se crea la ontología.
WSMO Studio	Mediante plug-ins	WSML,RDF	En BBDD, WSML	Vía Web a través de navegador	Mediante plugins e intercambio de anotaciones vía Web
SWoop editor	Mediante plug-ins en formato .jar	Pellet, RDQL, HTMl	En BBDD, Compatible varios formatos y ficheros de texto. Permite variós tipos de ontologías	Vía Web a través de navegador	Utiliza plug-ins para actualizarse. Su punto fuerte es su función correctora y reparadora. Además de su interfaz.

Tabla 1 Editores Ontológicos

- **Extensión de la Educación.** El Sitio Web de conocimientos tiene como objetivo trabajar en Pro de la creación de una Asociación Europea para la "Web Semántica Educación", que actuará como el principal foco de las actividades educativas sobre la Web Semántica.
- **Coordinación de Investigación.** El objetivo de un Sitio Web de conocimientos sería garantizar que la investigación realizada por los grupos líderes en este ámbito será suficientemente coordinados para evitar la duplicación y la fragmentación. Esta coordinación

es particularmente importante para la Web Semántica: ya que es una zona de ínter disciplinario de conjuntos de colaboración necesaria entre la investigación y las distintas comunidades. El objetivo del conocimiento es la Web para coordinar el esfuerzo de investigación europeo y para hacer la "Web Semántica" y los "Servicios Web Semántica" una realidad.

- El Sitio Web de conocimientos está coordinado por la Universidad de Innsbruck,(Austria) y consta de 19 principales socios.
- El Sitio Web de conocimientos forma parte de las ITS Internacional, una iniciativa que busca establecer una semántica como pilar básico de la moderna ingeniería informática.
- El Sitio Web de conocimientos es un miembro de la agrupación ESSI proyecto, que busca fortalecer la investigación europea y la industria en tecnologías de la Semántica.

Multitud de proyectos de investigación se centran en el desarrollo sobre las ontologías, semántica y herramientas para tales utilidades. Entre ellas podemos destacar Knowledge Web, con temas tan variados como "Métodos para la modularización de metodologías", "Aproximación del razonamiento y ontologías", "Herramientas comparativas de ontologías", "Modelado de procesos de consenso y aplicación de una Web semántica","Definición de negociación y técnicas de argumentación para los agentes que cumplen con las diferentes ontologías con el fin de permitir la comunicación entre ellos", "Investigación sobre técnicas de integración y aplicaciones"… entre muchos otros. De igual forma hay otros proyectos como Annotea, CORES, DBin, FOMI etc. se encuentran en situación similar de investigación y desarrollo, así como mejoras adaptación de herramientas ante las nuevas necesidades.

En este proceso hay áreas con un resultado más gratificante que otras, y el énfasis puesto en proyecto a desarrollar tiene una respuesta frente a los éxitos y oportunidades. Un logro importante que se busca es en el ámbito de la divulgación y creación de comunidades. Se busca atraer a diversos participantes de varios países, especialidades, e investigadores de código abierto y a las comunidades empresariales. Se busca la elaboración de un software más atractivo en las esferas de la semántica. Se están haciendo contribuciones sustanciales a la edición, normas fundamentales, y estudios de implementación de Buenas Prácticas de grupo de trabajo. Las principales vías se centran en buscar respuestas detalladas a preguntas sobre desarrolladores RDF, consultas, almacenamiento (análisis de cuestiones de escalabilidad; consultas de idiomas; API), clasificación, blogs semánticos, compatibilidad de ontologías, herramientas y aplicaciones que permitan editar y detectar errores en ontologías etc.

Pero los trabajos fundamentales se centran en la importancia y lo gratificante que es, tanto proporcionar un entorno en el que los miembros de la comunidad que rodea W3C pueden interactuar

como explorar conjuntamente las cuestiones y prácticas en torno a la semántica. Se tiende a un consenso al decidir la utilización de ficheros de configuración basados en XML (o clases Java para las más avanzadas aplicaciones).

Entre las características más importantes que se aprecian en los nuevos proyectos resaltamos:

- Sobre la base de los estándares abiertos y la filosofía de la iniciativa Web Semántica de W3C. Los datos pueden ser exportados y será ínter operable con cualquier otra herramienta de Web Semántica.
- P2P algoritmo (RDFGrowth) destinado a causar la mínima carga computacional a otros en la red. (Número de consultas distribuidas).
- En el concepto de seguridad, Firmas Digitales en cada una de las anotaciones. En este sentido, se puede utilizar para distribuir la información oficial y no oficial.
- Almacenamientos locales que permite un máximo de operaciones (no hay tiempo de espera de red, trabaja tanto fuera de línea como en línea).
- Almacenamientos locales de medios, el usuario puede aplicar reglas de filtrado y políticas de confianza a nivel local como considere oportuno.
- Almacenamiento local de los metadatos, este almacenamiento significa que se puede integrar en una sola vista la información que el usuario tiene a nivel local (y no quiere dar públicamente). El resultado es que se unen a las consultas (se pueden realizar en una visión integrada de los datos de P2P) las fuentes de datos locales (como la intranet y el usuario de escritorio DBS) archivos y recursos.
- Interfaces de respuesta, con multitud de herramientas integradas para buscar, explorar y editar anotaciones, así como los datos (por ejemplo, fotografías) que se le atribuye. Edición de metadatos como guía para las ontologías.
- Instalaciones integradas para publicar y recuperar los datos directamente a y desde la publicación de cuentas de la Web.
- Herramientas para el dominio de los expertos (no programadores) para crear el entorno para los usuarios finales, Brainlets, que se encargan de todos los conocimientos y proporcionar detalles de ingeniería y se centra en funcionalidades y herramientas.
- Licencias de libre soporte, aunque en esto no hay consenso.
- Se apoyarán en la reutilización de aplicaciones y adaptación de lo mantenido por las principales organizaciones y las iniciativas de normalización.

- Esquemas XML ➔ convertir a ontología OWL (u otras)
- Paso de unas ontologías a otras.

Existe un trabajo formal del W3C basado en pequeños grupos de trabajo, muy centrados en que los individuos dediquen tiempo a la creación de nuevos estándares semánticos, ontológicos y Web (por afluencia y necesidad de las nuevas tecnologías). Con esto lo que están buscando es obtener un mayor número de individuos dispersos geográficamente y pudiendo participar (a través del correo electrónico, IRC, talleres y la Web) en la iniciativa de la nueva Web Semántica. Se persigue con esta evolución de la Web semántica un mejor trabajo de normalización, lo que demostraría que el éxito puede ser complementado por un modelo de participación más amplio que permitiría a investigadores y ejecutores a hacer una contribución real a la implantación de las normas existentes y la creación de otras nuevas. El reto para el futuro es trabajar en Pro de una Web en la que todos los esfuerzos europeos de investigación contribuyen a las comunidades que sustentan la evolución de los estándares Web.

Precedentes

- **Especificación de Requisitos**
- **Estudio de Viabilidad**
- **Validación de Requisitos**
- **Metodología**

3. Precedentes

En este apartado se reúnen los documentos creados para el posterior análisis del sistema, así como los resultados de la validación de los requisitos del editor ontológico.

3.1. Especificación de requisitos

Esta parte es una especificación de requisitos del software (ERS) para la creación de un editor de ontologías, que surge por la necesidad de poder poblar e interactuar con los elementos de una ontología, con la finalidad de poder facilitar el trabajo a profesionales y técnicos que trabajen en un entorno definido. Todo su contenido ha sido elaborado teniendo en cuenta los requerimientos y necesidades observados en las especificaciones proporcionadas tras una serie de reuniones con los tutores del proyecto. Esta especificación ha sido estructurada en base al estándar IEEE Recommended Practice for Software Requirements Specification (IEEE-STD-830-1998).

3.1.1. Propósito

Esta especificación de requisitos software proporciona una descripción completa de todas las funciones y especificaciones del sistema APO. Explica el propósito y las características del sistema, las interfaces del sistema, la funcionalidad del sistema y las restricciones sobre las cuales debe operar.

3.1.2. Alcance

Ante la complejidad de poblar una ontología, tanto específica como genérica surge la necesidad de crear APO, una herramienta ontológica que gestionará tanto la creación, modificación y borrado de conceptos, así como sus relaciones y grados de afinidad entre relaciones. Es un sistema que trabajará con una amplia base de datos y no contempla en primera instancia la aplicación Web.

3.1.3. Definiciones, acrónimos y abreviaturas

La identificación de definiciones, acrónimos y abreviaturas se realizará mediante un código formado por dos letras específicas y dos dígitos que se irán incrementando en una unidad con cada uno de los casos.

- Las definiciones se identificarán DE-xx siendo xx números comprendidos entre 0 y 9.

- Los acrónimos se identificarán AC-xx siendo xx números comprendidos entre 0 y 9.
- Las abreviaturas se identificarán AB-xx siendo xx números comprendidos entre 0 y 9.

3.1.3.1. Definiciones

CÓDIGO	CONCEPTO	DESCRIPCIÓN
DE-01	Servidor	Nodo que atiende consultas y recibe datos.
DE-02	Cliente	Nodo que genera consultas.
DE-03	Consulta	Petición de un subconjunto de descripciones expresada en un lenguaje normalizado.
DE-04	Tupla	Secuencia ordenada de objetos (datos).

Tabla 2: Tabla definiciones

3.1.3.2. Acrónimos

CÓDIGO	CONCEPTO	DESCRIPCIÓN
AC-01	ERS	Documento de especificación de Requisitos Software.
AC-02	ANSI	American National Standard Institute.
AC-03	IEEE	Institute of Electrical and Electronics Engineers.
AC-04	STD	Software Test Documentation.
AC_05	SQL	Structure Query Lenguage.
AC-06	ANSI	American National Standards Institute.

Tabla 3: Tabla Acrónimos

3.1.3.3. Abreviaturas

CÓDIGO	CONCEPTO	DESCRIPCIÓN
AB-01	Std	Estándar.
AB-02	APO	Aplicación para la Población de Ontologías

Tabla 4: Tabla Abreviaturas

3.1.3.4. Referencias

CÓDIGO	TÍTULO	FECHA-DOCUMENTO
REF-1	ANSI/IEEE std. 830	1998
REF-2	ANSI SQL 99	1999

Tabla 5: Tabla Referencias

3.1.4. Visión General del Documento

Esta sección consta de tres subsecciones. Esta primera es la "Introducción" y proporciona una visión general de la ERS. En la "Sección 2", se da una descripción general del sistema, con el fin de conocer las principales funciones que debe realizar, los datos asociados, los factores, restricciones, supuestos y dependencias que afectan al desarrollo, sin entrar en excesivos detalles. En la "Sección 3", se definen detalladamente los requisitos que debe satisfacer el sistema.

3.1.5. Descripción Global

A continuación se hace la descripción global del sistema a desarrollar.

3.1.5.1. Perspectiva del Producto

APO es un sistema completo e independiente que no se encuentra relacionado con ningún otro producto, aunque no se cierran las puertas para en un futuro poder interactuar con otros módulos o herramientas. Por ello, no es necesario detallar ningún diagrama con las estructuras de componentes e interconexiones con interfaces externas.

3.1.5.2. Interfaces del Sistema

No aplicable. El sistema no se comunicará con ningún otro Sistema de Información externo.

3.1.5.3. Interfaces del Usuario

El sistema contará con una interfaz gráfica, tipo Windows, de acceso para el usuario. Consistirá en una ventana desde la que se facilitará el acceso a las siguientes operaciones según las primeras indicaciones:

- *Autenticación*: acceso al sistema.
- *Gestión de Conceptos*: alta concepto, edición de concepto.
- *Gestión de Término-Concepto*: asignar términos, borrar términos, modificar términos.
- *Gestión de Leguajes*: asignación de lenguajes, desasignación de lenguajes.

- *Gestión de Peso*: asignación de peso, modificación de peso.
- *Gestión Favoritos*: asignación de favoritos, desasignación de favoritos.
- *Gestión Barra de Menú*: despliegue de opciones de menú
- *Gestión de Búsqueda*: búsqueda exacta, búsqueda aproximada, filtrado de búsqueda.
- *Gestión de Relaciones*: asignación y desasignación de relación de conceptos.
- *Gestión de Jerarquías*: creación y modificación de la jerarquía de conceptos.
- *Gestión de la Función Restrictiva*: asignación, modificación, eliminación de compatibilidad de conceptos.

Las funciones restrictivas y descriptivas de la ontología no se establecen, se considera trabajo futuro.

3.1.5.4. Interfaces del Hardware

No aplicable. El sistema no se comunicará con ningún otro Sistema Hardware.

3.1.5.5. Interfaces del Software

El sistema se comunicará con bases de datos, mediante sentencias SQL, a gestores que lo soporten (ya que se trata de gestores consistentes así como seguros).Cabe la posibilidad de que tenga que recibir información del sistema Cognos (se encuentra en fase de desarrollo).

No tiene interfaces con otros sistemas software.

3.1.5.6. Interfaces de Comunicaciones

- SQL: Protocolo de Base de Datos que trabaja por el puerto 1433 por defecto.
- ODBC: Interfaz que requiere el sistema para soportar MySQL y Oracle, incorporado en la mayoría de sistemas operativos (Windows, Unix, Apple…)

3.1.5.7. Restricciones de Memoria

Se trata de una aplicación sencilla que no requiere de grandes requisitos. Se trabajará con un sistema de bases de datos Oracle, y para ello requiere de unos requisitos mínimos. Necesitará para poder funcionar correctamente al menos 128 Mb de RAM (recomendable 256 si se quiere trabajar

con Oracle 10g o versiones superiores). Necesita un área de SWAP (o área de intercambio) y para ello se recomienda el doble de la RAM (si se cambia, habrá que reiniciar el equipo para ver los cambios) .Por último se recomienda suficiente espacio en el archivo temporal.

El sistema de ficheros debería de ser NTFS (si fuera fat32 se necesitarían más requisitos) .Espacio en disco duro (Básica, Enterprise. aproximadamente 1.5 GB).

3.1.5.8. Operaciones

Las copias de seguridad, que se realicen de todos los datos que maneja el sistema, serán realizadas por el propio gestor de la base de datos; por tanto quedan fuera del alcance de APO. No se especifican otras operaciones en primera instancia.

3.1.5.9. Requisitos de Adaptación del Sitio

Adaptaciones necesarias según necesidades de las herramientas utilizadas para el desarrollo (al no haber requisitos iniciales por este tema se decidirá con posterioridad) para las bases de datos.

3.1.5.10. Funciones del Producto

NOMBRE	DESCRIPCIÓN
Gestión de Acceso	El sistema validará los accesos de usuarios y comprobará permisos en la base de datos.
Gestión de conceptos	El sistema gestionará las altas, modificaciones de conceptos creados y sus características principales.
Gestión de términos-Conceptos	El sistema gestionará la creación, asignación, borrado y modificaciones en las relaciones de conceptos y términos.
Gestión de peso	El sistema permitirá asignar, modificar o borrar los pesos asignados en relaciones de conceptos con términos.
Gestión de favoritos	El sistema según el usuario que accede, recopila los términos favoritos asignados a cada concepto, así como modificación, asignación y control de un término favorito por concepto.
Gestión barra menú	El sistema tiene una barra menú (no se han especificado propiedades ni funciones a realizar) debe poder ser visible y desplegable desde todas las pantallas, y común a todas.

Gestión de búsqueda	El sistema permitirá hacer búsquedas y filtrar las búsquedas de conceptos de forma aproximada o exacta.
Gestión de relaciones	El sistema permitirá visualizar, cambiar, asignar y borrar las relaciones entre conceptos.
Gestión de jerarquías	El sistema permitirá visualizar, cambiar, asignar y borrar jerarquías entre conceptos.
Gestión de función restrictiva	No se especifican funciones en esta primera fase. Solo que se despliegue la pantalla correspondiente.
Gestión de función descriptiva	No se especifican funciones en esta primera fase. Solo que se despliegue la pantalla correspondiente.

Tabla 6: Tabla Funciones del Producto

3.1.5.11. Características del Usuario

Los usuarios no requieren de unos conocimientos técnicos específicos al ser un sistema automatizado y de fácil uso. Deben estar familiarizados con el entorno a tratar y cierta familiarización con la herramienta y su funcionamiento.

3.1.5.12. Restricciones

1. **Políticas de regulación:** No aplicable.
2. **Limitaciones Hardware:** No aplicable.
3. **Interfaces a otras aplicaciones:** No aplicable.
4. **Operaciones en paralelo:** No aplicable.
5. **Funciones de auditoria:** No aplicable.
6. **Funciones de control:** No aplicable.
7. **Requisitos del lenguaje de alto nivel:** Cualquier lenguaje de programación de 3ª generación que permita desarrollar aplicaciones con entornos de ventanas Windows y acceso a bases de datos mediante sentencias SQL.

8. **Protocolos de acuerdo de señal:** No aplicable.
9. **Requisitos de fiabilidad:** No especificados, se consideran inherentes al producto y la base de datos.
10. **Criticidad de la operación:** Ha de poder soportar cargas de gran tamaño.
11. **Consideraciones de seguridad:** Control de acceso mediante clave de usuario, además del control de acceso será el proporcionado por el sistema operativo y los permisos concedidos por la base de datos. Cabe la posibilidad de aplicar algún tipo de cifrado.

3.1.5.13. Suposiciones y Dependencias

En cuanto a software podemos instalarlo en cualquier versión de Windows de 32 bits, pero obviamente funcionará mucho mejor en un sistema operativo preparado para dar servicios como puede ser Windows 2000 y 2003 Server (98 o XP también sirve). Requiere de gran capacidad de almacenamiento debido al volumen de datos que se desean guardar.

3.1.6. Desglose de Requisitos

Los requisitos se han etiquetado para posibilitar su seguimiento a lo largo del transcurso del proyecto si se necesitase referenciarlos. Muchos de los requisitos debido al tipo de proyecto, pueden sufrir modificaciones, se pueden añadir otros e incluso algunos pasarán a un estado de desuso. La identificación de cada requisito se realizará mediante un código formado por dos letras específicas y tres dígitos que se irán incrementando en una unidad con cada uno de los requisitos. Los requisitos sufrirán cambios a lo largo del ciclo de vida del proyecto.

De esta forma se evitarían errores en las referencias que más adelante en el proyecto se haga de los mismos.

- **Requisitos funcionales**: Se utilizarán las letras RF.
- **Requisitos de operación**: Se utilizan las letras RO.
- **Requisitos de interfaz de usuario**: Se utilizarán las letras RUS.

La especificación del requisito se hará mediante un nombre que resuma de forma clara y rápida la funcionalidad contenida en el requisito, una descripción en la que se defina la funcionalidad propiamente dicha. Para los requisitos funcionales además se añadirán las entradas proporcionadas, limitaciones o funciones de aplicación y salidas obtenidas.

3.1.7. Los Requisitos Específicos

A continuación se detallan los requisitos específicos del sistema, empezando por los requisitos no funcionales.

3.1.7.1. Requisitos No Funcionales

Estos requisitos a su vez se van a subdividir

3.1.7.1.1. Requisitos de Rendimiento

La capacidad del sistema dependerá de la comunicación con la base de datos y el volumen de datos intercambiado. Los requerimientos de memoria varían dependiendo del tamaño de la BD, el número de usuarios conectados concurrentemente y el uso que se le dé a la BD.

3.1.7.1.2. Requisitos de Diseño

Se requieren pantallas similares a las del Windows, así como botones visibles e intuitivos, menús desplegables, listados por pantalla, opciones de chequeo y selección. Vistoso, intuitivo y con variedad de funciones. Debe se un diseño consistente, y fiable (aunque no se ha especificado, se considera inherente al diseño posterior), debe ser eficaz y se valorará la eficiencia del sistema, modularizable, fácil de modificar y actualizar.

3.1.7.1.3. Requisitos de Interfaces Externas

No aplicable.

3.1.7.1.4. Atributos del Software del Sistema

Deba ser un lenguaje orientado a objetos, y con posibilidades de modularizar las funciones.

3.1.7.1.5. Requisitos de Operación

RO – 001: Instalación: La aplicación final se compondrá de una carpeta ejecutable (no aplicable a esta primera versión), es posible tener que hacer alguna modificación de la base de datos y posteriormente compilar de nuevo. Para ello se instalará primero la base de datos tipo Oracle.

RO – 002: Arranque manual: La aplicación se arrancará manualmente.

RO – 003: Arranque de la base de datos: La base de datos ha de montarse y subirse para poder funcionar la herramienta.

3.1.7.1.6. Requisitos de Interfaz de Usuario

Código	RUS-1
Descripción	Identificación de usuarios
Resumen	El usuario deberá identificarse para poder acceder a las operaciones para las cuales tiene permisos. Para ello dispondrá de un nombre de usuario y una contraseña.
Importancia	Vital

Código	RUS-2
Descripción	Menú de desplegables, con la finalidad de poder desplegar una ventana por función de la ontología, un total de 7.
Resumen	El sistema facilitará las operaciones de las que el usuario dispone mediante un menú de pantalla, que contará con una línea por cada opción.
Importancia	Vital

Código	RUS-3
Descripción	Menú en pantalla
Resumen	El sistema facilitará las operaciones de las que el usuario dispone mediante un menú de pantalla, que contará con una línea por cada opción.
Importancia	Vital

Código	RUS-4
Descripción	Ventana de búsqueda
Resumen	Es sistema puede desplegar una ventana de búsqueda, dicha ventana permite hacer una búsqueda de conceptos de forma única o general, permitiendo aplicar varios filtros sobre los resultados.
Importancia	Vital

Código	RUS-5
Descripción	Tablas de conceptos
Resumen	El sistema mostrará tablas sobre los datos de la base de datos, permitirá selecciones unitarias o múltiples dependiendo de la tabla.
Importancia	Optativo

Código	RUS-6
Descripción	Formato tabla
Resumen	El sistema permitirá visualizar información con tablas de filas por columnas de celdas, con caracteres alfanuméricos. El sistema mostrará una primera fila de celdas con los nombres de cada campo, las siguientes filas mostrarán los datos a visualizar. La tabla tendrá una barra de desplazamiento en caso de sobrepasar los datos el espacio reservado a la tabla en memoria. Se procurará que no sea necesaria una barra de desplazamiento horizontal, mediante la limitación del ancho de columna. (opción seleccionar y duplicaje)
Importancia	Opcional

Código	RUS-7
Descripción	Indicar error
Resumen	El sistema enviará al usuario un mensaje de error cuando se realice una solicitud incorrecta. Dicho mensaje de error consistirá en una ventana de tamaño reducido con un texto que indique al usuario el tipo de error cometido; no se permitirá la interacción con cualquier otro elemento del sistema hasta que la ventana del error sea cerrada.
Importancia	Opcional

Código	RUS-8
Descripción	Botones
Resumen	Se crearan botones con texto según la función a realizar, se pueden añadir imágenes a los botones e indicaciones al pasar el ratón por encima de ellos.
Importancia	Opcional

3.1.7.2. Requisitos Funcionales

Una vez vistos los requisitos no funcionales del sistema, es el momento de detallar los requisitos funcionales del mismo.

3.1.7.2.1. Gestión de autenticación

Código	RF-1
Descripción	Autenticación correcta
Resumen	El sistema permite a los usuarios autentificarse, para ello proporcionará su nombre de usuario y contraseña. El sistema comprobará que el usuario es el correcto y tiene permiso, dándole paso al acceso a la herramienta.
Entrada	Nombre de usuario y contraseña.
Proceso	Autenticación.
Salida	Pantalla visible de la herramienta.
Importancia	Vital.

Código	RF-2
Descripción	Aviso de autenticación incorrecta.
Resumen	El sistema permite a los usuarios autentificarse, para ello proporcionará su nombre de usuario y contraseña. El sistema comprobará que el usuario es el correcto y tiene permiso .Si no se produce mostrará un mensaje emergente.
Entrada	Nombre de usuario y contraseña.
Proceso	Autenticación.
Salida	Pantalla emergente informando del fallo de acceso.
Importancia	Vital.

Código	RF-3
Descripción	Contraseña enmascarada.
Resumen	El sistema ha de permitir que la contraseña introducida no muestre los caracteres reales “****” en vez de “PaCo”
Entrada	Contraseña
Proceso	Autenticación.
Salida	Texto en formato de asteriscos
Importancia	Opcional.

3.1.7.2.2. Gestión de la pantalla principal

Código	RF-4
Descripción	Menú desplegable
Resumen	El sistema permitirá desplegar el menú desde todas sus pantallas
Entrada	Selección del menú
Proceso	Utilización del menú
Salida	No especificadas. Visualizar menú y submenús creados
Importancia	Secundaria.

Código	RF-5
Descripción	Gestión de pestañas desplegables
Resumen	El sistema permitirá desplegar cada una de sus funciones ontológicas pestañas desplegables, siempre y cuando exista un concepto en el cuadro de texto Concepto.
Entrada	Despliegue de funciones ontológicas
Proceso	Navegar por las pestañas
Salida	Cada una de las pantallas correspondientes a la función ontológica
Importancia	Vital

Código	RF-6
Descripción	Generación de concepto
Resumen	El sistema permite generar la creación de un nuevo concepto. Creará un identificador para un posible nuevo concepto.
Entrada	Pulsar nuevo concepto.
Proceso	Creación de un identificador para la creación de un nuevo concepto
Salida	Creación del nuevo concepto.
Importancia	Vital.

Código	RF-7
Descripción	Selección del portapapeles
Resumen	El sistema permite elegir un elemento del portapapeles para operar con el.
Entrada	Concepto del portapapeles
Proceso	Utilización del portapapeles
Salida	Concepto visible desde cuadro de texto de conceptos.
Importancia	Vital.

3.1.7.2.3. Gestión ventana semiótica

Código	RF-8
Descripción	Alta concepto
Resumen	El sistema permite dar de alta un concepto. No se podrá dar de alta un concepto ya creado, ya que son únicos y es el sistema quién asigna su identificador. El sistema permite guardarlo o cancelar. Debe existir un concepto en la ventana textual del concepto para poder operar con el.
Entrada	Término, lenguaje y peso asignados para poder ser creados.
Proceso	Alta de concepto si se guarda.
Salida	Inserción del concepto a la base de datos.
Importancia	Vital.

Código	RF-9
Descripción	Asignación de término a concepto
Resumen	El sistema permite dar de alta un término. Se podrá dar de alta un término si hay seleccionado un concepto. Los términos pueden estar asignados a varios conceptos.
Entrada	El término, un peso y un lenguaje.
Proceso	Asignación de término a concepto.
Salida	Inserción en la base de datos. En pantalla en el listado de términos del concepto.
Importancia	Vital.

Código	RF-10
Descripción	Modificar término de concepto
Resumen	El sistema permite modificar las características de un concepto, términos asignados, relación con lenguajes y pesos.
Entrada	Concepto a modificar.
Proceso	Modificación de concepto.
Salida	Actualización de la base de datos.
Importancia	Vital.

Código	RF-11
Descripción	Gestión de favoritos
Resumen	El sistema permite asignar uno de los términos como favorito, en caso de seleccionar otro el primero se desmarcará como favorito. En caso de no asignarse ninguno el sistema asignará el de mayor peso.
Entrada	Selección de favorito
Proceso	Selección de favorito
Salida	Actualización de la base de datos. Visualmente aparecerá como marcado el término favorito.
Importancia	Vital.

Código	RF-12
Descripción	Control de creación de nuevo concepto
Resumen	El sistema controlará que en la creación de un nuevo concepto, al menos se añada un término relacionado con el concepto, de la misma forma un lenguaje y un peso.
Entrada	Concepto y términos asignados
Proceso	Selección de favorito
Salida	Actualización de la base de datos. Visualmente aparecerá como marcado el término favorito.
Importancia	Vital.

Código	RF-13
Descripción	Listar términos relacionados con un concepto
Resumen	El sistema permite visualizar los términos que están relacionados con un concepto.
Entrada	Ha de existir un concepto en el cuadro de diálogo de concepto, y se pulsará buscar.
Proceso	Listado por pantalla con todos los términos relacionados.
Salida	Listado por pantalla
Importancia	Vital.

Código	RF-14
Descripción	Filtrar términos asociados
Resumen	El sistema permite hacer un filtrado sobre los términos asociados a un concepto
Entrada	Listado de términos asociados, y selección de filtros
Proceso	Listado aplicando filtros.
Salida	Listado por pantalla filtrado
Importancia	Vital.

3.1.7.2.4. Gestión de la ventana búsqueda

Código	RF-15
Descripción	Cierre de la ventana
Resumen	El sistema debe dejar cerrar la ventana emergente aunque no se produjese una búsqueda.
Entrada	Pulsar salir
Proceso	Cierre de la ventana emergente
Salida	Pantalla principal del programa
Importancia	Opcional.

Código	RF-16
Descripción	Expansión de ventana
Resumen	El sistema expandirá la ventana en caso de querer filtrar los resultados, con la finalidad de poder ver las nuevas opciones de filtrado de la pantalla. Solo se podrá filtrar si se ha hecho una búsqueda de un término, en caso contrario este botón no está activa.
Entrada	Pulsar filtrar
Proceso	Expande la ventana de búsqueda
Salida	Pantalla principal del programa
Importancia	Vital

Código	RF-17
Descripción	Reducción de ventana
Resumen	El sistema reducirá al tamaño original la ventana de búsqueda tras haber realizado una filtración de datos.
Entrada	Tras pulsar volver
Proceso	Reduce la ventana de búsqueda
Salida	Pantalla estándar de búsqueda con los datos filtrados.
Importancia	Opcional

Código	RF-18
Descripción	Búsqueda Exacta
Resumen	El sistema permite buscar un término específico
Entrada	Término
Proceso	Búsqueda de término
Salida	Listado de todos los conceptos relacionados con el término.
Importancia	Vital.

Código	RF-19
Descripción	Búsqueda aproximada
Resumen	El sistema permite buscar un término por búsqueda aproximada
Entrada	Término.
Proceso	Búsqueda de término.
Salida	Listado de todos los conceptos relacionados con el término.
Importancia	Vital.

Código	RF-20
Descripción	Filtrado de búsqueda
Resumen	El sistema permite filtrar las búsquedas
Entrada	Filtros seleccionados desde listado
Proceso	Filtrado de búsqueda.
Salida	Listado de búsqueda con los filtros aplicados.
Importancia	Vital.

Código	RF-21
Descripción	Selección de concepto por filtración de término
Resumen	El sistema permite seleccionar un concepto relacionado con un término
Entrada	Selección de uno de los conceptos del listado
Proceso	Selección de concepto
Salida	Paso del concepto al portapapeles del escritorio
Importancia	Vital.

3.1.7.2.5. Gestión de Pantalla Sort

Código	RF-22
Descripción	Visualización de diagrama
Resumen	El sistema permite mostrar un diagrama de dependencia de Conceptos, donde un nodo hijo es el concepto, sus hijos serán los conceptos que dependen de el y sus padres de los que el depende.
Entrada	Un concepto existente en la base de datos desde el cuadro de diálogo de concepto
Proceso	Visualización de diagrama
Salida	Diagrama por pantalla
Importancia	Vital.

Código	RF-23
Descripción	Cambiar raíz del gráfico de dependencia
Resumen	El sistema permite seleccionar uno de los nodos hoja o padre que pasarán a ser en nodo raíz con sus correspondientes nodos hoja y padres.
Entrada	Selección de un objeto del diagrama y validación.
Proceso	Cambio de raíz.
Salida	Nuevo diagrama con un nuevo nodo raíz
Importancia	Vital.

Código	RF-24
Descripción	Borrado de nodos
Resumen	El sistema permite borrar los nodos seleccionados de un gráfico (tanto padres como hijos)
Entrada	Un gráfico con uno o varios nodos seleccionados.
Proceso	Borrado de nodos.
Salida	Nuevo diagrama sin los nodos que se han borrado y modificación en la base de datos
Importancia	Vital.

Código	RF-25
Descripción	Añadir nodos
Resumen	El sistema permite añadir nodos al grafico, para ello ha de tener varios seleccionados y sobre ellos añadirá uno que esté en el portapapeles. Los nodos seleccionados desaparecerán , pasando a ser hijos del nuevo nodo, y el nuevo nodo se insertará en el gráfico
Entrada	Varios nodos seleccionados sobre el gráfico y una entrada en el portapapeles
Proceso	Añadir nodos.
Salida	Nuevo diagrama con los nodos añadidos y modificación en la base de datos
Importancia	Vital.

Código	RF-26
Descripción	Cierre transitivo
Resumen	El sistema permite hacer un cierra transitivo sobre un nodo seleccionado, eliminado toda relación existente con otros nodos de forma transitiva.
Entrada	Selección de un nodo del gráfico
Proceso	Cierre transitivo
Salida	Eliminación de las entradas correspondientes en la base de datos
Importancia	Opcional

3.1.8. Validación de Requisitos

El resultado del trabajo realizado es una consecuencia de la ingeniería de requisitos (especificación del sistema e información relacionada) y es evaluada su calidad en la fase de validación. La validación de requisitos examina las especificaciones para asegurar que todos los requisitos del sistema han sido establecidos sin ambigüedad, sin inconsistencias, sin omisiones, que los errores detectados hayan sido corregidos, y que el resultado del trabajo se ajusta a los estándares establecidos para el proceso, el proyecto y el producto.

Para llevar a cabo dicha labor se establecen qué requisitos pueden ser evaluados y cuales no. Se establecerá el conjunto de pruebas que se realizarán para la comprobación de los mismos, así como su seguimiento, detección de errores y corrección de los mismos. Esta parte difiere de la verificación del sistema y sus comprobaciones son distintas.

Los requisitos no funcionales son más difíciles de verificar, debido a que muchos de ellos no son cuantificables ni medibles, no podemos decir si una herramienta es segura o amigable si no la podemos cuantificar y comparar con algo, estos requisitos serán evaluados con el cliente para que notifique su aprobación o descontento. Muchos de los requisitos pueden ser comprobados en conjunto con las mismas pruebas, e incluso es aconsejable hacer un seguimiento de su verificación en cada versión del sistema.

A parte de las pruebas de validación se establecen una serie de actuaciones de forma global para todos los requisitos.

- Reviews. Lectura de la documentación para ver la correcta interpretación. Más que una prueba como tal, esta forma de actuar servirá para unificar puntos de vista, y podrá ser utilizada para el caso de requisitos no medibles ni verificables.
- Inspecciones. Su finalidad es buscar defectos de usabilidad. A su vez son de dos tipos:
    - Conformidad. Verificar si se sigue lo establecido.
    - Consistencia. Se trata de ver si existen errores en lo formalizado

A continuación se establecen una serie de pruebas que se realizarán para controlar la exactitud de cada uno de los requisitos establecidos. La forma de reverenciarlos de aquí en adelante será mediante la siguiente nomenclatura: PR-xx, donde xx será el número de prueba.

- PR-01 Prototipo. Se caracteriza porque no tiene la totalidad de las funcionalidades, pero facilita hacerse una idea de su comportamiento y la corrección de posibles errores o mal interpretaciones de los requisitos. Sirve como partida para muchas de las validaciones. Habrán varios prototipos a o largo del ciclo de vida del producto. De la misma manera se harán prototipos de la base de datos y del editor.
- PR-02 Pruebas de ejecución. Consisten en la comprobación de funcionamiento de una funcionalidad específica. Consiste en la simulación detallada de una determinada funcionalidad.
- PR_03 Pruebas Visuales. Su finalidad es comprobar la correcta salida por pantalla ante una determinada entrada. Es una prueba visual que busca verificar la navegabilidad ante ciertos eventos, según los requisitos establecidos.
- PR-04 Verificación de formatos. Comprobación del correcto formato de elementos de la interface. Prueba visual que busca comprobar la existencia de los elementos establecidos y según las características previstas.
- PR-05 Comprobaciones de Entrada de la base de datos. Se finalidad es comprobar si se produce de forma correcta la creación de nuevos datos en la base de datos. Se extraerán los datos creados por el sistema de forma manual desde la base de datos y se comprobará su correcta ubicación y formato.
- PR-06 Comprobaciones de Salida de la base de datos. Se finalidad es comprobar si se produce de forma correcta la extracción de datos desde la base de datos. Se formularán consultas en la base de datos simulando las salidas esperadas, y comparándolas con las producidas por el sistema.
- PR-07 Simulación de situación. Ciertas comprobaciones conllevan la simulación de un conjunto de tareas entrelazadas para poder comprobar su correcta reproducción. Se procederá

a reproducir la situación y la correcta navegabilidad por cada paso, así como la consistencia de los estados tanto iniciales, intermedios y finales.

- PR-08 Control de ventanas, objetos existentes, instalación (aplicada a control de documentación también). Se ha de verificar qué objetos se mantienen para un determinado requisito. Para ello se pueden comprobar desde el administrador de tareas que procesos están activos.
- PR-09 Carga de datos. Se establecen cargas de datos de distinto formato para comprobar si es el correcto.

3.2. Estudio de viabilidad

En esta parte se estudia la viabilidad del sistema a desarrollar, mostrando el estado actual del sistema y posteriormente analizando sus deficiencias. A continuación se confecciona un catálogo de requisitos, apoyados por la especificación de requisitos previa, y se plantean diversas alternativas para solucionar el problema planteado. Finalmente, se elige de manera razonada una de las soluciones propuestas.

Mientras que el Plan de Sistemas de Información tiene como objetivo proporcionar un marco estratégico que sirva de referencia para los Sistemas de Información de un ámbito concreto de una organización, el objetivo del Estudio de Viabilidad del Sistema es el análisis de un conjunto concreto de necesidades para proponer una solución a corto plazo, que tenga en cuenta restricciones económicas, técnicas, legales y operativas. La solución obtenida como resultado del estudio puede ser la definición de uno o varios proyectos que afecten a uno o varios sistemas de información ya existentes o nuevos. Para ello, se estudian los requisitos que se ha de satisfacer y se estudia, si procede, la situación actual.

3.2.1. Estudio de la Solicitud

Se realiza una descripción general de la necesidad planteada por el usuario, y se estudian las posibles restricciones de carácter económico, técnico, operativo y legal que puedan afectar al sistema.

La proliferación de estudios de nuevas formas de comunicación y herramientas de trabajo con ontologías, juegan un importante papel a la hora del intercambio de conocimiento. Se requieren herramientas que comuniquen el lenguaje natural utilizado por los usuarios con la representación lógica de las ontologías. Hay una necesidad creciente de emular el comportamiento humano para facilitar la comunicación hombre-máquina. Se pretende crear una ontología genérica, meta ontología, que permita identificar información en cualquier ámbito, así como su correcto uso y generar nuevo conocimiento, simulando el comportamiento humano. Una vez identificados los objetos y relaciones

que existen para crear esta ontología, surge un problema para poder poblarla, es decir crear una base de datos lo suficientemente amplia para poder realizar pruebas suficientes. El léxico castellano es muy amplio, a la vez que tiene multitud de elementos combinaciones entre ellos, al mismo tiempo la afinidad entre términos y los roles distinto que pueden desempeñar objetos concretos en entornos distintos. Para poder poblar esta ontología se requiere de una interfaz, clara y sencilla, que permita agilizar este trabajo, que de manera manual es bastante tediosa, y cabe la posibilidad de que se produzcan errores durante el proceso.

Continuos estudios buscan una estandarización de las ontologías aplicable a cualquier entorno, basada en un conjunto de relaciones entre objetos reutilizables, y la creación de un sistema capaz de poder poblar y proporcionar riqueza a la ontología.

El sistema planteado requiere de ayuda externa para la catalogación e identificación de significados según contextos distintos. Estas ayudas externas deben ser de libre uso, si es posible, y además han de poder soportar trabajar con volúmenes de información considerables. Se busca que el sistema pueda ser utilizado para que se aprenda de las conductas y comportamientos humanos, para extraer la información requerida.

3.2.2. Identificación del Alcance del Sistema

En esta sección se estudia la necesidad planteada para el presente proyecto, concretamente en dotar a la aplicación APO de un interfaz gráfico intuitivo, funcional y portable para facilitar su utilización por parte del usuario. Su finalidad, un interfaz capaz de poblar cualquier ontología tanto a través de ficheros generados, como de forma manual, así como las modificaciones y eliminación de relaciones. En una primera fase únicamente se busca poder poblarla de forma manual. Pero no deja de lado el hecho de la entrada de ficheros comentada, esta parte se considera como trabajo futuro al igual que el desarrollo completo de la herramienta, ya que un primer acercamiento las posibilidades del sistema son muy amplias para comprimirlas en un único proyecto.

3.2.3. Descripción General del Sistema

En este apartado se pasa a describir el contexto en el que va a operar el interfaz requerido y una descripción general de la aplicación APO.

3.2.4. Contexto

El presente proyecto se encuadra dentro del marco de la investigación, la manipulación de bases de datos y un ámbito lingüístico, siendo de gran carga la fase de investigación con respecto al

resto. Se trata de una herramienta creada sobre una estructura de reglas Prolog, pero que ha de poder trabajar con bases de datos en que soporten sentencias SQL, como Oracle, y montado todo sobre un interfaz creada en Java. Esta interfaz es la que ha de poder extraer la información necesaria para poblar nuestra ontología. La entrada ha de poder se manual también, para así poder contrastar datos, especificar relaciones y poder realizar los cambios que sean oportunos.

La población de la ontología en un futuro próximo se podrá hacer gracias a una aplicación en proceso de desarrollo, Cognos, herramienta que permite identificar roles en un texto. Posteriormente esta información sería recibida por nuestro sistema, que se encarga de establecer las relaciones oportunas con el resto de elementos de nuestra base de datos, así como el grado de afinidad con unos u otros términos. Esta entrada ha de estar en un formato estándar para mayor afinidad, por lo que o es transformada antes de su entrada o ya está estandarizada (es algo que no esta aún definido). Se pretende utilizar un estándar del tipo DAM-LR **[25] y [26],** no se descarta la posibilidad de un formato XML debido a lo extendido que se encuentra hoy día, en caso de que la herramienta termine teniendo un uso Web seria más adecuado emplear OWL, pero estas suposiciones escapan del alcance actual del proyecto .

3.2.5. Especificación del Alcance del Sistema

Estos objetivos dan una visión global de lo que se pretende con el presente proyecto.

- En primer lugar se debe permitir ejecutar la aplicación desde el interfaz con los parámetros deseados.
- El interfaz debe permitir configurar un nuevo sistema para el análisis de manera cómoda y clara.
- Se debe mantener actualizada la información sobre el progreso del análisis en una representación gráfica intuitiva.
- El análisis debe poder detenerse en cualquier momento.
- El sistema ha de poder permitir comprobar las estructuras introducidas.
- El sistema ha de poder permitir consultar las estructuras creadas.
- El sistema ha de permitir cambiar roles y relaciones entre objetos.
- El sistema ha de ser capaz de extraer y generar nueva información.
- No existirá duplicidad de datos.
- Ha de ser un sistema integro, completo y seguro.

- Se debe aplicar un plan de calidad continuo en su desarrollo, aunque no se haya especificado.
- Será un interfaz predecible y amigable.
- Debe poder ser ampliada o modificada bien por módulos o Plug-ins nuevos.

3.2.6. Definición de requisitos del sistema

A continuación se especifican unos requisitos preliminares acerca de la aplicación a desarrollar. Estos requisitos indicarán los aspectos que el sistema debe cumplir obligatoriamente.

3.2.6.1. Requisitos funcionales

Además de los requisitos funcionales aparecidos en el estudio de requisitos, se han de considerar otros que o bien no se han dejado claros, o bien se consideran inherentes al propio proyecto, para garantizar la calidad del producto, entendiendo calidad como… "*cumplimiento de los requisitos deseados por el cliente*". Estos requisitos son de un carácter más abstracto y generales que de los específicos vistos en el estudio de requisitos. Para seguir la normativa antes utilizada serán numerados siguiendo el mismo esquema de nombrado y numeración:

- RF-27 Creación de una relación de objetos de la base de datos desde la misma interfaz.
- RF-28 Capacidad para cargar ficheros de texto.
- RF-29 Opción de configurar los parámetros de ejecución.
- RF-30 Almacenamiento de los parámetros para próximas ejecuciones.
- RF-31 Posibilidad de ejecutar la aplicación con los parámetros deseados.
- RF-32 Posibilidad de detener la aplicación.
- RF-33 Información acerca de la tarea en curso.
- RF-34 Notificación de errores.
- RF-35 Menús claros y detallados.
- RF-36 Nombre de las etiquetas claras y sencillas.

3.2.6.2. Requisitos no funcionales

Para denominar estos requisitos la sintaxis será RNF-xx, siendo xx el número de orden. Estos requisitos son recopilación de los aparecidos en el Estudio de Requisitos y otros que han aparecido tras el visto bueno de la Especificación de Requisitos.

3.2.6.2.1. Rendimiento

- RNF-1 Gran capacidad de almacenamiento, debido a la riqueza del léxico castellano (también aplicable a otros léxicos).
- RNF-2 La herramienta deba ser capaz de soportar la carga de gran número de datos.
- RNF-3 Las cargas han de ser rápidas, evitando los accesos innecesarios a memoria.
- RNF-4 Diseño de pantallas por ventanas, similar a Windows o S.O Linux de interfaz gráfica.
- RNF-5 Ventanas intuitivas.
- RNF-6 Similitud de formato entre ventanas, así como tablas etc.
- RNF-7 Fácil de actualizar.
- RNF-8 Eficiente (aunque no es requisito establecido por cliente)
- RNF-9 Requisitos de interfaz de usuario de la especificación de requisitos.

No se especificaron más detalles.

3.2.6.2.2. Frecuencia de tratamiento

- RNF-10 Frecuencia de actualización del análisis adecuada con la evolución del proceso.

No se especificaron más detalles.

3.2.6.2.3. Requisitos de seguridad

- RNF-11 Confidencialidad de claves de usuario.
- RNF-12 Accesibilidad al sistema.
- RNF-13 Accesibilidad a la base de datos.
- RNF-14 Información fiable.

- RNF-15 Información completa.
- RNF-16 No duplicidades en los datos.

3.2.6.2.4. Requisitos especiales

- RNF-17 La aplicación debe ser portable.
- RNF-18 Se ha de documentar de forma clara y concisa.
- RNF-19 Se establecerán estándares de datos y formatos.
- RNF-20 Sistema Modularizado.
- RNF-21 Ampliable y adaptable.

3.2.6.2.5. Requisitos relativos al entorno tecnológico

Desde le punto de vista del hardware se tiene los siguientes requisitos:

- RNF-22 Al menos 2 GB de almacenaje necesarios para Oracle.
- RNF-23 Se recomienda un procesador Pentium IV para un correcto desempeño.
- RNF-24 Se recomienda una memoria de 126-258 megas de RAM.

Desde el punto de vista del Software:

- RNF-25 Entorno Java para su elaboración y máquina virtual Java.
- RNF-26 Sistema operativo Unix, Linux, W2000, NT, XP, OpenVMS o W2003.Que son soportados por Oracle. Preferible Windows
- RNF-27 Oracle 9.0 o posterior (recomendamos 10g)
- RNF-28 Herramienta Prolog (aunque no es necesaria).

3.2.6.3. Prioridad de los requisitos

Las prioridades de requisitos se valorarán de 1-5, siendo el máximo valor 1 y el mínimo 5, a la hora de priorizarlas. Aquellos requisitos opcionales o no previstos en el origen se marcarán con Op+N donde N representaría su prioridad como en los no opcionales. Se muestra el identificador del requisito, la prioridad en una escala de 1 a 5 (en caso de ser opcional se marca como Op) y una breve descripción.

REQUISITO	PRIORIDAD	DESCRIPCIÓN
RF-1	4	Autenticación correcta de usuarios.
RF-2	5	Aviso de autenticación incorrecta.
RF-3	Op-4	Enmascaramiento de la contraseña.
RF-4	1	Despliegue del menú y submenús desde todas las pantallas.
RF-5	1	Gestión de pestañas desplegables para ver las funciones ontológicas.
RF-6	4	Generación de identificador de concepto
RF-7	5	Selección de un elemento del portapapeles.
RF-8	3	Alta de Concepto nuevo.
RF-9	3	Asignación de términos a un concepto.
RF-10	4	Modificar términos de un concepto.
RF-11	5	Gestión de los favoritos de un concepto.
RF-12	3	Control de la creación de un nuevo concepto.
RF-13	2	Listar términos relacionados con un concepto.
RF-14	2	Filtrar términos asociados a un concepto.
RF-15	Op-1	Cierre de la ventana de búsqueda al pulsar salir.
RF-16	2	Expansión de ventana de búsqueda para filtrar.
RF-17	Op-2	Reducción de la ventana de búsqueda tras selección de filtros.
RF-18	2	Búsqueda exacta de un término.
RF-19	2	Búsqueda aproximada de un término.
RF-20	2	Filtrado de búsqueda.

RF-21	3	Selección de concepto tras búsqueda de conceptos relacionados con un término
RF-22	Op-2	Visualización de diagrama de dependencia de conceptos.
RF-23	5	Cambiar raíz del gráfico de dependencia al seleccionar otro nodo.
RF-24	5	Borrado de nodos seleccionados.
RF-25	5	Añadir nodos al gráfico.
RF-26	Op-5	Cierre transitivo, eliminando todas las relaciones de un nodo.
RF-27	2	Creación de relaciones de objetos desde la interface.
RF-28	Op-5	Capacidad de cargar datos mediante ficheros de texto.
RF-29	Op-4	Capacidad de configurar los parámetros de ejecución
RF-30	1	Almacenamiento de parámetros para futuras utilizaciones.
RF-31	1	Posibilidad de ejecutar la aplicación con los parámetros deseados.
RF-32	Op-1	Posibilidad de detener la aplicación.
RF-33	4	Información sobre la tarea en curso
RF-34	Op-1	Notificación de errores.
RF-35	1	Menús claros y detallados.
RF-36	1	Etiquetas claras y sencillas
RNF-1	1	Gran capacidad de almacenamiento.
RNF-2	3	Se ha de poder soportar gran número de datos.
RNF-3	1	Cargas rápidas evitando accesos innecesarios a memoria.
RNF-4	1	Presentación de la interface por ventanas, similar a Windows u otros sistemas operativos de inteface gráfica.
RNF-5	1	Ventanas intuitivas.

RNF-6	1	Similitud del diseño de ventanas, botones etc.
RNF-7	2	Fácil de actualizar.
RNF-8	4	Eficiente.
RNF-9	1	Requisitos establecidos para la interfaz de usuario en la especificación de requisitos (RUS-1 al RUS-8)
RNF-10	1	Frecuencia de actualización del análisis adecuada con la evolución del proceso.
RNF-11	Op-2	Confidencialidad de claves de usuario.
RNF-12	2	Accesibilidad al sistema.
RNF-13	1	Accesibilidad a la base de datos.
RNF-14	1	Información fiable.
RNF-15	1	Información completa.
RNF-16	1	No duplicación de datos.
RNF-17	5	Aplicación portable.
RNF-18	1	Documentación clara y concisa durante todo el ciclo.
RNF-19	Op-2	Se establecerán estándares en datos y formatos.
RNF-20	1	Sistema modularizado.
RNF-21	Op-4	Ampliable y adaptable
RNF-22	1	Al menos 2 GB de almacenaje, necesarios para Oracle.
RNF-23	Op-1	Al menos un Pentium IV
RNF-24	1	Recomendación de 128 a 256 de memoria RAM.
RNF-25	1	Instalación de entorno Java y máquina virtual Java para implementación y desarrollo de la Herramienta.

RNF-26	1	Sistema operativo que soporte Oracle.
RNF-27	1	Versión de Oracle mínimo 9.0.
RNF-28	Op-3	Herramienta Prolog

Tabla 7 Tabla Prioridad de Requisitos

3.2.7. Estudio de la Situación Actual

La situación actual es el estado en el que se encuentran los sistemas de información existentes en el momento en el que se inicia su estudio. Teniendo en cuenta el objetivo del estudio de la situación actual, se realiza una valoración de la información existente acerca de los sistemas de información afectados. En función de dicha valoración, se especifica el nivel de detalle con que se debe llevar a cabo el estudio.

3.2.8. Valoración del estudio de la situación actual

En la fase actual el sistema se encuentra en los preliminares de codificación. Se establecerán en Prolog las instrucciones necesarias para identificar las relaciones de objetos. De la misma forma se establecerá el diseño preliminar de diagramas de clases del sistema, en el que se identificarán los objetos y sus relaciones correspondientes. Por otra parte la herramienta Cognos está en proceso de desarrollo, pero no afecta al desempeño del resto de actividades, ya que las primeras poblaciones de la ontología se podrían hacer manualmente, probando una vez terminada la herramienta con volúmenes de datos mayores tras las adaptaciones oportunas para soportar dicha carga.

A fecha de hoy quedan por definir algunas especificaciones de la herramienta, así como posible mejoras, nos encontramos en un proceso evolutivo, y de retroalimentación.

3.2.9. Identificación de los Usuarios Participantes

En función del nivel de detalle establecido para el estudio de la situación actual, se identifican los usuarios participantes de cada una de las unidades organizativas afectadas por dicho estudio. Se informa a los usuarios identificados como implicados en el Estudio de la Situación Actual, se solicita su aceptación y se espera su confirmación.

No aplicable.

3.2.10. Descripción de los Sistemas de Información Existentes

En esta tarea se describen los sistemas de información existentes afectados, según el alcance y nivel de detalle establecido en la tarea Valoración del Estudio de la Situación Actual (EVS 4.1), mediante sesiones de trabajo con los usuarios designados para este estudio.

No aplicable.

3.2.11. Realización del Diagnóstico de la Situación Actual

Con el fin de elaborar el diagnóstico de la situación actual se analiza la información de los sistemas de información existentes, y se identifican problemas, deficiencias y mejoras.

Nos encontramos ante el problema de coordinar varios lenguajes, varios formatos de bases de datos y la búsqueda de una estandarización para datos de entrada y salida. Partiendo del diagrama de clases, adaptando el módulo generado con Prolog, y estableciendo correctamente la interfaz de comunicación entre los distintos componentes, podremos solucionar muchos de los problemas que se plantean. A parte de eso debido a la capacidad de información que se ha de almacenar, la opción de poblarlas manualmente se presenta tediosa, por lo que para utilizar correctamente las salidas generadas por Cognos, se ha de llegar a un consenso, para acordar una metodología común o modelo de estandarización. En caso de no ser posible la disponibilidad de Cognos, se plantea la posibilidad de permitir la entrada de otros tipos de fichero, y debido a su extensión podríamos considerar formatos XML.

3.2.12. Estudio de Alternativas de Solución

En función de los requisitos planteados y el estudio de la situación actual, se proponen diversas alternativas de solución. Sobre ellas se valorará cuál es la solución óptima, en base al impacto de carácter tecnológico y de operación que cada medida supone.

3.2.12.1. Alternativas de lenguaje de programación.

Se describen a continuación las alternativas de solución propuestas para el presente proyecto. Los requerimientos no se han especificado de manera muy concreta, el único aspecto en el que pueden aparecer diferentes alternativas es en el lenguaje de programación en el que se va a implementar el interfaz gráfico para la aplicación APO. En primera instancia se ha plantea utilizar

Java, debido a que gran parte de las herramientas para crear bases de datos con Oracle y trabajar en Prolog, tienen interfaces que soportan Java. Aún así se han valorado otras alternativas.

Alternativas estudiadas:

- Java
- C++
- Phyton
- C#

3.2.12.2. Valoración de Alternativas de lenguaje de programación.

A continuación se evalúan las alternativas descritas en el apartado anterior:

- Java es portable, interpretado y orientado a objetos. Es muy sencillo, ya que omite muchas características complejas como los punteros y la liberación de memoria. Lleva mucho tiempo en circulación y tiene mucha documentación y librerías ya escritas. Sun proporciona una amplia documentación sobre todos los paquetes Java, además de tutoriales gratuitos. Además dispone de los API Swing y AWT que facilitan la creación de interfaces gráficas, así como la creación de applets para dar carácter Web a las creaciones.
- C++ no es portable, y sí orientado a objetos. Existe código y documentación abundante, pero su programación es mucho más compleja, ya que incorpora punteros, referencias, redefinición de tipos, macros y la necesidad de liberar memoria. Al no ser portable habría que realizar dos versiones compiladas, una para Windows y otra para Linux.
- Phyton es un lenguaje portable, interpretado y orientado a objetos, pero mucho menos potente y extendido que Java y C++. No esta pensado para usarse en grandes proyectos y se pueden tener problemas a la hora de buscar documentación o código fuente.
- C# es un lenguaje orientado a objetos, basado en C++, por lo que no portable. Simplifica el uso de algunas características de C++ como la inclusión de un liberador de memoria transparente al programador. El problema es que es muy reciente (2001), y apenas hay documentación ni tutórales sobre un uso avanzado del mismo. Existiría el mismo problema que con C++ respecto a su nula portabilidad.
- Se podrían considerar otros tipos de lenguajes no orientados a objetos, pero supondrían una gran dificultad a la hora de establecer la gran cantidad de interrelaciones que se han de crear.

3.2.12.3. Alternativa de lenguaje de programación.

Seguidamente se presenta la solución adoptada para la resolución del problema planteado. Para ello, se han estudiado, previamente a la resolución final, las valoraciones de las distintas alternativas posibles. El factor determinante en dicha selección ha sido la evaluación de prestaciones, la flexibilidad del lenguaje y el cumplimiento por completo de los requisitos funcionales y no funcionales del sistema.

3.2.12.3.1. Alternativa escogida del lenguaje.

La alternativa escogida finalmente ha sido la Alternativa 1: Diseño de un interfaz gráfico en Java.

3.2.12.3.2. Características de la alternativa escogida.

- Portabilidad: más allá de la portabilidad por ser de arquitectura independiente, Java implementa otros estándares de portabilidad para facilitar el desarrollo. Por ejemplo, los enteros son siempre enteros de 32 bits en complemento a 2. Además, Java construye sus interfaces de usuario a través de un sistema abstracto de ventanas de forma que éstas puedan ser implantadas en entornos Windows, o Linux si se necesitase modelizar en un futuro, sin olvidar las funciones de seguridad y transformación en applets que proporciona
- Orientado a Objetos: Java implementa la tecnología básica de C++ con algunas mejoras y elimina aspectos concretos para mantener la simplicidad del lenguaje. Java trabaja con sus datos como objetos y con interfaces a esos objetos. Soporta las tres características propias del paradigma de la orientación a objetos: encapsulación, herencia y polimorfismo. Las plantillas de objetos son llamadas, como en C++ clases, y sus copias, instancias. Estas instancias necesitan ser construidas y destruidas en espacios de memoria
- Simplicidad: Java ofrece toda la funcionalidad de un lenguaje potente, pero sin las características menos usadas y más confusas de éstos. Elimina muchas de las características de otros lenguajes como C++, para mantener reducidas las especificaciones del lenguaje y añadir características muy útiles como el “garbage collector”. No es necesario liberar memoria y se limita la fragmentación de la memoria. Se han eliminado además la aritmética de punteros, las referencias, los registros, la definición de tipos y las macros.

- Interpretado: El intérprete Java puede ejecutar directamente el código objeto. Enlazar un programa consume menos recursos que compilarlo. A pesar de que obviamente Java es más lento que otros lenguajes de programación ejecutados, la situación se acerca mucho a la de programas compilados, sobre todo en lo que a la rapidez en la ejecución de código se refiere.
- Multitarea: Java permite realizar muchas actividades simultáneas en un programa .

Una vez elegido el lenguaje de programación a explotar, debemos concretar qué entorno de desarrollo se aprovechará. Se buscará uno de libre uso y con la opción de poder trabajar tanto por línea de comandos como por ventanas. Esta fase de búsqueda está aún en proceso.

3.2.12.4. Alternativas para la interface de comunicación

Nos encontrábamos ante un problema al tener que trabajar con varios entornos distintos. Se requería la búsqueda de un entorno a ser posible de libre acceso, que proporcionara una interfaz capaz de enlazar Prolog y Oracle, ya que Prolog tenía limitaciones de memoria para el alcance de nuestro proyecto. Además esta comunicación debía ser intuitiva, fácil de implementar y consistente. Se evaluaron diversos entornos y sistemas:

- Ciao Prolog.
- Sistemas Datalog.
- Quintus Prolog
- Swi Prolog
- DES
- Visual Prolog

3.2.12.5. Valoración de las Alternativas de interfaces de comunicación

Tras descartar Ciao Prolog, debido a que no trabajaba con Oracle, fueron multitud de herramientas las analizadas, tanto en requisitos funcionales con respecto a nuestro trabajo, interfaces de comunicación, licencias etc. Finalmente encontramos DES, que se ajustaba en gran parte a los requisitos establecidos. Posteriormente descubrimos que tenía ciertas limitaciones que no hacía viable la selección, y que detallo a continuación:

DES Está implementado en Prolog **[27],** incorpora recursión no lineal (a diferencia de SQL) y negación estratificada. Se puede usar tanto desde un intérprete Prolog (por lo tanto, sobre cualquier

plataforma hardware/software que admita tal intérprete) como desde ejecutables para Windows, Linux y Unix. Aquí es donde encontramos las primeras limitaciones, por un lado si utilizamos un interprete Prolog hemos de regirnos por las limitaciones del mismo, y en caso de realizarse desde el ejecutable, las limitaciones serán las del mismo (el problema que en la documentación del DES, al menos en la versión de Windows, no especifican requerimientos ni restricciones de ningún tipo).

El sistema se distribuye desde Sourceforge bajo licencia GPL, y su código es abierto y gratuito.

Los datos se almacenan en una base de datos Datalog, incluidos los hechos y las normas, y todas las consultas.

- Números. Permite trabajar con enteros y flot, con las limitaciones que tenga la plataforma Prolog que se utilice.
- Constantes alfanuméricas. Cualquier secuencia de caracteres alfanuméricos (incluyendo el guión de subrayado _), comenzando con una letra minúscula o cualquier secuencia de caracteres alfanuméricos y espacios delimitados por una sola comilla.
- La negación de extractos no garantiza la exactitud de una consulta.

Se aprecian además una serie de limitaciones con respecto a SQL:

- Untyped data system. Table columns are untyped and no data type declaration is needed when creating a table.
- Set-oriented (in contrast to multiset) answers. A select all statement performs equal to a select distinct one, even when duplicates occur.
- No aggregates. There is no provision of aggregate functions (as min, max, and avg) as well as grouping (group by clause).
- No null values. The underlying logic system is two valued.
- No constraints. There is no way to impose database constraints as primary keys, referential integrity constraints, to name a few.
- No provision for ordering results (order by clause)
- No insertions/deletions/updates into views.
- No syntax error reports. The parser does not inform about the possible syntax error
- Non-linear recursive queries (using the standard syntax).

DES trabaja con una base de datos deductiva. El modelo usado para las bases de datos deductivas está estrechamente relacionado con el modelo de datos relacional y en particular con el

formalismo de cálculo relacional de dominios. Está también relacionado con el lenguaje Prolog y, principalmente, con el lenguaje Datalog, un subconjunto de Prolog que evita el uso de estructuras potencialmente infinitas.

Las ventajas que tiene Des y que han de mencionarse son las siguientes:

- Sistema multiplataforma. Se ejecuta en la mayoría de los sistemas operativos actuales, bien sea a partir de su código fuente Prolog o mediante los ejecutables proporcionados.
- Recursión no lineal. Admite múltiples llamadas recursivas en el cuerpo de una regla, incluyendo llamadas recursivas con negación.
- Negación estratificada. La negación estratificada significa *grosso modo* que un objetivo negado no puede llamarse a sí mismo en ningún camino de cómputo con objeto de asegurar la corrección de la respuesta y así evitar situaciones como la conocida paradoja de Russell. La negación puede aparecer en uno o varios literales negados del cuerpo de las reglas.
- El sistema implementa un algoritmo ascendente guiado por el objetivo y con cómputos descendentes que calcula de forma segura las respuestas.
- Vistas transitorias. Permiten emitir consultas conjuntivas al vuelo, es decir, desde el inductor de comandos:

Tiene una serie de limitaciones que son expuestas a continuación:

- Agregados. El sistema no dispone de funciones de agregación, que se demuestran muy útiles en la práctica. Su implementación no es sencilla, aunque está prevista.
- Aritmética. Aún no se ha implementado la posibilidad de incluir primitivas aritméticas.
- La interfaz de usuario del sistema es un intérprete de comandos que acepta tanto consultas Datalog como comandos del sistema. Sin embargo, se debería ampliar esta interfaz a una interfaz gráfica de usuario.
- Depuración. Actualmente no hay herramientas para la depuración de programas Datalog.
- Incapacidad de representar términos compuestos. Los términos compuestos pueden introducir problemas de no terminación. La terminación es una característica que se exige *de facto* a los sistemas de bases de datos, y se podría asegurar limitando la profundidad de los términos a una cota finita. Una correspondencia entre constantes y un conjunto finito de términos construidos podría dar lugar a una implementación sencilla.
- Dado que Datalog se derivó de Prolog, se han adoptado prácticamente todos los convenios sintácticos de Prolog para la escritura de programas Datalog. Sin embargo, los comandos son diferentes a los que están acostumbrados los programadores de Prolog.

El sistema se puede usar a partir de su código fuente Prolog sobre cualquiera de los siguientes sistemas Prolog: GNU Prolog, Ciao Prolog, Sicstus Prolog y SWI Prolog. No obstante, el método recomendado es instalar uno de los ejecutables para Windows, Linux o Unix.

El modo habitual de uso del sistema es crear mediante cualquier editor de texto en el sistema operativo un fichero con extensión .dl para guardar el programa Datalog que se desarrolle.

Una consulta es el nombre de una relación del programa con tantos argumentos como la aridad de la relación y que se escribe en el inductor DES. Cada uno de ellos puede ser una variable o una constante, pero no un término compuesto. No se pueden escribir consultas conjuntivas, como a(X), b(X) para calcular la intersección de ambas relaciones. Sin embargo, esta situación se puede abordar desde dos puntos de vista: (1) escribir en el programa una regla r(X):-a(X), b(X) (o insertarla con el programa cargado mediante el comando /assert) y emitir la consulta r(X), y (2) usar vistas transitorias.

En este sistema se ha introducido un concepto nuevo en los sistemas de consulta a bases de datos deductivas, las vistas transitorias, que permiten escribir consultas conjuntivas al vuelo. Una vista transitoria es una regla que se añade a la base de datos y su cabeza (parte izquierda) se considera como una consulta que se ejecuta. Después de su evaluación, la regla se elimina de la base de datos. Las vistas transitorias son útiles para emitir rápidamente consultas conjuntivas.

Las reglas Datalog son similares a las cláusulas Prolog con las mismas limitaciones indicadas para las consultas.

A diferencia de muchas implementaciones de Datalog, que usan una evaluación ascendente (*bottom-up*), este intérprete usa una evaluación ascendente guiada por una descendente (*topdown*). En lugar de calcular el significado del programa completo, calcula el significado de la consulta emitida usando tablas de extensión como técnica de optimización.

Todos los operadores son infijos salvo la negación. Los infijos se deben entender como tests en lugar de relaciones para asegurar finitud en la respuesta. Esto significa que la lectura declarativa de las reglas con estos operadores no es clara. No se aseguran respuestas correctas para los programas que usen primitivas sin sus argumentos cerrados (i.e., sin variables) dado que en general el orden de los objetivos que incluyan estos operadores afecta a la semántica. Es una consecuencia de las reglas inseguras, las cuales se pueden caracterizar sintácticamente aunque la versión actual del sistema no advierte de su presencia.

Hay algunas limitaciones más como que no permite ordenar los resultados, no se muestran los errores de sintaxis etc., pero estos detalles se pueden ver en su manual.

3.2.12.6. Descripción de la alternativa seleccionada para el entorno de trabajo

Finalmente no se ha terminado de seleccionar una alternativa para este punto, bien por la complejidad de las herramientas, por las licencias o los complejos manuales. Las opciones de elegir DES como herramienta de implementación se vio truncada debido:

- DES es un sistema sin tipos declarados al igual que la mayoría de los sistemas Prolog. Sin embargo, trabaja con tipos de datos numéricos (enteros y reales) y átomos (que podrían interpretarse como cadenas de caracteres). Estos tipos dependen estrechamente del sistema Prolog sobre el que se ejecute (por ejemplo, en cuanto a la precisión de los reales). Los ejecutables de Windows, Linux y Unix se han compilado con Sicstus Prolog y heredan sus características y limitaciones.
- En cuanto al tamaño de las "tablas" (en este caso predicados que definen relaciones), depende también del sistema Prolog sobre el que se ejecute. DES es una base de datos en memoria (es decir, no ofrece acceso persistente como un SGBD relacional - Oracle, Access,...) y la limitación de su tamaño la impone el espacio de direccionamiento del sistema operativo. Hay que tener en cuenta que este sistema guarda en memoria las reglas y hechos consultados, así como la información inferida (tanto explícita como implícita), lo que supone que de la información explícita se mantienen, a lo sumo, dos copias.
- DES está orientado a la educación y no ha sido objeto de un estudio de su alcance en cuanto a rendimiento y espacio. No obstante, hacer las pruebas de carga para un sistema en concreto no sería difícil. Podríamos generar mediante programación un fichero de texto con tantos hechos Datalog como consideremos necesarios, y a partir de ahí consultarlo y emitir consultas.

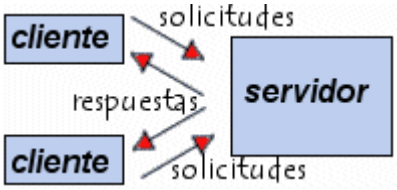
3.2.12.7. Elección del Software y el Hardware.

Para desarrollar la herramienta ontológica requerimos la utilización de un sistema cliente/servidor.

En el modelo cliente/servidor, cuando un proceso desea un servicio que proporciona otro proceso, le envía un mensaje solicitando ese servicio: una petición. El proceso que cumple el servicio se llama servidor y el solicitante se llama cliente (ver figura 3.1).

En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión

de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.



3.1 Cliente Servidor

Estos sistemas tienen una serie de ventajas:

- Centralización del control: los accesos, recursos y la integridad de los datos son controlados por el servidor de forma que un programa cliente defectuoso o no autorizado no pueda dañar el sistema. Esta centralización también facilita la tarea de poner al día datos u otros recursos (mejor que en las redes P2P).
- Escalabilidad: se puede aumentar la capacidad de clientes y servidores por separado. Cualquier elemento puede ser aumentado (o mejorado) en cualquier momento, o se pueden añadir nuevos nodos a la red (clientes y/o servidores).
- Fácil mantenimiento: al estar distribuidas las funciones y responsabilidades entre varios ordenadores independientes, es posible reemplazar, reparar, actualizar, o incluso trasladar un servidor, mientras que sus clientes no se verán afectados por ese cambio (o se afectarán mínimamente). Esta independencia de los cambios también se conoce como encapsulación.
- Existen tecnologías, suficientemente desarrolladas, diseñadas para el paradigma de C/S que aseguran la seguridad en las transacciones, la amigabilidad del interfaz, y la facilidad de empleo.

Pero también tiene una serie de desventajas que vemos a continuación:

- La congestión del tráfico ha sido siempre un problema en el paradigma de C/S. Cuando una gran cantidad de clientes envían peticiones simultaneas al mismo servidor, puede ser que cause muchos problemas para éste (a mayor número de clientes, más problemas para el servidor). Al contrario, en las redes P2P como cada nodo en la red hace también de servidor, cuantos más nodos hay, mejor es el ancho de banda que se tiene.
- El paradigma de C/S clásico no tiene la robustez de una red P2P. Cuando un servidor está *caído*, las peticiones de los clientes no pueden ser satisfechas. En la mayor parte de redes P2P, los recursos están generalmente distribuidos en varios nodos de la red. Aunque algunos

salgan o abandonen la descarga; otros pueden todavía acabar de descargar consiguiendo datos del resto de los nodos en la red.

- El software y el hardware de un servidor son generalmente muy determinantes. Un hardware regular de un ordenador personal puede no poder servir a cierta cantidad de clientes. Normalmente se necesita software y hardware específico, sobre todo en el lado del servidor, para satisfacer el trabajo. Por supuesto, esto aumentará el coste.
- El cliente no dispone de los recursos que puedan existir en el servidor. Por ejemplo, si la aplicación es una Web, no podemos escribir en el disco duro del cliente o imprimir directamente sobre las impresoras sin sacar antes la ventana previa de impresión de los navegadores.

El Servidor será en este caso de Base de Datos (Database Server), instalada en un equipo y el cliente, el editor ontológico, que accede a los servicios que ofrece el servidor (la base de datos).

Para nuestro caso cliente y servidor están en la misma máquina, y para ello necesita de unos requisitos.

Se trabaja con un equipo en el que hay instalado Windows XP (para otras versiones u otros sistemas operativos debería verificarse los requisitos necesarios y compatibilidades). Se decidió trabajar con Oracle Database 10 g Express Edition. Por el conocimiento de Oracle y su entorno, además de que dispone de su propio driver para trabajar con Java y se puede utilizar en Eclipse (herramienta utilizada para la codificación del código del editor)

Oracle Database 10g Express Edition (Oracle Database XE) es una base de datos de entrada de footprint pequeño, creada sobre la base de código Oracle Database 10g Release 2 que puede desarrollarse, implementarse y distribuirse sin cargo; es fácil de descargar y fácil de administrar. Permite trabajar con Java.

Oracle Database XE puede instalarse en máquinas host de cualquier tamaño con cualquier cantidad de CPUs (una base de datos por máquina), no obstante XE almacenará hasta 4GB de datos de usuarios, utilizará hasta 1GB de memoria, y utilizará una sola CPU en la máquina host.

Para ejecutar el programa se requiere un máquina virtual instalada en el equipo (hay múltiples disponibles en la red y cada una reúne una serie de requisitos mínimos), para este caso se ha utilizado Java (JRE) y necesitará también un procesador Pentium 166 MHz o superior con un mínimo de 125 MB de espacio libre en el disco y un mínimo de 32 MB de RAM.También permite trabajar en XP (Para sistemas Vista, si fuera el caso, se recomienda ver los requisitos tanto de Oracle como de las máquinas virtuales).

3.3. Validación de Requisitos

En este apartado se recoge de forma gráfica las pruebas aplicadas para la comprobación de los requisitos, es independiente de las pruebas funcionales y modulares del sistema. La finalidad es recopilar la información del cumplimiento de los requisitos, así cómo la forma en la que se han hecho las comprobaciones. En la tabla se plasman en el eje horizontal las pruebas y en le vertical los requisitos del sistema (sobre todo los funcionales, ya que algunos requisitos no funcionales no son del todo viables para verificar). En caso de error se plasma en qué prueba falla, y si se decide una corrección o no al problema, todo dependerá de la importancia del requisito y de lo que afecte a la globalidad del sistema. Se marcarán las pruebas realizadas con OK, ERROR según se apliquen y funcionen, en caso de aparecer NI significa que no se ha implementado, pero esas serían las pruebas a realizar. Por último, en caso de no aparecer nada esa prueba no se ha aplicado a ese requisito.

	PR-1	**PR-2**	**PR-3**	**PR-4**	**PR-5**	**PR-6**	**PR-7**	**PR-8**	**PR-9**
RF-1	OK	OK	OK	OK	OK	OK		OK	OK
RF-2	OK	OK	OK	OK		OK	OK		
RF-3	OK		OK						
RF-4	OK	OK		OK					
RF-5	OK	OK		OK					
RF-6	OK				OK		OK		
RF-7	OK	OK	OK			OK	OK		
RF-8	OK	OK				OK			
RF-9	OK	OK				OK	OK		OK
RF-10	OK	OK			OK	OK	OK		
RF-11	OK	OK			OK	OK			
RF-12	OK	OK			OK	OK			
RF-13	OK		OK			OK			
RF-14	OK		OK			OK			
RF-15	OK	OK					OK	OK	
RF-16	OK			OK				OK	
RF-17	OK			OK				OK	
RF-18	OK	OK	OK			OK	OK		
RF-19	OK	OK	OK			OK	OK		
RF-20	OK	OK	OK			OK	OK		
RF-21	OK	OK					OK		

RF-22	OK		OK	OK		OK			
RF-23	OK	OK				OK	OK		
RF-24	OK	OK			OK	OK	OK	OK	
RF-25	OK	OK			OK	OK	OK		
RF-26	NI	NI			NI	NI	NI		
RF-27	OK	OK				OK	OK		
RF-28	NI	NI			NI				NI
RF-29	NI	NI	NI						
RF-30						OK			
RF-31	OK	OK							OK
RF-32	NI	NI	NI				NI		
RF-33	OK		OK						
RF-34	OK	OK	OK						
RF-35	OK		OK	OK					
RF-36	OK		OK	OK					
RNF-2					OK				OK
RNF-4			OK					OK	
RNF-14	OK				OK				
RNF-15	OK				OK				
RNF-18				OK					
RNF-22			OK					OK	
RNF-23			OK					OK	
RNF-24			OK					OK	
RNF-25			OK					OK	
RNF-26			OK					OK	
RNF-27			OK					OK	
RNF-28			NI					NI	

Tabla 8 Validación de Requisitos

3.4. Metodología

La evolución de la disciplina de ingeniería de software ha traído consigo propuestas diferentes para mejorar los resultados del proceso de construcción. La Ingeniería de software, se vale y establece de una serie de modelos que establecen y muestran las distintas etapas y estados por lo que pasa un producto software, desde su concepción inicial, pasando por su desarrollo, puesta en marcha y posterior mantenimiento, hasta la retirada del producto. Las metodologías tradicionales hacen énfasis en la planeación, y las metodologías ágiles haciendo énfasis en la adaptabilidad del proceso. Decantarse por una u otra es en ocasiones complicado. Seguir una metodología específica ayuda a definir las fases que tendrán lugar, qué criterios se seguirán para pasar de fase, establecer resultados intermedios y finales. Para el técnico ayuda a comprender el problema, a establecer los recursos y los costes, así como las necesidades requeridas. Para el cliente es una garantía de calidad del producto, y una confianza en los plazos establecidos.

Una de las tareas más importantes de realizar es establecer el ciclo de vida que va a seguir el proyecto, El Ciclo de vida definirá los estados que pasará y qué hacer para transformar el producto. Para su elección se han de tener en cuente ciertos aspectos, antes de decidir cual es el que mejor se adapta a nuestra situación.

Nos encontramos ante un proyecto que no tiene todos los requisitos establecidos desde el principio, además puede sufrir modificaciones y existen ciertos riesgos no controlables, riesgos como por ejemplo requisitos no comprendidos, posibles diseños erróneos, errores en la implementación, cambios importantes de estructuras, herramientas, adaptación a las herramientas utilizadas etc.

Siguiendo estas características el modelo de ciclo de vida que mejor se adapta sería un modelo evolutivo en espiral. Este modelo tiene en cuenta fuertemente el riesgo que aparece a la hora de desarrollar software. Para ello, se comienza mirando las posibles alternativas de desarrollo, se opta por la de riesgo más asumible y se hace un ciclo de la espiral. Si el cliente quiere seguir haciendo mejoras en el software, se vuelve a evaluar las distintas nuevas alternativas y riesgos y se realiza otra vuelta de la espiral, así hasta que llegue un momento en el que el producto software desarrollado sea aceptado y no necesite seguir mejorándose con otro nuevo ciclo. Permite volver a atrás y asegura una calidad, es ideal para procesos no cerrados, aunque es complicado, de riesgo y no están claras sus entradas y salidas. Es por todas estas característica por las que se adapta mejor a nuestro trabajo.

Este modelo se caracteriza por las siguientes ventajas:

- No necesita una definición completa de los requisitos para empezar a funcionar.
- Al entregar productos desde el final de la primera iteración es más fácil validar los requisitos.

- El riesgo en general es menor, porque si todo se hace mal, solo se ha perdido el tiempo y recursos invertidos en una iteración (las anteriores iteraciones están bien).
- El riesgo de sufrir retrasos es menor, ya que al identificar los problemas en etapas tempranas hay tiempo de subsanarlos.

Como inconvenientes principales:

- Es difícil evaluar los riesgos.
- Necesita de la participación continua por parte del cliente.
- Cuando se subcontrata hay que producir previamente una especificación completa de lo que se necesita, y esto lleva tiempo.

En cada vuelta o iteración hay que tener en cuenta (figura 3.2).

3.2 Ciclo de Vida en Espiral

- Los Objetivos: Que necesidad debe cubrir el producto.
  - Fijar también los productos definidos a obtener: requerimientos, especificación, manual de usuario.
  - Fijar las restricciones.
  - Identificación de riesgos del proyecto y estrategias alternativas para evitarlos.
- Análisis del riesgo**:** Se estudian todos los riesgos potenciales y se seleccionan una o varias alternativas propuestas para reducir o eliminar los riesgos.
- Desarrollar, verificar y validar (probar)
  - Tareas de la actividad propia y de prueba.
  - Análisis de alternativas e identificación resolución de riesgos.

- Dependiendo del resultado de la evaluación de los riesgos, se elige un modelo para el desarrollo, el que puede ser cualquiera de los otros existentes, como el formal, el evolutivo, el de cascada, etc. Así si por ejemplo si los riesgos en la interfaz de usuario son dominantes, un modelo de desarrollo apropiado podría ser la construcción de prototipos evolutivos. Si lo riesgos de protección son la principal consideración, un desarrollo basado en transformaciones formales podría ser el más apropiado.

- Planificar**:** Revisamos todo lo hecho, evaluándolo, y con ello decidimos si continuamos con las fases siguientes y planificamos la próxima actividad.

Además es posible tener en cuenta mejoras y nuevos requerimientos sin romper con la metodología, ya que este ciclo de vida no es rígido ni estático.

También genera muchos riesgos

- Genera mucho tiempo en el desarrollo del sistema
- Modelo costoso. Sobre todo en nuestro caso, de tiempo
- Requiere experiencia en la identificación de riesgos

La finalización de cada iteración ha culminado con una reunión con el cliente (tutores del proyecto). Se ha optado en cada periodo del ciclo por la realización de un prototipo para un mejor seguimiento de los requisitos y sus mejoras.

Un prototipo debe tener el objetivo de mostrar al cliente o a la gerencia del proyecto el resultado que se obtendrá de la implementación de cada uno de los requerimientos del cliente una vez terminado el desarrollo. Con los prototipos se tiene la posibilidad de obtener retroalimentación de manera temprana.

En cada reunión se efectúan las siguientes tareas:

- Se establecen si los resultados se cumplen con respecto a lo establecido en el ciclo anterior y se analiza la desviación que se está produciendo en caso de existir esta.
- En cada fase se analizan los objetivos, tanto los cumplidos como los nuevos.
- Se discuten unas alternativas iniciales para llevar a fin la siguiente fase.
- Se establecen plazos teniendo en cuenta los riesgos controlables.
- Se establece qué se va a hacer en la siguiente fase.

- Documentar aquello realizado y lo necesario para las siguientes fases.
- Un compromiso. Decisiones de gestión sobre cómo continuar.

Por su parte cada ciclo reúne además de las cuatro fases comentadas con anterioridad, un seguimiento, una documentación y una serie de pruebas de verificación y validación, tanto de la funcionalidad como de los requisitos.

Nuestro ciclo de vida en espiral se compone de las siguientes vueltas:

- Las primeras fases del ciclo han servido para establecer las herramientas a utilizar así como los primeros prototipos de la interface.
- Una vez establecida las características del editor y sus comunicaciones, las fases siguientes se centraron en establecer el modelo de base de datos a utilizar.
- Se establece el esquema de la base de datos y se verifica el cumplimiento de la seguridad, consistencia y exactitud de la misma.
- En la siguiente vuelta del ciclo se formaliza un prototipo refinado de la interface (sin interactuar con la base de datos).
- Se crea la ventana de autenticación (ya se accede a la base de datos) y se realiza un refinamiento de lo existente.
- Formalizar los últimos cambios de la ventana semiótica y su comunicación con la base de datos.
- Funcionalidad de la ventana de búsqueda y la cohesión con la base de datos y las ventanas principal y semiótica.
- Ventana correspondiente a la dimensión Sort y su navegabilidad con el resto de ventanas así como con la base de datos.
- Prototipo final para posibles mejoras.
- Versión definitiva.
- Recopilación de documentación y líneas futuras para el editor.
- Unificación de toda la documentación y verificación final de pruebas

Análisis y Diseño

- **Diagrama de Casos de Uso**
- **Descripción de Escenarios**
- **Arquitectura del Sistema**
- **Diseño de la Interface**
- **La base de Datos**

4. Análisis y Diseño

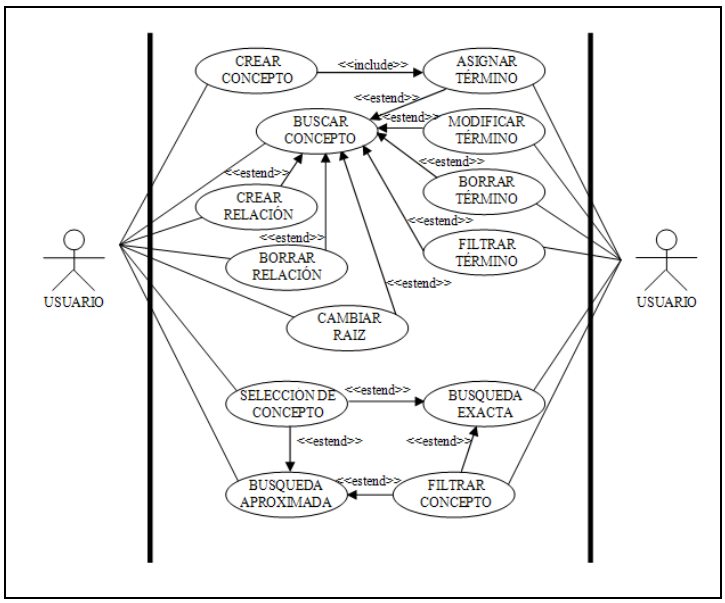
En este apartado se procede a analizar qué debe hacer nuestro sistema y cómo ha de hacerlo de la misma forma se establecen las partes que componen el sistema.

El análisis es un apartado que requiere una gran dedicación ya que es de suma importancia. Un estudio poco exhaustivo de alguna determinada opción podría conducir a problemas más adelante, ya en la fase de implementación, lo que provocaría retrasos y muchas mas complicaciones de las que de por sí pueden aparecer.

El diseño, es una de las partes más complicadas, ya que no se limita a encontrar una solución a un determinado problema, sino que además hay que intentar buscar la mejor solución, aquella que sea lo más coherente posible. La conclusión de estas fases, es que un mal modelo supondría modificar todo el trabajo realizado hasta el momento. Partiendo de la especificación de requisitos y otros datos se estable tanto el "qué" ha de hacer y posteriormente el "cómo" ha de hacerse.

4.1. Diagrama de Casos de Uso

Partiendo de la especificación de requisitos, y ayudados por UML, vamos a describir el comportamiento del sistema mediante un Diagrama de Casos de Uso (figura 4.1), donde cada caso de uso especifica un uso que se puede ejecutar en el sistema.



4.1 Diagrama de Casos de Uso

Pero esta información no es suficiente para explicar "qué" hace el sistema, para profundizar un poco más haremos una descripción algo más detallada de cada caso de uso.

4.2. Descripción de Casos de Uso

La descripción de los casos de uso se puede hacer de varias formas. Se pueden utilizar plantillas más o menos estandarizadas, o bien mediante lenguaje natural. Para este proyecto se toma la decisión de utilizar la plantilla del "caso de uso simple". A continuación se describe cada uno de los casos de uso. Consideramos que para poder realizar cualquiera de las acciones el usuario ha tenido que acceder al sistema autenticándose, por lo que se omitirá la mención de autenticación.

Crear concepto

Caso de uso 1: Crear Concepto.

Actores: Usuario.

Propósito: Crear un nuevo concepto en la base de datos.

Visión General: El usuario seleccionará la opción crear un nuevo concepto, añadirá al concepto los términos que quiera (al menos un término) y por último guardará el concepto.

Tipo: Primario y esencial.

Referencias: RF-06,RF-08,RF12

Curso típico de eventos:

1. Este caso de uso comienza cuando el usuario selecciona crear nuevo concepto.
2. Accede a la pantalla de edición de concepto.
3. Asigna de "1" a "n" términos al concepto.
4. Selecciona la opción guardar.
5. La información se guarda en la base de datos.
6. El sistema informa de la creación del concepto (opcional).

Alternativas:

- Línea 3: No se introduce ningún término, el sistema no dejará guardar.

Asignar Término

Caso de uso 2: Asignar Término.

Actores: Usuario.

Propósito: Asignar un término a un concepto.

Visión General: El usuario ha seleccionado crear o ha buscado un concepto, en ese momento el usuario puede añadir un término al concepto, ha de seleccionar un lenguaje y un peso para el término a añadir.

Tipo: Primario y esencial.

Referencias: RF-09.

Curso típico de eventos:

1. Este caso de uso comienza cuando el usuario accede a la pantalla reedición de concepto.
2. El usuario inserta un nuevo término.
3. El usuario selecciona un lenguaje de los disponibles.
4. El usuario asigna un peso.
5. El usuario seleccionar añadir.
6. El usuario selecciona guardar (validar).
7. El nuevo término se podrá ver entre los asignados.
8. La base de datos se actualiza con los nuevos datos.

Alternativas:

- Línea 2: No se inserta ningún término. El sistema no permitirá el resto de opciones, y al guardar no actualiza nada.
- Línea 3: No se selecciona un lenguaje. El sistema notifica que ha de asignarse.
- Línea 4: No se selecciona un peso. El sistema notifica que a de asignarse.

Modificar Término

Caso de uso 3: Modificar Término.

Actores: Usuario.

Propósito: Modificar las características de un término asignado a un concepto.

Visión General: El usuario accede a la pantalla de edición de concepto, selecciona un término de los asignados a un concepto, cambia los atributos que quiera y selecciona añadir.

Tipo: Primario y esencial.

Referencias: RF-10.

Curso típico de eventos:

1. Este caso de uso comienza cuando el usuario selecciona un término del listado de términos asociado a un concepto.
2. Cambia los valores que se requieran.
3. Pulsa el botón añadir (validar).
4. Se actualiza en el listado las características del término.
5. Se actualiza la base de datos.

Alternativas:

- Línea 3: No se valida. El sistema no se actualiza y los cambios no tiene lugar.

Borrar Término

Caso de uso 4: Borrar Término.

Actores: Usuario.

Propósito: Eliminar la asociación de un término específico con un concepto.

Visión General: El usuario accede a la pantalla de edición de concepto, selecciona un término de los asignados a un concepto y selecciona eliminar.

Tipo: Primario y esencial.

Referencias: RF-12, RF-10.

Curso típico de eventos:

1. Este caso de uso comienza cuando el usuario selecciona un término del listado de términos asociado a un concepto.
2. Pulsa el botón eliminar.
3. Se actualiza en el listado de términos del concepto.
4. Se actualiza la base de datos.

Alternativas:

- Línea 2: No valida el cambio. El sistema no se actualiza y los cambios no tienen lugar.

Filtrar Término

Caso de uso 5: Filtrar Término.

Actores: Usuario.

Propósito: Realizar un filtrado de los Términos que aparezcan por pantalla.

Visión General: El usuario accede a la pantalla de edición de concepto y visualiza los términos relacionados con un concepto, selecciona los filtros que vea necesario y pulsa filtrar.

Tipo: Primario y esencial.

Referencias: RF-13, RF-14.

Curso típico de eventos:

1. Este caso de uso comienza cuando el usuario tiene visible un listado de términos asociados a un concepto.
2. Selecciona uno o varios filtros de los disponibles.
3. Pulsa filtrar.
4. Se actualiza el listado original, según los filtros aplicados.

Alternativas:

- Línea 2: No se selecciona ningún filtro. No cambia el listado original.

Buscar Concepto

Caso de uso 6: Buscar Concepto.

Actores: Usuario.

Propósito: Buscar un concepto y cargar todas sus características en el sistema.

Visión General: Para buscar un concepto ha de existir un concepto en el portafolios de la pantalla principal, el usuario pulsa editar en la pantalla principal, el concepto pasa al cuadro de texto del lenguaje y carga los valores del concepto.

Tipo: Primario y esencial.

Referencias: RF-07.

Curso típico de eventos:

1. Este caso de uso comienza cuando el usuario selecciona editar de la pantalla principal.
2. El concepto que había en el portapapeles pasa al cuadro de texto de Concepto.
3. El sistema carga las características del concepto donde corresponda.

Alternativas:

- Línea 2: No hay concepto en el portapapeles. No cambia el valor del cuadro de texto Concepto.

Crear Relación

Caso de uso 7: Crear Relación.

Actores: Usuario.

Propósito: Asignar una nueva relación entre conceptos.

Visión General: El usuario ha de estar en la pantalla de relaciones de conceptos, pulsa crear relación y el concepto que esté en el portapapeles pasará a formar parte de la relación.

Tipo: Primario y esencial.

Referencias: RF-21, RF-25, RF-27.

Curso típico de eventos:

1. Este caso comienza cuando el usuario se encuentra en la pantalla de asociación de conceptos.
2. El usuario selecciona la opción añadir relación.
3. Se actualizan las relaciones y se muestran de nuevo todas las relaciones.
4. Se actualiza la base de datos.

Alternativas:

- Línea 2: No hay ningún concepto en el portapapeles. No se actualizan las relaciones.

Borrar Relación

Caso de uso 8: Borrar Relación.

Actores: Usuario.

Propósito: Borrar una relación existente entre conceptos.

Visión General: El usuario se encuentra en la pantalla de relaciones de conceptos, selecciona alguno de los conceptos relacionados con el concepto actual, pulsamos eliminar relación y se actualiza las relaciones.

Tipo: Primario y esencial.

Referencias: RF-24.

Curso típico de eventos:

1. Este caso comienza cuando el usuario se encuentra en la pantalla de asociación de conceptos.
2. El usuario selecciona alguno de los conceptos relacionados con el concepto raíz.
3. Se selecciona eliminar relación.
4. Se actualiza por pantalla la relación de conceptos con el concepto raíz.
5. Se actualiza la base de datos.

Alternativas:

- Línea 2: No se selecciona ningún concepto. No se produce actualización.

Cambiar Raíz

Caso de uso 9: Cambiar Raíz.

Actores: Usuario.

Propósito: Poder ver las relaciones que tiene un concepto de los relacionados con el actual seleccionado.

Visión General: El usuario se encuentra en la pantalla de relaciones de conceptos, selecciona alguno de los conceptos relacionados con el concepto actual, pulsamos cambiar raíz y se cargan las relaciones de este concepto.

Tipo: Primario y esencial.

Referencias: RF-23.

Curso típico de eventos:

1. Este caso comienza cuando el usuario se encuentra en la pantalla de asociación de conceptos.
2. El usuario selecciona alguno de los conceptos relacionados con el concepto raíz.
3. Se selecciona cambiar raíz.
4. Se actualiza por pantalla la relación de conceptos, pasando a ser la raíz de la relación el concepto seleccionado, y mostrando sus relaciones.

Alternativas:

- Línea 2: No se selecciona ningún concepto. No se actualiza la pantalla.

Búsqueda Exacta

Caso de uso 10: Búsqueda Exacta.

Actores: Usuario.

Propósito: Realizar una búsqueda de los conceptos relacionados con un término específico

Visión General: El usuario ha de haber seleccionado la opción buscar de la pantalla principal. Escribirá un término en el cuadro de texto correspondiente, pulsará búsqueda exacta y el sistema le mostrará todos los conceptos relacionados con el término.

Tipo: Primario y esencial.

Referencias: RF-18.

Curso típico de eventos:

1. Este caso comienza cuando el usuario se encuentra en la pantalla de búsqueda.
2. El usuario introduce un término en el cuadro de texto correspondiente.
3. Se selecciona Búsqueda exacta.
4. El sistema muestra los conceptos relacionados con ese término.

Alternativas:

- Línea 2: No se introduce ningún término. Se muestran todos los conceptos que hay en la base de datos.
- Línea 4: No hay conceptos relacionados. El sistema notificará que no hay términos conceptos relacionados con el término.

Selección de Concepto

Caso de uso 11: Selección de Concepto.

Actores: Usuario.

Propósito: Seleccionar un concepto y pasarlo al portapapeles.

Visión General: El usuario se ha de encontrar en la ventana de búsqueda tras haber hecho una búsqueda, se selecciona un concepto y se pulsa aceptar, entonces el concepto pasa al portapapeles de la ventana principal.

Tipo: Primario y esencial.

Referencias: RF-21.

Curso típico de eventos:

1. Este caso comienza cuando el usuario se encuentra en la pantalla de búsqueda tras una búsqueda de conceptos.
2. El usuario selecciona un concepto de los del listado.
3. Se pulsa aceptar.
4. La ventana búsqueda se cierra y pasa el concepto al portapapeles de la ventana principal.

Alternativas:

- Línea 2: El usuario no selecciona ningún concepto. Se cierra la ventana y no cambia el portapapeles.
- Línea 3: Se pulsa cancelar. Se cierra la ventana y no cambia el portapapeles.

Búsqueda Aproximada

Caso de uso 12: Búsqueda Aproximada.

Actores: Usuario.

Propósito: Se pretende buscar todos los conceptos relacionados con el término introducido o con los términos que se parezcan al introducido.

Visión General: El usuario ha de haber seleccionado la opción buscar de la pantalla principal. Escribirá un término en el cuadro de texto correspondiente, pulsará búsqueda aproximada y el sistema le mostrará todos los conceptos relacionados con el término y con los términos similares al introducido.

Tipo: Primario y esencial.

Referencias: RF-19.

Curso típico de eventos:

1. Este caso comienza cuando el usuario se encuentra en la pantalla de búsqueda.
2. El usuario introduce un término en el cuadro de texto correspondiente.
3. Se selecciona Búsqueda aproximada.
4. El sistema muestra los conceptos relacionados con ese término o con términos similares.

Alternativas:

- Línea 2: No se introduce ningún término. Se muestran todos los conceptos que hay en la base de datos.
- Línea 4: No hay conceptos relacionados. El sistema notificará que no hay términos conceptos relacionados con el término.

Filtrar Conceptos

Caso de uso 13: Filtrar Conceptos.

Actores: Usuario.

Propósito: Filtrar el listado de conceptos para facilitar su selección.

Visión General: Se parte de una búsqueda de concepto, el usuario selecciona la opción filtrar, selecciona los filtros que quiere aplicar y pulsa aceptar, apareciendo los conceptos filtrados para facilitar su selección.

Tipo: Primario y esencial.

Referencias: RF-16, RF-17, RF-20.

Curso típico de eventos:

1. Este caso comienza cuando el usuario se encuentra en la pantalla de búsqueda.
2. El usuario selecciona la opción filtrar.
3. Se seleccionan los filtros.
4. Se selecciona aplicar filtros.
5. Se visualizan los conceptos tras aplicar los filtros.

Alternativas:

- Línea 3: No se selecciona ningún filtro. No se actualiza el listado de conceptos.
- Línea 4: Se selecciona cancelar. No se actualiza el listado de conceptos.

4.3. Descripción de Escenarios

Hasta aquí podríamos decir que tenemos la parte de análisis del sistema. Hemos partido de los requisitos establecidos y hemos llegado a la conclusión de "qué" debe hacer el sistema. Ahora podemos profundizar un poco más viendo "cómo" se ha de comportar el sistema que nos encontramos diseñando.

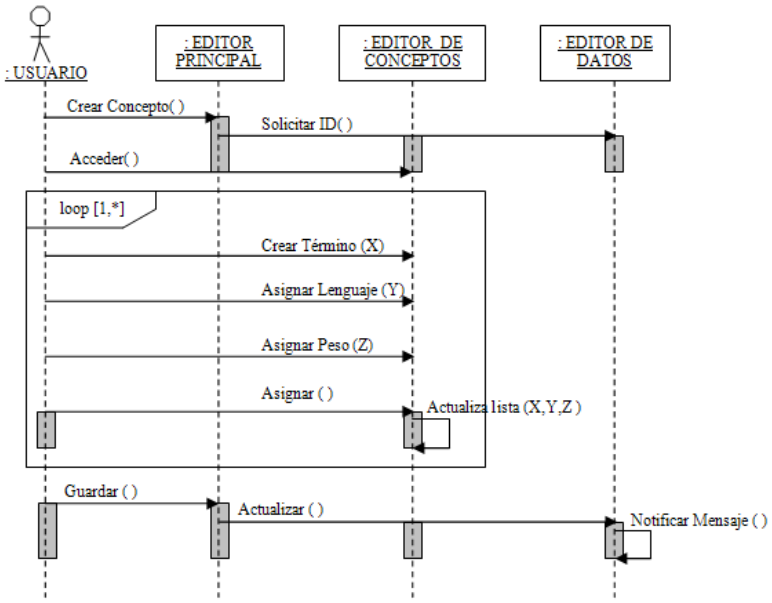
Para representar el comportamiento del sistema, se sigue empleando UML (Unified Modeling Language), en esta parte se extiende a UML 2, debido al empleo de ciertas notaciones que UML no es capaz de plasmar (como el caso de representar alternativas y opciones). Teníamos una descripción detallada de los casos de uso del sistema, ahora estableceremos los escenarios típicos e importantes que describen las diferentes situaciones que se van a poder dar y cómo el sistema debe actuar en cada una de ellas. Para ello se han confeccionado trece diagramas que plasman los correspondientes escenarios en los que se puede manejar el sistema.

Se puede decir que un escenario es una representación grafica de la ejecución del sistema en el que se refleja todo el comportamiento del mismo. Cada uno de los escenarios descritos tiene relación con uno de los casos de uso anteriores, y los objetos que intervienen se corresponden a objetos del sistema diferenciados.

4.3.1. Escenario Crear Concepto

En este escenario, se representa el comportamiento del sistema a la hora de crear un nuevo concepto por parte del usuario (figura 4.2). El usuario selecciona la opción crear concepto, desde la pantalla principal, en ese momento se solicita a la base de datos un nuevo identificador para dicho concepto (ver nota). Tras reservarse ese identificador, para el nuevo concepto, se tiene acceso al editor de conceptos que es el momento en el que el usuario puede acceder al editor de conceptos. Una vez que tiene visible la pantalla que le permite editar los conceptos (dimensión semiótica), crea

los términos relacionados con el concepto, al menos uno, y de la misma manera, por cada término creado, asigna un lenguaje y un peso a cada término creado, finalizando cada asignación tras seleccionar la opción asignar, momento en el cual se mostrará en la lista de términos relacionados con el concepto el nuevo término. Nota: Se cambia la petición de ID al principio, haciéndose a la hora de guardar, para evitar un acceso innecesario en caso de cancelar antes de guardar.



4.2 Escenario Crear Concepto

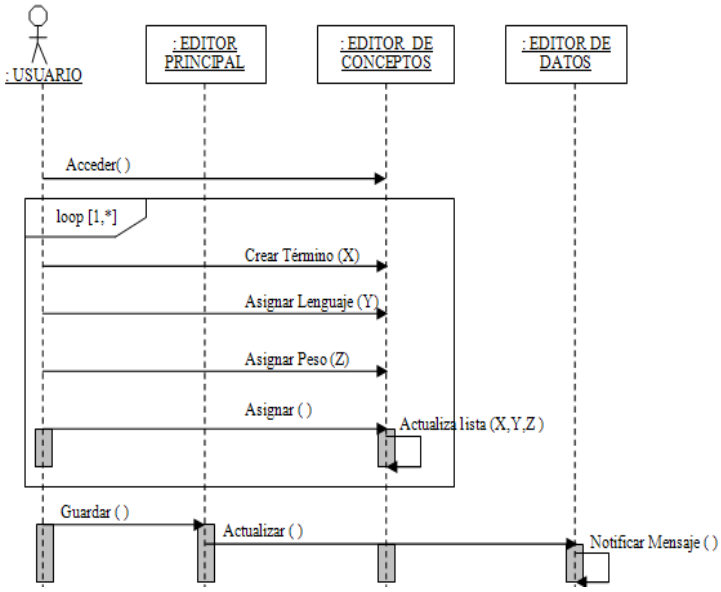
Para que todo tenga lugar el usuario ha de seleccionar la opción guardar, produciéndose la actualización de la base de datos y se mostrará un mensaje avisándonos del nuevo concepto creado (este mensaje es opcional).

4.3.2. Escenario Añadir Término

Con este escenario se pretende representar la interacción del sistema y el usuario cuando este pretende añadir un término a un concepto desde la pantalla de edición de conceptos (figura 4.3).

El usuario accede a la ventana semiótica, que le permite poder editar un concepto, desde este punto puede añadir desde uno hasta “n” términos, con sus correspondientes pesos y lenguajes, finalizando cada asignación con la actualización del listado de términos relacionados, tras seleccionar la opción asignar. Una vez que se tienen asignados los nuevos términos, si se quiere que tenga lugar la asignación, se selecciona la opción Guardar, desde la ventana principal, produciéndose la actualización de la base de datos (en el caso de la ventana de edición los cambios se realizan sin necesidad de guardar, son en tiempo de ejecución).

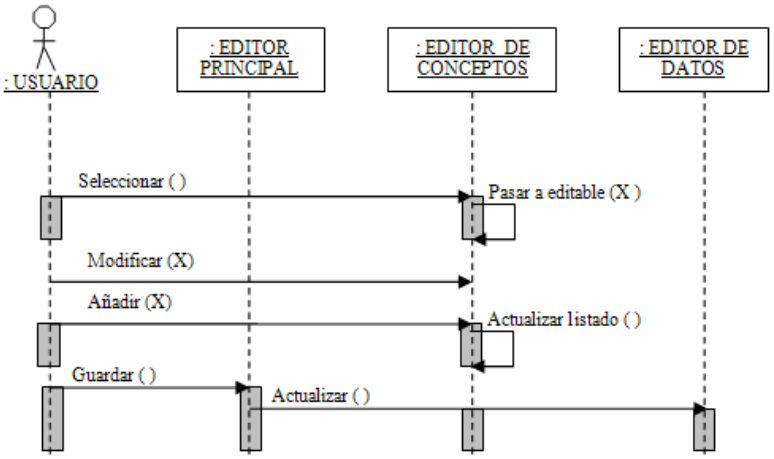
Si se aprecia, se puede ver que este escenario es similar al anterior, pero su diferencia radica en que este escenario describe el hecho de añadir nuevos términos a un concepto, independientemente de que sea tras crear un nuevo concepto o tras seleccionar modificar uno existente.



4.3 Escenario Añadir Término

4.3.3. Escenario Modificar Término

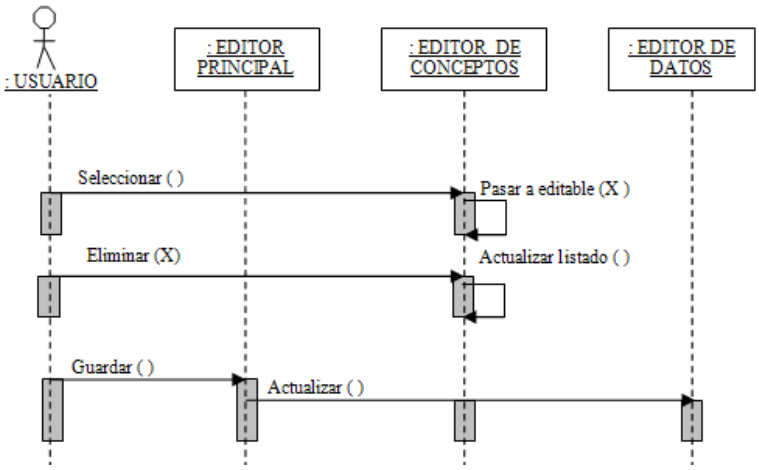
Este escenario representa el conjunto de acciones que se producen entre usuario y sistema, cuando se modifica algún término (figura 4.4). Se parte de que el usuario se encuentra en la ventana semiótica, el usuario selecciona un término del listado posible, el editor lo pasa en ese momento a editable, es en ese momento cuando el usuario puede modificar aquellas cualidades que quiera del término seleccionado, selecciona añadir término, entonces el sistema actualiza el listado de términos con el término modificado. Para que todo tenga lugar el usuario selecciona guardar desde la ventana principal, momento en el cual el editor de principal envía la actualización de la base de datos. Nota: se omite el punto de guardar, se guarda al validar.



4.4 Escenario Modificar Término

4.3.4. Escenario Borrar Término

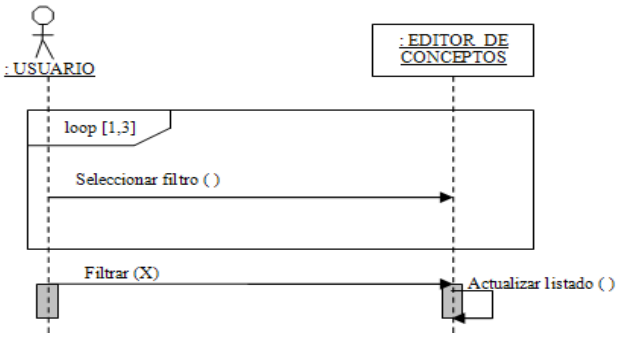
En este escenario se reproducen los acontecimientos que tiene lugar para producirse el borrado de un término. Se parte de que el usuario se encuentra en la pantalla semiótica, y selecciona un término de los disponibles, en se momento el Editor de Conceptos pasa a editable el término seleccionado (figura 4.5). A continuación el usuario selecciona la opción borrar (desasignar), momento en el que se actualiza el listado de términos disponible eliminando el seleccionado. Para finalizar la cadena de actividades, y que sea completa, el usuario selecciona guardar, y en ese momento se actualiza la base de datos del sistema. Nota: El borrado es automático al validarlo, se omite el paso guardar.



4.5 Escenario Borrar Término

4.3.5. Escenario Filtrar Término

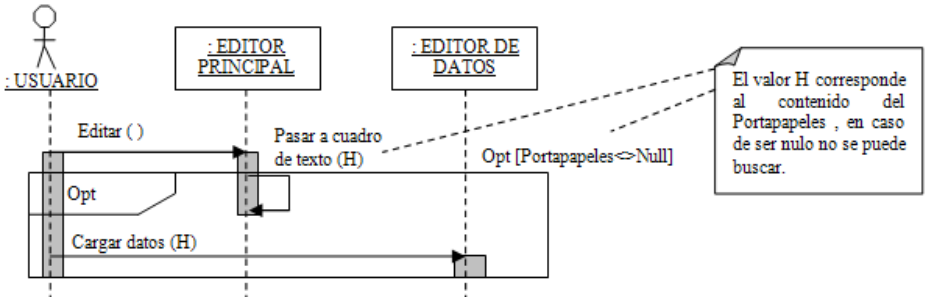
Este escenario representa el ciclo de actividades que se dan a la hora de aplicar un filtro sobre un listado de términos asociado a un concepto. Se parte de que el usuario se encuentra accediendo al listado de términos (figura 4.6). Con el listado visible, el usuario selecciona aquellos filtros que precise. Tras seleccionar los filtros seleccionará la opción filtrar y se actualizará el listado de pantalla según los filtros aplicados.



4.6 Escenario Filtrar Término

4.3.6. Escenario Buscar Concepto

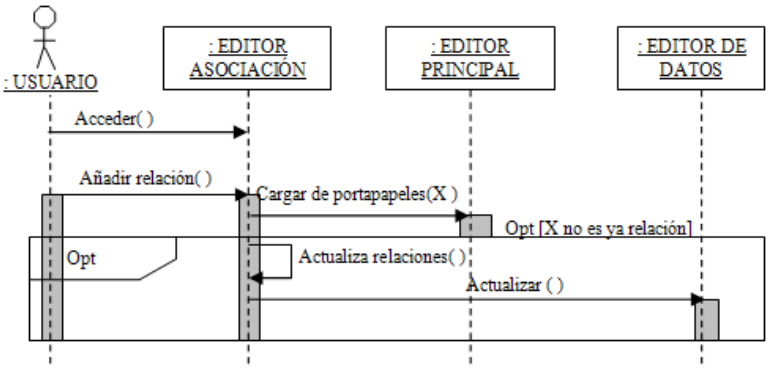
Con la descripción de este escenario, se plasma el funcionamiento que tiene lugar a la hora de realizar la búsqueda de un concepto (figura 4.7). Comienza el acontecimiento cuando el usuario selecciona editar desde la pantalla principal, en ese momento el sistema lleva a cabo una comprobación, dicha comprobación consiste en ver si hay algún concepto en el portapapeles de la ventana principal, si es así continua con la búsqueda y carga los datos del concepto desde la base de datos.



4.7 Escenario Buscar Concepto

4.3.7. Escenario Crear Relación

En este punto describiremos cómo se ha de comportar el sistema a la hora de crear una relación de conceptos, así cómo la manera que tendrá de interactuar con el usuario (figura 4.8).

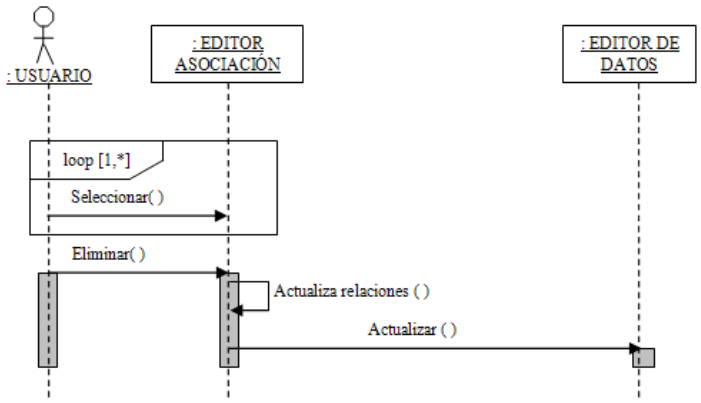


4.8 Escenario Crear Relación

El usuario tiene acceso a la ventana "Sort" (clasificativa) del sistema, en ese momento tiene ante sí un gráfico con las relaciones de un concepto. Selecciona la opción de añadir una relación al esquema, el sistema carga de la pantalla principal el concepto que hay en el portapapeles auxiliar, y compraba que no sea ya una relación del concepto raíz. Una vez comprobado podrá seguir con su funcionamiento normal y actualizar tanto por pantalla como en la base de datos la nueva relación.

4.3.8. Escenario Borrar Relación

El siguiente ciclo de acciones, entre usuario y sistema, plasma el modo de trabajo a la hora de querer borrar una relación entre conceptos (figura 4.9).



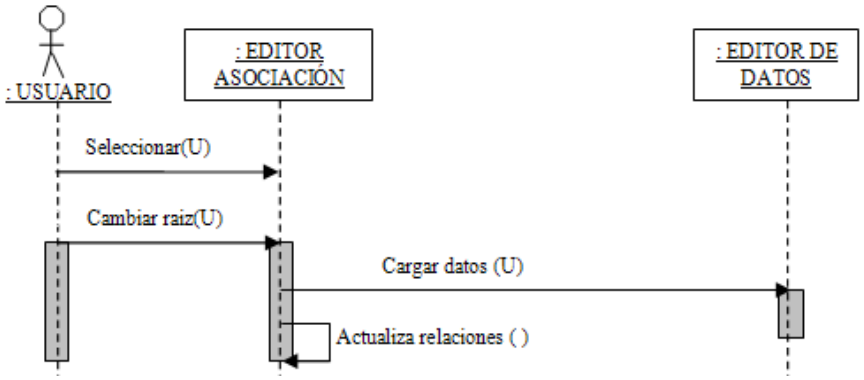
4.9 Escenario Borrar Relación

Se parte de que el usuario se encuentra en la ventana "Sort" y tiene visible el árbol de relaciones que tiene un concepto específico. El usuario selecciona los conceptos que vea oportunos, a

continuación selecciona eliminar relaciones, y en ese momento se actualiza tanto por pantalla como en la base de datos las relaciones afectadas.

4.3.9. Escenario Cambiar Raíz

La siguiente sucesión de actividades describen el escenario típico que representa el cambio de la raíz, y por tanto del árbol de relaciones, visibles en la pantalla "Sort" (figura 4.10).



4.10 Escenario Cambiar Raíz

El usuario en primera instancia ha de seleccionar uno de los conceptos del árbol mostrado por pantalla, en ese momento si pulsa la opción cambiar raíz, se cargarán los datos del concepto seleccionado y por lo tanto se actualizará el árbol de relaciones, pasando el concepto seleccionado a ser la raíz del árbol.

4.3.10. Escenario Búsqueda Exacta

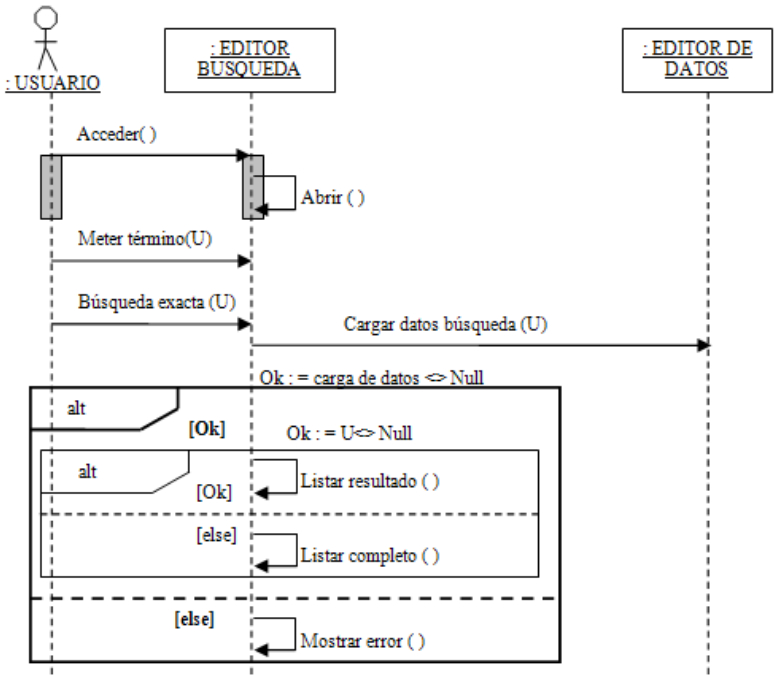
El siguiente escenario describe el ciclo de actividades que se dan cuando el usuario decide hacer una búsqueda exacta (figura 4.11).

El usuario accede a la ventana de búsqueda, momento en el que se despliega en pantalla como ventana emergente la misma. A continuación introduce un término a buscar y selecciona la opción búsqueda exacta. El sistema accederá a la base de datos para recuperar la información necesaria. Tras recibir la información de la base de datos el sistema puede actuar de dos maneras:

1. Se han recuperado datos de la base de datos. En ese momento se pueden dar dos opciones según la entrada que se produjo por parte del usuario:
    - Si el usuario introdujo un término se mostrará el listado de conceptos relacionados con ese término.

- Si el usuario no introdujo ningún valor, entonces el sistema muestra el listado de todos los conceptos de la base de datos.

2. No se recuperan datos de la base de datos, entonces se muestra un error de búsqueda.



4.11 Escenario Búsqueda Exacta

4.3.11. Escenario Búsqueda Aproximada

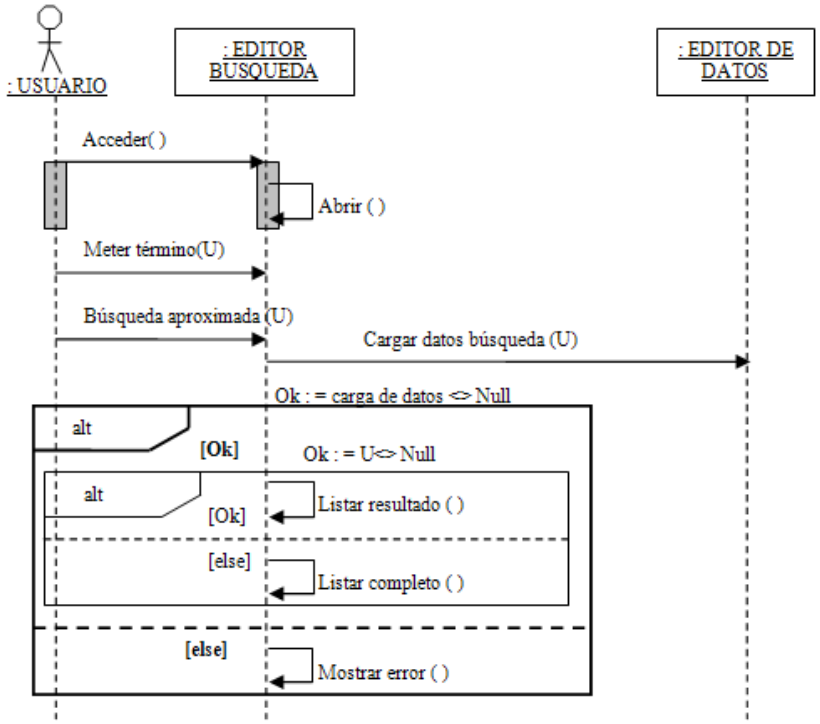
El siguiente escenario es muy similar al anterior (figura 4.12). El usuario accede a la ventana de búsqueda, momento en el que se despliega en pantalla como ventana emergente la misma. A continuación introduce un término a buscar y selecciona la opción búsqueda aproximada. El sistema accederá a la base de datos para recuperar la información necesaria. Tras recibir la información de la base de datos el sistema puede actuar de dos maneras:

1. Se han recuperado datos de la base de datos. En ese momento se pueden dar dos opciones según la entrada que se produjo por parte del usuario:

   - Si el usuario introdujo un término se mostrará el listado de conceptos relacionados con ese término o con los similares a el.

   - Si el usuario no introdujo ningún valor, entonces el sistema muestra el listado de todos los conceptos de la base de datos.

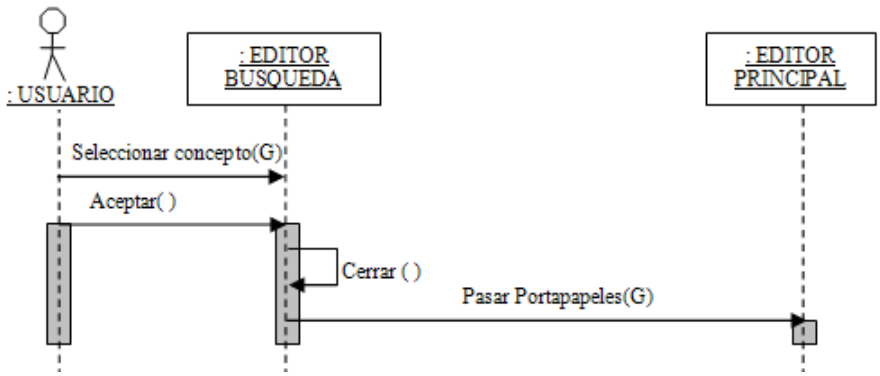
2. No se recuperan datos de la base de datos, entonces se muestra un error de búsqueda.



4.12 Escenario Búsqueda Aproximada

4.3.12. Escenario Selección de Concepto

Con este escenario tan simple se trata de aclarar el funcionamiento previsto a la hora de seleccionar un concepto desde la ventana de búsqueda (figura 4.13).

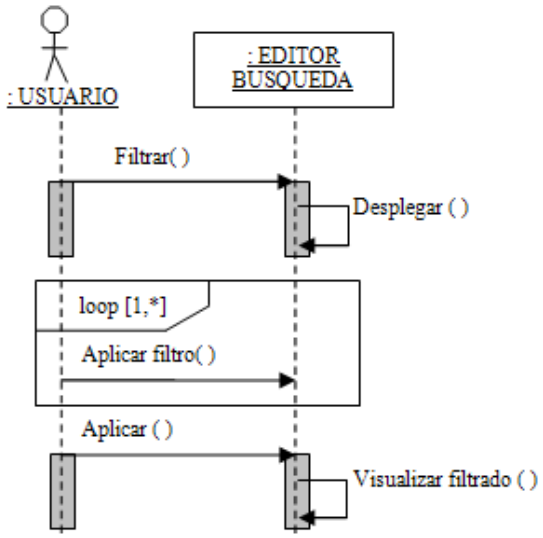


4.13 Escenario Selección de Concepto

El usuario se encuentra en la pantalla de búsqueda, y ante él se encuentra el listado de conceptos relacionados con un término, en ese momento selecciona un concepto cualquiera de los disponibles y elige la opción “Aceptar”. Al producirse tal acción la ventana de búsqueda se cerrará y el concepto seleccionado pasa al portapapeles de la ventana principal.

4.3.13. Escenario Filtrar Conceptos

Este último escenario describe de manera secuencial al detalle, el conjunto de actividades que se han de producir para realizar un filtrado de conceptos en la ventana de búsqueda (figura 3.14).



4.14 Escenario Filtrar Conceptos

El usuario se encuentra en la ventana de búsqueda, tras una búsqueda. Decide realizar un filtrado de los conceptos obtenidos. Para ello selecciona la opción filtrar, y en ese momento la ventana de búsqueda se expande mostrando la opciones de filtrado. El usuario selecciona los filtros que desea aplicar, y por último selecciona aplicar los filtros. El sistema actualiza los conceptos visibles según los filtros seleccionados.

4.3.14. Aclaraciones sobre los escenarios

En los puntos anteriores se han mostrado los escenarios característicos del sistema, pudiéndose dar alternativas que varíen el acontecer de los pasos descritos. Han sufrido ligeras modificaciones a lo largo del proceso de desarrollo, paro en sí mantiene la esencia del caso de uso correspondiente. Se ha de mencionar uno de los escenarios que no aparecen en los casos de uso, y aunque evidente no por ello debo olvidar, me refiero al escenario típico de identificación de usuario.

Este escenario no descrito es importante para iniciar el sistema. Comienza cuando el usuario arranca el sistema, en ese momento tiene acceso a la pantalla de autenticación. Introducirá su nombre de usuario y contraseña y seleccionará comenzar, es en ese momento cuando se contrasta con la base

de datos si está o no autorizado y si es así accederá a la pantalla principal del sistema, y en caso contrario se mostrará un mensaje de error.

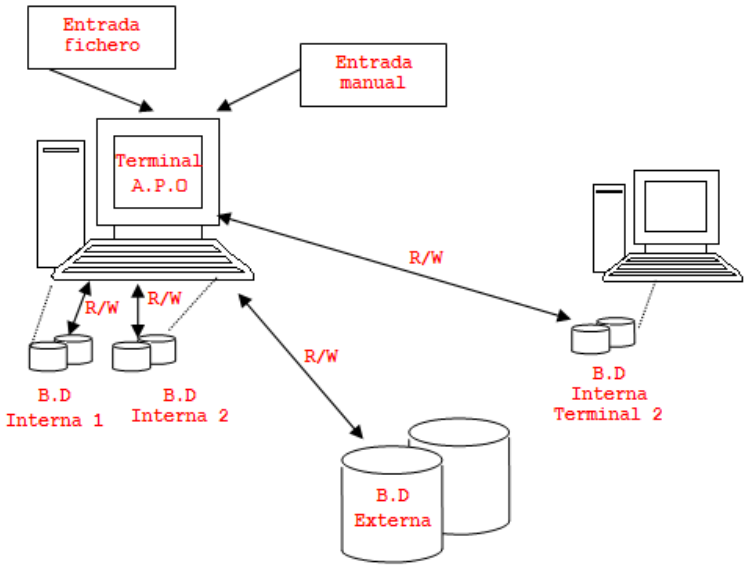
Otro punto importante a aclarar es el hecho de que el sistema no permite mostrar ninguna de las pantallas de las dimensiones (semiótica, sort, etc.) si antes no tiene ningún concepto asignado en la ventana de texto de concepto, ya sea por creación de un nuevo concepto o bien por edición de un concepto.

4.4. Arquitectura del Sistema

Este sistema tiene una estructura sencilla, en la que el usuario a través de la herramienta tiene acceso de escritura y consulta sobre una base de datos. En la siguiente figura se muestra la estructura requerida para el sistema (figura 4.15).

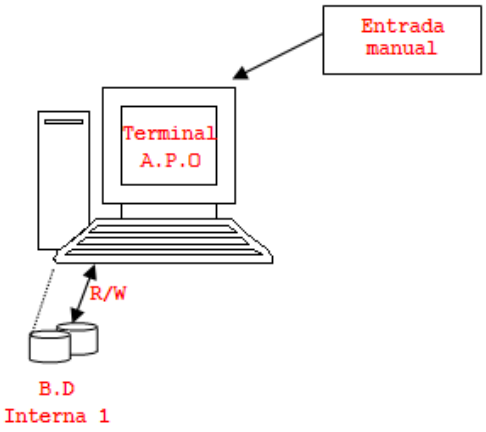
Se caracteriza por:

- Un terminal en el que ha de ejecutarse la herramienta de población ontológica.
- El terminal debería poder tener acceso a su propia base de datos.
- Se podría acceder a varias bases de datos en el mismo equipo.
- Se podría acceder a la base de datos de otros equipos, así como a datos almacenados en otras unidades.
- Las entradas de información serían tanto manuales como por medio de ficheros obtenidos de otras herramientas en el formato correcto.



4.15 Arquitectura del Sistema I

El sistema mostrado en la figura anterior describe el ideal para nuestro sistema. Podría acceder a cualquier base de datos que contenga la información necesaria para trabajar con ontologías. Para ello se requieren de otros módulos que permitan estas comunicaciones, pero eso escapa al alcance de este proyecto. A continuación (figura 4.16) se muestra el esquema que sigue el sistema actual.



4.16 Arquitectura del Sistema II

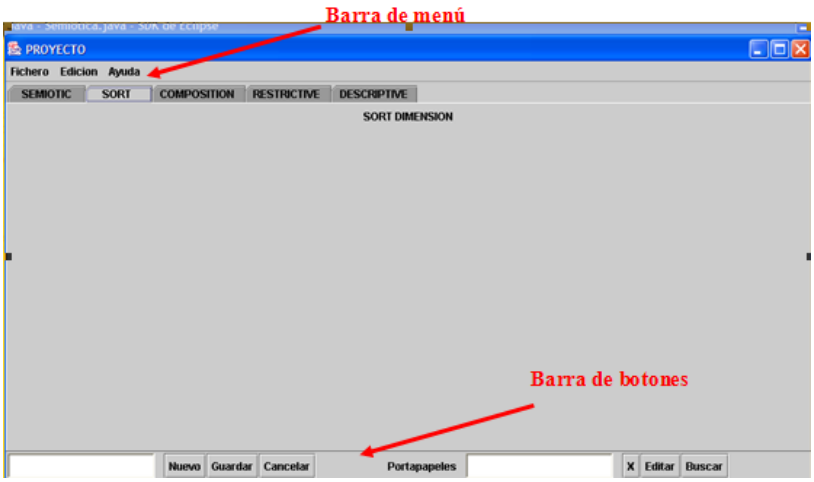
Es más sencillo, pero no por ello menos importante, ya que es la base para el resto de módulos que se formalicen en un futuro. En este esquema nos centramos en el acceso a una única base de datos interna del sistema (cabría poder acceder a varias bases de datos, haciendo modificaciones en el código, cuando lo ideal sería un módulo que permitiese configurar el acceso a la base de datos necesaria).

4.5. Diseño de la interface

Se establecen una serie de características que ha de reunir el editor ontológico, para darle cierta amigabilidad y vistosidad cara al usuario.

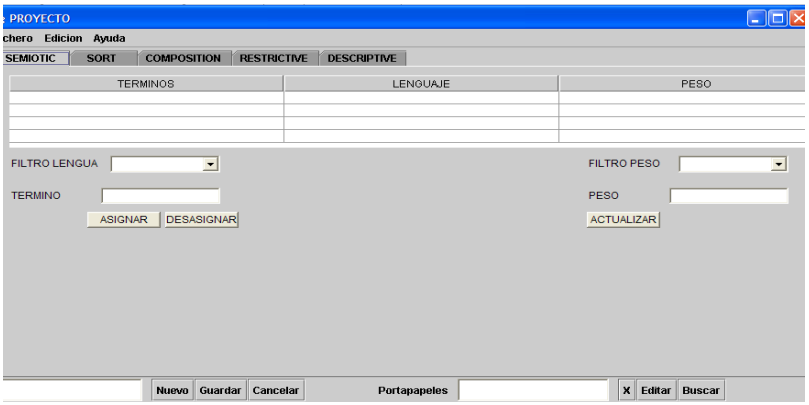
El editor gira en torno a una ventana principal, que se caracteriza por una barra de menú y una barra de botones comunes para el resto de ventanas emergentes (figura 4.17).

Las imágenes que se muestran a continuación son una aproximación inicial de la herramienta, para ver las definitivas ver el Anexo (A) Manual de Usuario.



4.17 Imagen Pantalla Principal

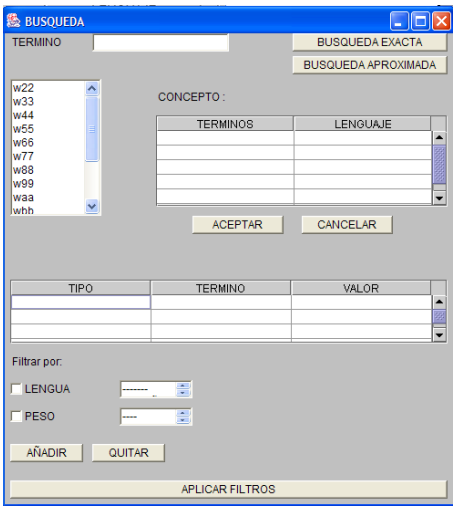
Por cada dimensión, de las que se componga la ontología, debe poderse desplegar una ventana (figura 4.18), y cada una de ellas ha de poder acceder tanto a la barra de menú, como a la barra de botones.



4.18 Ejemplo de Pantalla de Dimensión

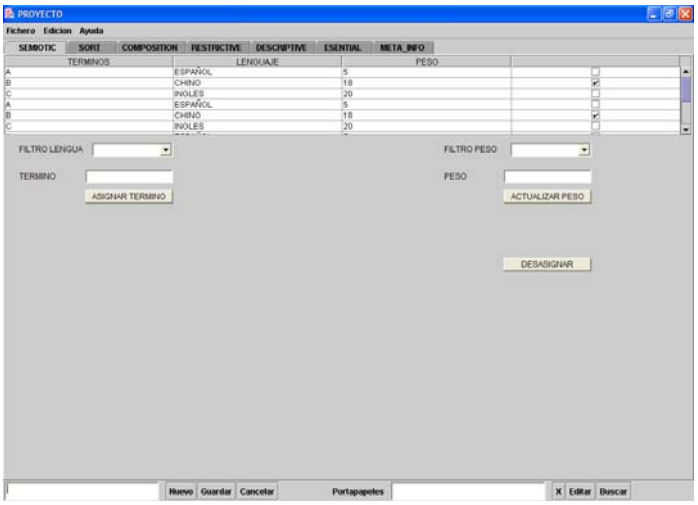
En la barra de botones deben existir opciones para crear, guardar y cancelar, teniendo para cada ventana de dimensión una funcionalidad u otra, según corresponda. Además debe de contener un portapapeles y una opción de buscar (figura 4.19). La opción de buscar debería desplegar una ventana especial para esta acción.

Como se puede apreciar en la imagen piloto de “búsqueda”, se requieren listados de elementos, que han de realizarse mediante tablas, así como elementos que permitan selección, ya sea mediante listados seleccionables o chequeos seleccionables. Esta operativa se seguirá en el resto de pantallas.



4.19 Ventana de Búsqueda

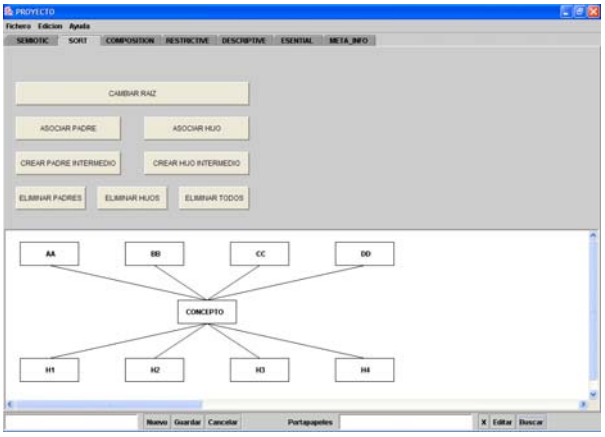
Además de la ventana principal se realiza el diseño de la "Semiotic dimension" (figura 4.20) y la "Sort dimension" (figura 4.21). La primera permite la creación y modificación de conceptos, permitiendo añadir términos relacionados con el concepto, así como otras características Esta ventana semiótica debe poder aplicar filtros.



4.20 Ventana Semiótica

La ventana "Sort" tendrá como finalidad crear relaciones entre conceptos, así como ver qué conceptos se encuentran relacionados, y todo aquello que afecte a las relaciones entre Conceptos a este nivel.

Se debe tener en cuenta que debido a la cantidad de información que puede haber, es posible que los listados y tablas sean más grandes que la pantalla en la que se encuentran, por lo que se añadirán barras de desplazamientos a listas y tablas para poder visualizar todo el contenido.



4.21 Ventana Dimensión Tipo

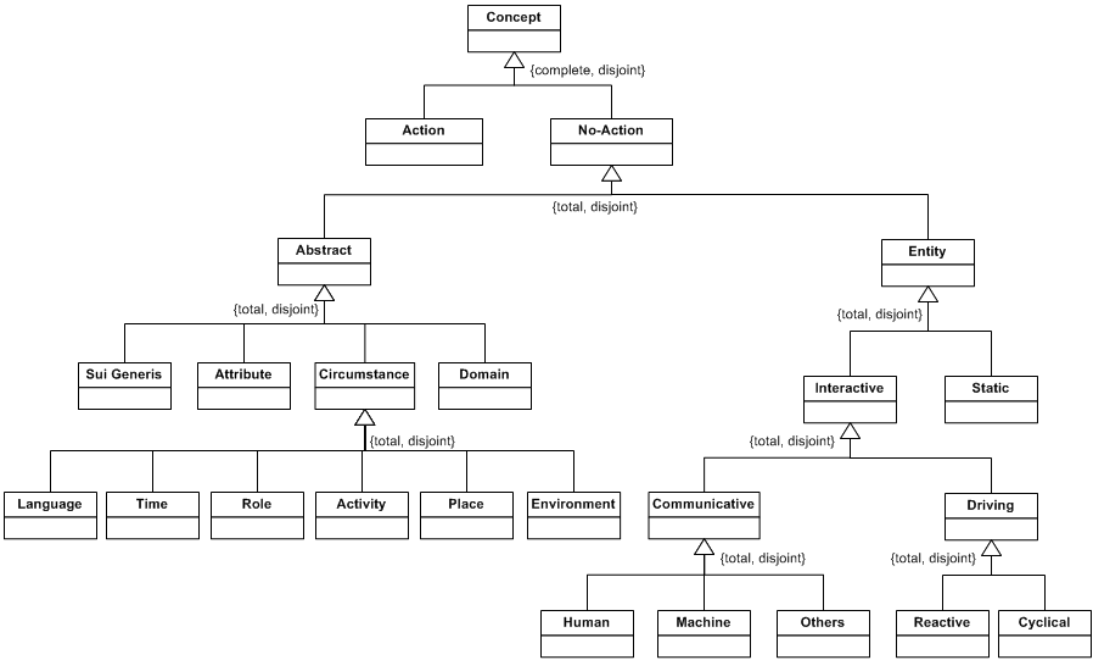
4.6. La Base de Datos

Para el editor de ontología se necesita una base de datos, la cual ha de recoger una serie de características y un formato específico para que los módulos del sistema puedan interpretar y obtener aquello que necesiten. Por todo en esta base de datos se recoge un esquema de ontología general, para poder ser reutilizado en cualquier medio, y tiene que recoger todo lo necesario para recopilar tanto los conceptos como sus relaciones, interacciones y características.

Partiendo de esta idea se crea una serie de dimensiones que cumplirán con estas características. Se parte de un diseño ya establecido **[49],** donde se ha adaptado ligeramente ciertas características, y se ha pasado a un esquema UML debido a la complejidad visual que entramaba crear su esquema ER Estas dimensiones son expuestas a continuación, así como sus diagramas de clases para una mayor compresión. No es parte del proyecto el adentrarse en las conclusiones para llegar a dichos esquemas, han sido evaluados y creados a partir de peticiones específicas de los tutores del proyecto. La importancia radica en la fase de implementación, que es dónde se ha de dar cuerpo a estos esquemas siguiendo las características que se verán a continuación.

4.6.1. Dimensión Esencial

Referenciada como Essential Dimensión (figura 4.22), representa la taxonomía de esta ontología genérica, y lo que plasma es la clasificación de los conceptos.

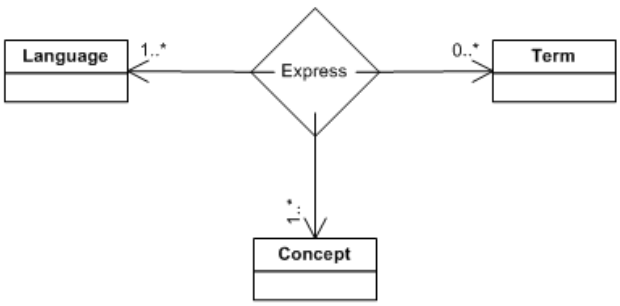


4.22 Dimensión Esencial

Se trata como se puede ver de la jerarquía existente de todos los tipos de concepto.

4.6.2. Dimensión Semiótica

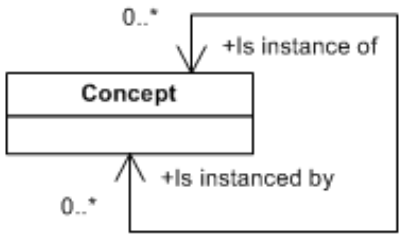
Referenciada como Semiotic dimension (figura 4.23), lo que dice es que en general toda la colección de conceptos puede ser representada como símbolos o términos, que son utilizados para referir la interacción del conocimiento. Se considera como norma que todos los conceptos pueden ser referenciados por uno o más términos y cada término puede estar usado específicamente por uno o más conceptos. Los términos son clasificados además en lenguajes que definen la dimensión semiótica. De la misma forma los lenguajes son por si mismos conceptos abstractos que pueden ser relacionados con otros.



4.23 Dimensión Semiótica

4.6.3. Dimensión Tipo

Referenciada como Sort dimension (figura 4.24), representa una relación padre-hijo entre relación de conceptos, donde un concepto puede ser instancia de otro y a su vez puede estar instanciado por otros objetos



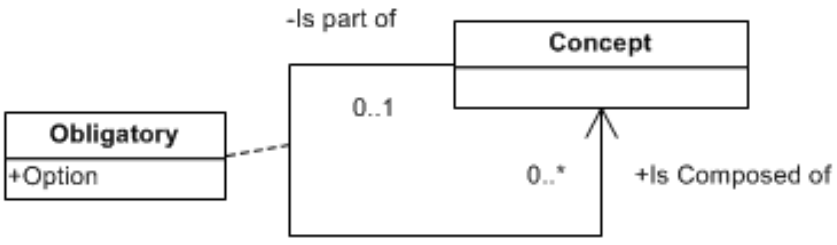
4.24 Dimensión Tipo

4.6.4. Dimensión Composición

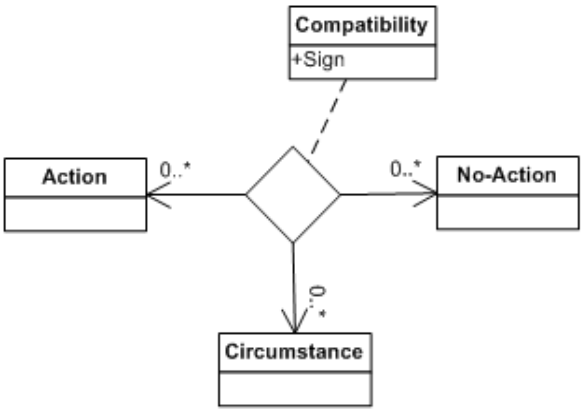
Referenciada como Composition dimension (figura 4.25), con esta dimensión representaremos como un concepto puede componerse de otros conceptos.

4.6.5. Dimensión Restrictiva

Referenciada como Restricitive dimension (figura 4.26), que describe la compatibilidad entre conceptos acción y el resto. Hay que resaltar que en estas relaciones existe un factor de compatibilidad entre los conceptos relacionados.



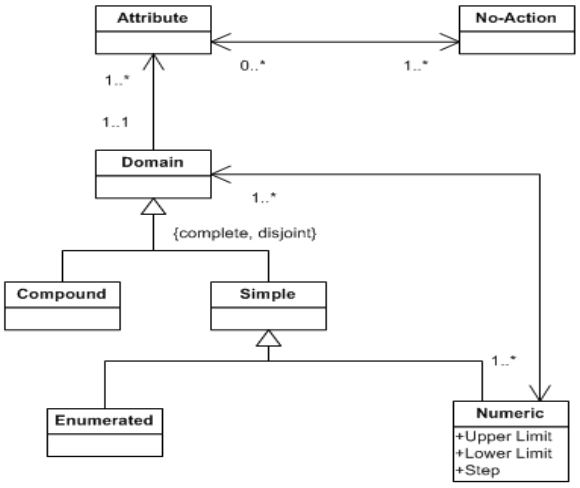
4.25 Dimensión Composición



4.26 Dimensión Restrictiva

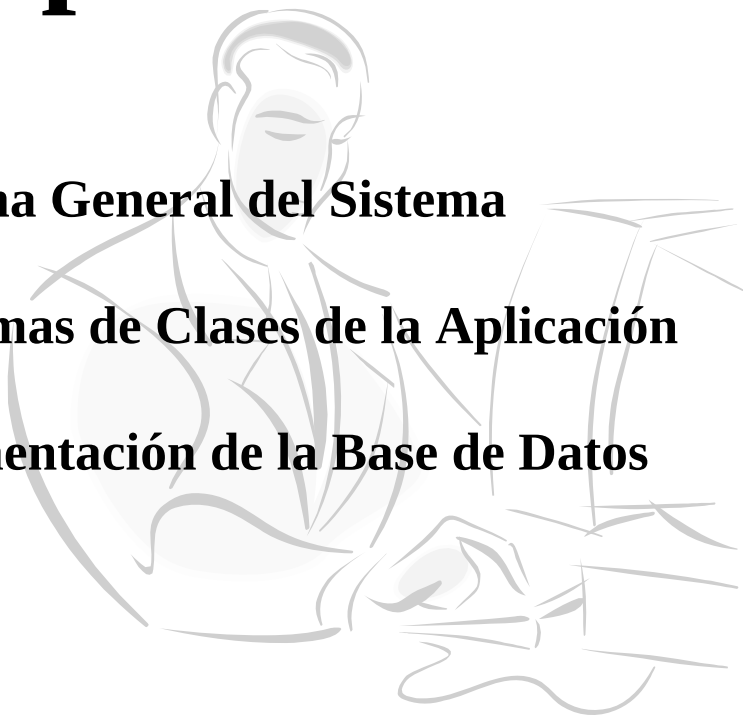
4.6.6. Dimensión Descriptiva

Referenciada como Description dimension (figura 3.28), muestra la relación entre conceptos con los conceptos del tipo abstracto del tipo atributo.



4.27 Dimensión Descriptiva

Implementación



- **Esquema General del Sistema**
- **Diagramas de Clases de la Aplicación**
- **Implementación de la Base de Datos**

5. Implementación

En esta fase se formaliza lo descrito en la fase de análisis y diseño. Se construyen las estructuras de datos necesarias, así como las interfaces que se necesitan.

Es importante en la implementación seguir un orden y comprobar la equivalencia entre estructuras, así como los procesos que se van a tratar. Es una fase que busca la cohesión entre todos los procesos.

Para la implementación se ha utilizado el lenguaje Java, por los motivos vistos anteriormente (*Estudio de viabilidad*). Además de las características mencionadas con anterioridad, Java tiene la cualidad de estar orientado a objetos, y permite reutilizar y editar las clases existentes, además de ser de libre acceso y disponer en la red de multitud de ejemplos de interfaces gráficas. Se utilizarán los componentes de los paquetes AWT y SWING, que disponen de gran cantidad de objetos relacionados con componentes para la implementación gráfica. La diferencia entre ambos radica en que AWT es dependiente del sistema operativo a utilizar, y las ventanas y objetos declarados toman características de dichos sistemas, mientras que SWING es independiente del sistema operativo, por lo que le da autonomía a la hora de la portabilidad. Como ideas de futuro se plantea el hecho de unificar todos los componentes AWT utilizados y adaptarlos a SWING, pero eso se explicará más adelante.

Como he comentado dichos componentes pueden además adaptarse a nuestras necesidades, reescribiendo los métodos correspondientes a las clases utilizadas, eso hace que se pueden crear nuevas clases con otras funcionalidades no disponibles así como formatos y plantillas propias para nuestro trabajo.

En cuanto al sistema gestor de base de datos se ha utilizado Oracle 10g, para implementar la base de datos, es sencillo de manejar, versátil y cubre todas las necesidades de la aplicación. Además Oracle distribuye los “Plug-ins” necesarios para comunicar Java con la base de datos, evitándonos tener que crearlos o utilizar alguno no compatible.

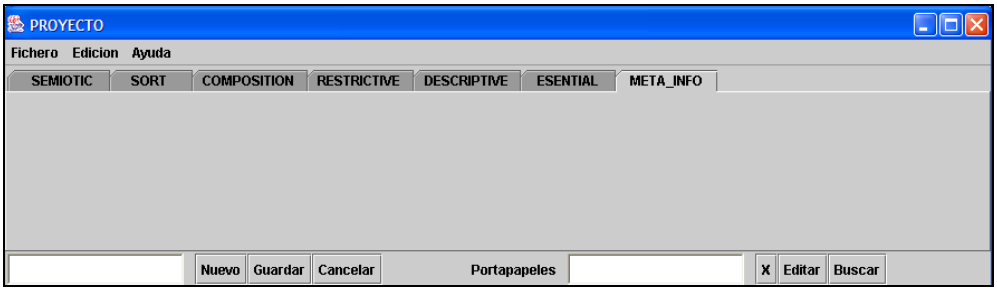
5.1. Esquema General del Sistema

Una representación del esquema general del sistema y que nos da una aproximación de su funcionamiento se podía observar en el capítulo anterior.

Tenemos una base de datos sobre la que la herramienta ejecuta continuas consultas y modificaciones. El usuario se encuentra ante una interface gráfica, intuitiva y clara.

Una vez validado el usuario, este tiene acceso a la pantalla principal del sistema (figura 5.1), que es la raíz de todas las operaciones disponibles.

En ella se hay una barra de menú (JMenuBar) y un abarra con botones común para todas las pantallas. Además dispone de un "JTabbedPane" que permite tener una serie de pestañas en pantalla, donde cada pestaña corresponde a una de las dimensiones de la ontología, para que cuando se seleccione se despliegue el contenido dentro de la pantalla principal, y así poder tener visible la barra de menú y la barra de botones con las opciones generales sobre las pantallas.



5.1 Pantalla Principal

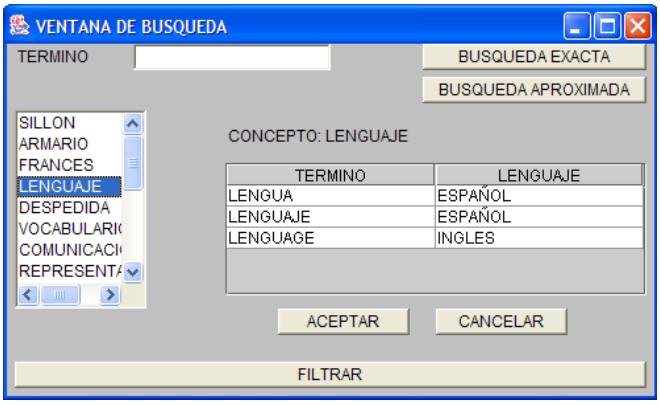
La barra de menú es orientativa, ya que las funcionalidades desarrolladas no requieren de ella, pero se ha creado con la finalidad de futuras ampliaciones (actualmente no tiene operatividad, salvo la opción de salir).

En cuanto a la barra inferior de opciones (figura 5.2), es de mencionar el cuadro de texto y el portapapeles. El primero no es editable, y únicamente muestra información cuando se crea un nuevo concepto o cuando se pasa un concepto del portapapeles para editarlo, en cuanto al segundo tampoco es editable y muestra información cuando se realiza alguna búsqueda de conceptos en la base de datos. Los botones creados tienen las funcionalidades descritas en sus etiquetas, pero podemos comentar un poco más a fondo cómo se han implementado sus roles.

5.2 Barra de Botones

- El botón Nuevo genera un identificador nuevo, obteniendo información sobre el último identificador dado de alta en la base de datos. Mostrará al usuario en el primer cuadro de texto "NUEVO CONCEPTO". Antes de crear un nuevo concepto o de buscar uno el sistema mantiene inactivas las ventanas de las dimensiones, ya que no se puede operar sobre ellas si no se tiene cargado datos de un concepto. Así que una vez se presiona nuevo, el sistema desbloquea las pestañas del JTabbedPanel.

- El botón Guardar guarda las modificaciones que se han hecho, así como en el caso de haber creado un nuevo concepto. Dependiendo de en qué pantalla estemos trabajando guardará una información u otra en la base de datos.
- El botón Cancelar limpia la pantalla en la que estemos, en caso de haber estado creando un nuevo concepto dejará limpio el cuadro de texto.
- El botón Editar pasará el concepto seleccionado en el Portapapeles al primer cuadro de texto, de esa forma el concepto seleccionado podrá ser consultado o editado según queramos.
- El botón Buscar despliega una ventana de búsqueda (figura 5.3), cuya finalidad es buscar conceptos a través de la relación que hay con términos.

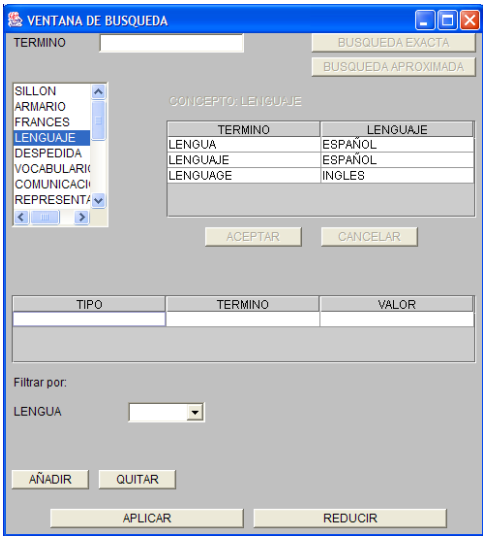


5.3 Ventana de Búsqueda

En esta ventana de Búsqueda, el usuario puede operar de varias formas:

- Introduce un término y selecciona búsqueda exacta, entonces se le mostrará en una lista todos los conceptos que se encuentran relacionados con ese término.
- Introduce un término y selecciona búsqueda aproximada, entonces se le mostrará en una lista todos los conceptos que se encuentran relacionada con ese término o alguno similar.
- No introduce ningún término y hace una búsqueda, lo que se mostrará en ese momento en el listado serán todos los conceptos de la base de datos.
- Cuando tiene disponible el listado de conceptos puede seleccionar uno y en la tabla con cabecera (“TERMINOS LENGUAJE”) se mostrarán todos los términos y su lenguaje asignado relacionados con el concepto (la utilidad de esta tabla es ver si el concepto seleccionado es el deseado, se puede pensar en conceptos científicos o de lenguajes muy técnicos que podrían ser complejos de entender y esto ayudaría).
- Si se pulsa aceptar y hay algún concepto seleccionado este pasa al portapapeles de la ventana principal y se cierra la de búsqueda.

- Si se pulsa cancelar se cierra la ventana de búsqueda sin hacer nada.
- Si se pulsa Filtrar (figura 5.4), la ventana se expande permitiendo aplicar los filtros que queramos sobre los términos de un concepto (de nuevo esto sirve para afinar la compresión de un concepto seleccionado).

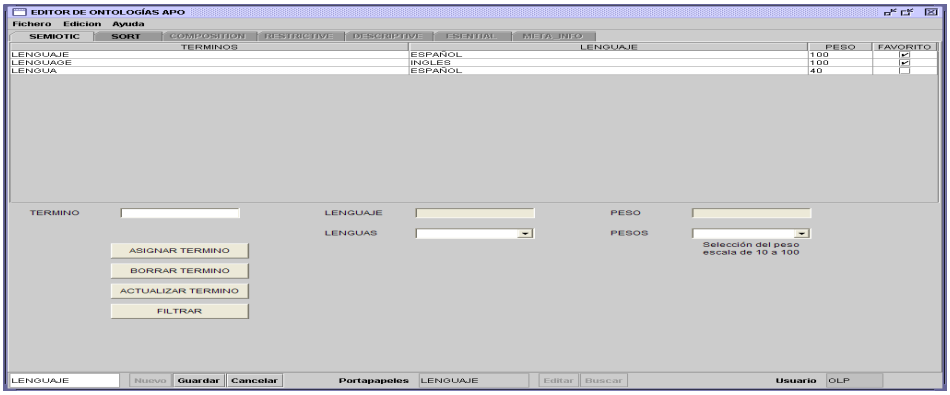


5.4 Ventana de Búsqueda Expandida

- Desde esta ventana expandida podemos con los CKECK creados seleccionar qué tipo de filtro a aplicar, y con listas desplegables seleccionar un valor a aplicar de esos filtros.
- Una vez seleccionado los filtros se añaden o quitan de la tabla con cabeceras (“TIPO TERMINO VALOR”), donde se van añadiendo los filtros a aplicar.
- Al seleccionar Aplicar la tabla de términos del concepto se actualiza con el nuevo filtrado.
- Si se selecciona Reducir no se aplican los filtros y se eliminan de la tabla de filtros los seleccionados.

Tanto en esta pantalla como en el resto los botones, tablas y listas están inactivas en caso de que no se realicen los pasos previos para su utilización, de esta manera para el caso de la ventana de búsqueda, no se puede seleccionar la opción filtrar si no hay concepto seleccionado, de la misma forma no se puede seleccionar añadir si no hay filtro seleccionado.

Para las dimensiones se han desarrollado 2 de ellas, la dimensión semiótica (figura 5,5), y la dimensión tipo (figura 5.6).



5.5 Ventana Semiótica

La utilidad de la ventana semiótica es asignar y modificar los términos a los conceptos, permite actualizar términos seleccionados.

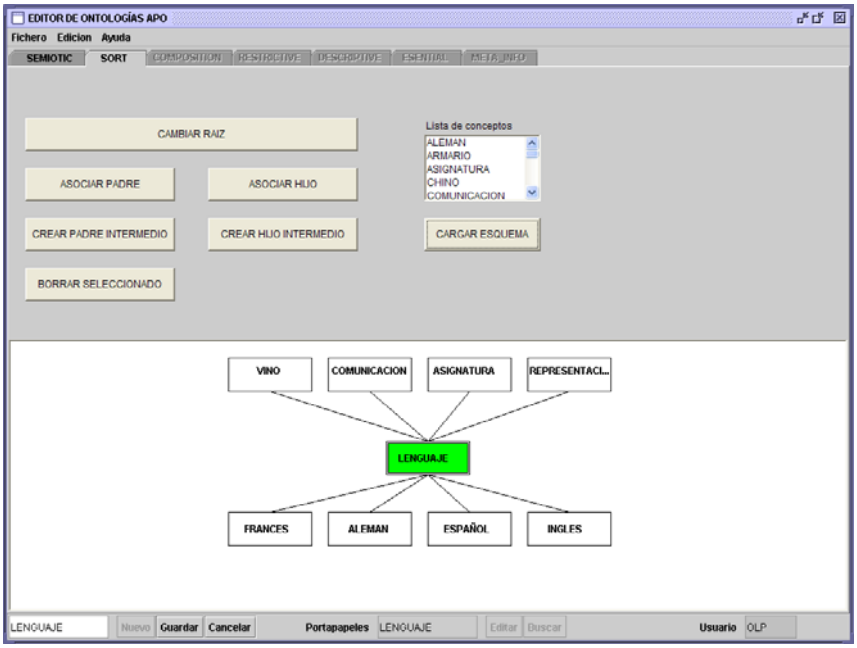
- Se compone de una tabla (JTable) que ha sido modificada para que permita chequeos en sus datos de la columna “FAVORITO” (permite asignar como favorito un término a un concepto). Esta tabla muestra todos los términos relacionados con un concepto. Se han sobrescrito sus métodos para cambiar tamaño de columnas, permitir check en una de sus columnas, controlar que sólo exista un favorito por lenguaje en ese momento etc.
- Permite añadir nuevos términos al concepto, y una vez se pulsa asignar pasan a la tabla.
- Permite eliminar términos seleccionando el término y pulsando desasignar.
- Cuando se crea un concepto nuevo esta tabla está vacía, y ha de introducirse al menos un término para guardar el concepto.
- Se añaden unos “Choice” que son listas desplegables en las que se pueden seleccionar un elemento de un listado para seleccionar lenguajes, peso y hacer los filtrados. Los lenguajes serán cargados cuando arranque la herramienta, para evitar accesos a la base de datos, por lo que cualquier cambio de lenguajes obliga a reiniciar el sistema.
- Permite seleccionar varios filtros a aplicar en la tabla de términos.

La ventana Clasificativa o sort, muestra en la parte inferior un canvas (lienzo de dibujo) en el que se muestra en una jerarquía a dos niveles (padres e hijos directos) las relaciones con respecto a un concepto. En la parte central del diagrama se encuentra la raíz de la jerarquía.

- Permite añadir relaciones de conceptos (para ello ha de existir un concepto en el portapapeles.
- Permite borrar relaciones, tras seleccionar varios conceptos (existe un escuchador que detecta los movimientos del ratón, y al pasar por encima de un recuadro y si hacemos click, pondrá en otro color el recuadro), y al seleccionar borrar eliminará las relaciones seleccionadas y

actualizará el gráfico (se hace una llamada al método repaint() de la clase canvas que borra el lienzo y vuelve a cargarlo con las modificaciones.

- Dispone de una barra de desplazamiento para el "lienzo", ya que puede darse la situación de ser muchas las relaciones existentes.
- Por último permite seleccionar uno de los conceptos relacionados y hacer que ese pase a ser a raíz del esquema. Se han sobrescrito sus métodos para redibujar y poder utilizar como botones los nodos creados (apareciendo seleccionados y deseleccionados).



5.6 Sort Dimension

Toda la comunicación con la base de datos se hace mediante el empleo de un driver ODBC, que permite realizar la conexión con la base de datos. Mediante un objeto de la clase Statement se ejecuta la sentencia SQL correspondiente, y los resultados son recogidos en un objeto de la clase ResultSet, que a modo de lista puede ser recorrido y extraídos los datos que sean oportunos (así como modificarlos). Por cada acceso es aconsejable realizar una conexión (figura 5.7) mediante instrucciones del tipo de la figura.

En primer lugar indicamos el nombre del driver a utilizar, y posteriormente la dirección en la que está dicho driver, incluido usuario y contraseña. Después se crea la consulta correspondiente, y por último se cierra para liberar recursos.

```
try{
            Connection cn;
            Statement st;
            Class.forName("oracle.jdbc.driver.OracleDriver");
            cn=DriverManager.getConnection("jdbc:oracle:thin:@localhost
            :1521:XE","system","sebas");
            st=cn.createStatement();
            String tsql;
            tsql="Insert into NN values('AA')";
            st.execute(tsql);
            cn.close();
        }
```

5.7 Ejemplo de Conexión

5.2. Diagrama de Clases de la Aplicación

El diagrama de clases sirve para visualizar las relaciones entre clases que involucran el sistema. Está compuesto de clase y relaciones. Una clase es la unidad básica que encapsula toda la información de un Objeto (un objeto es una instancia de una clase). A través de ella podemos modelar el entorno de estudio y se representa con una caja con tres subdivisiones.

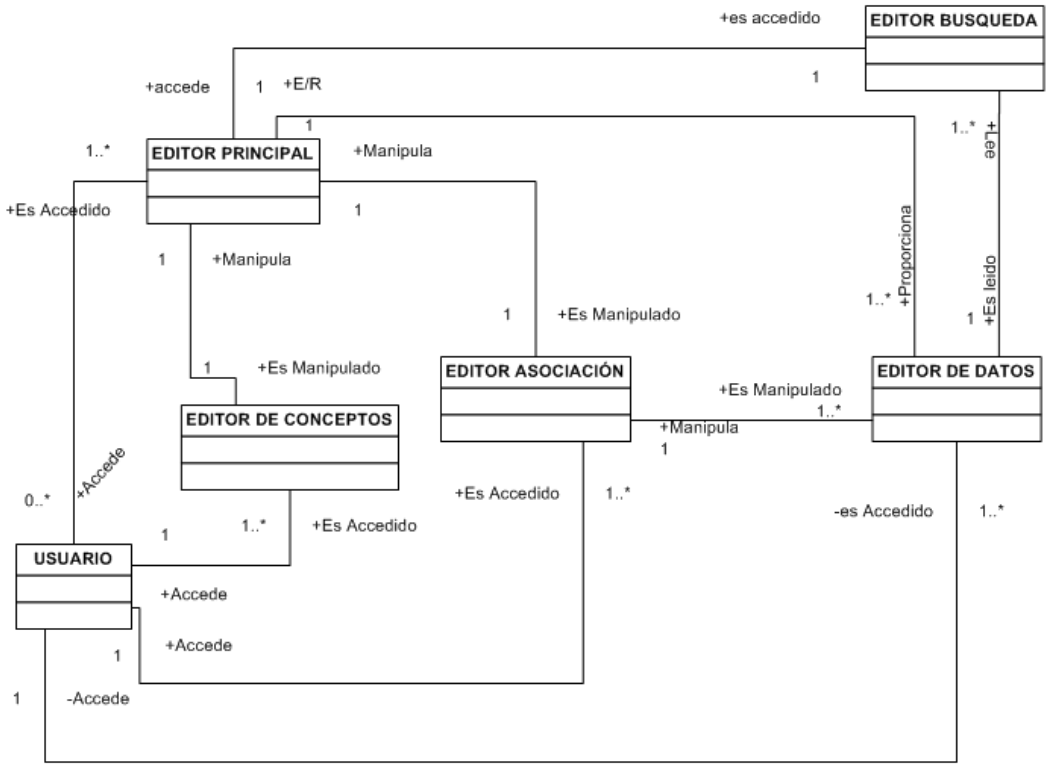
Según la funcionalidad de las clases que componen el sistema se agrupan en las siguientes clases (abstracción), donde cada una de estas clases se corresponde con varias de las creadas finalmente.

- Clase Editor principal. Recopila las clases relacionadas con la creación y manipulación de la pantalla principal del programa.
- Clase Editor de datos. Recopila las clases relacionadas con el acceso a la base de datos.
- Clase Editor de Conceptos. Recopila todas aquellas clases relacionadas con la creación y manipulación de la ventana semiótica.
- Clase Editor de Asociaciones. Recopila todas aquellas clases relacionadas con la creación y manipulación de la ventana Sort.
- Clase Editor de Búsqueda. Recopila todas aquellas clases relacionadas con la creación y actividades proporcionadas por la ventana de búsqueda.

Mediante el diagrama de Clases siguiente (figura 5.8), donde las clases son abstracciones de otras, como ya he comentado, y mediante las relaciones que las unen (también abstracciones de las relaciones posibles), representamos las relaciones que involucran al sistema.

El diagrama es una abstracción del final, ya que debido al conjunto de clases creadas para la implementación se complica y hace ilegible un cómputo de clases mayor, sobre todo por la cantidad de interacciones que se dan entre clases. Con este esquema podemos aclarar para cada clase las

principales funciones que desempeña. Para ello solo tenemos que ver los escenarios descritos en el tema de Análisis y Diseño.



5.8 Diagrama de Clases

5.3. Implementación de la Base de Datos

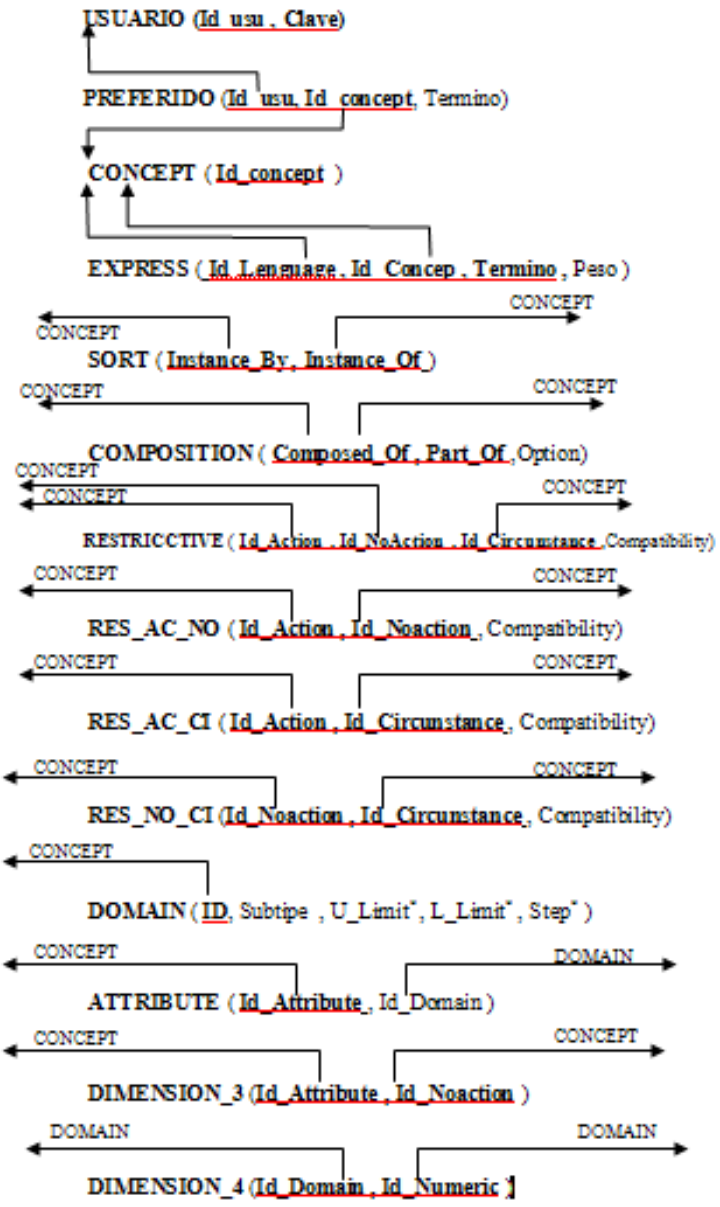
El SGBD que se va a utilizar es el de Oracle, que se basa en un lenguaje relacional (SQL), por lo que se necesitará una descripción relacional de la base de datos.

Para la implementación de la base de datos se ha construido el modelo relacional partiendo del diagrama de clases mostrado con anterioridad en la fase de análisis y diseño, se partió en primer lugar desde el esquema entidad relación, pero se complicaba demasiado por la cantidad de relaciones que existían, por ese motivo se creó un diagrama de clases para la base de datos por cada dimensión de la ontología.

El modelo relacional muestra cada tabla de la base de datos y sus atributos, en el esquema siguiente (figura 5.9) se representan las claves primarias (identificadores de cada tupla), mediante atributos subrayados (en color rojo) y mediante flechas se representan las claves ajenas, para no complicar el diagrama, algunas de estas líneas no llegan a su destino (para no hacerlo engorroso), indicando en ellas a qué tabla hacen referencia.

Para resolver cada una de las dimensiones se resuelve de la siguiente manera:

- Semiotic Dimension. Se resuelve con la tabla EXPRESS. Esta tabla va a representar la relación ternaria. Donde el atributo "peso" contempla un valor entre 0 y 100 que plasma que peso (relación) tiene un término con respecto a un concepto en un lenguaje determinado. Un lenguaje es un concepto en sí mismo.



5.9 Diagrama Relacional

- Sort Dimension. Para resolver esta situación se ha creado la tabla SORT.
- Composition Dimension. Para este caso se crea la tabla COMPOSITION, donde el atributo "part Of" ha de ser "UNIQUE", ya que según el esquema un concepto únicamente puede ser parte de otro.

  Para resolver cada una de las dimensiones se resuelve de la siguiente manera:

- Restrictive Dimensión. Para este caso emplearemos las tablas RESTRICTIVE, que plasman la ternaria completa (el aso de existir un “Action”, un “No-Action” y un “Circumstance”).La tabla RES_AC_NO, que plasma el caso “Action” y “No-Actin”. La tabla RES_AC_CI , que plasma el caso “Action” y “Circunstance”.La tabla RES_NO_CI, para el caso “No_Action” y “Circunstance”.

- Descriptive Dimension. Este es el caso más complejo, pero lo que se ha resuelto de la siguiente manera.

    - DOMAIN. Referencia a concepto (ha de ser el concepto del tipo “domain”), este a su vez tiene un subtipo (compound, enumerated y numeric) y los atributos upper, lower y step serán opcionales, únicamente se rellenarán en el caso de ser del subtipo “Numeric”.

    - ATTRIBUTE. Referencia a un concepto “Attribute” y además ha de tener un atributo dominio (ya que es 1..1). Por lo tanto en la tabla concepto ha de haber un concepto del tipo atributo que además está en la tabla ATTRIBUTE.

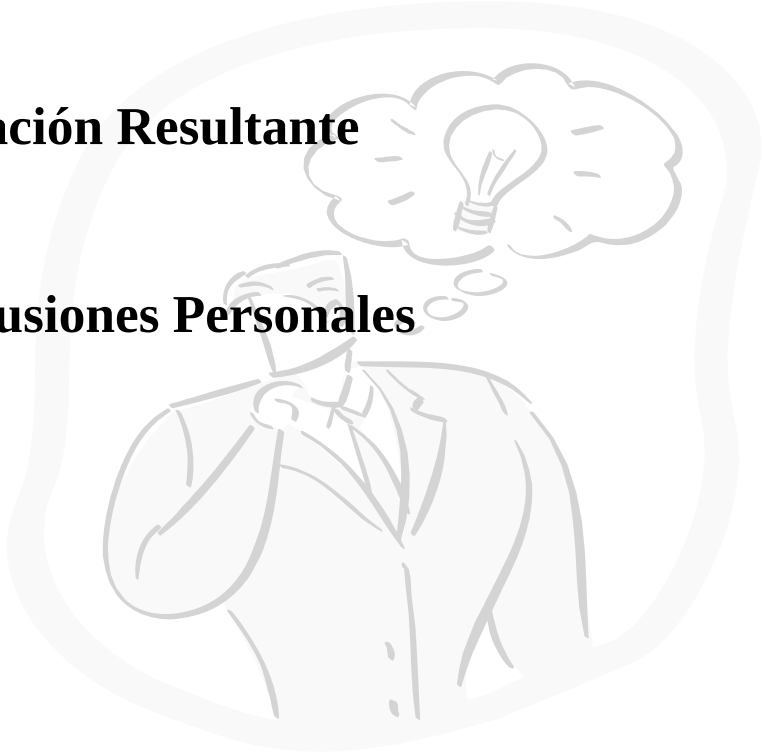
    - DIMENSION_3 (Id_Attribute , Id_Noaction). Plasma la relación entre un concepto “Attribute” y otro “No_action”.

    - DIMENSION_4 (Id_Domain , Id_Numeric) . Reflejo el caso de la relación entre un dominio u un numeric. (relación entre dos dominios, donde uno de ellos ha de ser un numeric)

En último lugar se añaden las tablas USUARIO y PREFERIDO, donde únicamente es de mencionar la utilidad e la tabla PREFERIDO, la cual recoge por cada concepto un término preferido (le falta el atributo Idioma, ya que para cada idioma un concepto tendrá un término favorito distinto, aunque no aparece en el esquema es algo que se ha implementado a última hora).

Conclusiones

- **Aplicación Resultante**
- **Conclusiones Personales**



6. Conclusiones

En este capítulo se va a realizar una valoración del trabajo realizado y los resultados conseguidos con la realización de este proyecto. En primer lugar se hace un comentario sobre la aplicación obtenida y su funcionalidad, para posteriormente realizar una valoración a un nivel más personal.

6.1. Aplicación Resultante

El resultado del trabajo realizado durante este periodo ha dado como fruto la aplicación que se ha tratado de explicar en los capítulos anteriores. Se considera que con los resultados obtenidos se han satisfecho los objetivos iniciales planteados en el nacimiento del proyecto. Han sido multitud de cambios y modificaciones las que se han producido, y se ha pretendido crear los pilares de esta herramienta, con la finalidad de desarrollarla y posiblemente mejorarla en futuros proyectos. Finalmente se ha conseguido desarrollar una herramienta capaz de proporcionar al usuario un editor de ontologías, con la que se pueda crear tanto las relaciones como características de los componentes que integran una ontología. Es un sistema seguro, amigable y fácilmente modulable. Se dejan las puertas abiertas para multitud de funcionalidades nuevas a implementar. De la misma forma se ha establecido un esquema ontológico representado en una base de datos, lo que facilita la adaptabilidad de cualquier ontología poblada para poder ser reutilizada en el sistema.

6.2. Conclusiones Personales

La realización de este proyecto ha supuesto un reto personal. La idea de poder crear una herramienta que facilitase la población y edición de ontologías, así como sus posibles usos en líneas futuras hicieron que me decantase por este proyecto. Pero no solo la idea me debía convencer, se me planteaba otro reto, el tener que trabajar en Java, lenguaje no había visto en mi vida, y me suponía una experiencia adquirida a la finalización de mi trabajo.

Fue Java una de mis primeras barreras a lo largo de este periplo. Tuve que dedicar mucho tiempo a familiarizarme con el lenguaje, a la vez que me decantaba por una herramienta que me facilitase crear mi código de forma segura y amigable, y es aquí donde surgió mi segunda barrera. Dediqué gran tiempo a conocer Java y ha familiarizarme con Eclipse, leí multitud de documentación e hice cientos de pruebas. Aunque todas estas trabas ralentizaron el trabajo, de la misma forma suponían la adquisición de nuevos conocimientos y un enriquecimiento personal.

Al mismo tiempo que me peleaba con Java y la herramienta Eclipse, tenía que ir dando forma a mi sistema, buscando información, leyendo documentación de otros expertos y valorar otras aplicaciones existente (sin salirme de lo especificado en los requisitos establecidos), con la finalidad de poder centrarme y comenzar el desarrollo del proyecto.

Han sido muchas horas pegado a mi ordenador, y un gran sacrificio a nivel personal y familiar, pero cada día que avanzaba un poco veía mas cerca el desenlace de esta historia. La parte más destacable ha sido la cantidad de documentación leída y analizada, ya que ante todo se han querido consolidar los cimientos de la herramienta, con la finalidad de facilitar el trabajo futuro para otras personas. Ha habido muchos vuelcos y virajes durante las primeras fases de análisis y diseño, debido a las limitaciones de las herramientas existentes. Han sido muchas las personas a las que he tenido que recurrir para documentarme, tanto docentes de la universidad como personas ajenas a ella, y a todos ellos he de agradecer la ayuda prestada.

No puedo más que decir que me ha parecido un proyecto interesante, y que me gustaría en un futuro cercano seguir trabajando en entornos similares, ya que son muchas las posibilidades que se despliegan de este trabajo. Considero por último que es un proyecto muy completo, en el sentido de que para su desarrollo he tenido que poner en práctica muchos de los conocimientos adquiridos en la carrera, como diseño e implementación de bases de datos, programación orientada a objetos, diseño software, ect .

Líneas Futuras

7. Líneas Futuras

El sistema es muy amplio y conllevaría varios proyectos o incluso una tesis. La parte implementada es bastante completa y supone todos los principios para el posterior trabajo. Este proyecto ha llevado una gran parte de investigación, siendo el desarrollo menos vistoso. Aún así se ha dejado definida la estructura de las ontologías a emplear, así como varias de las dimensiones diseñadas. Con todo esto lo que pretendo es detallar la necesidad de mejoras y nuevos perfeccionamientos que se han planteado e incluso que se podrían considerar. Es una herramienta que da mucho juego a la hora de insertar nuevos módulos, e incluso mejorar los existentes.

En primer lugar, una de las primeras modificaciones que se le pueden aplicar al sistema, sería la opción multilingüe. Con ello se pretende dotar a la herramienta de nuevos atractivos para su futura comercialización (si fuera el caso), e incluso para su utilización de investigación. Esta mejora consistiría en la posibilidad de poder seleccionar entre varias lenguas, y que tanto menús, botones, etc. Cambiasen según el idioma seleccionado. Conlleva quizás no solo un cambio de etiquetas, sino también de dimensiones de algunos de los objetos visibles por pantalla, para que se respete la estética del producto.

Siguiendo por la línea de mejoras se puede considerar el hecho de aplicar seguridad a las claves utilizadas por los usuarios, **[Anexo D]** .Utilizando el sistema de encriptación descrito en el anexo, y realizando ciertas mejoras se proporciona mayor seguridad, pudiéndose incluso habilitar permisos específicos a la hora de borrados o modificación de las bases de datos a las que se acceden (independientemente de los permisos proporcionados por el administrador de la base de datos).

Seguimos con las posibles mejoras, y las más importantes antes de nada sería terminar del todo este sistema. Aún quedan varias ventanas de las dimensiones por desarrollar, y aunque el esquema de la base de datos ya se encuentra afianzado, deben de introducirse mejoras de control de datos, ya sea mediante chequeos y disparadores, o bien mediante módulos en Java que analicen y valoren la información existente y la introducida.

Si continuamos por el camino referente a la terminación del editor ontológico, se debe dar funcionalidad completa a la barra de menú. Se debe permitir la opción de poder salvar y cargar ontologías o partes de ellas, se podrían añadir opciones respecto a las vistas de menús, configuración, impresión, conexión (en caso de habilitar nuevas propiedades de conexión), ayuda etc.

Retomando lo comentado en el párrafo anterior, con respecto a la carga de ontologías, se debería crear un módulo que permita importar información para poblar las ontologías desde ficheros externos. Estos ficheros deberían estar en un formato estándar, sería adecuado XML, y permitirían poblar fácilmente la base de datos. Estos ficheros a su vez podrían ser creados por otras herramientas

ontológicas, e incluso sería una buena idea para un futuro proyecto, crear una herramienta que trabaje con nuestro editor para proporcionarle ficheros para poblar la ontología.

Si nos mantenemos en torno a lo anterior se puede sopesar el hecho de crear un módulo que exporte nuestra ontología, sendo múltiples las posibilidades, ya que se podría exportar la ontología en los formatos que quisiésemos, incluso para pasarlos a formatos reutilizables por otros editores o herramientas ontológicas.

Siguiendo las tendencias actuales se debería dotar a este editor ontológico de versatilidad a la hora de trabajar vía Web (con las mejoras de comunicación y seguridad que ello conlleva), no supone mucho esfuerzo, ya que al estar implementado en Java, se puede trabajar con applets. De esa forma nos adaptaríamos a la nueva tendencia de herramientas que permitan el intercambio entre usuarios vía Web.

Siendo ambiciosos se puede considerar por último (aunque la lista de nuevas funcionalidades y mejoras podría ser casi infinita) la implementación de un módulo que permitiese la detección e incluso corrección de errores en las ontologías, e incluso la compatibilidad entre varias a la hora de utilizarlas.

Otros menos importantes serían unificar los objetos de las pantallas en SWING, para evitar problemas de refresco de pantalla y superposición de objetos.

Como se puede ver son muchas las funcionalidades y mejoras que se le podrían añadir y creo que puede suponer un filón para futuros proyectos de compañeros, que espero disfruten tanto, como yo lo he hecho

Bibliografía

8. Bibliografía

[1] J. Heflin. “OWL Web Ontology Language Use Cases and Requirements”. W3C Recommendation. Febrero 2004.

[2] I. Horrocks and P. F. Patel-Schneider. “Three theses of representation in the semantic Web”. In Proc. of the Twelfth International World Wide Web Conference (WWW 2003). 2003

[3] M. Horridge, H. Knublauch et al. “A Practical Guide To Building OWL Ontologies Using The Protégé-OWL Plugin and CO-ODE Tools”. The University Of Manchester. Reino Unido. Agosto 2004.

[4] Philippe Martin. “Knowledge Representation, Sharing and Retrieval on the Web”. Distributed System Technology Centre. Australia. 2002.

[5] S. Bechhofer. “The DIG Description Logic Interface: DIG/1.1”. Febrero 2003.

[6] http://www.cogsci.princetown.edu/ .wn/

[7] L. Aroyo, D. Dicheva. “The New Challenges for E-learning: The Educational Semantic Web”. Educational Technology & Society, 7 (4), Pp: 59-69. 2004

[8] http://www.topicmaps.org

[9] http://citeseer.ist.psu.edu/

[10] http://www.topicmaps.org/xtm

[11] http://www.isotopicmaps.org

[12] http://www.uml.org

[13] Página principal de RDF en W3c. http://www.w3c.org/rdf

[14] http://www.ruleml.org

[15] G. Gupta, E.Pontelli, K.Ali, M. Carlsson, M. Hermenegido. “Parallel Execution of Prolog Programs: A Survey”. ACM Transanctions on

Programming Languages and Systems, Vol 23, Num 4, Pp: 472, 602.

ACM Press, Julio 2001.

[16] Gerd Wagner, Said Tabet, Harold Boley "MOF-RuleML: The Abstract

Sytax of RuleML as a MOF Model". OMG Meeting. EEUU. Octubre

2003.

[17] "Ontological Engineering" Asunción Gómez-Pérez, Mariano Fernández-López, Oscar Corcho Ed. Springer, 2004

[18] "Resource Description Framework". http://www.w3.org/RDF/

[19] "DAML" http://www.cs.man.ac.uk/~horrocks/Publications/download/2002/ieeede2002.pdf

[20] "Language and Computing" http://www.landcglobal.com/pages/linkfactory.php

[21] "Oiled Web site"http://oiled.man.ac.uk/

[22] "The OTK tool repository" http://www.ontoknowledge.org/tools/ontoedit.shtml

[23] "Web Ontology Design Environment " http://webode.dia.fi.upm.es/WebODEWeb/index.html

[24] www.ksl.stanford.edu/software/**ontolingua**/

[25] www.mpi.nl/**DAM-LR**/

[26] BROEDER, D.; CLAUS, A.; OFFENGA, F. [*et al.*] (2006). «*LAMUS*

– the Language Archive Management and Upload System».

http://www.lat-mpi.eu/papers/papers-2006/lamus-paperfinal2.pdf

[27] http://www.fdi.ucm.es/profesor/fernan/fsp/index.html

[28] Pérez Hernández, C. 2000: *Explotación de los Córpora Textuales Informatizados para la Creación de Bases de Datos Terminológicas*. Tesis doctoral. Dpto. de Filología Inglesa. Universidad de Málaga.

[29] http://www.wsmostudio.org/

[30] Moreno Ortíz, 2000: OntoTerm: un sistema abierto de representación conceptual. Actas del XVI Congreso de la SEPLN. Vigo.

[31] Moreno Ortíz, 1999: El Lexicón en la Lexicografía Computacional: Adquisición y Representación de Información Léxica. Alfinge.

[32] Galinski y Picht, H. 1997: Graphic and Other Semiotic Forms of Knowledge Representation in Terminology Management. En Wright, S.E. y Budin, G. eds.: 1997: *Handbook of Terminology Management : Basic Aspects of Terminology Management.* Amsterdam/Philadelphia: John Benjamins: 42-61.

[33] Mahesh, K. 1996: Ontology Development for Machine Translation: Ideology and Methodology. NMSU. Computing Research Laboratory. Technical Report MCCS: 96-292.

[34] Knowles, F. 1989: The Computer in Lexicography. En Hausmann, F. J. et al. eds.: 1989: *Dictionaries: An International Encyclopedia of Lexicography.* Vol. 2. Berlin/Nueva York: Walter de Gruyler: 1645-1665.

[35] Langlois, L. 1996b: *Bitexte, bi-concordance et collocation*. Tesis Doctoral. Universidad de Ottawa. [Documento disponible en la red:langlois] [Consulta: 26 agosto 1998].

[36] Meyer, I. 1996: Refining the terminographer's concept-analysis methods: How can phraseology help?. Terminology 3 (1): 1-26.

[37] Tercedor Sánchez, Mª I. 1999: *La Fraseología en el Lenguaje Biomédico: Análisis desde las Necesidades del Traductor.*

[38] Teubert W. 1999: Corpus Linguistics -A Partisan View. International Journal of Corpus Linguistics 4 (1).

[39] Heid, U.1998/99: A Linguistic Bootstrapping approach to the Extraction of Term Candidates from German Text. Terminology vol. 5(2): 161-181.

[40] Ramos, L. y Villarroel, O. (2007). Estudio teórico sobre Objetos de Aprendizaje, Ontologías y Web semánticas en el marco del Proyecto AMBAR (Generador de AMBientes Constructivistas de Enseñanza - ApRendizaje). Seminario de Base de Datos. Escuela de Computación. Facultad de Ciencias. Universidad Central de Venezuela. Caracas, Venezuela.

[41] Página de Protégé http://protege.stanford.edu/ accedido a fecha de Julio de 2009.

[42] Página de Swoop http://www.mindswap.org/2004/SWOOP/

[43] [Sensus, 1994] “Ontology Creation and Use: SENSUS”

http://www.isi.edu/natural-language/resources/sensus.html

[44] [CyC, 2002] “Cyc Corporation”

http://www.cyc.com/

[45] "Ontology research and development". Part 1-a review of ontology generation. Journal of Information Science, 2002, vol 28, nº 2, p. 123-126)

[46] DAML+OIL "www.w3.org/TR/daml+oil-reference"

[47] Moreno Ortiz 1999:13 "http://elies.rediris.es/elies19/cap232.html"

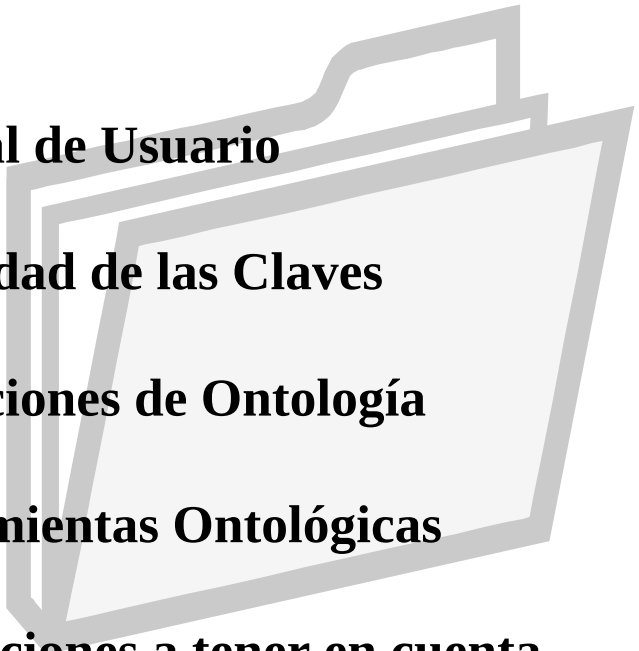
[48] Langlois 1996b: 10 http://ideas.repec.org/e/pla2.html

[49] Calle F.J,Castro E., Cuadra, D. "Ontological dimensions applied to Natural Interation"

This paper appears in: Ontologies in Interactive Systems, 2008. ONTORACT '08. First International Workshop on**.**

http://www2.computer.org/portal/web/csdl/doi/10.1109/ONTORACT.2008.11.

Anexos



- **Manual de Usuario**
- **Seguridad de las Claves**
- **Definiciones de Ontología**
- **Herramientas Ontológicas**
- **Instalaciones a tener en cuenta**
- **Ejecución**

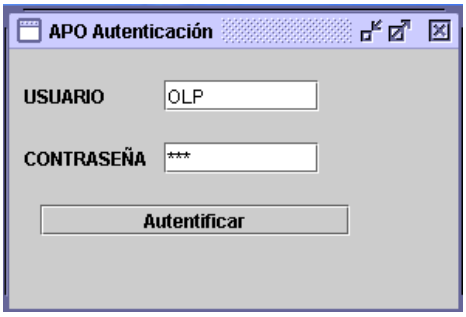
(Anexo A) Manual de usuario

Introducción

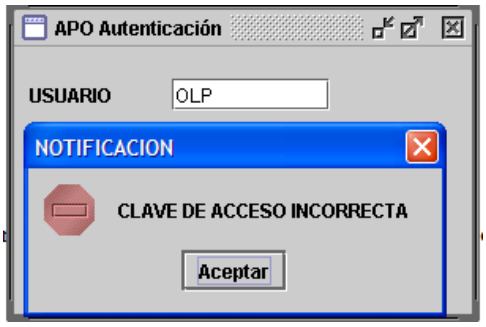
En esta documentación se describen el funcionamiento del editor ontológico APO. Debido a que es un amplio proyecto, en estas líneas se explican dos de las dimensiones de las que se compone la herramienta. Para poder trabajar se ha de crear una base de datos en Oracle por defecto con la clave "sebas".

Autenticación

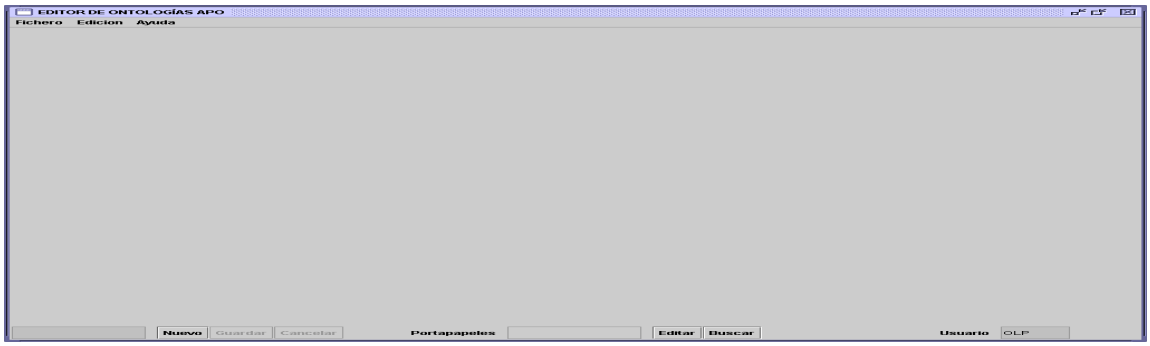
Una vez instalada y configurada nuestra máquina, iniciamos APO y nos muestra en primera instancia la ventana de autenticación.



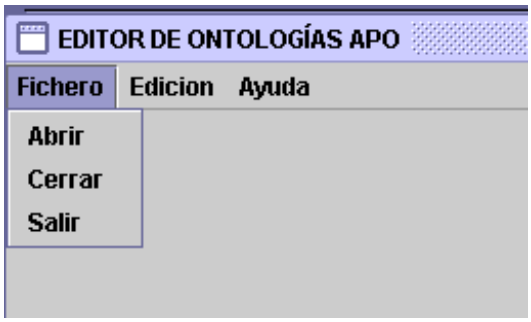
Se ha de introducir un usuario y contraseña válidos, en caso contrario nos mostrará un error de acceso.



En caso de que la autenticación sea correcta se mostrará un mensaje de bienvenida y se mostrará la ventana principal del programa.



Esta ventana se caracteriza por un menú en la parte superior y una barra de botones en la inferior. El menú no está operativo, salvo fichero> Salir, que permite cerrar la herramienta.



Barra de botones

A continuación se describen los elementos que componen la barra inferior y su utilización, que depende de la ventana en la que nos encontremos.

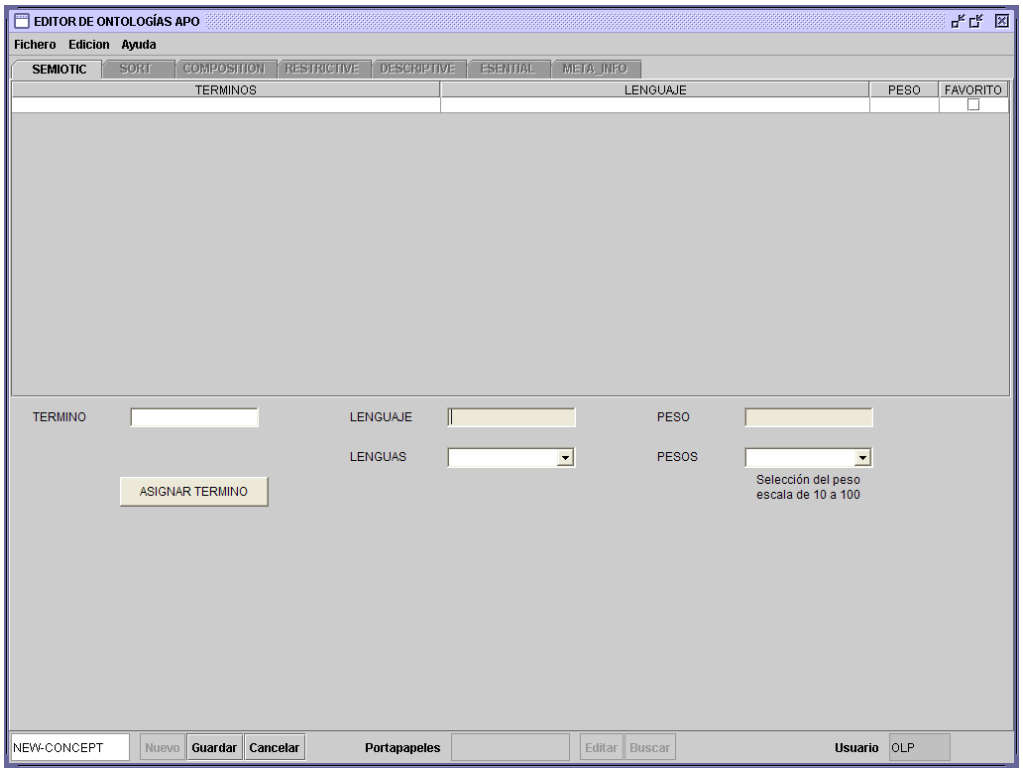
De izquierda a derecha estos son los elementos:

- Un campo de texto que nos indicará en cada momento con qué concepto trabajamos, ya sea uno nuevo o uno ya existente. Es un campo de texto no editable.
- El botón nuevo desplegará la ventana semiótica para crear un nuevo concepto.
- El botón guardar tiene como finalidad pasar a la base de datos los nuevos elementos creados (en esta ventana de inicio no está activo)
- El campo de texto portapapeles mostrará un elemento buscado en la base de datos sobre el que podemos operar. Es un campo no editable.
- El botón editar permite modificar el elemento que hay en el portapapeles, en caso de no haber un concepto en el portapapeles nos avisará de que no es posible dicha operación.

- El botón “Buscar” desplegará una ventana emergente que nos permitirá seleccionar un concepto o hacer una búsqueda sobre el término que busquemos.
- El campo de texto “usuario” muestra el usuario que se encuentra conectado.

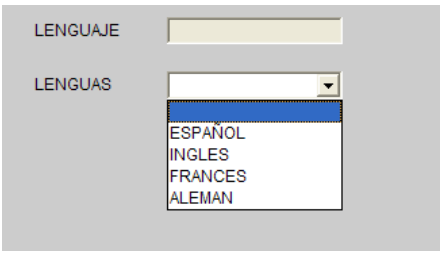
Nuevo concepto

Desde la ventana principal pulsamos el botón “Nuevo”, desplegándose la ventana correspondiente a la creación de un nuevo concepto.

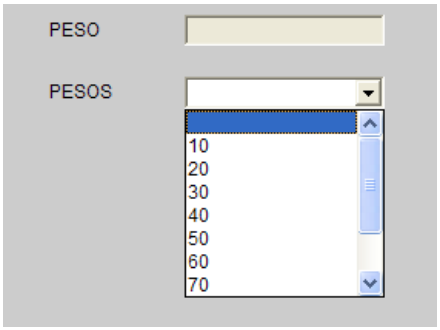


Cuando se despliega esta ventana se muestra una tabla en la que iremos añadiendo los términos relacionados con el nuevo concepto. Para poder crear un concepto se debe al menos asignarle un término, en caso contrario no permite crear el concepto. La secuencia de creación es la siguiente:

- Introducir el término a asignar en el campo de texto “Termino”.
- Seleccionar uno de los lenguajes disponibles desde el listado “Lenguas”

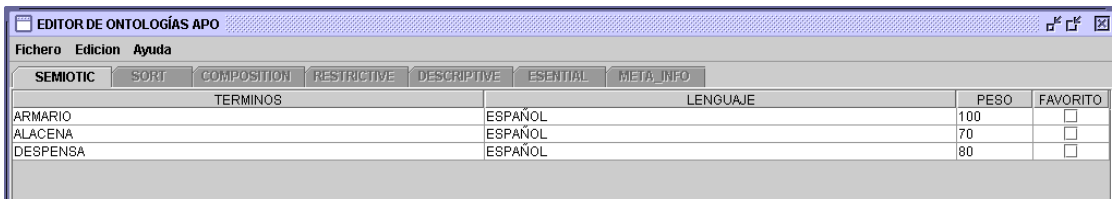


- Seleccionar un peso, que indica en una escala de 10 a 100, siendo 10 el menor y 100 el máximo valor, la afinidad del concepto con ese término.



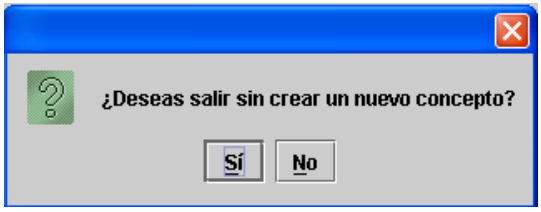
- Una vez seleccionados todos los datos en los campos de texto se ven los datos seleccionados para cada caso.

- Pulsaremos el botón “asignar término”, en caso de no haber alguno de los datos requeridos, el programa muestra el error correspondiente. Si todo está ok subirá el término a la tabla.



- En la tabla de términos se observa una columna “FAVORITO”, que no se encuentra activa en esta ventana, si probamos a pulsar veremos que no hace nada. Su utilización será en el caso de la edición de conceptos.

- Para guardar todo se ha de pulsar el botón “GUARDAR”, es entonces cuando se crea el nuevo concepto y se le asignan los términos favoritos, para cada lengua, según las entradas hechas en la tabla.
- En caso de pulsar “CANCELAR” el sistema nos preguntará si deseamos salir sin crear el concepto, si pulsamos “SI” el sistema vuelva a la ventana principal, en caso contrario muestra la ventana de creación de conceptos tal cual la teníamos.

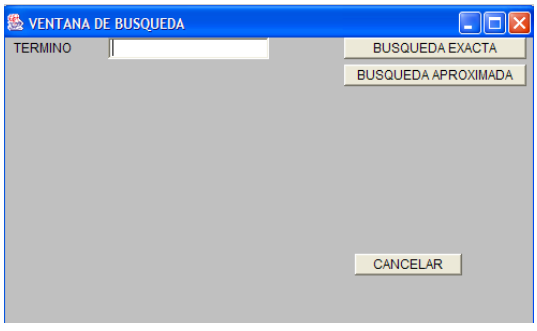


- Si pulsamos “GUARDAR” se pasan a la base de datos la información introducida y volvemos a la ventana principal.

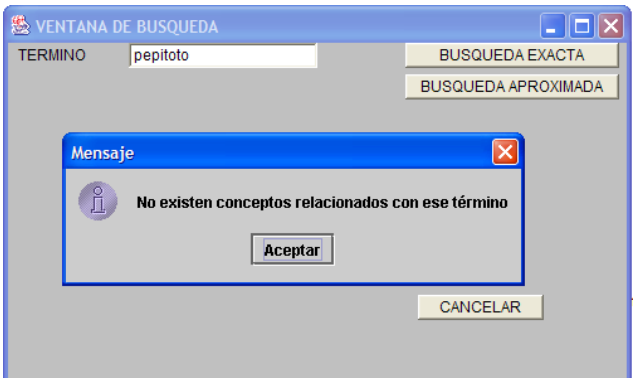
Búsqueda de concepto

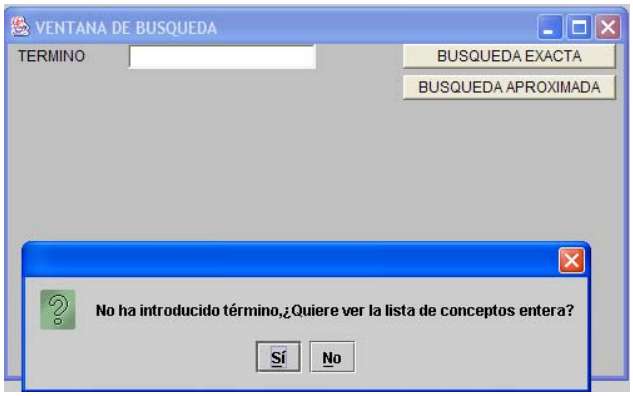
Esta ventana sirve para ver la información sobre los conceptos y sus términos, así como buscar un concepto a través de uno de sus términos. Desde esta ventana podemos pasar un concepto al portapapeles para poder editarlo (en caso de partir desde la ventana principal), o de pasar el concepto al portapapeles auxiliar en la ventana de edición Clasificativa o Sort. Para acceder a esta ventana se ha de pulse el botón “BUSCAR”.

Lo primero que muestra esta ventana en un campo de texto editable que nos permite introducir el término que queremos buscar. Además muestra dos botones de búsqueda y uno de cancelación.



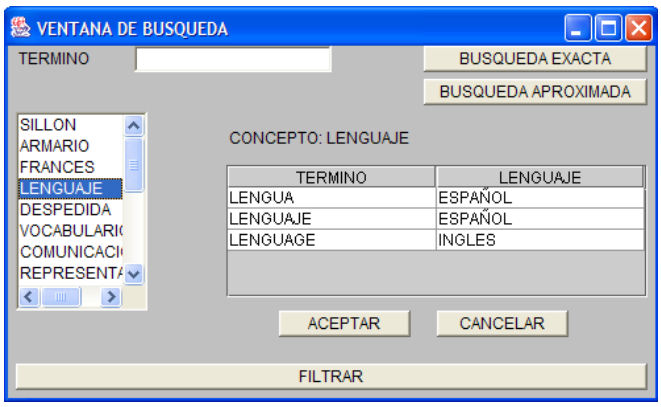
- El botón “BUSQUEDA EXACTA” permite buscar los conceptos relacionados con el término introducido. En caso de no haber ninguno lo indicará, y en caso de no meter ningún valor en el campo de texto nos dará lo opción de poder ver todos los conceptos de la base de datos.





- El botón de cancelación permite salir de la ventana y volver a la ventana principal.
- El botón "BUSQUEDA APROXIMADA" realiza la búsqueda de aquellos conceptos cuyo término contenga la palabra introducida en el campo de texto.

Una vez hecha una búsqueda la ventana cambia apareciendo de la siguiente manera:



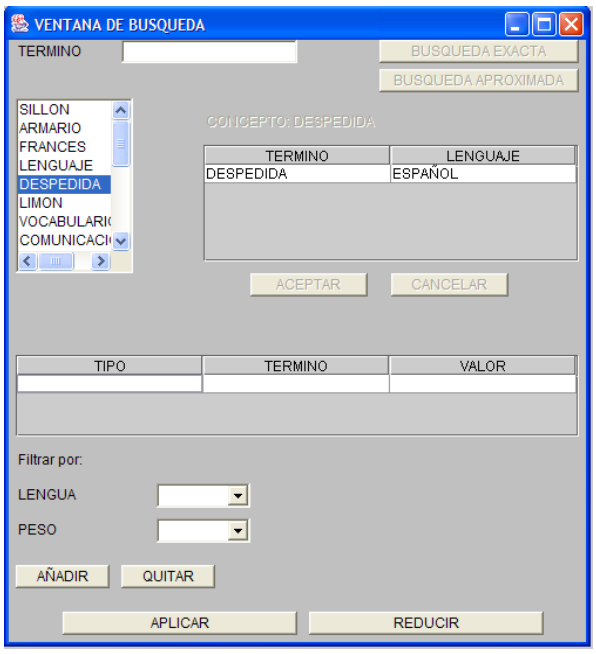
Donde en la ventana se ven de izquierda a derecha y de arriba abajo os siguientes elementos:

- Una lista de los conceptos que se han encontrado relacionados con el término buscado.

- Una tabla que contiene los términos relacionados con el concepto que seleccionemos de la lista anterior, según vamos seleccionando los conceptos de la lista anterior se nos muestran sus términos relacionados.
- El botón “ACEPTAR”, que al presionarlo, y si hay algún concepto seleccionado de la lista anterior, lo pasa al portapapeles de la ventana principal y cierra la ventana de búsqueda.
- El botón filtrar, que sirve para afinar la búsqueda del concepto que buscamos, al filtrar los términos relacionados con cada uno de los conceptos, es útil para el caso de listados grandes.

En caso de pulsar “FILTRAR” se expande la ventana de búsqueda. Se pueden ver los siguientes objetos en la ventana:

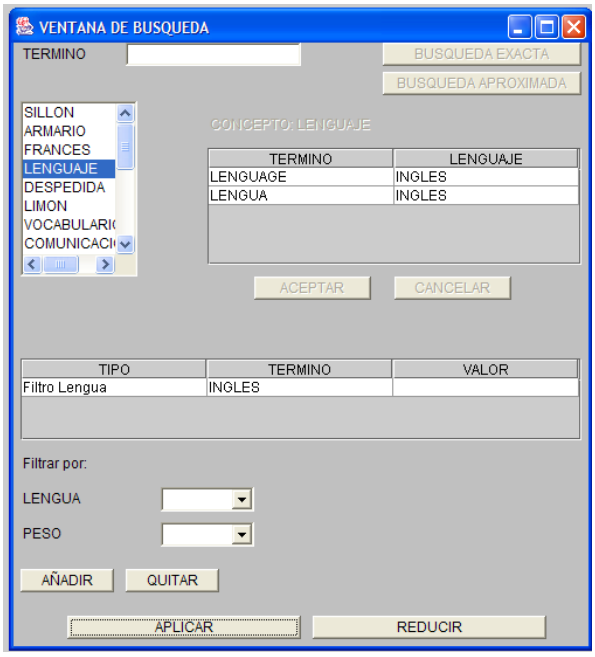
- Una tabla en la que se mostrarán los filtros aplicados.
- Una lista con el filtro de idioma a elegir.
- Un botón “AÑADIR” que permite añadir el filtro, en caso de estar repetido nos lo indicará.
- Un botón “QUITAR” que permite eliminar el último filtro añadido.
- Un botón “APLICAR” que aplica los filtros a la tabla de términos del concepto.
- Un botón “REDUCIR” que vuelve la ventana a su estado anterior.



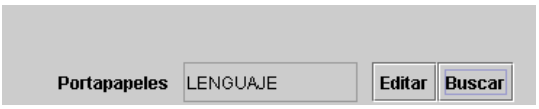
Mientras está en esta fase no pueden hacerse búsquedas ni seleccionar los conceptos del listado hasta que se pulsa “REDUCIR”.

De esta forma para aplicar un filtro a los términos de un concepto, se operaría de la siguiente manera:

Seleccionamos de la lista de lenguas “INGLES”, pulsamos “AÑADIR” y después “APLICAR”, quedando la ventana de la siguiente manera:



Pulsaremos a continuación “REDUCIR” y “ACEPTAR” pasando el concepto a la ventana principal, al portapapeles.

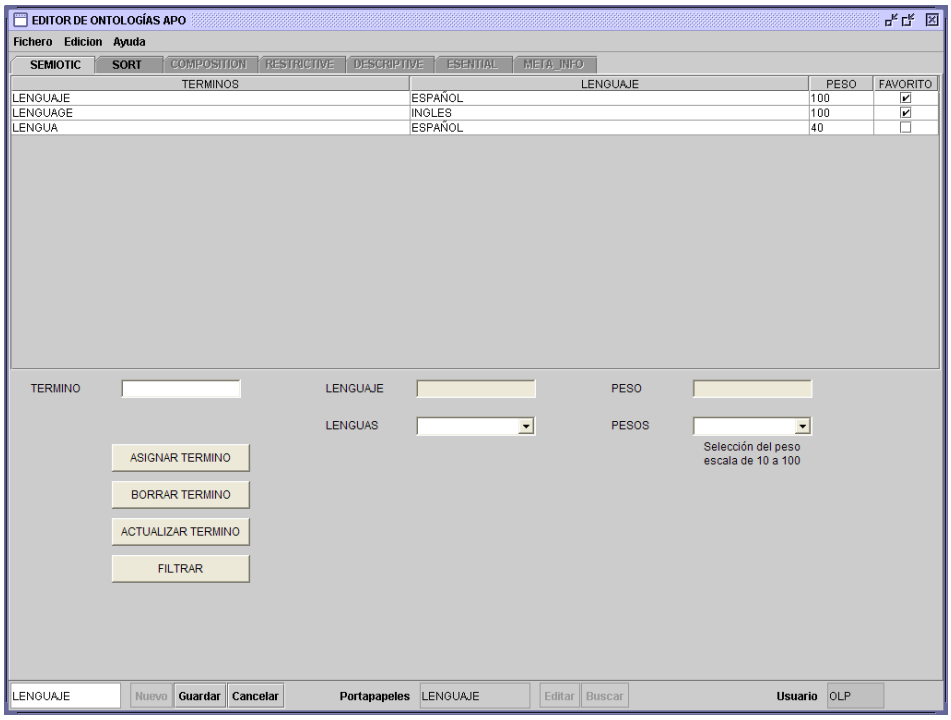


Edición de conceptos

Pulsamos desde la ventana principal el botón “EDITAR”, desplegándose la ventana de edición, que se compone de dos pestañas, una de la ventana “semiotic” y otra de la ventana “sort”. La primera permite modificar, añadir y borrar términos relacionados con un concepto, así como cambiar el término favorito que tiene un concepto. La segunda pestaña permite añadir y borrar las relaciones existentes entre conceptos existentes en la base de datos. Los cambios que se realicen tienen efecto directamente en la base de datos sin tener que guardar.

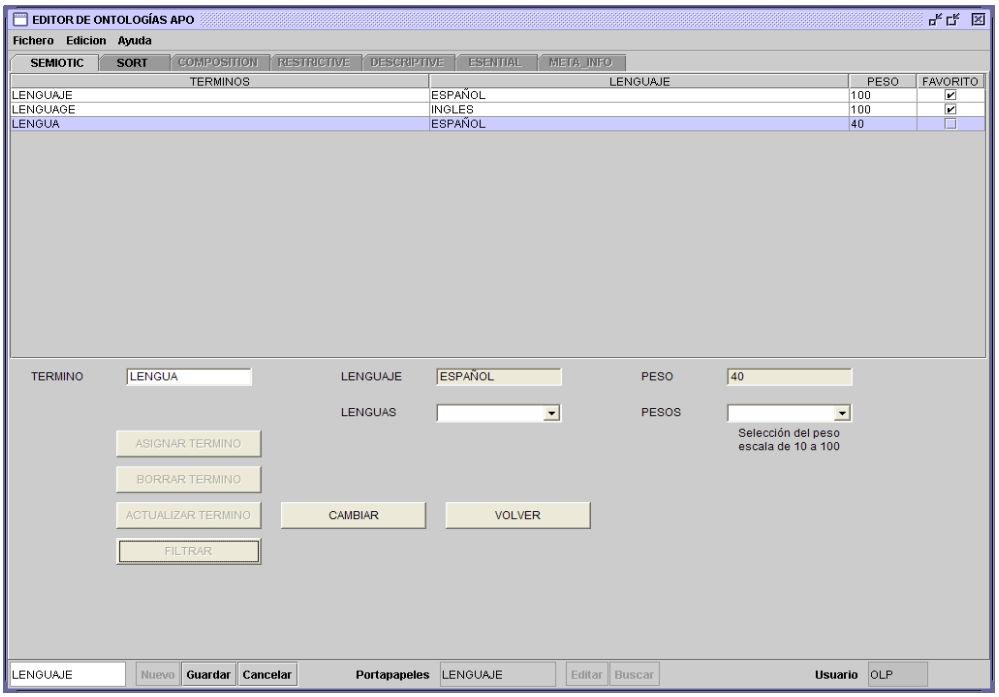
Edición desde la ventana “semiotic”

Si seleccionamos la pestaña “SEMIOTIC” se despliega la siguiente ventana.

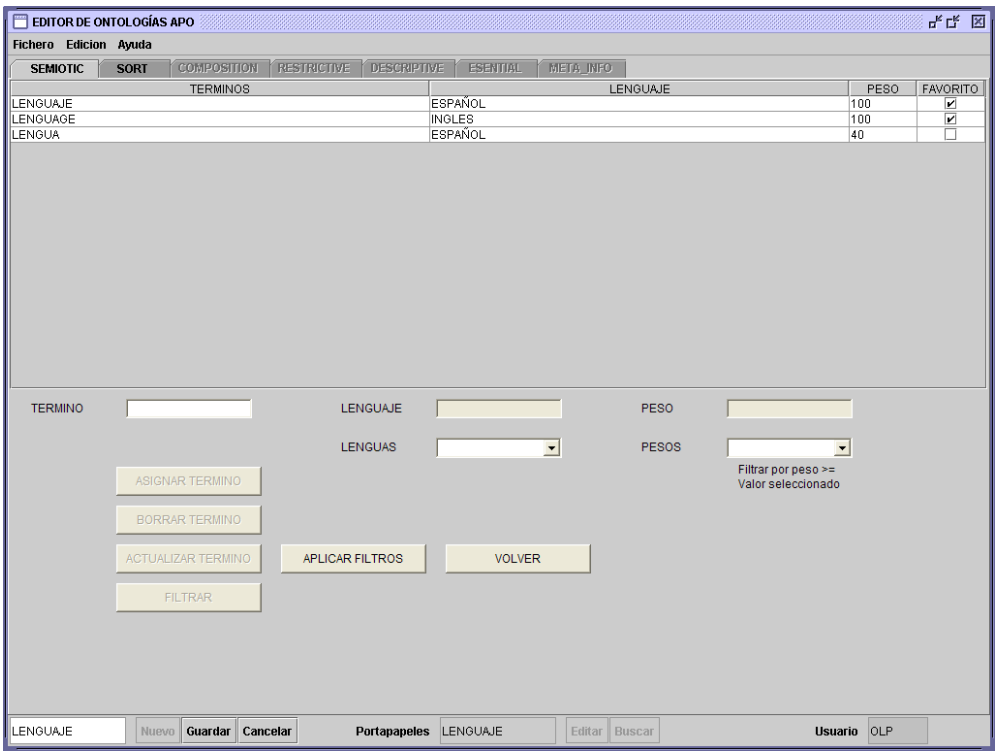


Se pueden ver los siguientes objetos de izquierda a derecha y de arriba abajo (se explican solo aquellos que difieran de la ventana de creación de un concepto).

- La tabla de los términos. En ella se muestran todos los términos relacionados con un concepto, además permite editar desde la columna “FAVORITO” los favoritos de un concepto, estos conceptos aparecen marcados, el sistema controla que sólo exista no por lenguaje, y los cambios se producen en tiempo de ejecución, al abrir esta ventana se cargarán los que existan en la base d datos, y si se cambia algo se produce también en la base de datos, sin necesidad de guardar.
- El botón “ASIGNAR TERMINO” tiene el mismo uso que en la ventana de creación.
- El botón “BORRAR TERMINO” permite borrar un término de los existentes en la tabla, para ello ha de seleccionarse una de las filas.
- El botón “ACTUALIZAR TERMINO”, permite seleccionar una de las filas de la tabla de términos y cambiar sus valores. Al pulsarla se muestran unos cambios. Pasará a los campos de texto de término, lenguaje y peso el elemento seleccionado, además se mostrarán uso botones de cambio y cancelación. En caso de hacer una modificación y pulsar “CAMBIAR” se actualiza el valor existente, en caso de pulsar “VOLVER” se retorna a la ventana “semiotic” inicial.



- El botón “FILTRAR” despliega nuevas opciones sobre la ventana “semiotic”.

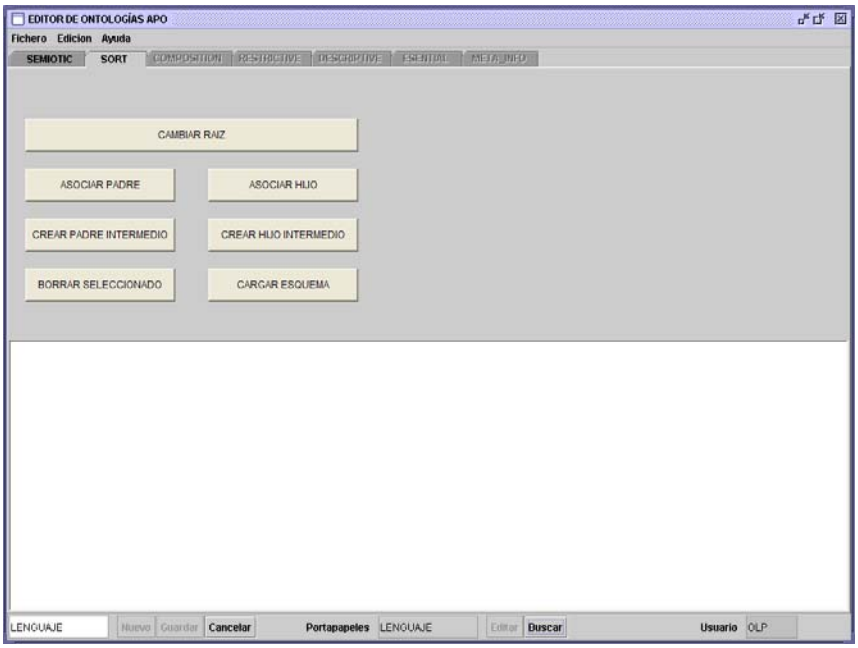


Se podrán elegir aplicar un filtro de lengua y un peso >= que a los términos que existan. Para ello se seleccionan los filtros y el botón “APLICAR FILTROS”, mostrándose la tabla con los filtros aplicados, pero no siendo editable durante este proceso, para volver al estado editable se pulsa “VOLVER” y la ventana pasa al estado anterior.

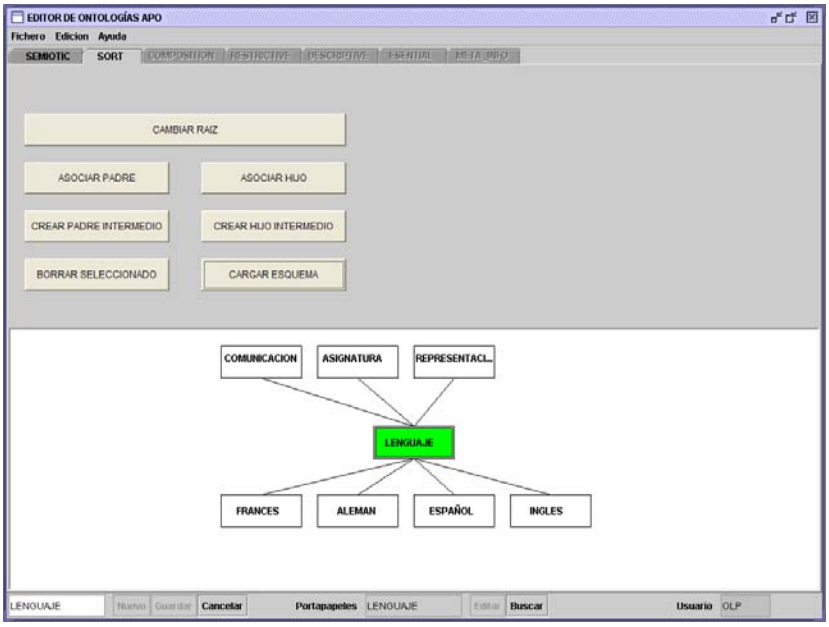
- El botón “GUARDAR” no tiene efecto, ya que los cambios son al instante.
- El botón “CANCELAR” nos indica si queremos salir de la edición, tanto dese esta pestaña como desde la pestaña “sort”, volviendo a la ventana inicial del editor.

Edición desde la ventana “sort”

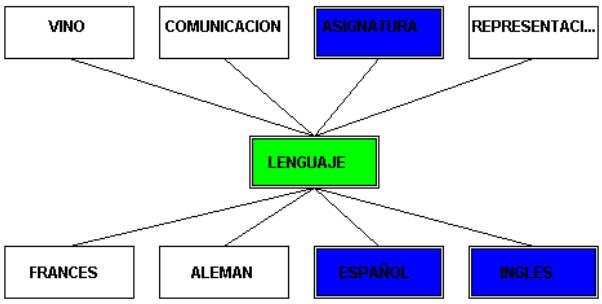
Cuando pulsamos “EDITAR” es la primera pestaña que se despliega, y muestra lo siguiente:



Para poder opera pulsaremos “CARGAR ESQUEMA”, que cargará el esquema de relaciones hijos y padres con el concepto del portapapeles en la parte inferior.



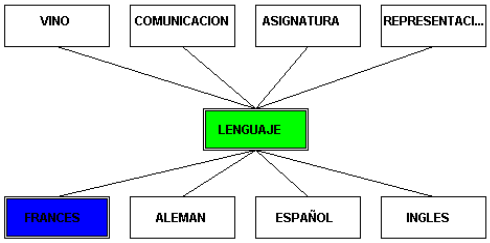
El funcionamiento es sencillo, se selecciona el nodo sobre el que se quiere operar, o el concepto a añadir y se selecciona la operación a ejecutar. Al seleccionar cualquiera de los nodos hijos o padres su color cambia a azul, lo que nos indica que se encuentra seleccionado.



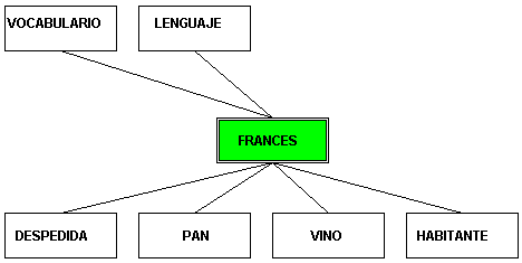
Las opciones disponibles son las siguientes:

- Asociar Padre. Permite añadir una relación padre-hijo (no es una jerarquía en una relación instanciado por-instancia de) nueva. Siendo el padre el concepto seleccionado desde la ventana de búsqueda. y el hijo el concepto central (siempre se mostrará en verde el concepto central).
- Asociar Hijo. Permite añadir una relación padre-hijo (no es una jerarquía en una relación instanciado por-instancia de) nueva. Siendo el padre el concepto del medio (siempre se muestra en color verde el concepto central) y el hijo el concepto seleccionado desde la ventana de búsqueda.
- Asociar Padre intermedio. Su funcionalidad es similar a la de asociar padre, salvo que introduce un nuevo concepto padre entre el concepto central y los conceptos padres seleccionados.
- Asociar Hijo intermedio. Su funcionalidad es similar a la de asociar hijo, salvo que introduce un nuevo concepto hijo entre el concepto central y los conceptos hijos seleccionados.
- Borrar seleccionado. Elimina las relaciones seleccionadas en azul.
- Cambiar raíz. Permite cambiar la raíz central entre uno de los nodos que aparecen, esto nos permite navegar por las distintos niveles de las relaciones. Solo permite aplicarse si se ha seleccionado uno y solo uno de los nodos. Veremos un ejemplo:

  Seleccionamos el nodo “Frances” y pulsamos “CAMBIAR RAIZ”

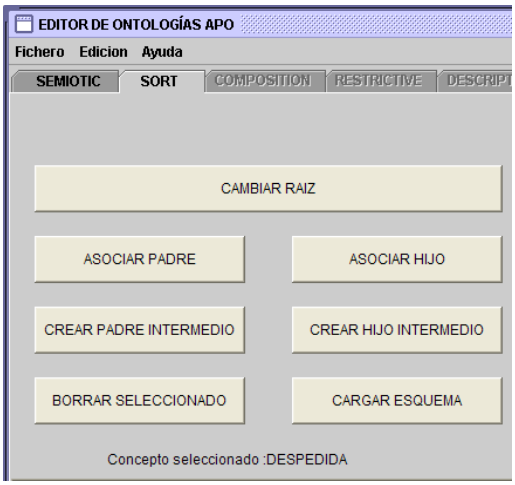


Vemos que cambia el esquema.



De esta forma se puede navegar por los distintos nodos.

Cuando se carga un concepto desde la ventana de búsqueda este pasa a un portapapeles auxiliar debajo de los botones de la ventana, con la forma “Concepto seleccionado XXX”



Nota: El botón “CARGAR ESQUEMA” nos permite volver al esquema inicial desde cualquier punto que nos encontremos, seleccionará el esquema correspondiente al concepto seleccionado en el portapapeles.

(Anexo B) Seguridad de las Claves

Necesidades de seguridad

- ***Encriptación*.** Las claves de usuario se deben guardar encriptadas. No sirve conque no se visualicen en configuración sino que tampoco deben ser visibles en la base de datos. Hay que definir un algoritmo de encriptación para codificar la clave y guardarla codificada.
- ***Privacidad.*** La clave pertenece al usuario y nadie más que él debe conocerla. Por tanto, además del punto anterior el usuario debe poder cambiar la clave cuando lo desee.
- ***Trazabilidad***. La consulta a los informes debe guardar una traza. En la actualidad se tiene traza de las modificaciones realizadas. Las consultas a los datos por medio de la aplicación deben ser registradas.
- ***Caducidad***. Las claves de acceso deben caducar bajo la periodicidad marcada por el administrador del sistema y no se podrán repetir.

Solución propuesta

- ***Encriptación***. Se definirá un algoritmo único con semilla única. Más adelante se explica el algoritmo (podría utilizarse otro similar).
- ***Privacidad.*** El administrador puede ver la clave, ya que es quién administra el sistema. Hay que crear una opción para que el usuario pueda cambiar la clave sin tener que recurrir al administrador. Incorporarla al menú de Usuario. Para cambiar la clave hay que solicitar la clave previa y se debe introducir por duplicado. Tampoco en este punto debe ser visible.
- ***Trazabilidad.*** Cada vez que se modifique algo de la base de datos se debe grabar el usuario con la fecha y hora de acceso.

  Hay que crear una tabla (Logs) para grabar el log de modificaciones que contenga:

  - Fecha
  - Usuario
  - Tablas modificadas
  - Módulo desde el que se accede

- ***Caducidad.*** La caducidad se debe definir en la tabla de usuarios. El administrador puede asignar a cada usuario una caducidad para sus claves. Habrá que crear un campo para introducir este parámetro. Si está a cero la clave nunca caduca.

Las claves usadas se guardarán en un campo memo encriptada igual que las claves de usuario. Cada clave estará separada por el carácter '\t'.

Solución adoptada

- ***Encriptación***. Se encriptan las claves del usuario guardándose en la base de datos encriptadas. Si la clave está en blanco no se encripta. De este modo, si un usuario se ha olvidado de la clave podemos darle acceso borrando el contenido del campo clave de la tabla "usu" para ese usuario. El usuario puede entrar sin contraseña y puede crear una nueva contraseña. La instrucción para borrar la clave de un usuario es:

  update usu set clave='' where abr='*cod_usu*'

- ***Trazabilidad***. Cada vez que se modifique se graba el usuario con la fecha y hora de acceso. Sólo en los accesos a través de la pantalla de resultados auque se la llame desde cualquier punto como validación.

  Hay que crear la tabla "Logs" que graba los accesos a ver un informe:

```
CREATE TABLE Log (
      nid NUMBER(10) NOT NULL ,
      usu NUMBER(10) NULL ,
      tablas VARCHAR2(100) NULL ,
      fecha date NULL ,
      pc varchar2(20) NULL
);
```

  En Oracle además hay que crear una secuencia.

```
      CREATE SEQUENCE OL_SEQ_LOG;
```

- ***Caducidad.*** La caducidad se define en la tabla de usuarios. El administrador puede asignar a cada usuario una caducidad para sus claves. Se han creado los siguientes campos:

  1. Clave_ini (datetime) Contiene la fecha en que se asigno la clave actual.
  2. Expira (int) Contiene el periodo en días para que caduque la clave a contar desde la fecha inicial (contenida en clave_ini).

  Las siguientes instrucciones permiten modificar la base de datos para soportar estos cambios:

```
Alter table usu add clave_ini date;
```

```
Alter table usu add expira number(10) default(0);
```

Modelo de encriptación

La encriptación se realiza de la siguiente manera:

1. Se pone un punto medio al principio para indicar que es una cadena codificada: "·"
2. Para cada carácter se toma su valor ASCII y se le resta su posición (empezando por 0) y se codifica en hexadecimal con dos dígitos en mayúsculas.
3. A la cadena resultante se le cambian las F's opr Z's. (para despistar).

Ej 1. Encriptación

ODETE ➔ ·4Z45475749

Carácter	ASCII	Ajuste	Dec	Hex
O	79	+ 0	79	4F ➔4Z
D	68	+ 1	69	45
E	69	+ 2	71	47
T	84	+ 3	87	57
E	69	+ 4	73	49

Ej 1. Desencriptación

·564E ➔ VM NOTA: Si hubiera una Z, se sustituiría por una F

Hex	Dec	Ajuste	ASCII	Carácter
56	86	- 0	86	V
4E	78	- 1	77	M

Ej 2. Desencriptación

·61746C3C3435 ➔ asj900 NOTA: Si hubiera una Z, se sustituiría por una F

Hex	Dec	Ajuste	ASCII	Carácter
61	97	- 0	97	A
74	116	- 1	115	s
6C	108	- 2	106	j
3C	60	- 3	57	9
34	52	- 4	48	0
35	53	- 5	48	0

(Anexo C) Definiciones de Ontologías

Las siguientes definiciones han sido extraídas de:

http://es.wikipedia.org/wiki/Ontolog%C3%ADa

Según **Husserl** la ontología es una ciencia de las esencias que puede ser formal o material. La primera se dedica a las esencias formales, es decir, a las propiedades de todas las esencias. Las ontologías materiales tratan de esencias materiales y se restringen según los modos de sus objetos. Por tanto, son llamadas también "ontologías regionales". Obviamente la ontología formal abarca todas las materiales e incluso las del ser...

Heidegger afirma que existe una ontología fundamental que es llamada "metafísica de la existencia" que se encarga de descubrir "la constitución del ser de la existencia". La ontología se refiere entonces a las condiciones de posibilidad de las existencias o al ser mismo en su apertura originaria.

Además, insiste en diferenciar la metafísica de la ontología, alegando que son radicalmente distintas, pues la primera confunde ser con ente; mientras que la segunda, parte precisamente del hecho de que son diferentes.

Partiendo de una crítica de la noción de ontología como metafísica y con ella de toda la escolástica, **Hartmann** afirma que la ontología es en realidad la crítica que permite descubrir los límites de la metafísica y qué contenidos pueden ser considerados racionales o inteligibles.

El término ontología en informática hace referencia a la formulación de un exhaustivo y riguroso esquema conceptual dentro de uno o varios dominios dados; con la finalidad de facilitar la comunicación y la compartición de la información entre diferentes sistemas y entidades. Aunque toma su nombre por analogía, ésta es la diferencia con el punto de vista filosófico de la palabra ontología.

Un uso común tecnológico actual del concepto de ontología, en este sentido semántico, lo encontramos en la inteligencia artificial y la representación del conocimiento. En algunas aplicaciones, se combinan varios esquemas en una estructura de facto completa de datos, que contiene todas las entidades relevantes y sus relaciones dentro del dominio.

Los programas informáticos pueden utilizar así este punto de vista de la ontología para una variedad de propósitos, incluyendo el razonamiento inductivo, la clasificación, y una variedad de técnicas de resolución de problemas.

Típicamente, las ontologías en las computadoras se relacionan estrechamente con vocabularios fijos --una ontología fundacional -- con cuyos términos debe ser descrito todo lo demás. Debido a que esto puede ocasionar representaciones pobres para ciertos dominios de problemas, se deben crear esquemas más especializados para convertir en útiles los datos a la hora de tomar decisiones en el mundo real.

Dichas ontologías son valórales comercialmente, creándose competencia para definirlas. **Peter Murray-Rust** se ha quejado de que esto conduce a *"una guerra semántica y ontológica debido a la competencia entre estándares"*. Por consiguiente, cualquier estándar de ontología fundacional es posible que sea contestado por los agentes políticos o comerciales, cada uno con su propia idea de 'lo que existe' (en el sentido filosófico de ontología).

Una variación ha sido propuesta recientemente, véase el sitio de **MathWorld**, cuyo autor propugna que el universo se modela mejor en los términos de los programas informáticos (computacionales) que con los términos matemáticos convencionales.

(Anexo D) Herramientas Ontológicas

Las ontologías, son pues, vocabularios comunes que, junto con otras tecnologías que proveen de herramientas y lenguajes para generar marcado y procesamiento semántico. Es, pues, necesario una semántica formalizada en ontologías con el fin de que este conocimiento, sea intercambiado por los agentes de *software*.

Quizás, el mayor repositorio de ontologías y recursos sobre ontologías, que también ofrece un buscador y un navegador internos, se encuentra en:

SchemaWeb http://www.schemaweb.info/schema/BrowseSchema.aspx

Aunque también podemos encontrar en la red numerosos recursos sobre ontologías: herramientas, aplicaciones y *software*, tutoriales y acceso a ontologías publicadas en OpenDirectory: **"**http://dmoz.org/Reference/Knowledge_Management/Knowledge_Representation/Ontologies/" y en otras muchas *webs* como Kaon: http://kaon.semanticweb.org/ontologies

También existen herramientas y programas para realizar anotaciones en páginas *Web* con lenguajes de marcado propios. La mayoría de estos programas permiten describir el contenido de los documentos en forma de metadatos, soportados sobre una ontología representada en RDF Schema (RDFS) o basados en grafos conceptuales. Existen algunas listas exhaustivas de este tipo de herramientas, como las ofrecidas por xml.com en:

http://www.xml.com/2002/11/06/Ontology_Editor_Survey.html

o por *Mondeca* en:

http://mondeca-publishing.com/s/anonymous/title10403.html

He aquí algunas de ellas:

- **CORESE:** una herramienta RDF basada en grafos conceptuales. http://www-sop.inria.fr/acacia/soft/corese/
- **DOME:** Distributed Ontology Management Environment. http://dome.sourceforge.net/
- **JENA:** es un formato no propietario que ofrece un marco de recursos Java para construir aplicaciones de la Web Semántica. Ofrece un entorno para RDF, esquemas RDF y OWL e incluye un motor basado en reglas de inferencia: http://jena.sourceforge.net/

- **KAON**: es un gestor de ontologías de código abierto. Incluye un conjunto de herramientas para crear y gestionar ontologías y otras herramientas para construir aplicaciones basadas en ontologías. http://kaon.semanticweb.org/
- **KIM:** es una plataforma que permite la anotación semántica y que soporta RDF, RDFS y OWL Lite. http://www.ontotext.com/kim/index.html
- **Kowari**: escrita en Java y de código abierto que soporta RDF y OWL. http://kowari.org/
- **KPOntology:** es una biblioteca para gestionar ontologías que permite usar diferentes ontologías. Permite utilizar los lenguajes RDF, OWL y ODE. http://kpontology.isoco.net/
- **Mindswap:** un editor de ontologías hipermedia basado en OWL: http://www.mindswap.org/2004/SWOOP/
- **OntoEdit:** para construir ontologías usando significaciones gráficas. http://www.ontoknowledge.org/tools/ontoedit.shtml
- **OntoLingua:** provee, en un entorno colaborativo, un buscador, generador y modificador de ontologías: http://www.ksl.stanford.edu/software/ontolingua/
- **Ontomat**: herramienta de código abierto que soporta marcado OWL. http://annotation.semanticweb.org/ontomat/index.html
- **Ontopia:** ofrece un conjunto de herramientas para desarrollar y mantener ontologías. http://www.ontopia.net/solutions/products.html
- **OntoShare:** es una ontología para la WWW basada en RDF y pensada para compartir conocimiento en un ambiente compartido. http://semanticweb2002.aifb.uni-karlsruhe.de/proceedings/Research/davies.pdf
- **OntoWeaver**: permite el diseño y desarrollo de sitios Web basándose en ontologías. http://kmi.open.ac.uk/projects/akt/ontoweaver/
- **OntoWebber**: sistema de gestión de sitios Web basado en ontologías. http://www-db.stanford.edu/OntoAgents/OntoWebber/
- **pOWL:** plataforma que utiliza RDFS/OWL. http://powl.sourceforge.net/
- **OpenCyC**: http://www.cyc.com/opencyc
- **OWLIM:** es un repositorio semántico para la capa de almacenamiento e inferencia de la base de datos RDF de Sesame: http://www.ontotext.com/owlim/index.html

- **PROTÉGÉ:** editor de ontologías de código abierto para construir ontologías sobre RDFS, OWL y XML Schema. Es uno de los editores más utilizados. http://protege.semanticweb.org.
- **PROTON Ontology (PROTo ONtology):** plataforma que sirve para construir ontologías, la anotación semántica, la indización y la recuperación de información. http://proton.semanticweb.org/ Esta herramienta ha sido desarrollada dentro del programa SEKT (Semantically-Enabled Knowledge Technologies) de la Unión Europea (http://www.sekt-project.com/).
- **Sesame:** es una base de datos RDF de fuente abierta con soportes de inferencia y consulta para esquemas RDF. Originalmente fue desarrollado por Aduna como un prototipo de desarrollo para el proyecto de la Unión Europea On-To-Knoledge. Ahora hay muchos voluntarios que lo desarrollan, entre ellos Ontotext. http://www.openrdf.org/
- **SUMO (Suggestd Upper Merged Ontology)**: http://www.ontologyportal.org/
- **SWOOP**: editor de ontologías hipermedia. http://www.mindswap.org/2004/SWOOP/
- **TAP Knowledge Base**: http://tap.stanford.edu
- **TM-Builder:** es un constructor de ontologías basado en Topic Maps: http://www.di.uminho.pt/~jcr/XML/publicacoes/artigos/2003/clei.pdf
- **WebODE:** permite desarrollar ontologías sobre ingeniería http://webode.dia.fi.upm.es/WebODEWeb/index.html
- **WebOnto: c**onsiste en un *applet* de Java que permite navegar y editar modelos de conocimiento sobre la Web. http://kmi.open.ac.uk./projects/webonto
- **WordNet**: http://wordnet.princeton.edu
- **WSMO Studio:** es un editor de ontologías para modelado de servicios de la Web Semántica: http://www.wsmostudio.org/ desarrollado por el Web Service Modeling Ontology Group: http://www.wsmo.org/

Además, para potenciar el uso de ontologías, se han desarrollado aplicaciones específicas de búsqueda de ontologías en la Web, tales como **OntoAgent** http://www.i-u.de/schools/eberhart/ontoagent/ para que indiquen a los usuarios las ontologías ya existentes y sus características para poder utilizarlas en su sistema; o herramientas como **OntoJava** http://www.i-u.de/schools/eberhart/ontoJava/ un compilador que traslada ontologías realizadas con Protégé a bases de datos de objetos Java, **OntoSQL** que permite usar una base de datos relacional como una base de

datos deductiva http://www.i-u.de/schools/eberhart/ontosql/ o **RDFCrawler** para buscar y escanear datos RDF en la Web: http://www.i-u.de/schools/eberhart/rdf/

Los proyectos para desarrollar ontologías son muchos y de muy distinto signo. He aquí algunos proyectos de desarrollo de ontologías:

- **Annotea:** el proyecto Annotea desarrolló varias herramientas para compartir anotaciones de metadatos basados en Web, esquemas RDF, etc. http://www.w3.org/2001/Annotea/
- **CORES.** Un foro para compartir vocabularios de metadatos. http://www.cores-eu.net
- **DBin**: http://www.dbin.org/
- **FOMI:** Formal Ontologies Meet Industry es un foro para compartir ontologías relacionadas con la industria, la ingeniería, los negocios, el comercio electrónico, etc. http://fandango.cs.unitn.it/fomi/
- **KAON**: El proyecto KAON1 desarrolló algunas herramientas basabas en RDF http://kaon.semanticweb.org/, mientras que el proyecto KAON 2 desarrolla herramientas basadas en OWL-DL.http://kaon2.semanticweb.org/
- **Knowledge Web**: http://knowledgeweb.semanticweb.org/
- **OntoWeb:** es una red de intercambio de información sobre ontologías. http://ontoweb.org/
- **SchemaWeb:** se trata de un directorio de esquemas RDF expresado en lenguajes de esquemas RDFS, OWL y DAML+OIL con numerosos recursos y herramientas sobre ontologías. http://www.schemaweb.info/
- **SWAD- Europe:** un proyecto de la Unión Europea para el desarrollo de la Web Semántica http://www.w3.org/2001/sw/Europe que ofrece un weblog http://esw.w3.org/mt/esw/
- **W3C Semantic Web:** que a través del Semantic Web Interest Group ofrece los estándares establecidos por el W3C: http://www.w3.org/2001/sw

(Anexo E): Instalaciones a tener en cuenta

Instalación de Eclipse

En caso de querer cambiar alguno de los módulos o tener que volver a compilar el programa, se puede instalar Eclipse, ya que ha sido la herramienta utilizada, pero hay otros como:

- NetBeans http://www.netbeans.org/downloads/index.html.
- JBuilder http://www.borland.co/us/products/builder/index.html.
- Jdeveloper http://www.oracle.com/technology/products/jdev/index.html.

Eclipse es una plataforma de software de Código abierto independiente de una plataforma para desarrollar Aplicaciones. Esta plataforma, es usada para desarrollar entornos integrados de desarrollo (IDE), como el IDE de Java (Java Development Toolkit JDT).

Eclipse fue desarrollado originalmente por IBM como el sucesor de VisualAge. Ahora es desarrollado por la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

En primer lugar descargaremos Eclipse, desde la Web:

http://www.eclipse.org/downloads/ (unos 121 MB de tamaño), si estamos trabajando en entorno Windows pulsaremos sobre *Download now: Eclipse SDK 3.2.2,Windows (o la versión actualizada de ese momento)*:

Para trabajar con Eclipse en otras plataformas / sistemas operativos (Linux, Solaris, Mac, AIX,HP-UX,etc.) pulsaremos en

"Other downloads for 3.2.2”

(*http://download.eclipse.org/eclipse/downloads/drops/R-3.2.2-200702121330*):

La versión 3.2.2 de Eclipse necesita, al menos la versión 1.4 de Java runtime environment (JRE) o Java development kit (JDK), es recomendable tenerla instalada, para descargarla: http://www.Java.com/es/download/manual.jsp

Podemos descargar algunos ejemplos de aplicaciones desarrolladas en Eclipse, desde la Web:

http://www.eclipse.org/downloads/download.php?file=/eclipse/downloads/drops/R3.2.2-200702121330/eclipse-examples-3.2.2-win32.zip

Instalar paquete de idiomas Eclipse

Para traducir Eclipse al castellano descargaremos el fichero .zip de:

http://download.eclipse.org/eclipse/downloads/drops/L-3.2.1_Language_Packs-200609210945/index.php (Comprobar que sea el adecuado para la versión instalada).

Los ficheros descargados son:

Nombre fichero	Descripción	Tamaño
eclipse-SDK-3.2.2-win32.zip	Eclipse SDK 3.2.2 (Windows)	121 MB
NLpack1-eclipse-SDK-3.2.1-win32.zip	Paquete de idiomas (español, etc.) para Eclipse SDK 3.2.1	55 MB
eclipse-examples-3.2.2-win32.zip	Ejemplos de Eclipse 3.2.2	3,58 MB
jre-6u1-windows-i586-p-iftw.exe	Java runtime environment 1.6.0_01	360 KB

Empezaremos instalando JRE (si aún no lo tenemos), para ello ejecutaremos "jre-6u1-windows-i586-p-iftw.exe".

Eclipse no necesita instalación, es suficiente con descomprimir el fichero "eclipse-SDK-3.2.2-win32.zip" en la carpeta que deseemos:

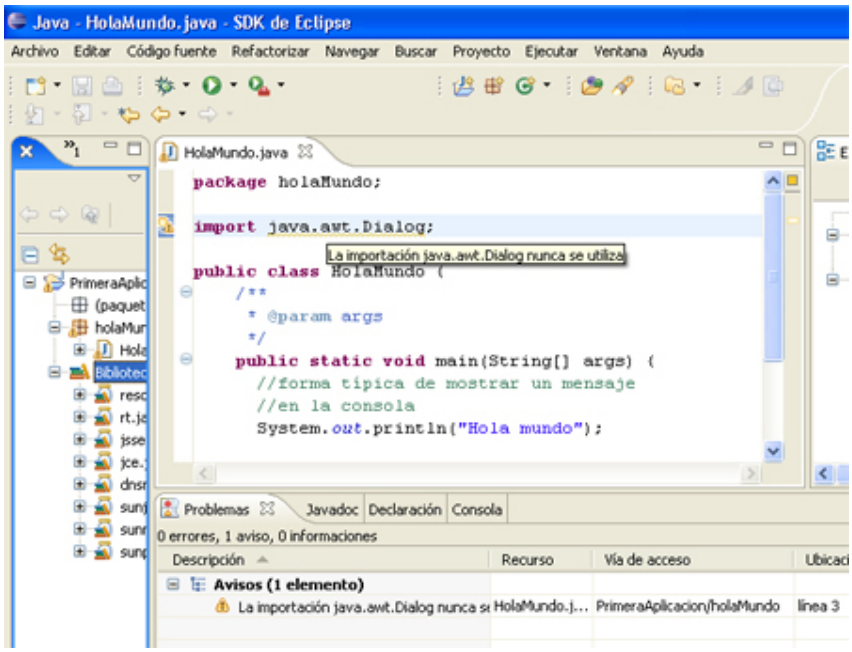
Antes de ejecutar Eclipse, descomprimiremos el fichero "NLpack1-eclipse-SDK-3.2.1-win32.zip" en la misma carpeta que Eclipse:

Reemplazaremos todos los ficheros, pulsando "Yes to All":

Una vez descomprimido Eclipse y el paquete de idiomas, en la carpeta donde hayamos descomprimido Eclipse ejecutaremos el fichero *eclipse.exe*:

Ventajas de eclipse

Cuando se produce un error o un aviso se puede ver en la pestaña "Problemas". Nos indicará una descripción del problema y la línea de código donde se produce. En la línea de código también nos la mostrará subrayada (rojo si es un error y amarillo si es un aviso). En nuestro caso nos está avisando de que hemos importado la clase "awt.Dialog" pero no la hemos utilizado:



Instalación y configuración de Eclipse SQL Explorer en Eclipse (Java)

A continuación se describe cómo configurar Eclipse SQL Explorer. Se trata de un Plug-in gratuito para Eclipse que permite acceder y manipular bases de datos (Oracle, MySQL, SQL Server, etc.) desde el propio IDE de Eclipse (perspectiva).

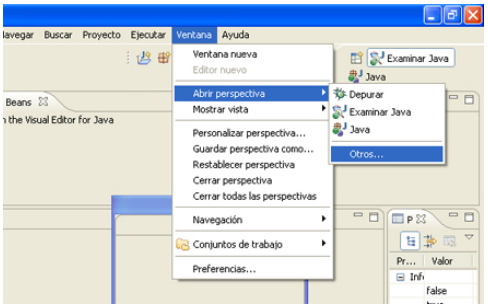
Eclipse SQL Explorer se trata de un Plug-in de código fuente abierto, disponible en: *http://sourceforge.net/projects/eclipsesql*

Que permite la manipulación de bases de datos desde el propio Eclipse (ejecutar consultas SQL de todo tipo mostrando el resultado (SQL Results), estructura de la base de datos (Database structure), detalles de la base de datos (Database Details), historia del SQL ejecutado (SQL History), admite múltiples conexiones a múltiples motores de base de datos (Axion, HSQLDB, IBM DB2, InstantDB, InterClient, Firebird, Microsoft SQL Server, Mimer SQL, MySQL, Oracle, Pointbase, PostgreSQL, Progress OpenEdge, SAPDB, Sunopsis XML, Sybase, ThinkSQL, etc.).

Lo descargaremos desde la Web anterior, será un fichero .zip que descomprimiremos en la carpeta de instalación de Eclipse, se descomprimirá la carpeta

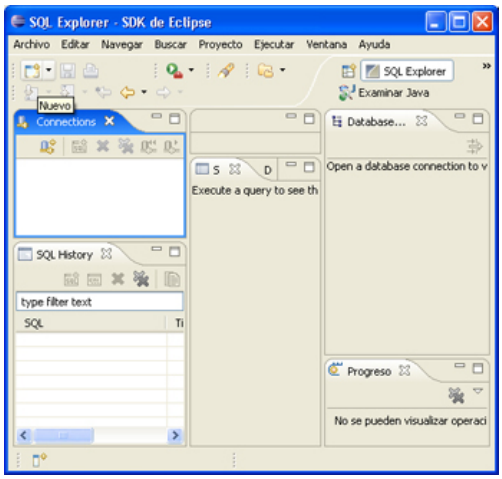
net.sourceforge.sqlexplorer_3.0.0.20060901 dentro de *Plug-ins*. Si tenemos abierto Eclipse lo cerraremos y lo volveremos a abrir.

Para poder visualizar la perspectiva de SQL Explorer en Eclipse, abriremos Eclipse y pulsaremos en "Ventana" - "Abrir perspectiva" - "Otros...":



En la ventana "Abrir perspectiva" seleccionaremos "SQL Explorer" y pulsaremos "Aceptar":

El IDE de Eclipse mostrará este aspecto:



A continuación indicaremos cómo configurarlo para MySQL y para Oracle. Para ello descargaremos e instalaremos (con descomprimirlo es suficiente) lo siguiente:

- Para MySQL ➔Connector/J de MySQL (JDBC), para ello accederemos a:

  http://dev.mysql.com/downloads/connector/j/5.1.html

- Para Oracle ➔ Obtener de Oracle 10g Release2 controlador JDBC de Oracle. Si usted ya tiene instalado Oracle 10g XE, entonces usted puede encontrar el controlador apropiado en:

  oraclexe_root\app\oracle\product\10.2.0\server\jdbc\lib\ojdbc14.jar.

  También se puede descargar el controlador JDBC 10g R2 en:

http://www.oracle.com/technology/software/tech/Java/sqlj_jdbc/index.html

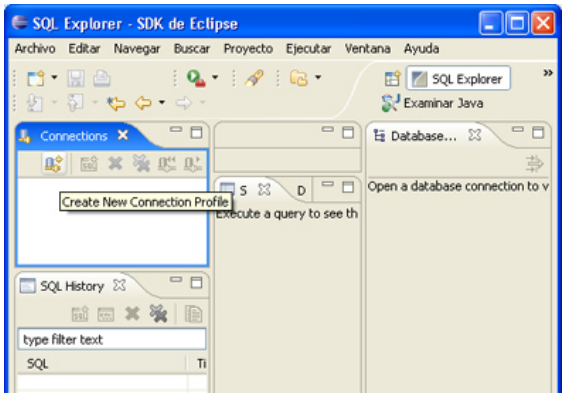
Copie el ***ojdbc14.jar datos.***

Oracle proporciona Oracle 10g Express Edition (XE), una edición gratuita de su servidor de base de datos. Para poder probar un controlador alternativo hay que configurar el controlador JDBC de 10g por Oracle para su uso. Este controlador se refiere a menudo como el Oracle Thin Client. A veces el comportamiento del controlador varía entre dos proveedores, por lo que su a menudo muy útil para tratar las versiones alternas.

Descomprimiremos el fichero JDBC descargado anteriormente en cualquier carpeta, en nuestro caso en

C:/eclipse/mysql-connector-Java-5.1.0

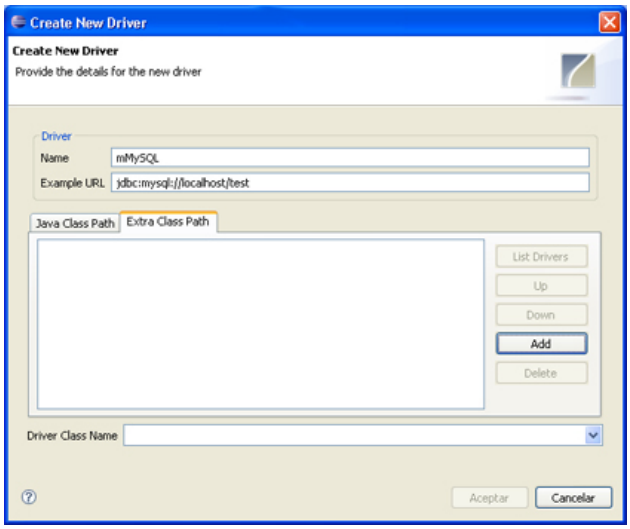
Para acceder a MySQL u Oracle desde Eclipse con SQL Explorer y el JDBC abriremos Eclipse, en la ventana "Connections" pulsaremos en el primer botón "Create New Connection Profile", la siguiente configuración es para MySQL, apareciendo la de Oracle más adelante:



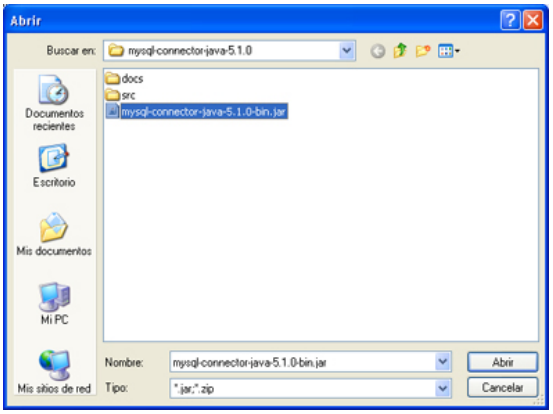
En la ventana de "Create New Connection Profile" introduciremos los siguientes datos:

- Name: el nombre del nuevo perfil de conexión, por ejemplo "pruebaConexionMySQL"
- Driver: en el desplegable nos aparecerán drivers por defecto, en nuestro caso, vamos a utilizar el JDBC descargado anteriormente, para ello pulsaremos en "New Driver":

Le pondremos un nombre al nuevo driver en "Name", por ejemplo "mMySQL", introduciremos el ejemplo de URL en "Example URL": *jdbc:mysql://localhost/test*. Pulsaremos en la pestaña / solapa "Extra Class Path" y pulsaremos en el botón "Add" que aparece en la parte derecha de la ventana:



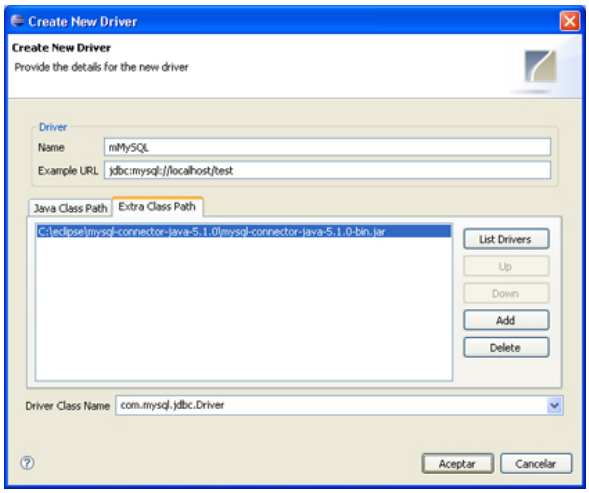
Seleccionaremos la ubicación del driver JDBC descargado anteriormente de la Web de MySQL, en concreto el fichero .jar: *mysql-connector-Java-5.1.0-bin.jar*:



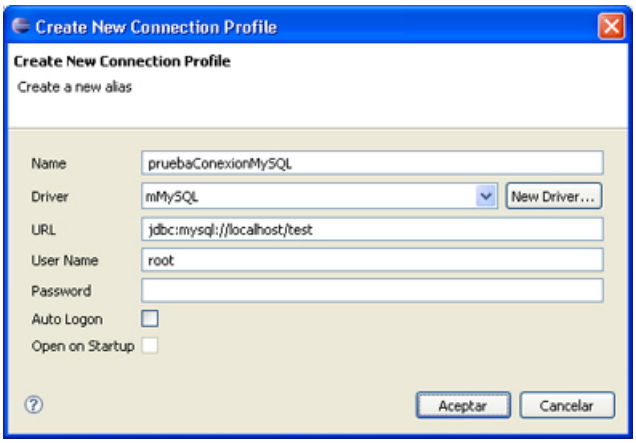
Por último en la ventana de "Create New Driver" indicaremos (en el desplegable que aparece en la parte inferior de la ventana) el "Driver Class Name": *com.mysql.jdbc.Driver*. Pulsaremos "Aceptar" para volver a la ventana "Create New Connection Profile":

- En "Driver" seleccionaremos el anteriormente dado de alta "mMySQL".
- URL: jdbc:mysql://localhost/test (donde "localhost" será el nombre o la IP del servidor de MySQL, "test" será el nombre de la base de datos (esquema) de MySQL al que nos conectaremos.
- User Name: nombre de usuario de la base de datos con el que nos conectaremos.
- Password: contraseña del usuario de la bd.
- Auto Logon: marcaremos esta opción si hemos introducido usuario y contraseña y queremos que la conexión se realice de forma automática, sin pedir estos datos.

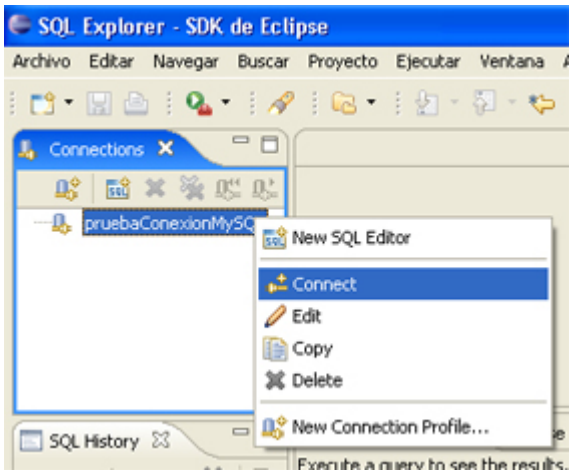
- Open on Startup: marcaremos este check para conectar directamente con el driver indicado al abrir el entorno Eclipse SQL Explorer.



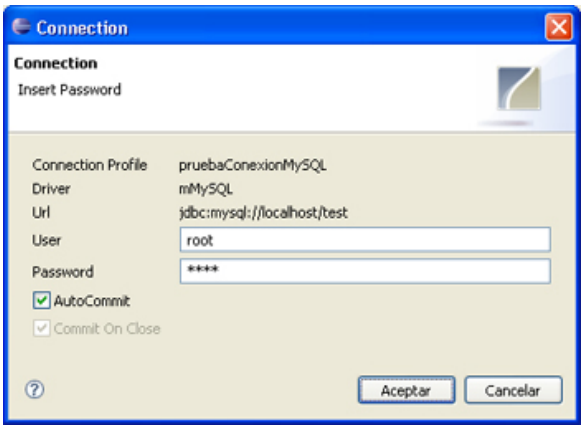
Tras rellenar todas las opciones pulsaremos en "Aceptar" para crear el nuevo perfil de conexión:



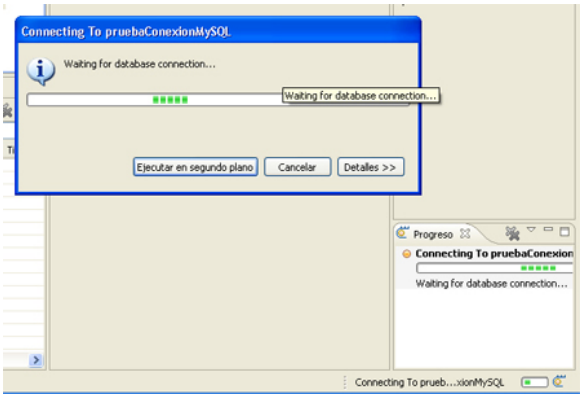
Para realizar una conexión pulsaremos con el botón derecho del ratón sobre el perfil de conexión creado "pruebaConexionMySQL" y seleccionaremos "Connect":



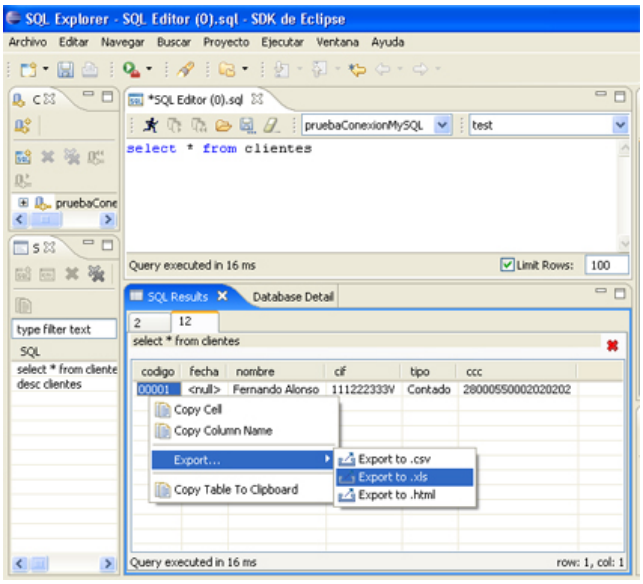
Indicaremos usuario y contraseña y pulsaremos "Aceptar":



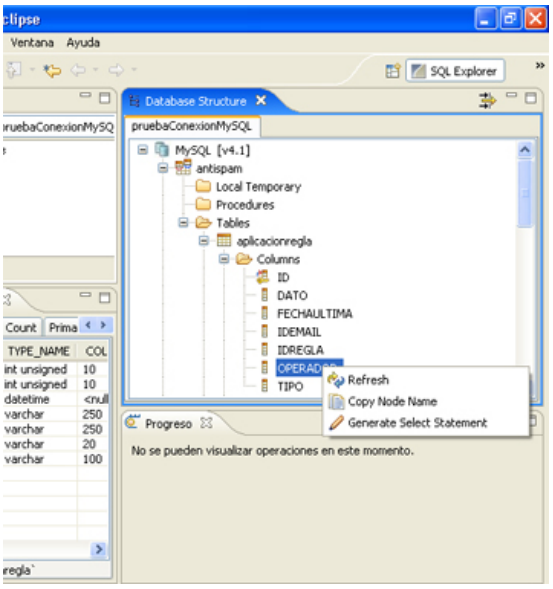
Mostrará una Eclipse SQL Explorer una ventana indicando el proceso de la conexión:



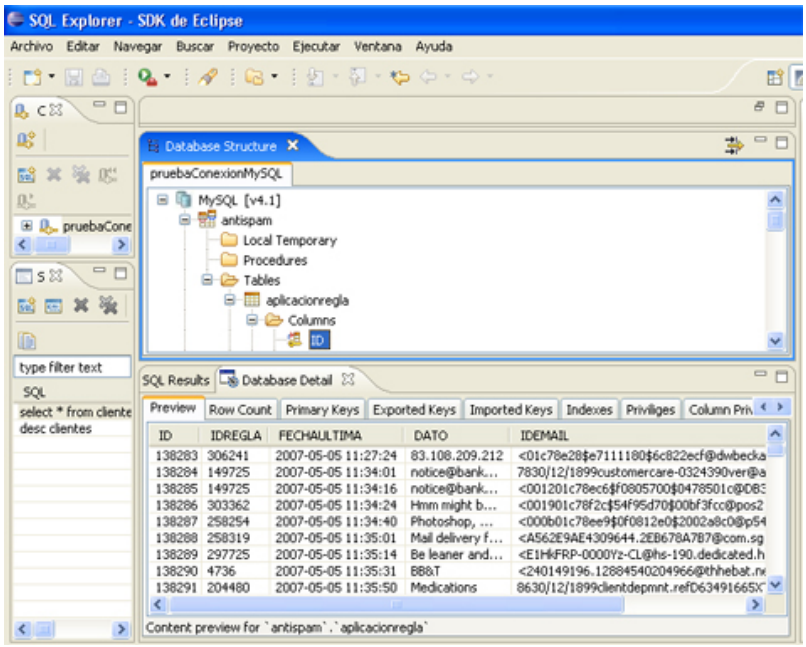
Incluso podremos exportar el resultado de las consultas a .xls (Excel), csv, html):



Con Eclipse SQL Explorer también podremos ver en forma de árbol todos los esquemas de nuestro servidor con todas las tablas y sus campos, índices, procedimientos, etc. Utilizando para ello "Database Structure":



En la ventana "Database detail" podremos ver todos los datos posibles del esquema y tabla seleccionados en "Database Structure": vista previa de todos los registros y campos (Preview), número de registros (Row count), claves primarias (Primary Keys), claves exportadas/foráneas (Exported Keys), índices (Indexes), privilegios (Priviliges), privilegios de columna (Column Priviliges), Row Ids, Versions, etc:



Con lo cual Eclipse SQL Explorer se convierte en una herramienta sumamente útil para realizar aplicaciones con acceso a base de datos. Permite de una forma rápida ejecutar consultas SQL, mostrar los resultados, ver la estructura de las tablas, etc.

Para realizar este manual hemos utilizado:

- Microsoft Windows XP SP2 (Sistema Operativo).
- Eclipse 3.2.0.
- SQL Explorer 3.0.0.200609.
- MySQL Server 4.1.

Archivos necesarios:

ARCHIVO	DESCRIPCIÓN	TAMAÑO
mysql-connector-Java-5.1.0.zip	Conector JDBC para MySQL y Java. Descargable desde *http://www.mysql.com*	7,72 MB
sqlexplorer_plugin_3.0.0.20060901.zip	Plugin para Eclipse y acceso a cualquier base de datos. Descargable desde *http://sourceforge.net/projects/eclipsesql*	1.41MB

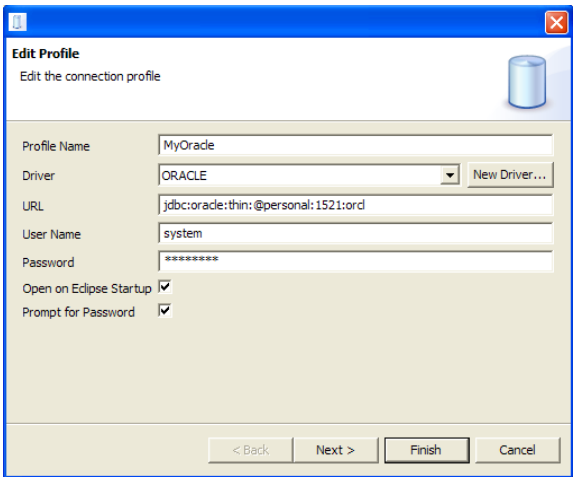
Configuración para Oracle

Seguiremos los mismos pasos que para MySQL, pero poniendo los valores siguientes:

- Profile Name: *MyOracle* (Nombre de usuario)
- Driver: *ORACLE* (del nuevo driver seleccionado)
- Name: *ORACLE*
- Example URL: *jdbc:oracle:thin:@[host]:1521:XE*
- Donde [host] -- aquí el host = personal
- XE = orcl{tomar de D:\oracle\product\10.2.0\db_1\NETWORK\ADMIN\tnsnames.ora}

D:\oracle\product\10.2.0\db_1\jdbc\lib\classes12.zip

- Driver Class Name : oracle.jdbc.OracleDriver
- URL : jdbc:oracle:thin:@personal:1521:orcl
- UserName : system
- Password : password



Instalar Oracle Database 10g Release 2 (10.2.0.1.0) en Windows XP

Oracle es un sistema de gestión de base de datos relacional (RDBMS Relational Data Base Management System), fabricado por Oracle Corporation. Oracle es uno de los sistemas de bases de datos más completos.

En primer lugar descargaremos el fichero de instalación de Oracle, desde su página Web (es gratuito siempre que no se utilice con fines comerciales), el enlace es:

http://www.oracle.com/technology/software/products/database/oracle10g/index.html

Seleccionaremos el enlace:

Oracle Database 10g Release 2 (10.2.0.1.0) for Microsoft Windows

O se seleccionará el correspondiente a la versión de Windows que se utilice. Será necesario ser usuario registrado de Oracle (es gratuito) para realizar la descarga. Se descargará el fichero 10201_database_win32.zip, de unos 655 MB.

Una vez descargado el fichero de instalación y descomprimido, ejecutaremos el fichero "setup.exe" para iniciar el programa de instalación que muestra varias opciones de instalación.

- **Instalación Básica**: seleccione este método de instalación si desea instalar rápidamente la base de datos Oracle 10g. Este método necesita una intervención mínima del usuario. Instala el software y, opcionalmente, crea una base de datos de uso general con el esquema SAMPLE y el tablespace EXAMPLE, con la información especificada en la pantalla inicial. Nota: Si no especifica toda la información necesaria, Installer muestra las pantallas de instalación avanzada. Utiliza los valores especificados como valores por defecto en las pantallas correspondientes.
- **Instalación Avanzada**: seleccione este método de instalación para cualquiera de las siguientes tareas: realizar una instalación personalizada del software o seleccionar una configuración diferente de la base de datos. Instalar Oracle Real Application Clusters. Actualizar una base de datos existente. Seleccionar un juego de caracteres de la base de datos o idiomas de producto diferentes. Crear una base de datos en otro sistema de archivos del software. Configurar la gestión automática de almacenamiento (ASM) o utilizar dispositivos raw para el almacenamiento en la base de datos. Especificar contraseñas diferentes para esquemas administrativos. Configurar copias de seguridad automáticas o notificaciones de Oracle Enterprise Manager.

Para la Instalación Básica, las opciones son:

- **Ubicación del Directorio Raíz de Oracle**: unidad y carpeta donde se realizará la instalación de Oracle 10g.

- **Tipo de instalación**:
    - **Enterprise Edition**: este tipo de instalación está diseñado para aplicaciones a nivel de empresa. Está diseñado para el Procesamiento de Transacciones en Línea (OLTP) de alta seguridad y de importancia crítica y para entornos de almacenes de datos. Si selecciona este tipo de instalación, se instalan todas las opciones de Enterprise Edition con licencias independientes.
    - **Standard Edition**: este tipo de instalación está diseñado para aplicaciones a nivel de departamento o grupo de trabajo o para pequeñas y medianas empresas. Está diseñado para proporcionar las opciones y servicios de gestión de bases de datos relacionales esenciales. Si selecciona este tipo de instalación, deberá adquirir licencias adicionales para instalar otras opciones de Enterprise Edition.
    - **Personal Edition** (sólo para Sistemas Operativos Windows): este tipo de instalación instala el mismo software que el tipo de instalación Enterprise Edition, pero sólo soporta un entorno de desarrollo y despliegue monousuario que debe ser totalmente compatible con Enterprise Edition y Standard Edition.
- **Crear base de datos inicial**: creará una base de datos de uso general durante la instalación. Si no la selecciona, Installer sólo instala el software. Si no desea crear una base de datos durante la instalación, puede utilizar el Asistente de Configuración de Bases de Datos (DBCA) para crearla después de instalar el software.
    - Nombre de la Base de Datos Global: nombre con el que se identificará la base de datos, máximo 8 caracteres.
    - Contraseña de Base de Datos: contraseña que se asignará a los usuarios SYS, SYSTEM, SYSMAN y DBSNMP.

(Anexo F): Ejecución

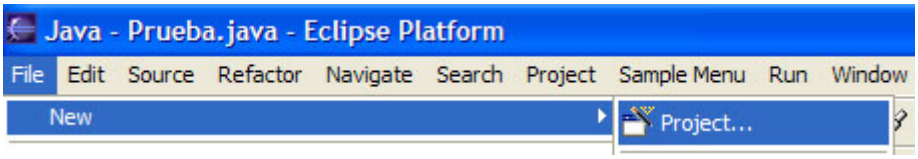
Para poder ejecutar el programa se han de cargar los script correspondientes de la base de datos.

El programa se encuentra compilado, para ejecutarlo podemos hacerlo desde Windows o desde Eclipse, u otro programa que permita abrir ficheros .class. Para ejecutar un fichero compilado ".class" desde Windows se necesita tener instalada una maquina virtual de Java, desde Internet hay múltiples descargables, y también hay que configurar los path.

Para ejecutar un .class (supongamosPrueba.class). Simplemente desde la línea de comandos " Java Prueba"

En el entorno de desarrollo Eclipse todo archivo se almacena dentro de un proyecto. Esto quiere decir que todo documento, carpeta, archivo de código fuente (.Java) y código compilado (.class) tiene que estar contenido dentro de un proyecto.

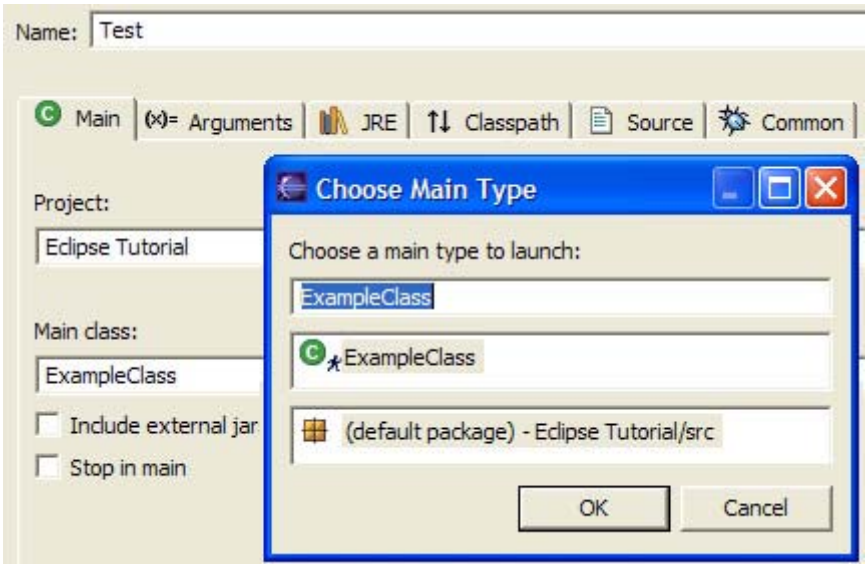
Es necesario crear un nuevo proyecto no sólo para desarrollar un nuevo programa de Java, sino para editar también archivos ya existentes (como por ejemplo, un programa ".Java" almacenado descargado o guardado en una carpeta). Para crear un nuevo proyecto, seleccione en la línea de menús principal "File > New > Project...". También es posible seleccionar "New > Project..." haciendo clic derecho en cualquier parte una vista de Eclipse (como por ejemplo, el Package Explorer o el Resource Navigator).



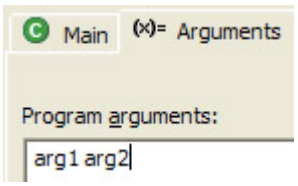
Ejecutar

Para ejecutar un programa dentro de Eclipse hay que seleccionar "Run > Run..." del menú principal. Dentro de "Configurations" se almacenan diferentes configuraciones de ejecución. Hay cuatro tipos de configuraciones de ejecución: Java Applet (para applets Web), Java Application (para programas normales de Java), JUnit (casos de prueba) y Run-Time Workbench (otras instancias de Eclipse que permiten probar nuevos módulos de Eclipse).

Así pues, para ejecutar un programa de Java normal debería seleccionarse "Java Application" y pulsar el botón "New" para crear una nueva configuración. Dentro de la pestaña "Main" hay que dar nombre a la nueva configuración seleccionar el proyecto que contiene la clase con el método main y seleccionar dicha clase. El método "main" es el punto de ejecución de un programa Java, y se representa como un pequeño icono de un hombre corriendo al lado del icono de la clase.



Si se desea pasar argumentos al método main (en la forma de "String[] args"), no hay más que hacer clic en la solapa de "Arguments" y escribir esos argumentos separados por espacio dentro de la zona en blanco de "Program Arguments".



Finalmente, hacer clic en el botón "Run" lanzará la ejecución del programa seleccionado. Una forma más rápida de arrancar la ejecución del programa recientemente ejecutado es pulsar en el icono de un hombre corriendo que aparece en el menú principal. Pulsar la flecha próxima a este icono mostrará otras configuraciones de ejecución recientemente utilizadas.

En Eclipse 3.0 el icono anterior ha sido reemplazado por una flecha blanca dentro de un círculo verde.